

# OCF Security Specification

VERSION 2.0.4 | July 2019



**OPEN** CONNECTIVITY  
FOUNDATION™

CONTACT [admin@openconnectivity.org](mailto:admin@openconnectivity.org)

Copyright Open Connectivity Foundation, Inc. © 2019.  
All Rights Reserved.

## 1 **LEGAL DISCLAIMER**

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

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

15 Copyright © 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 CONTENTS

18	1	Scope .....	1
19	2	Normative References .....	1
20	3	Terms, definitions, and abbreviated terms .....	3
21	3.1	Terms and definitions.....	3
22	3.2	Abbreviated terms.....	6
23	4	Document Conventions and Organization .....	10
24	4.1	Conventions.....	10
25	4.2	Notation.....	10
26	4.3	Data types .....	11
27	4.4	Document structure.....	11
28	5	Security Overview.....	12
29	5.1	Preamble .....	12
30	5.2	Access Control.....	14
31	5.2.1	ACL Architecture .....	15
32	5.2.2	Access Control Scoping Levels.....	19
33	5.3	Onboarding Overview .....	21
34	5.3.1	Onboarding General .....	21
35	5.3.2	Onboarding Steps.....	23
36	5.3.3	Establishing a Device Owner .....	24
37	5.3.4	Provisioning for Normal Operation .....	25
38	5.3.5	Device Provisioning for OCF Cloud and Device Registration Overview .....	25
39	5.3.6	OCF Compliance Management System.....	25
40	5.4	Provisioning.....	26
41	5.4.1	Provisioning General .....	26
42	5.4.2	Provisioning other services.....	26
43	5.4.3	Provisioning Credentials for Normal Operation .....	27
44	5.4.4	Role Assignment and Provisioning for Normal Operation .....	27
45	5.4.5	ACL provisioning .....	27
46	5.5	Secure Resource Manager (SRM).....	27
47	5.6	Credential Overview.....	28
48	6	Security for the Discovery Process .....	29
49	6.1	Preamble .....	29
50	6.2	Security Considerations for Discovery.....	29
51	7	Security Provisioning.....	32
52	7.1	Device Identity.....	32
53	7.1.1	General Device Identity .....	32
54	7.1.2	Device Identity for Devices with UAID [Deprecated].....	32
55	7.2	Device Ownership.....	32
56	7.3	Device Ownership Transfer Methods.....	33
57	7.3.1	OTM implementation requirements .....	33
58	7.3.2	SharedKey Credential Calculation .....	35
59	7.3.3	Certificate Credential Generation.....	35

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

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

148	13.3.3	Key Formatting .....	112
149	13.3.4	Credential Refresh Method Details .....	112
150	13.4	Certificate Revocation List .....	114
151	13.4.1	CRL Resource Definition .....	114
152	13.5	ACL Resources .....	114
153	13.5.1	ACL Resources General .....	114
154	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures. ....	114
155	13.5.3	ACL Resource .....	115
156	13.6	Access Manager ACL Resource .....	120
157	13.7	Signed ACL Resource .....	121
158	13.8	Provisioning Status Resource .....	122
159	13.9	Certificate Signing Request Resource .....	127
160	13.10	Roles Resource .....	128
161	13.11	Account Resource .....	129
162	13.12	Account Session Resource .....	131
163	13.13	Account Token Refresh Resource .....	132
164	13.14	Security Virtual Resources (SVRs) and Access Policy .....	133
165	13.15	SVRs, Discoverability and OCF Endpoints .....	133
166	13.16	Additional Privacy Consideration for Core and SVRs Resources .....	133
167	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ....	133
168	13.16.2	Privacy Protecting the Device Identifiers .....	136
169	13.16.3	Privacy Protecting the Protocol Independent Device Identifier .....	136
170	13.16.4	Privacy Protecting the Platform Identifier .....	137
171	13.17	Easy Setup Resource Device State .....	137
172	14	Security Hardening Guidelines/ Execution Environment Security .....	140
173	14.1	Preamble .....	140
174	14.2	Execution Environment Elements .....	140
175	14.2.1	Execution Environment Elements General .....	140
176	14.2.2	Secure Storage .....	140
177	14.2.3	Secure execution engine .....	143
178	14.2.4	Trusted input/output paths .....	143
179	14.2.5	Secure clock .....	143
180	14.2.6	Approved algorithms .....	143
181	14.2.7	Hardware tamper protection .....	144
182	14.3	Secure Boot .....	144
183	14.3.1	Concept of software module authentication .....	144
184	14.3.2	Secure Boot process .....	146
185	14.3.3	Robustness Requirements .....	146
186	14.4	Attestation .....	146
187	14.5	Software Update .....	146
188	14.5.1	Overview: .....	146
189	14.5.2	Recognition of Current Differences .....	147
190	14.5.3	Software Version Validation .....	148
191	14.5.4	Software Update .....	148

192	14.5.5	Recommended Usage.....	149
193	14.6	Non-OCF Endpoint interoperability.....	149
194	14.7	Security Levels .....	149
195	14.8	Security Profiles.....	150
196	14.8.1	Security Profiles General.....	150
197	14.8.2	Identification of Security Profiles (Normative) .....	150
198	14.8.3	Security Profiles .....	152
199	15	Device Type Specific Requirements.....	157
200	15.1	Bridging Security .....	157
201	15.1.1	Universal Requirements for Bridging to another Ecosystem .....	157
202	15.1.2	Additional Security Requirements specific to Bridged Protocols .....	158
203	Annex A	(informative) Access Control Examples.....	160
204	A.1	Example OCF ACL Resource .....	160
205	A.2	Example AMS .....	160
206	Annex B	(Informative) Execution Environment Security Profiles .....	162
207	Annex C	(normative) Resource Type definitions.....	163
208	C.1	List of Resource Type definitions .....	163
209	C.2	Account Token.....	163
210	C.2.1	Introduction .....	163
211	C.2.2	Well-known URI.....	163
212	C.2.3	Resource type .....	163
213	C.2.4	OpenAPI 2.0 definition.....	163
214	C.2.5	Property definition .....	166
215	C.2.6	CRUDN behaviour .....	167
216	C.3	Access Control List [DEPRECATED].....	167
217	C.4	Access Control List-2.....	167
218	C.4.1	Introduction .....	167
219	C.4.2	Well-known URI.....	167
220	C.4.3	Resource type .....	167
221	C.4.4	OpenAPI 2.0 definition.....	167
222	C.4.5	Property definition .....	176
223	C.4.6	CRUDN behaviour .....	177
224	C.5	Managed Access Control .....	177
225	C.5.1	Introduction .....	177
226	C.5.2	Well-known URI.....	177
227	C.5.3	Resource type .....	177
228	C.5.4	OpenAPI 2.0 definition.....	177
229	C.5.5	Property definition .....	181
230	C.5.6	CRUDN behaviour .....	181
231	C.6	Credential.....	182
232	C.6.1	Introduction .....	182
233	C.6.2	Well-known URI.....	182
234	C.6.3	Resource type .....	182
235	C.6.4	OpenAPI 2.0 definition.....	182

236	C.6.5	Property definition .....	192
237	C.6.6	CRUDN behaviour .....	192
238	C.7	Certificate Revocation.....	193
239	C.7.1	Introduction .....	193
240	C.7.2	Well-known URI .....	193
241	C.7.3	Resource type .....	193
242	C.7.4	OpenAPI 2.0 definition.....	193
243	C.7.5	Property definition .....	195
244	C.7.6	CRUDN behaviour .....	195
245	C.8	Certificate Signing Request.....	196
246	C.8.1	Introduction .....	196
247	C.8.2	Well-known URI .....	196
248	C.8.3	Resource type .....	196
249	C.8.4	OpenAPI 2.0 definition.....	196
250	C.8.5	Property definition .....	197
251	C.8.6	CRUDN behaviour .....	198
252	C.9	Device Owner Transfer Method.....	198
253	C.9.1	Introduction .....	198
254	C.9.2	Well-known URI .....	198
255	C.9.3	Resource type .....	198
256	C.9.4	OpenAPI 2.0 definition.....	198
257	C.9.5	Property definition .....	201
258	C.9.6	CRUDN behaviour .....	203
259	C.10	Device Provisioning Status .....	203
260	C.10.1	Introduction .....	203
261	C.10.2	Well-known URI .....	204
262	C.10.3	Resource type .....	204
263	C.10.4	OpenAPI 2.0 definition.....	204
264	C.10.5	Property definition .....	207
265	C.10.6	CRUDN behaviour .....	212
266	C.11	Asserted Roles .....	212
267	C.11.1	Introduction .....	212
268	C.11.2	Well-known URI .....	212
269	C.11.3	Resource type .....	212
270	C.11.4	OpenAPI 2.0 definition.....	213
271	C.11.5	Property definition .....	222
272	C.11.6	CRUDN behaviour .....	222
273	C.12	Signed Access Control List .....	222
274	C.12.1	Introduction .....	222
275	C.12.2	Well-known URI .....	222
276	C.12.3	Resource type .....	222
277	C.12.4	OpenAPI 2.0 definition.....	222
278	C.12.5	Property definition .....	230
279	C.12.6	CRUDN behaviour .....	230



280	C.13	Session.....	230
281	C.13.1	Introduction .....	230
282	C.13.2	Well-known URI .....	230
283	C.13.3	Resource type .....	230
284	C.13.4	OpenAPI 2.0 definition.....	231
285	C.13.5	Property definition .....	232
286	C.13.6	CRUDN behaviour .....	233
287	C.14	Security Profile .....	233
288	C.14.1	Introduction .....	233
289	C.14.2	Well-known URI .....	233
290	C.14.3	Resource type .....	233
291	C.14.4	OpenAPI 2.0 definition.....	234
292	C.14.5	Property definition .....	236
293	C.14.6	CRUDN behaviour .....	236
294	C.15	Token Refresh .....	236
295	C.15.1	Introduction .....	236
296	C.15.2	Well-known URI .....	236
297	C.15.3	Resource type .....	236
298	C.15.4	OpenAPI 2.0 definition.....	237
299	C.15.5	Property definition .....	239
300	C.15.6	CRUDN behaviour .....	239
301	Annex D (informative)	OID definitions .....	240
302	Annex E (informative)	Security considerations specific to Bridged Protocols .....	242
303	E.1	Security Considerations specific to the AllJoyn Protocol .....	242
304	E.2	Security Considerations specific to the Bluetooth LE Protocol.....	242
305	E.3	Security Considerations specific to the oneM2M Protocol .....	242
306	E.4	Security Considerations specific to the U+ Protocol .....	243
307	E.5	Security Considerations specific to the Z-Wave Protocol.....	243
308	E.6	Security Considerations specific to the Zigbee Protocol .....	244
309			

## FIGURES

310		
311	Figure 1 – OCF Interaction.....	10
312	Figure 2 – OCF Layers .....	12
313	Figure 3 – OCF Security Enforcement Points .....	14
314	Figure 4 – Use case-1 showing simple ACL enforcement.....	16
315	Figure 5 – Use case 2: A policy for the requested Resource is missing.....	17
316	Figure 6 – Use case-3 showing AMS supported ACL .....	18
317	Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS.....	19
318	Figure 8 – Example Resource definition with opaque Properties .....	20
319	Figure 9 – Property Level Access Control .....	20
320	Figure 10 – Onboarding Overview.....	22
321	Figure 11 – OCF Onboarding Process .....	24
322	Figure 12 – OCF's SRM Architecture .....	28
323	Figure 13 – Discover New Device Sequence.....	34
324	Figure 14 – A Just Works OTM .....	36
325	Figure 15 – Random PIN-based OTM .....	38
326	Figure 16 – Manufacturer Certificate Based OTM Sequence .....	41
327	Figure 17 – Vendor-specific Owner Transfer Sequence.....	42
328	Figure 18 – Establish Device Identity Flow.....	44
329	Figure 19 – Owner Credential Selection Provisioning Sequence .....	45
330	Figure 20 – Symmetric Owner Credential Provisioning Sequence .....	46
331	Figure 21 – Asymmetric Owner Credential Provisioning Sequence.....	47
332	Figure 22 – Configure Device Services .....	49
333	Figure 23 – Provision New Device for Peer to Peer Interaction Sequence.....	50
334	Figure 24 – Example of Client-directed provisioning.....	53
335	Figure 25 – Example of Server-directed provisioning using a single provisioning service .....	54
336	Figure 26 – Example of Server-directed provisioning involving multiple support services .....	57
337	Figure 27 – Device state model.....	59
338	Figure 28 – OBT Sanity Check Sequence in SRESET .....	63
339	Figure 29 – Client-directed Certificate Transfer.....	80
340	Figure 30 – Client-directed CRL Transfer.....	81
341	Figure 32 – Asserting a role with a certificate role credential. ....	85
342	Figure 33 – Device connection with OCF Cloud .....	88
343	Figure 34 – OCF Security Resources .....	98
344	Figure 35 – "/oic/sec/cred" Resource and Properties.....	99
345	Figure 36 – "/oic/sec/acl2" Resource and Properties.....	99
346	Figure 37 – "/oic/sec/amacl" Resource and Properties .....	100
347	Figure 38 – "/oic/sec/sacl" Resource and Properties .....	100
348	Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states .....	137

349	Figure 40 – Software Module Authentication .....	145
350	Figure 41 – Verification Software Module.....	145
351	Figure 42 – Software Module Authenticity .....	146
352	Figure 43 – State transitioning diagram for software download .....	147
353	Figure A-1 – Example "/oic/sec/acl2" Resource.....	160
354	Figure A-2 Example "/oic/sec/amacl" Resource.....	161
355	Figure E-1 Security Considerations for BLE Bridge .....	242
356	Figure E-2 Security Considerations for Z-Wave Bridge.....	244
357	Figure E-3 Security Considerations for Zigbee Bridge .....	245
358		

359	<b>Tables</b>	
360	Table 1 – Discover New Device Details.....	34
361	Table 2 – A Just Works OTM Details.....	36
362	Table 3 – Random PIN-based OTM Details.....	38
363	Table 4 – Manufacturer Certificate Based OTM Details.....	41
364	Table 5 – Vendor-specific Owner Transfer Details.....	43
365	Table 6 – Establish Device Identity Details.....	44
366	Table 7 – Owner Credential Selection Details.....	45
367	Table 8 – Symmetric Owner Credential Assignment Details.....	46
368	Table 9 – Asymmetric Owner Credential Assignment Details.....	47
369	Table 10 – Configure Device Services Detail.....	49
370	Table 11 – Provision New Device for Peer to Peer Details.....	50
371	Table 12 – Steps describing Client -directed provisioning.....	53
372	Table 13 – Steps for Server-directed provisioning using a single provisioning service.....	54
373	Table 14 – Steps for Server-directed provisioning involving multiple support services.....	57
374	Table 15 – Mapping of Properties of the "oic.r.account" and "oic.r.coapcloudconf"	
375	Resources.....	58
376	Table 16 – X.509 v1 fields for Root CA Certificates.....	69
377	Table 17 - X.509 v3 extensions for Root CA Certificates.....	70
378	Table 18 - X.509 v1 fields for Intermediate CA Certificates.....	70
379	Table 19 – X.509 v3 extensions for Intermediate CA Certificates.....	71
380	Table 20 – X.509 v1 fields for End-Entity Certificates.....	71
381	Table 21 – X.509 v3 extensions for End-Entity Certificates.....	72
382	Table 22 – Device connection with the OCF Cloud flow.....	88
383	Table 23 – ACE2 Wildcard Matching Strings Description.....	94
384	Table 24 – Definition of the "/oic/sec/doxm" Resource.....	101
385	Table 25 – Properties of the "/oic/sec/doxm" Resource.....	101
386	Table 26 – Properties of the "oic.sec.didtype" type.....	102
387	Table 27 – Properties of the "oic.sec.doxmtype" type.....	104
388	Table 28 – Definition of the "oic.r.cred" Resource.....	105
389	Table 29 – Properties of the "/oic/sec/cred" Resource.....	106
390	Table 30 – Properties of the "oic.sec.cred" Property.....	107
391	Table 31: Properties of the "oic.sec.credusagetype" Property.....	108
392	Table 32 – Properties of the "oic.sec.pubdatatype" Property.....	108
393	Table 33 – Properties of the "oic.sec.privdatatype" Property.....	108
394	Table 34 – Properties of the "oic.sec.optdatatype" Property.....	109
395	Table 35 – Definition of the "oic.sec.roletype" type.....	109
396	Table 36 – Value Definition of the "oic.sec.crmttype" Property.....	111
397	Table 37 – 128-bit symmetric key.....	112

398	Table 38 – 256-bit symmetric key .....	112
399	Table 39 – Definition of the "oic.r.crl" Resource .....	114
400	Table 40 – Properties of the "oic.r.crl" Resource .....	114
401	Table 41 – BNF Definition of OCF ACL .....	114
402	Table 42 – Value Definition of the "oic.sec.crudntype" Property .....	117
403	Table 43 – Definition of the "oic.sec.acl2" Resource .....	117
404	Table 44 – Properties of the "oic.sec.acl2" Resource .....	117
405	Table 45 – "oic.sec.ace2" data type definition. ....	118
406	Table 46 – "oic.sec.ace2.resource-ref" data type definition. ....	118
407	Table 47 – Value definition "oic.sec.conntype" Property.....	119
408	Table 48 – Definition of the "oic.r.amacl" Resource.....	121
409	Table 49 – Properties of the "oic.r.amacl" Resource .....	121
410	Table 50 – Definition of the "oic.r.sacl" Resource.....	121
411	Table 51 – Properties of the "oic.r.sacl" Resource .....	121
412	Table 52 – Properties of the "oic.sec.sigtype" Property .....	122
413	Table 53 – Definition of the "oic.r.pstat" Resource .....	122
414	Table 54 – Properties of the "oic.r.pstat" Resource .....	123
415	Table 55 – Properties of the "/oic/sec/dostype" Property.....	124
416	Table 56 – Definition of the "oic.sec.dpmttype" Property .....	126
417	Table 57 – Value Definition of the "oic.sec.dpmttype" Property (Low-Byte) .....	126
418	Table 58 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte).....	126
419	Table 59 – Definition of the "oic.sec.pomtype" Property .....	127
420	Table 60 – Value Definition of the "oic.sec.pomtype" Property .....	127
421	Table 61 – Definition of the "oic.r.csr" Resource .....	127
422	Table 62 – Properties of the "oic.r.csr" Resource .....	128
423	Table 63 – Definition of the "oic.r.roles" Resource .....	129
424	Table 64 – Properties of the "oic.r.roles" Resource .....	129
425	Table 65 – Definition of the "oic.r.account" Resource.....	130
426	Table 66 – Properties of the "oic.r.account" Resource .....	130
427	Table 67 – Definition of the "oic.r.session" Resource .....	131
428	Table 68 – Properties of the "oic.r.session" Resource .....	131
429	Table 69 – Definition of the "oic.r.tokenrefresh" Resource .....	132
430	Table 70 – Properties of the "oic.r.tokenrefresh" Resource .....	133
431	Table 71 – Core Resource Properties Access Modes given various Device States.....	135
432	Table 72 – Examples of Sensitive Data.....	141
433	Table 73 – Description of the software update bits.....	147
434	Table 74 – Definition of the "oic.sec.sp" Resource .....	151
435	Table 75 – Properties of the "oic.sec.sp" Resource.....	151

436	Table 76 – Dependencies of VOD Behaviour on Bridge state, as clarification of	
437	accompanying text.....	158
438	Table B.1 – OCF Security Profile .....	162
439	Table C.1 – Alphabetized list of security resources .....	163
440	Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account". .....	166
441	Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account".....	167
442	Table C.4 – The Property definitions of the Resource with type "rt" = "oic.r.acl2". .....	177
443	Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".....	177
444	Table C.6 – The Property definitions of the Resource with type "rt" = "oic.r.amacl". .....	181
445	Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl".....	181
446	Table C.8 – The Property definitions of the Resource with type "rt" = "oic.r.cred".....	192
447	Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred". .....	192
448	Table C.10 – The Property definitions of the Resource with type "rt" = "oic.r.crl".....	195
449	Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl". .....	195
450	Table C.12 – The Property definitions of the Resource with type "rt" = "oic.r.csr". .....	197
451	Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr". .....	198
452	Table C.14 – The Property definitions of the Resource with type "rt" = "oic.r.doxm". .....	201
453	Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".....	203
454	Table C.16 – The Property definitions of the Resource with type "rt" = "oic.r.pstat". .....	208
455	Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat". .....	212
456	Table C.18 – The Property definitions of the Resource with type "rt" = "oic.r.roles". .....	222
457	Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles". .....	222
458	Table C.20 – The Property definitions of the Resource with type "rt" = "oic.r.sacl". .....	230
459	Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl".....	230
460	Table C.22 – The Property definitions of the Resource with type "rt" = "oic.r.session".....	233
461	Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.session". .....	233
462	Table C.24 – The Property definitions of the Resource with type "rt" = "oic.r.sp". .....	236
463	Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp". .....	236
464	Table C.26 – The Property definitions of the Resource with type "rt" = "oic.r.tokenrefresh".	239
465	Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh"..	239
466	Table E.1 GAP security mode .....	242
467	Table E.2 TLS 1.2 Cipher Suites used by U+ .....	243
468	Table E.3 Z-Wave Security Class.....	244
469	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers .....	245
470		
471		

## 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](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](https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf)

497 OCF Wi-Fi Easy Setup, Information technology – Open Connectivity Foundation (OCF)  
498 Specification – Part 7: Wi-Fi Easy Setup specification  
499 Latest version available at:  
500 [https://openconnectivity.org/specs/OCF\\_Wi-Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)

501 OCF Cloud Specification, Information technology – Open Connectivity Foundation (OCF)  
502 Specification – Part 8: Cloud Specification  
503 Latest version available at:  
504 [https://openconnectivity.org/specs/OCF\\_Cloud\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf)

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

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

508 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September  
509 2000, <https://tools.ietf.org/html/rfc2898>

510 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November  
511 2000, <https://tools.ietf.org/html/rfc2986>

512 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December  
513 2005, <https://tools.ietf.org/html/rfc4279>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

548



549 **3 Terms, definitions, and abbreviated terms**

550 **3.1 Terms and definitions**

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

553 ISO and IEC maintain terminological databases for use in standardization at the following  
554 addresses:

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

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

557 **3.1.1**

558 **Access Management Service (AMS)**

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

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

562 **3.1.2**

563 **Access Token**

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

566 **3.1.3**

567 **Authorization Provider**

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

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

571 **3.1.4**

572 **Client**

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

574 **3.1.5**

575 **Credential Management Service (CMS)**

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

578 **3.1.6**

579 **Device**

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

581 **3.1.7**

582 **Device Class**

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

586 **3.1.8**

587 **Device ID**

588 a stack instance identifier.

589 **3.1.9**

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

591 a logical entity that establishes device ownership

592 **3.1.10**  
593 **Device Registration**  
594 a process by which Device is enrolled/registered to the OCF Cloud infrastructure (using Device  
595 certificate and unique credential) and becomes ready for further remote operation through the  
596 cloud interface (e.g. connection to remote Resources or publishing of its own Resources for  
597 access).

598 **3.1.11**  
599 **End-Entity**  
600 any certificate holder which is not a Root or Intermediate Certificate Authority.

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

602 **3.1.12**  
603 **Entity**

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

605 **3.1.13**  
606 **OCF Interface**

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

608 **3.1.14**  
609 **Intermediary**

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

612 **3.1.15**  
613 **OCF Cipher Suite**

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

617 **3.1.16**  
618 **OCF Cloud User**

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

621 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  
622 organization that onboarded that Device. The OCF Cloud User delegates, to the OCF Cloud authority, authority to route  
623 between Devices registered by the OCF Cloud User. The OCF Cloud delegates, to the OCF Cloud User, authority to  
624 select the set of Devices which can register and use the services of the OCF Cloud.

625 **3.1.17**  
626 **OCF Rooted Certificate Chain**

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

630 **3.1.18**  
631 **Onboarding Tool (OBT)**

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

633 **3.1.19**  
634 **Out of Band Method**

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

636 **3.1.20**  
637 **Owner Credential (OC)**

638 credential, provisioned by an OBT(3.1.18) to a Device during onboarding, for the purposes of  
639 mutual authentication of the Device and OBT(3.1.18) during subsequent interactions

640 **3.1.21**  
641 **Platform ID**  
642 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

643 **3.1.22**  
644 **Property**  
645 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

646 **3.1.23**  
647 **Resource**  
648 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

649 **3.1.24**  
650 **Role (Network context)**  
651 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

652 **3.1.25**  
653 **Role Identifier**  
654 a Property of an OCF credentials Resource or element in a role certificate that identifies a  
655 privileged role that a Server Device associates with a Client Device for the purposes of making  
656 authorization decisions when the Client Device requests access to Device Resources.

657 **3.1.26**  
658 **Secure Resource Manager (SRM)**  
659 a module in the OCF Core that implements security functionality that includes management of  
660 security Resources such as ACLs, credentials and Device owner transfer state.

661 **3.1.27**  
662 **Security Virtual Resource (SVR)**  
663 a resource supporting security features.  
664 Note 1 to entry: For a list of all the SVRs please see clause 13.

665 **3.1.28**  
666 **Server**  
667 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

668 **3.1.29**  
669 **Trust Anchor**  
670 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.  
671 a Device and an OBT(3.1.18)) can assume trust

672 **3.1.30**  
673 **Unique Authenticable Identifier**  
674 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is  
675 used to create the Device ID.  
676 Note 1 to entry: The ownership of a UAID may be authenticated by peer Devices.

677 **3.1.31**  
678 **Device Configuration Resource (DCR)**  
679 a Resource that is any of the following:

680 a) a Discovery Core Resource, or  
681 b) a Security Virtual Resource, or  
682 c) a Wi-Fi Easy Setup Resource ("oic.r.easyssetup", "oic.r.wificonf", "oic.r.devconf"), or  
683 d) a CoAP Cloud Configuration Resource ("oic.r.coapcloudconf"), or  
684 e) a Software Update Resource ("oic.r.softwareupdate"), or

685 f) a Maintenance Resource ("oic.wk.mnt").

686 **3.1.32**

687 **Non-Configuration Resource (NCR)**

688 a Resource that is not a Device Configuration Resource (3.1.31).

689 **3.1.33**

690 **Bridged Device**

691 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

692 **3.1.34**

693 **Bridged Protocol**

694 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

695 **3.1.35**

696 **Bridge**

697 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

698 **3.1.36**

699 **Bridging Platform**

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

701 **3.1.37**

702 **Virtual Bridged Device**

703 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

704 **3.1.38**

705 **Virtual OCF Device**

706 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

707 **3.1.39**

708 **OCF Security Domain**

709 set of onboarded OCF Devices that are provisioned with credentialing information for confidential  
710 communication with one another

711 **3.1.40**

712 **Owned (or "in Owned State")**

713 having the "owned" Property of the "/oic/sec/doxm" resource equal to "TRUE"

714 **3.1.41**

715 **Unowned (or "in Unowned State")**

716 having the "owned" Property of the "/oic/sec/doxm" resource equal to "FALSE"

717 **3.2 Abbreviated terms**

718 **3.2.1**

719 **AC**

720 Access Control

721 **3.2.2**

722 **ACE**

723 Access Control Entry

724 **3.2.3**

725 **ACL**

726 Access Control List

727 **3.2.4**

728 **AES**

729 Advanced Encryption Standard

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

731 **3.2.5**

732 **AMS**

733 Access Management Service

734 **3.2.6**

735 **CMS**

736 Credential Management Service

737 **3.2.7**

738 **CRUDN**

739 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

740 **3.2.8**

741 **CSR**

742 Certificate Signing Request

743 **3.2.9**

744 **CVC**

745 Code Verification Certificate

746 **3.2.10**

747 **ECC**

748 Elliptic Curve Cryptography

749 **3.2.11**

750 **ECDSA**

751 Elliptic Curve Digital Signature Algorithm

752 **3.2.12**

753 **EKU**

754 Extended Key Usage

755 **3.2.13**

756 **EPC**

757 Embedded Platform Credential

758 **3.2.14**

759 **EPK**

760 Embedded Public Key

761 **3.2.15**

762 **DOTS**

763 Device Ownership Transfer Service

764 **3.2.16**

765 **DPKP**

766 Dynamic Public Key Pair

767 **3.2.17**

768 **ID**

769 Identity/Identifier

770 **3.2.18**

771 **JSON**

772 JavaScript Object Notation.

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

774 **3.2.19**  
775 **JWS**  
776 JSON Web Signature.

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

778 **3.2.20**  
779 **KDF**  
780 Key Derivation Function

781 **3.2.21**  
782 **MAC**  
783 Message Authentication Code

784 **3.2.22**  
785 **MITM**  
786 Man-in-the-Middle

787 **3.2.23**  
788 **NVRAM**  
789 Non-Volatile Random-Access Memory

790 **3.2.24**  
791 **OC**  
792 Owner Credential

793 **3.2.25**  
794 **OCSP**  
795 Online Certificate Status Protocol

796 **3.2.26**  
797 **OBT**  
798 Onboarding Tool

799 **3.2.27**  
800 **OID**  
801 Object Identifier

802 **3.2.28**  
803 **OTM**  
804 Owner Transfer Method

805 **3.2.29**  
806 **OOB**  
807 Out of Band

808 **3.2.30**  
809 **OWASP**  
810 Open Web Application Security Project.

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

812 **3.2.31**  
813 **PE**  
814 Policy Engine

815 **3.2.32**  
816 **PIN**  
817 Personal Identification Number

818 **3.2.33**  
819 **PPSK**  
820 PIN-authenticated pre-shared key

821 **3.2.34**  
822 **PRF**  
823 Pseudo Random Function

824 **3.2.35**  
825 **PSI**  
826 Persistent Storage Interface

827 **3.2.36**  
828 **PSK**  
829 Pre Shared Key

830 **3.2.37**  
831 **RBAC**  
832 Role Based Access Control

833 **3.2.38**  
834 **RM**  
835 Resource Manager

836 **3.2.39**  
837 **RNG**  
838 Random Number Generator

839 **3.2.40**  
840 **SACL**  
841 Signed Access Control List

842 **3.2.41**  
843 **SBAC**  
844 Subject Based Access Control

845 **3.2.42**  
846 **SEE**  
847 Secure Execution Environment

848 **3.2.43**  
849 **SRM**  
850 Secure Resource Manager

851 **3.2.44**  
852 **SVR**  
853 Security Virtual Resource

854 **3.2.45**  
855 **SW**  
856 Software

857 **3.2.46**  
858 **UAID**  
859 Unique Authenticable Identifier

860 **3.2.47**  
861 **URI**  
862 Uniform Resource Identifier

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

864 **3.2.48**  
865 **VOD**  
866 Virtual OCF Device

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

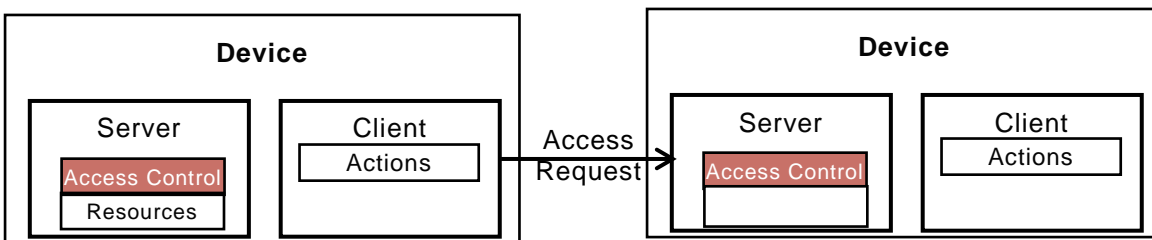
## 868 **4 Document Conventions and Organization**

### 869 **4.1 Conventions**

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

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

874 Figure 1 depicts interaction between OCF Devices.



875

876 **Figure 1 – OCF Interaction**

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

### 881 **4.2 Notation**

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

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

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

888 **Recommended** (or **should**).

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



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

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

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

899 **Conditionally allowed (CA)**

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

902 **Conditionally required (CR)**

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

906 **DEPRECATED**

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

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

914 Words that are emphasized are printed in italic.

### 915 **4.3 Data types**

916 See ISO/IEC 30118-1:2018.

### 917 **4.4 Document structure**

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

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

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

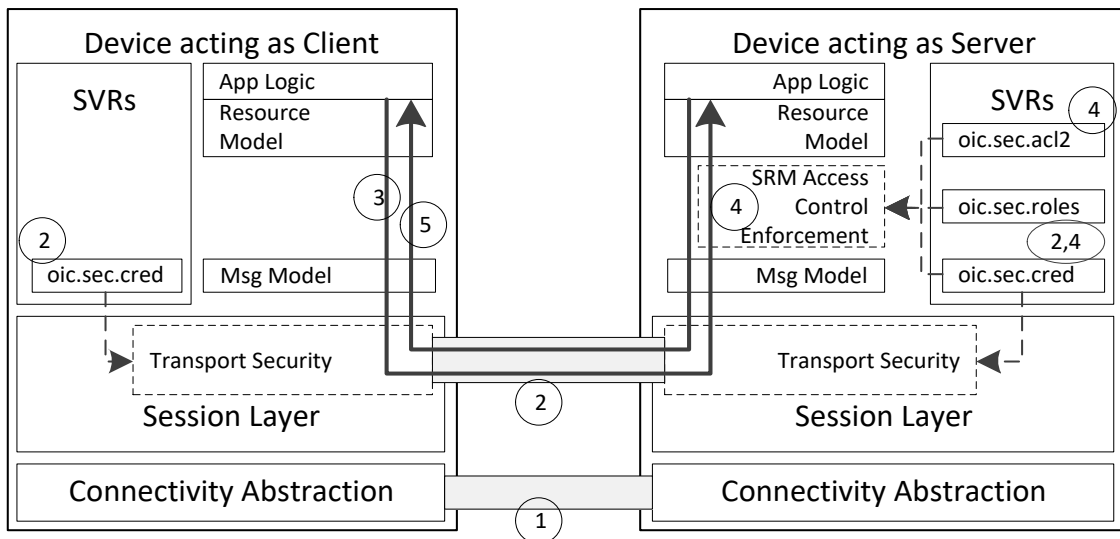
924

925 **5 Security Overview**

926 **5.1 Preamble**

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

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



936  
937

938 **Figure 2 – OCF Layers**

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

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

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

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

957 4) The Server receives the request.

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

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

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

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

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

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

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

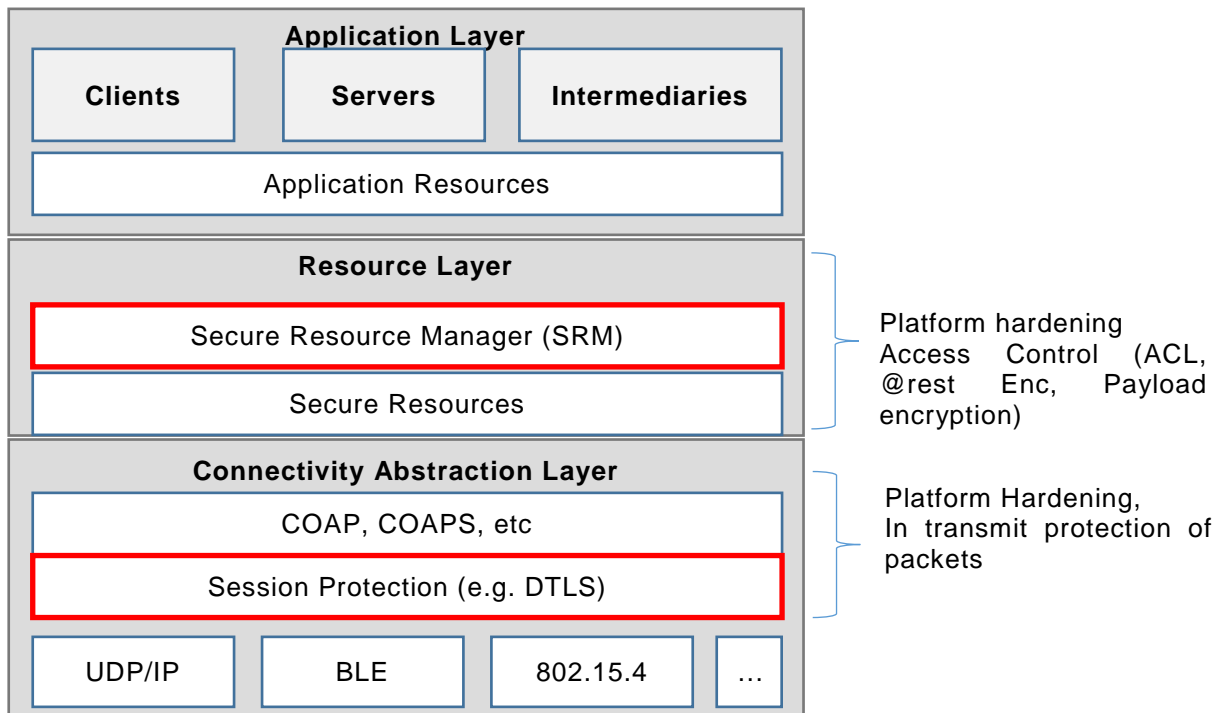
975 5) The Server sends a response back to the Client.

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

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

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

990 Figure 3 depicts OCF Security Enforcement Points.



991  
992  
993 **Figure 3 – OCF Security Enforcement Points**

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

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

1001 The OpenAPI 2.0 definitions (Annex C) used in this document are normative. This includes that  
1002 all defined payloads shall comply with the indicated OpenAPI 2.0 definitions. Annex C contains all  
1003 of the OpenAPI 2.0 definitions for Resource Types defined in this document.

1004 **5.2 Access Control**

1005 The OCF framework assumes that Resources are hosted by a Server and are made available to  
1006 Clients subject to access control and authorization mechanisms. The Resources at the end point  
1007 are protected through implementation of access control, authentication and confidentiality  
1008 protection. This clause provides an overview of Access Control (AC) through the use of ACLs.  
1009 However, AC in the OCF stack is expected to be transport and connectivity abstraction layer  
1010 agnostic.

1011 Implementation of access control relies on a-priori definition of a set of access policies for the  
1012 Resource. The policies may be stored by a local ACL or an Access Management Service (AMS)  
1013 in form of Access Control Entries (ACE). Two types of access control mechanisms can be applied:

- 1014 – Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of  
1015 requestor) of the requesting entity against the subject included in the policy defined for  
1016 Resource. Asserting the identity of the requestor requires an authentication process.

1017 – Role-based Access Control (RBAC), where each ACE will match a role identifier included in  
1018 the policy for the Resource to a role identifier associated with the requestor.

1019 Some Resources, such as Collections, generate requests to linked Resources when appropriate  
1020 Interfaces are used. In such cases, additional access control considerations are necessary.  
1021 Additional access control considerations for Collections when using the batch OCF Interface are  
1022 found in clause 12.2.7.3.

1023 In the OCF access control model, access to a Resource instance requires an associated ACE.  
1024 The lack of such an associated ACE results in the Resource being inaccessible.

1025 The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested  
1026 Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier  
1027 or (3) wildcard. The way in which the client connects to the server may be relevant context for  
1028 making access control decisions. Wildcard matching on authenticated vs. unauthenticated and  
1029 encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject  
1030 classes.

1031 Example Wildcard Matching Policy:

```
1032 "aclist2": [  
1033   {  
1034     "subject": {"conntype" : "anon-clear" },  
1035     "resources": [  
1036       { "wc": "*" }  
1037     ],  
1038     "permission": 31  
1039   },  
1040   {  
1041     "subject": {"conntype" : "auth-crypt" },  
1042     "resources": [  
1043       { "wc": "*" }  
1044     ],  
1045     "permission": 31  
1046   },  
1047 ]
```

1048 Details of the format for ACL are defined in clause 12. The ACL is composed of one or more  
1049 ACEs. The ACL defines the access control policy for the Devices.

1050 ACL Resource requires the same security protection as other sensitive Resources, when it comes  
1051 to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW  
1052 and SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may  
1053 have different scoping levels and thus hardening needs to be specially considered for each  
1054 scoping level. For instance, a physical device may host multiple Device implementations and thus  
1055 secure storage, usage and isolation of ACLs for different Servers on the same Device needs to  
1056 be considered.

## 1057 **5.2.1 ACL Architecture**

### 1058 **5.2.1.1 ACL Architecture General**

1059 The Server examines the Resource(s) requested by the client before processing the request. The  
1060 access control resource is searched to find one or more ACE entries that match the requestor  
1061 and the requested Resources. If a match is found, then permission and period constraints are

1062 applied. If more than one match is found, then the logical UNION of permissions is applied to the  
1063 overlapping periods.

1064 The server uses the connection context to determine whether the subject has authenticated or  
1065 not and whether data confidentiality has been applied or not. Subject matching wildcard policies  
1066 can match on each aspect. If the user has authenticated, then subject matching may happen at  
1067 increased granularity based on role or device identity.

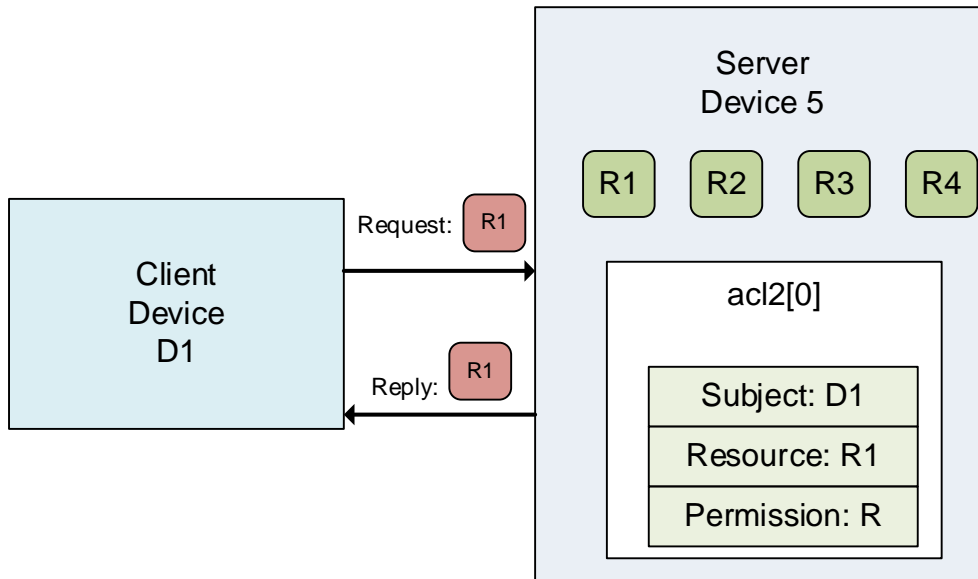
1068 Each ACE contains the permission set that will be applied for a given Resource requestor.  
1069 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY  
1070 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more  
1071 roles. Devices may acquire elevated access permissions when asserting a role. For example, an  
1072 ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally  
1073 accessible.

#### 1074 5.2.1.2 Use of local ACLs

1075 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control  
1076 processing than remote ACL processing by an AMS.  
1077 The following use cases describe the operation of access control

1078 Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4).  
1079 Client Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0]  
1080 corresponds to Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives  
1081 access to Resource R1 because the local ACL "/oic/sec/acl2/0" matches the request.

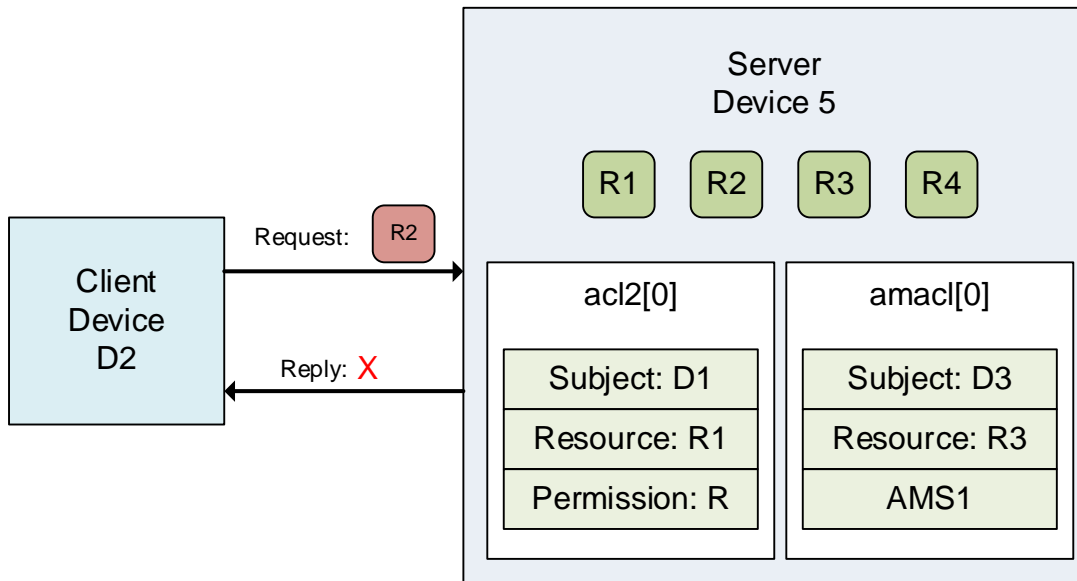
1082



1083

1084 **Figure 4 – Use case-1 showing simple ACL enforcement**

1085 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL  
1086 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.



1088

1089 **Figure 5 – Use case 2: A policy for the requested Resource is missing**1090 **5.2.1.3 Use of AMS**

1091 AMS improves ACL policy management. However, they can become a central point of failure.  
 1092 Due to network latency overhead, ACL processing may be slower through an AMS.

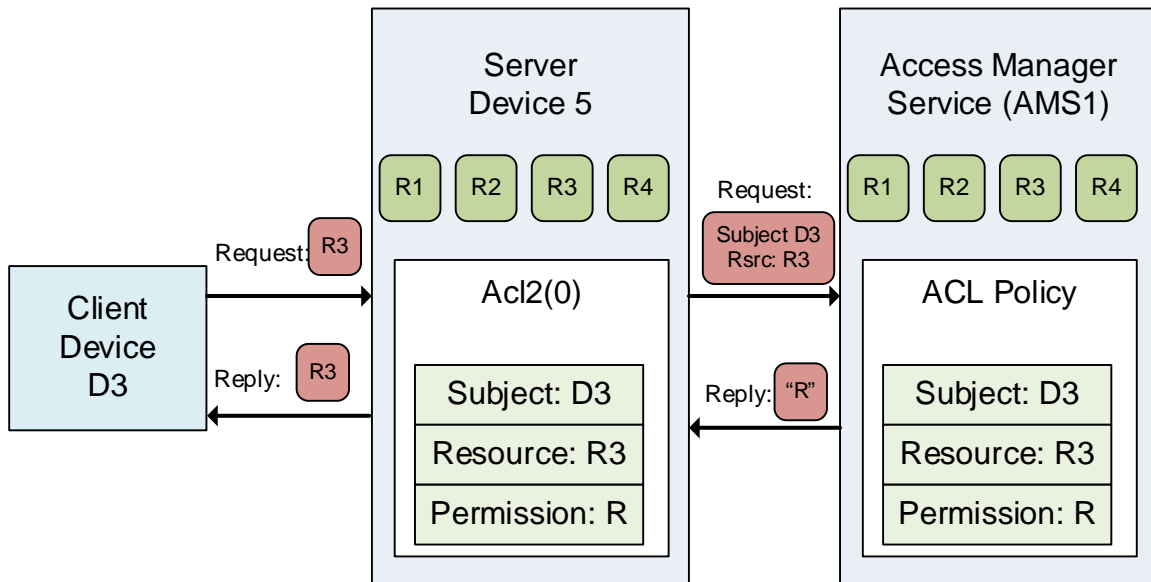
1093 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The  
 1094 Server shall determine which ACL mechanism to use for which Resource set. The  
 1095 "/oic/sec/amacl" Resource is an ACL structure that specifies which Resources will use an AMS to  
 1096 resolve access decisions. The "/oic/sec/amacl" may be used in concert with local ACLs  
 1097 ("/oic/sec/acl2").

1098 The AMS is authenticated by referencing a credential issued to the device identifier contained in  
 1099 "/oic/sec/acl2.rowneruuid".

1100 The Server Device may proactively open a connection to the AMS using the Device ID found in  
 1101 "/oic/sec/acl2.rowneruuid". Alternatively, the Server may reject the Resource access request with  
 1102 an error, ACCESS\_DENIED\_REQUIRES\_SACL that instructs the requestor to obtain a suitable  
 1103 ACE policy using a SACL Resource "/oic/sec/sacl". The "/oic/sec/sacl" signature may be  
 1104 validated using the credential Resource associated with the "/oic/sec/acl2.rowneruuid".

1105 The following use cases describe access control using the AMS:

1106 Use Case 3: As depicted in Figure 6, Device D3 requests and receives access to Resource R3  
 1107 with permission Perm1 because the "/oic/sec/amacl/0" matches a policy to consult the Access  
 1108 Manager Server AMS1 service



1109

1110

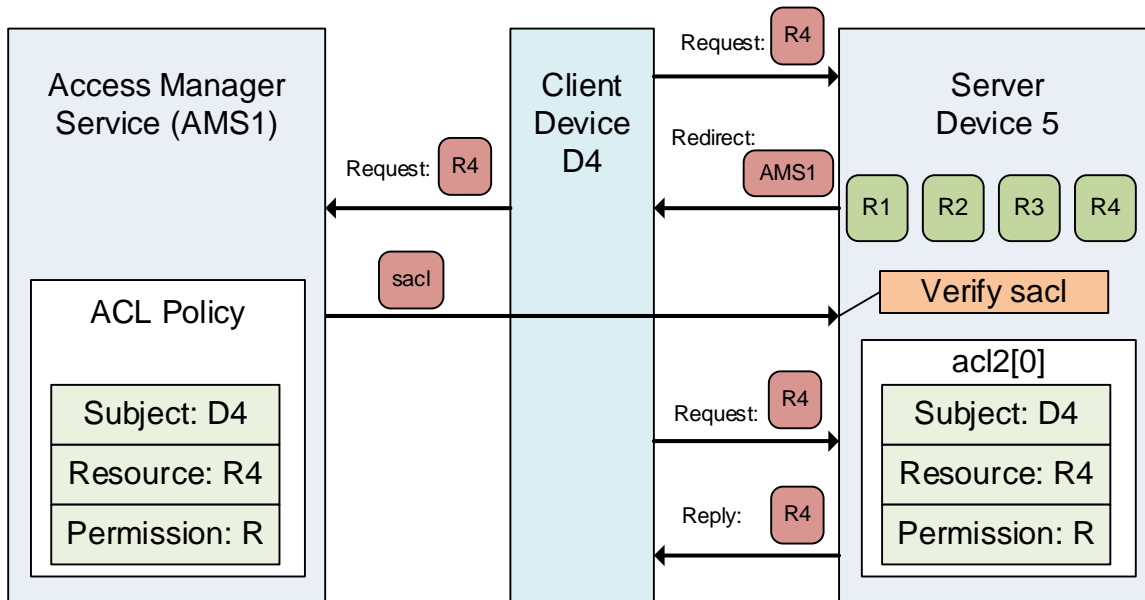
**Figure 6 – Use case-3 showing AMS supported ACL**

1111 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from  
 1112 Server Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1  
 1113 by returning an error identifying AMS1 as a "/oic/sec/sacl" Resource issuer. Device D4 obtains  
 1114 Sacl1 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the  
 1115 "/oic/sec/sacl" Resource and evaluates the ACE policy that grants Perm2 access.

1116 ACE redirection may occur when D4 receives an error result with reason code indicating no  
 1117 match exists (i.e. ACCESS\_DENIED\_NO\_ACE). D4 reads the "/oic/sec/acl2" Resource to find the  
 1118 "rowneruuid" which identifies the AMS and then submits a request to be provisioned, in this  
 1119 example the AMS chooses to supply a SACL Resource, however it may choose to re-provision  
 1120 the local ACL Resource "/oic/sec/acl2". The request is reissued subsequently. D4 is presumed to  
 1121 have been introduced to the AMS as part of Device onboarding or through subsequent credential  
 1122 provisioning actions.

1123 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.





1124

1125

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

1126

### 5.2.2 Access Control Scoping Levels

1127

1128

1129

1130

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

1131

1132

1133

1134

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

1135

1136

1137

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

1138

1139

1140

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

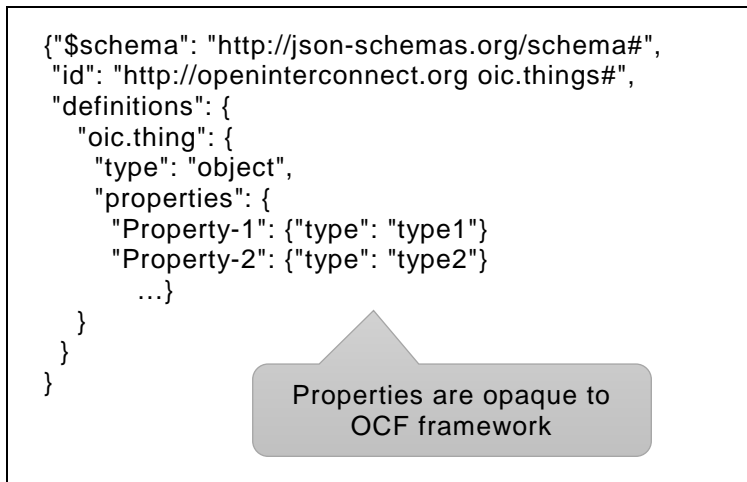
1141

1142

1143

1144

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

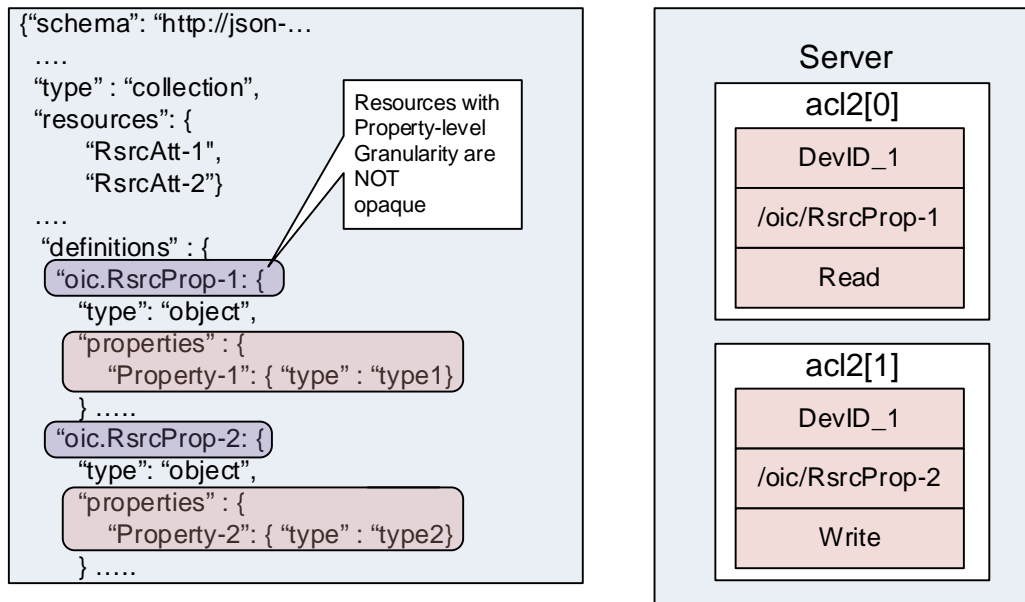


1145

1146

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

1147 Currently, OCF framework treats property level information as opaque; therefore, different  
 1148 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1  
 1149 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into  
 1150 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be  
 1151 achieved through use of Resource-level ACLs.



1152

1153

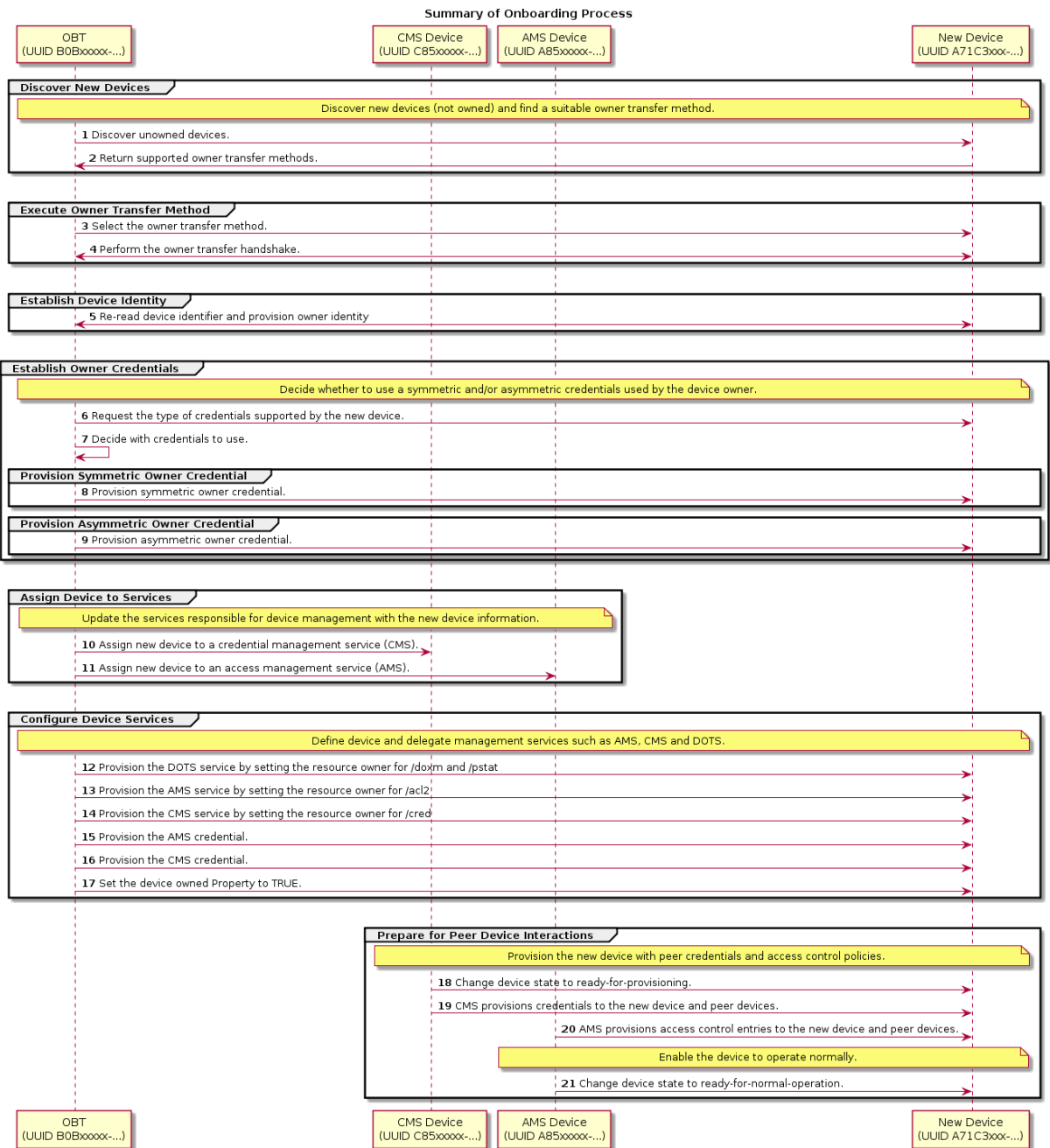
**Figure 9 – Property Level Access Control**

1154 **5.3 Onboarding Overview**

1155 **5.3.1 Onboarding General**

1156 Before a Device becomes operational in an OCF environment and is able to interact with other  
1157 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to  
1158 configure the ownership where the legitimate user that owns/purchases the Device uses an  
1159 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to  
1160 establish ownership. Once ownership is established, the OBT becomes the mechanism through  
1161 which the Device can then be provisioned, at the end of which the Device becomes operational  
1162 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an  
1163 OCF Device.

1164 Figure 10 depicts Onboarding Overview.



1165  
1166

**Figure 10 – Onboarding Overview**

1167 This clause explains the onboarding and security provisioning process but leaves the provisioning  
 1168 of non-security aspects to other OCF documents. In the context of security, all Devices are  
 1169 required to be provisioned with minimal security configuration that allows the Device to securely  
 1170 interact/communicate with other Devices in an OCF environment. This minimal security  
 1171 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified  
 1172 in 7.5.

1173 Onboarding and provisioning implementations could utilize services defined outside this  
 1174 document, it is expected that in using other services, trust between the device being onboarded  
 1175 and the various tools is not transitive. This implies that the device being onboarded will  
 1176 individually authenticate the credentials of each and every tool used during the onboarding

1177 process; that the tools not share credentials or imply a trust relationship where one has not been  
1178 established.

### 1179 **5.3.2 Onboarding Steps**

1180 The flowchart in Figure 11 shows the typical steps that are involved during onboarding. Although  
1181 onboarding may include a variety of non-security related steps, the diagram focus is mainly on  
1182 the security related configuration to allow a new Device to function within an OCF environment.  
1183 Onboarding typically begins with the Device becoming an Owned Device followed by configuring  
1184 the Device for the environment that it will operate in. This would include setting information such  
1185 as who can access the Device and what actions can be performed as well as what permissions  
1186 the Device has for interacting with other Devices.

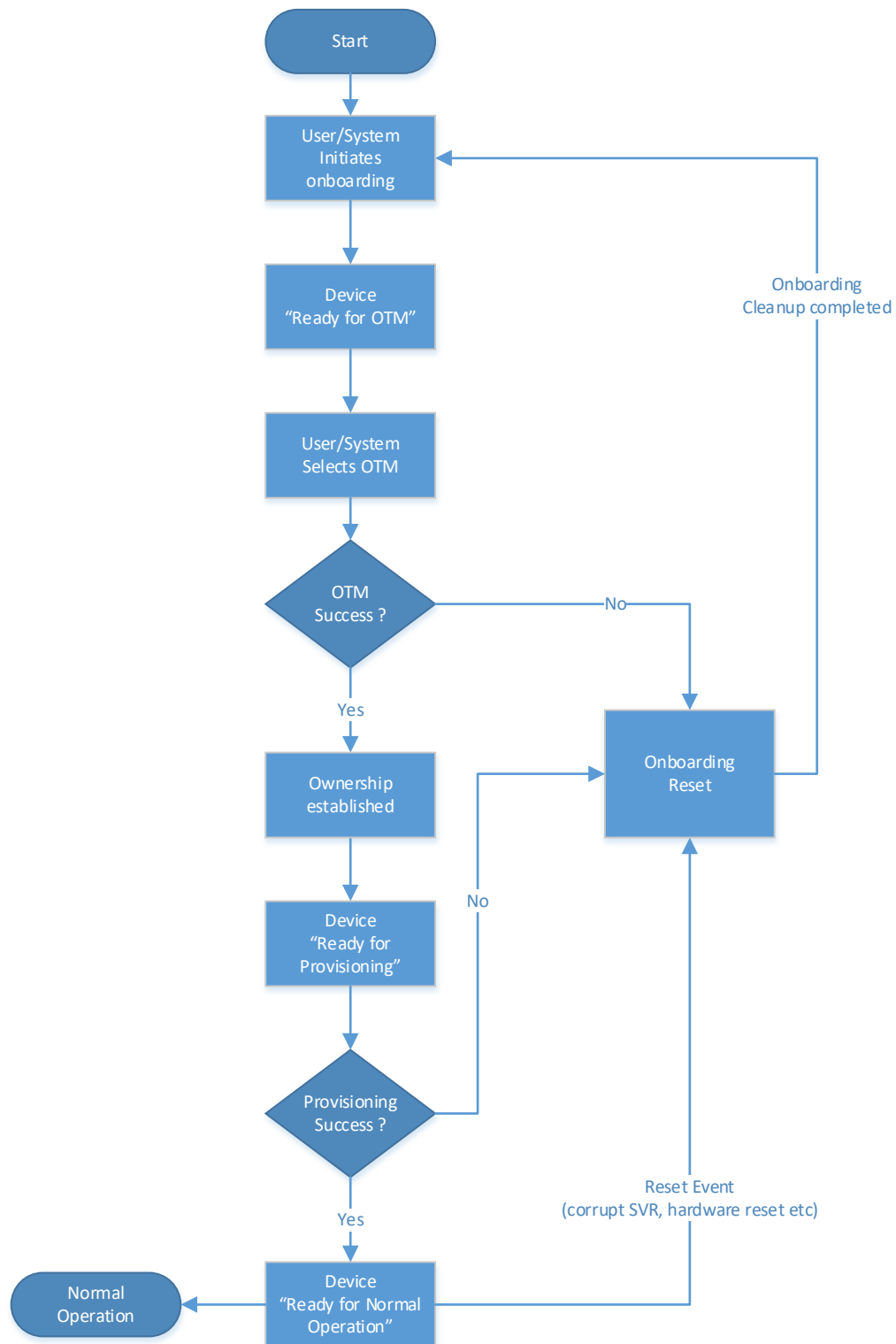


Figure 11 – OCF Onboarding Process

1187

1188

1189 **5.3.3 Establishing a Device Owner**

1190 The objective behind establishing Device ownership is to allow the legitimate user that  
 1191 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 24

1192 through the use of an OBT that includes the creation of an ownership context between the new  
1193 Device and the OBT tool and asserts operational control and management of the Device. The  
1194 OBT can be considered a logical entity hosted by tools/ Servers such as a network management  
1195 console, a device management tool, a network-authoring tool, a network provisioning tool, a  
1196 home gateway device, or a home automation controller. A physical device hosting the OBT will be  
1197 subject to some security hardening requirements, thus preserving integrity and confidentiality of  
1198 any credentials being stored. The tool/Server that establishes Device ownership is referred to as  
1199 the OBT.

1200 The OBT uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term  
1201 owner transfer is used since it is assumed that even for a new Device, the ownership is  
1202 transferred from the manufacturer/provider of the Device to the buyer/legitimate user of the new  
1203 Device.

1204 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.  
1205 Owner transfer establishes the following

- 1206 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"  
1207 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during  
1208 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property  
1209 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the  
1210 identity of Device as part of ownership transfer.
- 1211 – The Device owner establishes trust in the Device through the OTM.
- 1212 – Preparing the Device for provisioning by providing credentials that may be needed.

#### 1213 **5.3.4 Provisioning for Normal Operation**

1214 Once the Device has the necessary information to initiate provisioning, the next step is to  
1215 provision additional security configuration that allows the Device to become operational. This can  
1216 include setting various parameters and may also involve multiple steps. Also provisioning of  
1217 ACL's for the various Resources hosted by the Server on the Device is done at this time. The  
1218 provisioning step is not limited to this stage only. Device provisioning can happen at multiple  
1219 stages in the Device's operational lifecycle. However specific security related provisioning of  
1220 Resource and Property state would likely happen at this stage at the end of which, each Device  
1221 reaches the Onboarded Device "Ready for Normal Operation" State. The "Ready for Normal  
1222 Operation" State is expected to be consistent and well defined regardless of the specific OTM  
1223 used or regardless of the variability in what gets provisioned. However individual OTM  
1224 mechanisms and provisioning steps may specify additional configuration of Resources and  
1225 Property states. The minimal mandatory configuration required for a Device to be in "Ready for  
1226 Normal Operation" state is specified in 8.

#### 1227 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview**

1228 As mentioned in the start of clause 5, communication between a Device and OCF Cloud is  
1229 subject to different criteria in comparison to Devices which are within a single local network. The  
1230 Device is configured in order to connect to the OCF Cloud by a Mediator as specified in the  
1231 "CoAPCloudConf" Resource clauses in OCF Cloud Specification. Provisioning includes the  
1232 remote connectivity and local details such as URL where the OCF Cloud hosting environment can  
1233 be found and the OCF Cloud verifiable Access Token.

#### 1234 **5.3.6 OCF Compliance Management System**

1235 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that  
1236 provides Certification status and information for OCF Devices.

1237 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:  
1238 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1239 The OBT shall possess the Root Certificate needed to enable https connection to the URI  
1240 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1241 The OBT should periodically refresh its copy of the CPL via the URI  
1242 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security  
1243 Domain owner policy requirements.

## 1244 **5.4 Provisioning**

### 1245 **5.4.1 Provisioning General**

1246 In general, provisioning may include processes during manufacturing and distribution of the  
1247 Device as well as processes after the Device has been brought into its intended environment  
1248 (parts of onboarding process). In this document, security provisioning includes, processes after  
1249 ownership transfer (even though some activities during ownership transfer and onboarding may  
1250 lead to provisioning of some data in the Device) configuration of credentials for interacting with  
1251 provisioning services, configuration of any security related Resources and credentials for dealing  
1252 with any services that the Device need to contact later on.

1253 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to  
1254 be provisioned with proper security credentials and parameters for regular operation. These  
1255 parameters can include:

- 1256 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1257 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the  
1258 same OBT, but may be part of AMS in future.

1259 As mentioned, to accommodate a scalable and modular design, these functions are considered  
1260 as services that in future could be deployed as separate servers. Currently, the deployment  
1261 assumes that these services are all deployed as part of a OBT. Regardless of physical  
1262 deployment scenario, the same security-hardening requirement) applies to any physical server  
1263 that hosts the tools and security provisioning services discussed here.

1264 Devices are *aware* of their security provisioning status. Self-awareness allows them to be  
1265 proactive about provisioning or re-provisioning security Resources as needed to achieve the  
1266 devices operational goals.

### 1267 **5.4.2 Provisioning other services**

1268 To be able to support the use of potentially different device management service hosts, each  
1269 Device Secure Virtual Resource (SVR) has an associated Resource owner identified in the  
1270 Resource's rowneruuid Property.

1271 The DOTS shall update the rowneruuid Property of the "/oic/sec/doxm" and "/oic/sec/pstat"  
1272 resources with the DOTS resource owner identifier.

1273 The DOTS shall update the rowneruuid Property of the "/oic/sec/cred" resource with the CMS  
1274 resource owner identifier.

1275 The DOTS shall update the rowneruuid Property of the "/oic/sec/acl2" resource with the AMS  
1276 resource owner identifier

1277 When these OCF Services are configured, the Device may proactively request provisioning and  
1278 verify provisioning requests are authorized. The DOTS shall provision credentials that enable  
1279 secure connections between OCF Services and the new Device. The DOTS may initiate client-  
1280 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed  
1281 provisioning by setting tm Property of the "/oic/sec/pstat" Resource.



1282 **5.4.3 Provisioning Credentials for Normal Operation**

1283 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1284 – Pairwise symmetric keys
- 1285 – Group symmetric keys
- 1286 – Certificates
- 1287 – Raw asymmetric keys

1288 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS  
1289 credential provisioned by the DOTS.

1290 The following example describes how a Device updates a symmetric key credential involving a  
1291 peer Device. The Device discovers the credential to be updated; for example, a secure  
1292 connection attempt fails. The Device requests its CMS to supply the updated credential. The  
1293 CMS returns an updated symmetric key credential. The CMS updates the corresponding  
1294 symmetric key credential on the peer Device.

1295 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1296 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)  
1297 asserted by the Client requesting the Resource and compare that role identifier(s) with the  
1298 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned  
1299 with one or more role credentials.

1300 Each Device holds the role information as a Property within the credential Resource.

1301 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a  
1302 certificate role credential.

1303 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned  
1304 for a client, access to a Resource is granted if it would be granted under any of the roles.

1305 **5.4.5 ACL provisioning**

1306 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.  
1307 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL  
1308 policy by updating the Device's ACL Resources.

1309 The AMS shall digitally sign an ACL as part of issuing a "/oic/sec/sacl" Resource if the Device  
1310 supports the "/oic/sec/sacl" Resource. The public key used by the Device to verify the signature  
1311 shall be provisioned by the CMS as needed. A "/oic/sec/cred" Resource with an asymmetric key  
1312 type or signed asymmetric key type is used. The "PublicData" Property contains the AMS's public  
1313 key.

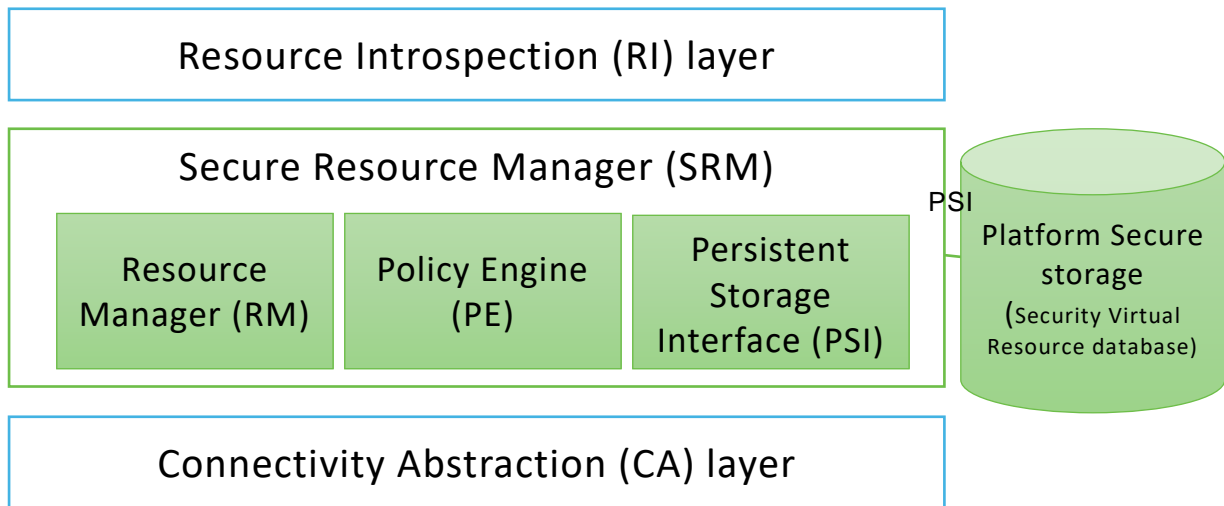
1314 **5.5 Secure Resource Manager (SRM)**

1315 SRM plays a key role in the overall security operation. In short, SRM performs both management  
1316 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3  
1317 main functional elements:

- 1318 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using  
1319 PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3)  
1320 Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a  
1321 format that is consistent with device-specific data store format. However, the RM will use  
1322 JSON format to marshal SVR data structures before being passed to PSI for storage, or travel  
1323 off-device.

- 1324 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control  
1325 policies responds to the requests with either "ACCESS\_GRANTED" or "ACCESS\_DENIED".  
1326 To make the access decisions, the PE consults the appropriate ACL and looks for best  
1327 Access Control Entry (ACE) that can serve the request given the subject (Device or role) that  
1328 was authenticated by DTLS.
- 1329 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in  
1330 its own memory and storage. The SRM design is modular such that it may be implemented in  
1331 the Platform's secure execution environment; if available.

1332 Figure 12 depicts OCF's SRM Architecture.



1333

1334

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

1335 **5.6 Credential Overview**

1336 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional  
1337 communication. Credentials can be symmetric or asymmetric. Each device stores secret and  
1338 public parts of its own credentials where applicable, as well as credentials for other devices that  
1339 have been provided by the DOTS or a CMS. These credentials are then used in the  
1340 establishment of secure communication sessions (e.g. using DTLS) to validate the identities of  
1341 the participating parties. Role credentials are used once an authenticated session is established,  
1342 to assert one or more roles for a device.

1343 Access Tokens are provided to an OCF Cloud once an authenticated session with an OCF Cloud  
1344 is established, to verify the User ID with which the Device is to be associated.

1345

## 1346 **6 Security for the Discovery Process**

### 1347 **6.1 Preamble**

1348 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,  
1349 called links) for the Resources hosted by the Server, complemented by attributes about those  
1350 Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-  
1351 1:2018)

### 1352 **6.2 Security Considerations for Discovery**

1353 When defining discovery process, care must be taken that only a minimum set of Resources are  
1354 exposed to the discovering entity without violating security of sensitive information or privacy  
1355 requirements of the application at hand. This includes both data included in the Resources, as  
1356 well as the corresponding metadata.

1357 To achieve extensibility and scalability, this document does not provide a mandate on  
1358 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on  
1359 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any  
1360 of the Resources.

1361 The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted  
1362 Resources. (See 13.5)

1363 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery  
1364 process itself needs to be secured. This document sets the following requirements for the  
1365 discovery process:

- 1366 1) Providing integrity protection for discovered Resources.
- 1367 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1368 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)  
1369 on the known "/oic/res" Resource.

1370 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a  
1371 Server cannot determine the identity of the requester. In such cases, a Server that wants to  
1372 authenticate the Client before responding can list the secure discovery URI (e.g.  
1373 coaps://IP:PORT/oic/res ) in the unsecured "/oic/res" Resource response. This means the secure  
1374 discovery URI is by default discoverable by any Client. The Client will then be required to send a  
1375 separate unicast request using DTLS to the secure discovery URI.

1376 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to  
1377 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN  
1378 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1379 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource  
1380 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1381 {  
1382   "aclist2": [  
1383     {  
1384       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1385       "resources": [{"href": "/door"}],  
1386       "permission": 2, // RETRIEVE  
1387       "aceid": 1  
1388     }  
1389   ],
```

```

1390     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1391   }
1392   {
1393     "aclist2": [
1394       {
1395         "subject": {"authority": "owner", "role": "owner"}
1396         "resources": [{"href": "/door"}],
1397         "permission": 2, // RETRIEVE
1398         "aceid": 2
1399       }
1400     ],
1401     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1402   }
1403   {
1404     "aclist2": [
1405       {
1406         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1407         "resources": [{"href": "/door/lock"}],
1408         "permission": 4, // UPDATE
1409         "aceid": 3
1410       }
1411     ],
1412     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1413   }
1414   {
1415     "aclist2": [
1416       {
1417         "subject": {"conntype": "anon-clear"},
1418         "resources": [{"href": "/light"}],
1419         "permission": 2, // RETRIEVE
1420         "aceid": 4
1421       }
1422     ],
1423     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1424   }

```

1425 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when  
1426 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will  
1427 include the URI of the "/door" Resource metadata. Client "d2" will have access to both the  
1428 Resources. ACE2 will prevent "d4" from update.

1429 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1430   [
1431     {
1432       "href": "/door",
1433       "rt": ["oic.r.door"],
1434       "if": ["oic.if.b", "oic.if.ll"],

```

```
1435     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1436   }
1437 ]
```

1438 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1439 [
1440   {
1441     "href": "/door",
1442     "rt": ["oic.r.door"],
1443     "if": ["oic.if.b", "oic.if.ll"],
1444     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1445   },
1446   {
1447     "href": "/door/lock",
1448     "rt": ["oic.r.lock"],
1449     "if": ["oic.if.b"],
1450     "type": ["application/json", "application/exi+xml"]
1451   }
1452 ]
```

1453 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

```
1454 [
1455   {
1456     "href": "/door/lock",
1457     "rt": ["oic.r.lock"],
1458     "if": ["oic.if.b"],
1459     "type": ["application/json", "application/exi+xml"],
1460     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1461   }
1462 ]
```

1463 Discovery results delivered to any device regarding d3's "/oic/res" Resource from the unsecure interface:

```
1465 [
1466   {
1467     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1468     "href": "/light",
1469     "rt": ["oic.r.light"],
1470     "if": ["oic.if.s"]
1471   }
1472 ]
1473
```

## 1474 **7 Security Provisioning**

### 1475 **7.1 Device Identity**

#### 1476 **7.1.1 General Device Identity**

1477 Each Device, which is a logical device, is identified with a Device ID.

1478 Devices shall be identified by a Device ID value that is established as part of device onboarding.  
1479 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. "urn:uuid"). Device IDs shall  
1480 be unique within the scope of operation of the corresponding OCF Security Domain, and should  
1481 be universally unique. The DOTS shall ensure Device ID of the new Device is unique within the  
1482 scope of the owner's OCF Security Domain. The DOTS shall verify the chosen new device  
1483 identifier does not conflict with Device IDs previously introduced into the OCF Security Domain.

1484 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"  
1485 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying  
1486 authentication credentials of a peer device.

1487 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of  
1488 credentials, both its own and other Device credentials, in the "/oic/sec/cred" Resource. The  
1489 device ID can be used to distinguish between a device's own credential, and credentials for other  
1490 devices. Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the  
1491 device.

1492 Device ID shall be:

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

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

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

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

#### 1504 **7.1.2 Device Identity for Devices with UAID [Deprecated]**

1505 This clause is intentionally left blank.

### 1506 **7.2 Device Ownership**

1507 This is an informative clause. Devices are logical entities that are security endpoints that have an  
1508 identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first  
1509 initialized. Establishing device ownership is a process by which the device asserts its identity to  
1510 the DOTS and the DOTS provisions an owner identity. This exchange results in the device  
1511 changing its ownership state, thereby preventing a different DOTS from asserting administrative  
1512 control over the device.

1513 The ownership transfer process starts with the OBT discovering a new device that is in Unowned  
1514 state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new  
1515 device. At the end of ownership transfer, the following is accomplished:

- 1516 1) The DOTS shall establish a secure session with new device.  
1517 2) Optionally asserts any of the following:  
1518 a) Proximity (using PIN) of the OBT to the Platform.  
1519 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific  
1520 attributes.  
1521 3) Determines the device identifier.  
1522 4) Determines the device owner.  
1523 5) Specifies the device owner (e.g. Device ID of the OBT).  
1524 6) Provisions the device with owner's credentials.  
1525 7) Sets the "Owned" state of the new device to TRUE.

1526 NOTE A Device which connects to the OCF Cloud still retains the ownership established at onboarding with the DOTS.

## 1527 **7.3 Device Ownership Transfer Methods**

### 1528 **7.3.1 OTM implementation requirements**

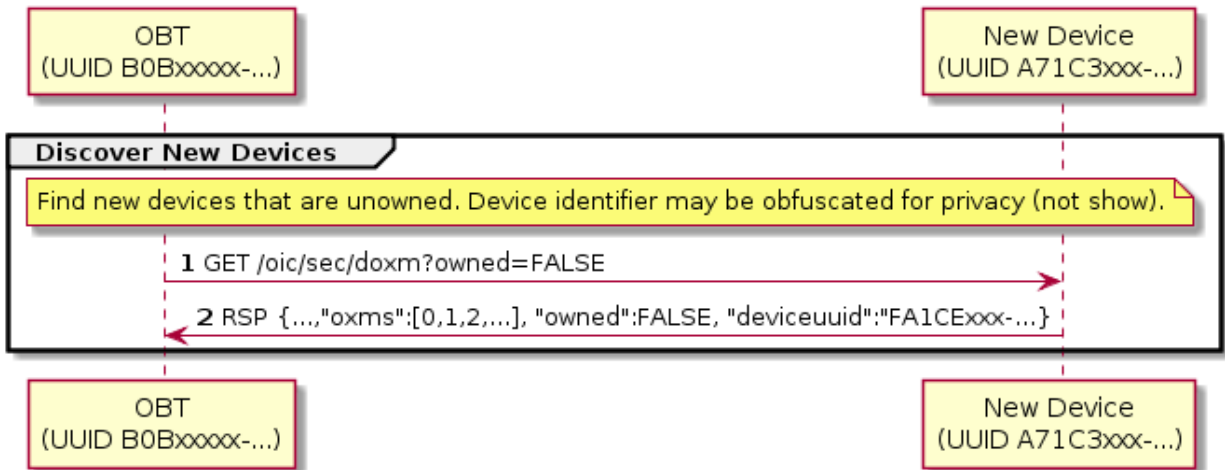
1529 This document provides specifications for several methods for ownership transfer.  
1530 Implementation of each individual ownership transfer method is considered optional. However,  
1531 each device shall implement at least one of the ownership transfer methods not including vendor  
1532 specific methods.

1533 All OTMs included in this document are considered optional. Each vendor is required to choose  
1534 and implement at least one of the OTMs specified in this document. The OCF, does however,  
1535 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability  
1536 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with  
1537 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the  
1538 preferred approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in  
1539 designing vendor-specific OTMs.

1540 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer  
1541 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.  
1542 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the  
1543 "/oic/sec/doxm" Resource. The DOTS shall query the Device's supported credential types using  
1544 the "credtype" Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision  
1545 credentials according to the credential types supported.

1546 Figure 13 depicts new Device discovery sequence.

### Discover New Devices Sequence



1547

1548

**Figure 13 – Discover New Device Sequence**

1549

1550

**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.

1551 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs  
 1552 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for  
 1553 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the  
 1554 new Device.

1555 The new device may have to perform some initialization steps at the beginning of an OTM. For  
 1556 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN  
 1557 value. The OBT shall POST to the oxmsel property of "/oic/sec/doxm" the value corresponding to  
 1558 the OTM being used, before performing other OTM steps. This POST notifies the new device that  
 1559 ownership transfer is starting.

1560 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT  
 1561 and the OBT to authenticate to the new device.

1562 The DOTS may perform additional provisioning steps subsequent to owner transfer success  
 1563 leveraging the established OTM session.

1564 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall  
 1565 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array  
 1566 includes a SVR.



### 1567 **7.3.2 SharedKey Credential Calculation**

1568 The SharedKey credential is derived using a PRF that accepts the key\_block value resulting from  
1569 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following  
1570 calculation to ensure interoperability across vendor products:

1571 SharedKey = PRF(Secret, Message);

1572 Where:

- 1573 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1574 - Secret is the key\_block resulting from the DTLS handshake
  - 1575 ▪ See IETF RFC 5246 clause 6.3
  - 1576 ▪ The length of key\_block depends on cipher suite.
    - 1577 • (e.g. 96 bytes for TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
    - 1578 40 bytes for TLS\_PSK\_WITH\_AES\_128\_CCM\_8)
- 1579 - Message is a concatenation of the following:
  - 1580 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
    - 1581 • See clause 13.2.4 for specific DoxmTypes
  - 1582 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
    - 1583 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
  - 1584 ▪ Device ID is new device's UUID Device ID
    - 1585 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1586 - SharedKey Length will be 32 octets.
  - 1587 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used.
  - 1588 DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

### 1589 **7.3.3 Certificate Credential Generation**

1590 The Certificate Credential will be used by Devices for secure bidirectional communication. The  
1591 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used  
1592 to mutually establish the authenticity of the Device. The onboarding details for certificate  
1593 generation will be specified in a later version of this document.

### 1594 **7.3.4 Just-Works OTM**

#### 1595 **7.3.4.1 Just-Works OTM General**

1596 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a  
1597 secure connection through which a device should be provisioned for use within the owner's OCF  
1598 Security Domain. Provisioning additional credentials and Resources is a typical step following  
1599 ownership establishment. The pre-shared key is called SharedKey.

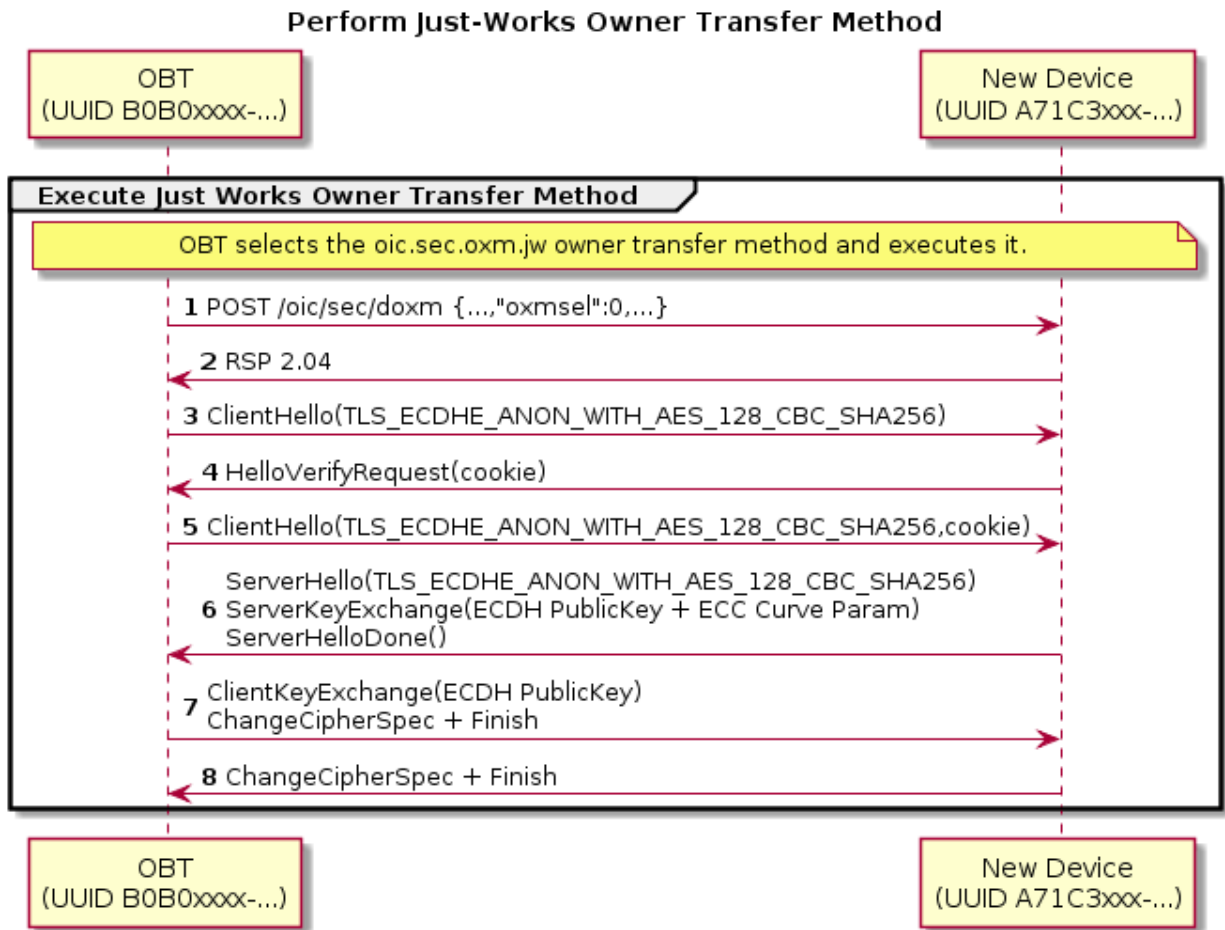
1600 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite  
1601 defined for the Just-works OTM.

1602 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1603 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,  
1604 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

1605 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for  
1606 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1607 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.



1608

1609

**Figure 14 – A Just Works OTM**

1610

1611

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

Step	Description
1, 2	The OBT notifies the Device that it selected the "Just Works" method.
3 - 8	A DTLS session is established using anonymous Diffie-Hellman. <sup>a</sup>
<sup>a</sup> This method assumes the operator is aware of the potential for man-in-the-middle attack and has taken precautions to perform the method in a clean-room network.	

**1612 7.3.4.2 Security Considerations**

1613 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this  
 1614 method presumes that both the OBT and the new device perform the "just-works" method  
 1615 assumes onboarding happens in a relatively safe environment absent of an attack device.

1616 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to  
 1617 the device.

1618 The new device should use a temporal device ID prior to transitioning to an owned device while it  
 1619 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-

1620 temporal device ID that could differ from the temporal value during the secure session in which  
1621 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not  
1622 conflict with a Device ID already in use. If it is already in use the existing credentials are used to  
1623 establish a secure session.

1624 An un-owned Device that also has established device credentials might be an indication of a  
1625 corrupted or compromised device.

### 1626 **7.3.5 Random PIN Based OTM**

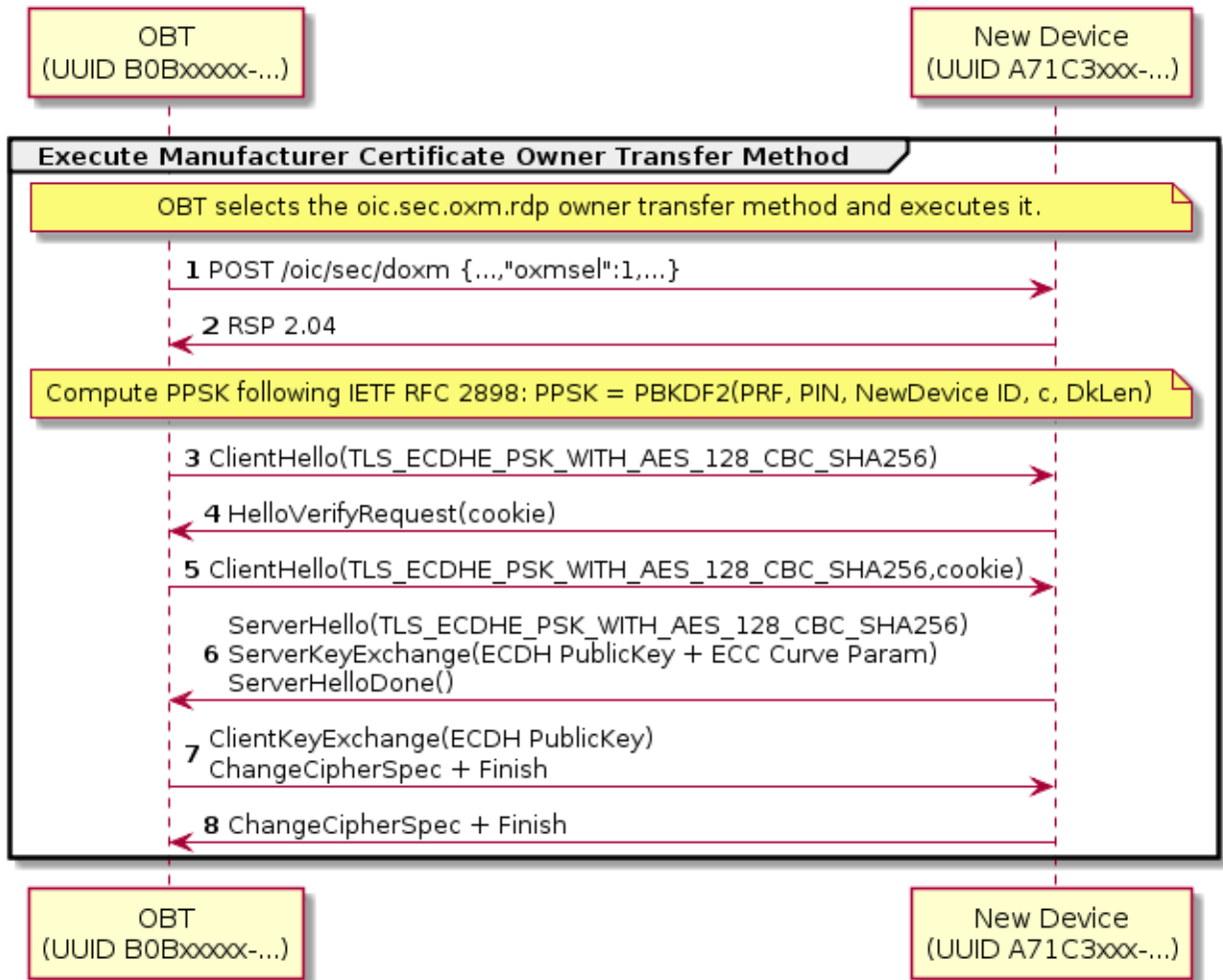
#### 1627 **7.3.5.1 Random PIN OTM General**

1628 The Random PIN method establishes physical proximity between the new device and the OBT  
1629 can prevent man-in-the-middle attacks. The Device generates a random number that is  
1630 communicated to the OBT over an out-of-band channel. The definition of out-of-band  
1631 communications channel is outside the scope of the definition of device OTMs. The OBT and new  
1632 Device use the PIN in a key exchange as evidence that someone authorized the transfer of  
1633 ownership by having physical access to the new Device via the out-of-band-channel.

#### 1634 **7.3.5.2 Random PIN Owner Transfer Sequence**

1635 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.

### Perform Random PIN Device Owner Transfer Method



1636

1637

**Figure 15 – Random PIN-based OTM**

1638

1639

**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.

1640

1641

1642

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

1643 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)  
1644 The PBKDF2 function has the following parameters:  
1645 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.  
1646 - PIN – obtain via out-of-band channel.  
1647 - Device ID – UUID of the new device.  
1648 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2  
1649 - c – Iteration count initialized to 1000  
1650 - dkLen – Desired length of the derived PSK in octets.

### 1651 **7.3.5.3 Security Considerations**

1652 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with  
1653 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials  
1654 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,  
1655 allows an attacker to masquerade as the onboarded device.

1656 It is recommended that the entropy of the PIN be enough to withstand an online brute-force  
1657 attack, 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-  
1658 9a-z), or a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack  
1659 (MITM) is when the attacker is active on the network and can intercept and modify messages  
1660 between the OBT and device. In the MITM attack, the attacker must recover the PIN from the key  
1661 exchange messages in "real time", i.e., before the peer's time out and abort the connection  
1662 attempt. Having recovered the PIN, he can complete the authentication step of key exchange.  
1663 The guidance given here calls for a minimum of 40 bits of entropy, however, the assurance this  
1664 provides depends on the resources available to the attacker. Given the parallelizable nature of a  
1665 brute force guessing attack, the attack enjoys a linear speedup as more cores/threads are added.  
1666 A more conservative amount of entropy would be 64 bits. Since the Random PIN OTM requires  
1667 using a DTLS ciphersuite that includes an ECDHE key exchange, the security of the Random PIN  
1668 OTM is always at least equivalent to the security of the JustWorks OTM.

1669 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN.  
1670 The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess  
1671 in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an  
1672 effective way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify  
1673 the reduction, since an X-fold increase in time spent by the honest peers does not directly  
1674 translate to an X-fold increase in time by the attacker. This asymmetry is because the attacker  
1675 may use specialized implementations and hardware not available to honest peers. For this  
1676 reason, when deciding how much entropy to use for a PIN, it is recommended that implementers  
1677 assume PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

1678 The Random PIN device OTM security depends on an assumption that a secure out-of-band  
1679 method for communicating a randomly generated PIN from the new device to the OBT exists. If  
1680 the OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN,  
1681 and the attacks described above apply. The out-of-band mechanism should be chosen such that  
1682 it requires proximity between the OBT and the new device. The attacker is assumed to not have  
1683 compromised the out-of-band-channel. As an example OOB channel, the device may display a  
1684 PIN to be entered into the OBT software. Another example is for the device to encode the PIN as  
1685 a 2D barcode and display it for a camera on the OBT device to capture and decode.

### 1686 **7.3.6 Manufacturer Certificate Based OTM**

#### 1687 **7.3.6.1 Manufacturer Certificate Based OTM General**

1688 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the  
1689 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device  
1690 and the OBT.

1691 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust  
1692 anchor.

1693 For some environments, policies or administrators, additional information about device  
1694 characteristics may be sought. This list of additional attestations that OCF may or may not have  
1695 tested (understanding that some attestations are incapable of testing or for which testing may be  
1696 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3  
1697 extension described in 9.4.2.2.6.

1698 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with  
1699 certificate data to authenticate their identities with the OBT in the process of bringing a new  
1700 device into operation on an OCF Security Domain. The onboarding process involves several  
1701 discrete steps:

1702 1) Pre-on-board conditions

1703 a) The credential element of the Device's credential Resource ("/oic/sec/cred") containing the  
1704 manufacturer certificate shall be identified by the "credusage" Property containing the  
1705 string "oic.sec.cred.mfgcert" to indicate that the credential contains a manufacturer  
1706 certificate.

1707 b) The manufacturer certificate chain shall be contained in the identified credential element's  
1708 "publicdata" Property.

1709 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1710 d) If the device requires authentication of the OBT as part of ownership transfer, it is  
1711 presumed that the OBT has been registered and has obtained a certificate for its unique  
1712 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.

1713 e) User has configured the OBT app with network access info and account info (if any).

1714 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the  
1715 Device may authenticate the OBT to verify the OBT signature.

1716 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready  
1717 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted  
1718 link in accordance with the negotiated cipher suite.

1719 **7.3.6.2 Certificate Profiles**

1720 See 9.4.2 for details.

1721 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

1722 In order for full, mutual authentication to occur between the device and the OBT, both the device  
1723 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This  
1724 implies that OCF may need to obtain services from a Certificate Authority (e.g. Symantec,  
1725 Verisign, etc.) to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors  
1726 are derived.

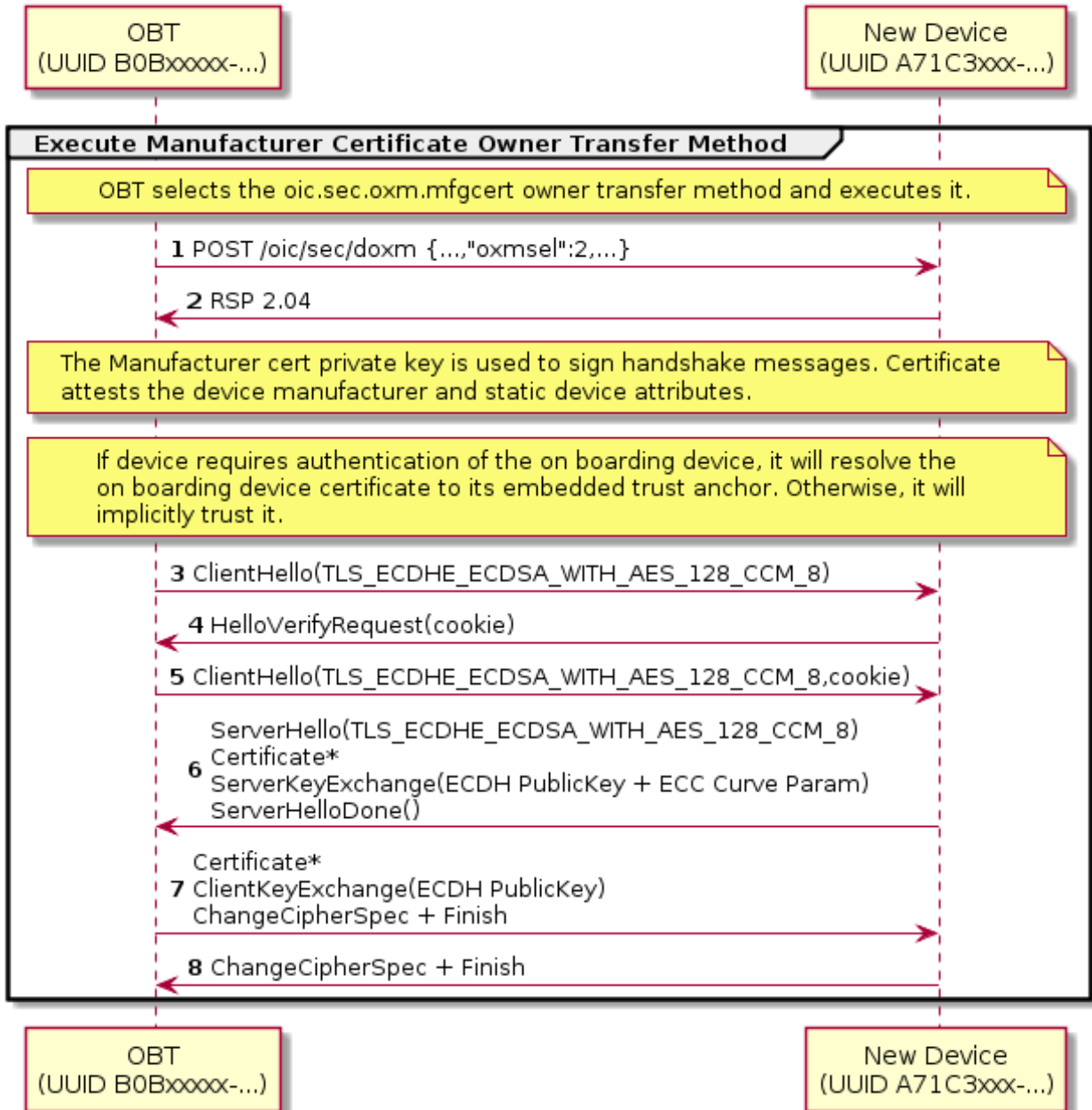
1727 The OBT shall authenticate the device during onboarding. However, the device is not required to  
1728 authenticate the OBT due to potential resource constraints on the device.

1729 In the case where the Device does NOT authenticate the OBT software, there is the possibility of  
1730 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,  
1731 which can compromise OCF Security Domain access credentials and/or personal information.

1732 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1733 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.

## Perform Manufacturer Certificate Owner Transfer Method



1734

1735

1736

1737

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

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

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

	attesting to the Device hardening and security properties.
--	--

1738 **7.3.6.5 Security Considerations**

1739 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of  
1740 assurance that the private key cannot be compromised.

1741 The Platform manufacturer issues the manufacturer certificate and attests the private key  
1742 protection mechanism.

1743 **7.3.7 Vendor Specific OTMs**

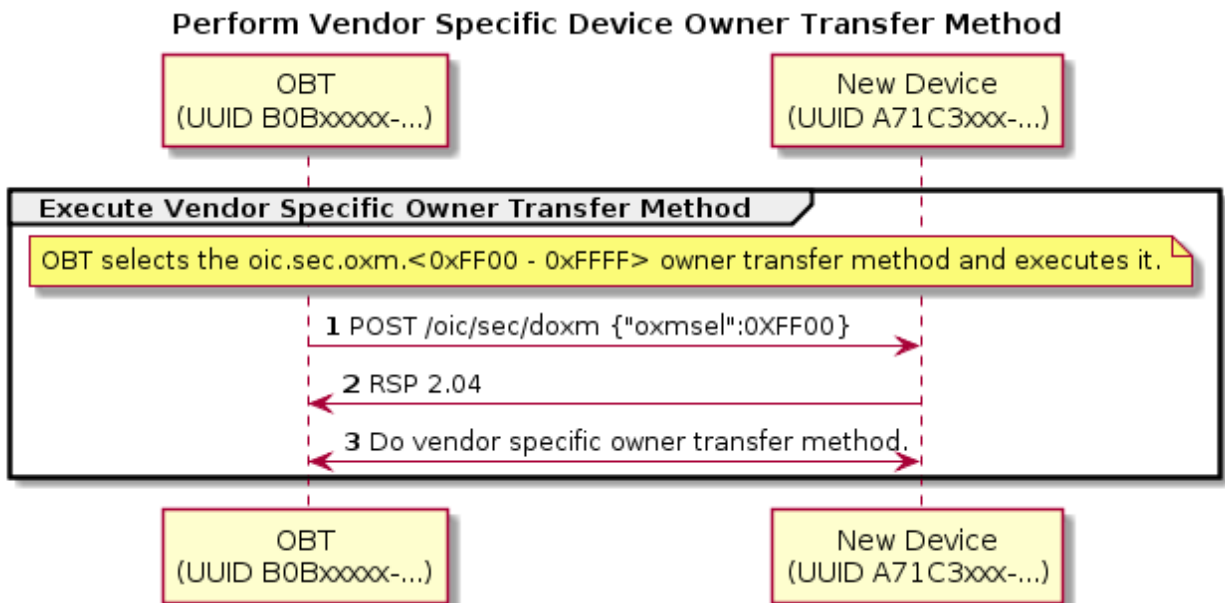
1744 **7.3.7.1 Vendor Specific OTM General**

1745 The OCF anticipates situations where a vendor will need to implement an OTM that  
1746 accommodates manufacturing or Device constraints. The Device OTM resource is extensible for  
1747 this purpose. Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1748 – The OBT must determine which credential types are supported by the Device. This is  
1749 accomplished by querying the Device's "/oic/sec/doxm" Resource to identify supported  
1750 credential types.
- 1751 – The OBT provisions the Device with OC(s).
- 1752 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1753 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security  
1754 Domain subsequent to ownership establishment.
- 1755 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks  
1756 to be performed by a security service.

1757 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1758 Vendor-specific OTM sequence example is shown in Figure 17 and steps described in Table 5.



1759

1760 **Figure 17 – Vendor-specific Owner Transfer Sequence**

1761



1762

**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

1763 **7.3.7.3 Security Considerations**

1764 The vendor is responsible for considering security threats and mitigation strategies.

1765 **7.3.8 Establishing Owner Credentials**

1766 Once the OBT and the new Device have authenticated and established an encrypted connection  
1767 using one of the defined OTM methods.

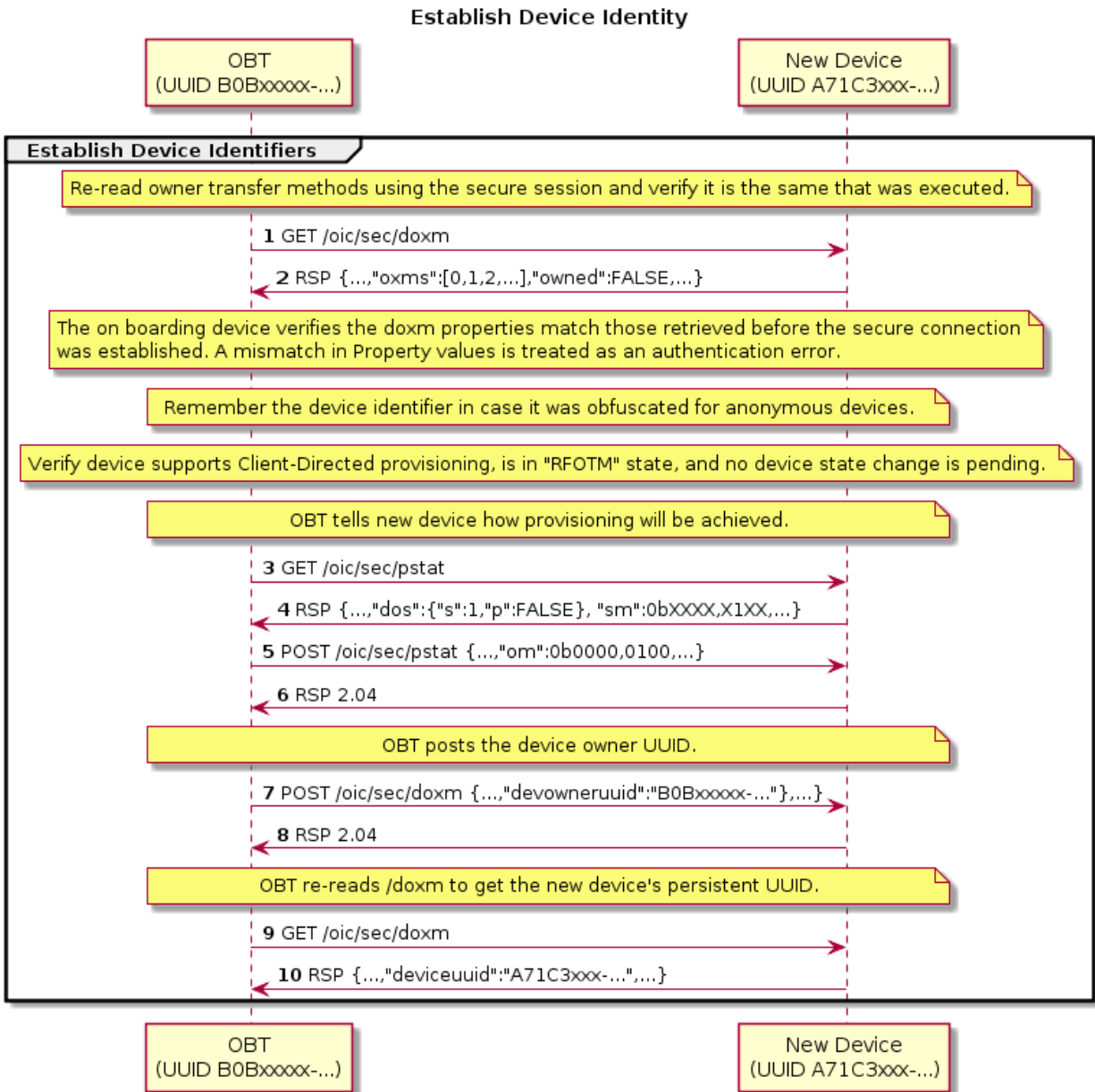
1768 Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security  
1769 Domain access information, provisioning functions, shared keys, or Kerberos tickets.

1770 The OBT might then provision the new Device with additional credentials for Device management  
1771 and Device-to-Device communications. These credentials may consist of certificates with  
1772 signatures, UAID based on the Device public key, PSK, etc.

1773 The steps for establishing Device's owner credentials (OC) are:

- 1774 1) The OBT shall establish the Device ID and Device owner uuid - See Figure 18 and Table 6.
- 1775 2) The OBT then establishes Device's OC - See Figure 19 and Table 7. This can be either:
  - 1776 a) Symmetric credential - See Figure 20 and Table 8.
  - 1777 b) Asymmetric credential - See Figure 21 and Table 9.
- 1778 3) Configure Device services - See Figure 22 and Table 10.
- 1779 4) Configure Device for peer to peer interaction - See Figure 23 and Table 11.

1780



1781

1782

1783

1784

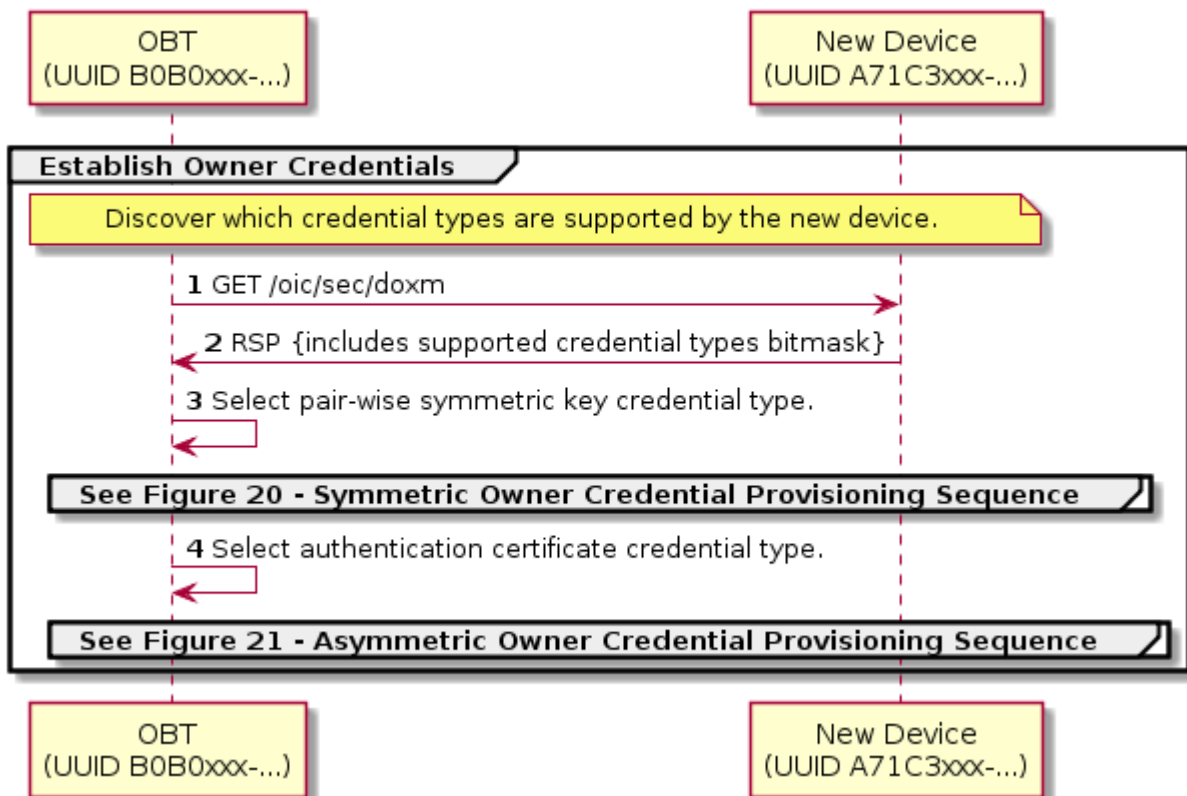
**Figure 18 – Establish Device Identity Flow**

**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



1785

1786

Figure 19 – Owner Credential Selection Provisioning Sequence

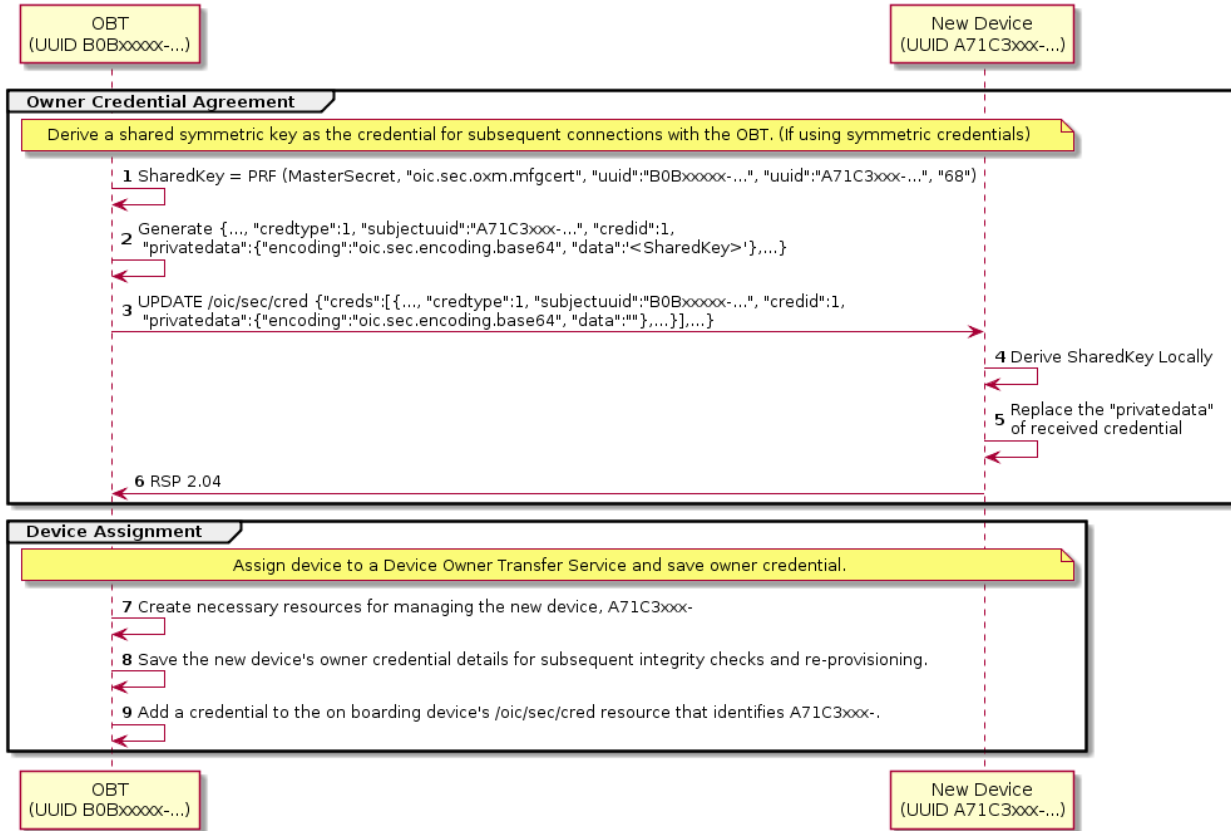
1787

1788

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



1789

1790

1791

1792

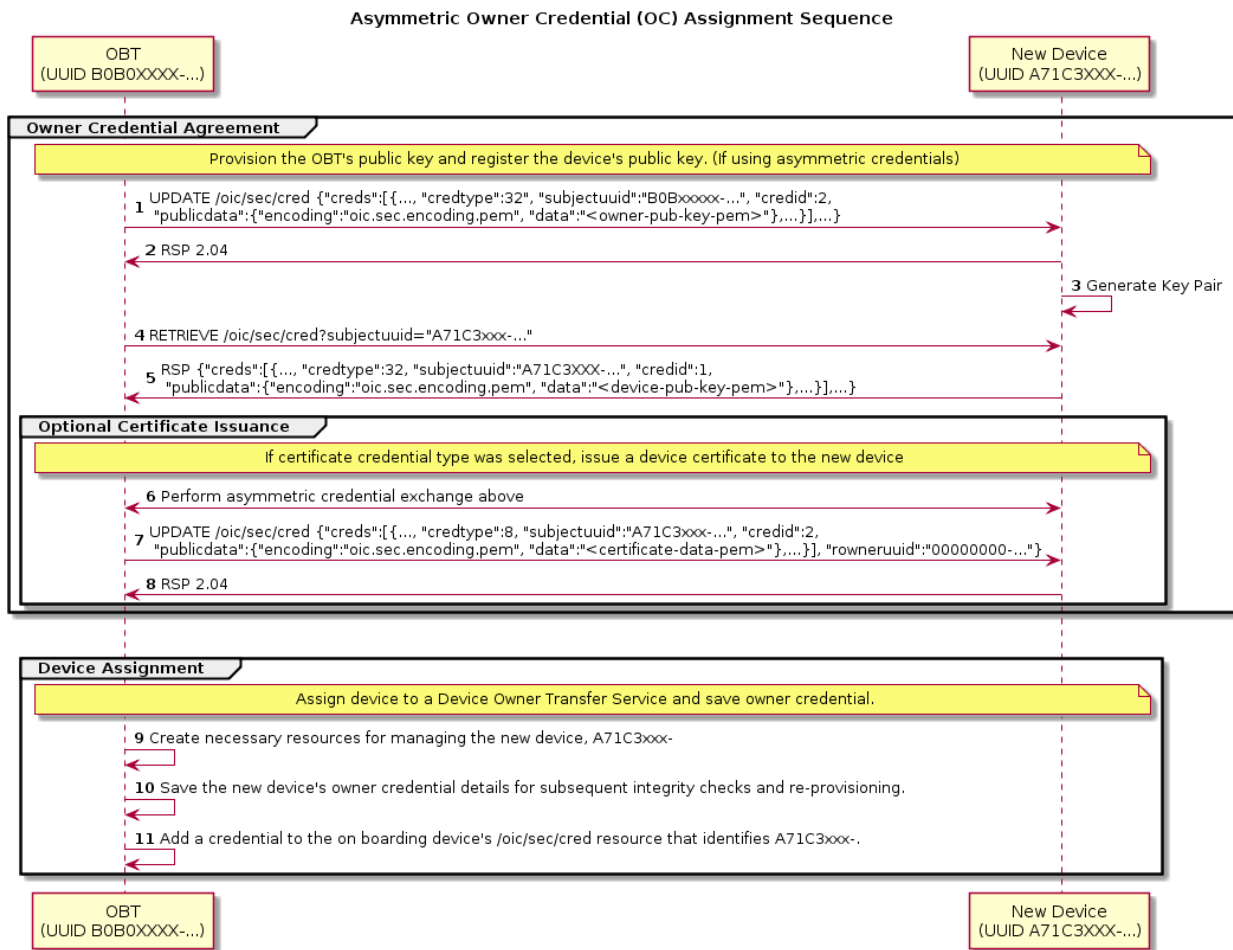
Figure 20 – Symmetric Owner Credential Provisioning Sequence

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.

- 1793 In particular, if the OBT selects symmetric owner credentials:
- 1794 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
  - 1795 described in 7.3.2.
  - 1796 – The OBT shall send an empty key to the new Device's "/oic/sec/cred" Resource, identified as
  - 1797 a symmetric pair-wise key.
  - 1798 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
  - 1799 generate the Shared Key using the SharedKey Credential Calculation method described in
  - 1800 7.3.2 and store it with the owner credential.
  - 1801 – The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred"
  - 1802 Resource to authenticate the owner during subsequent connections.



1803

1804

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

1805

1806

**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

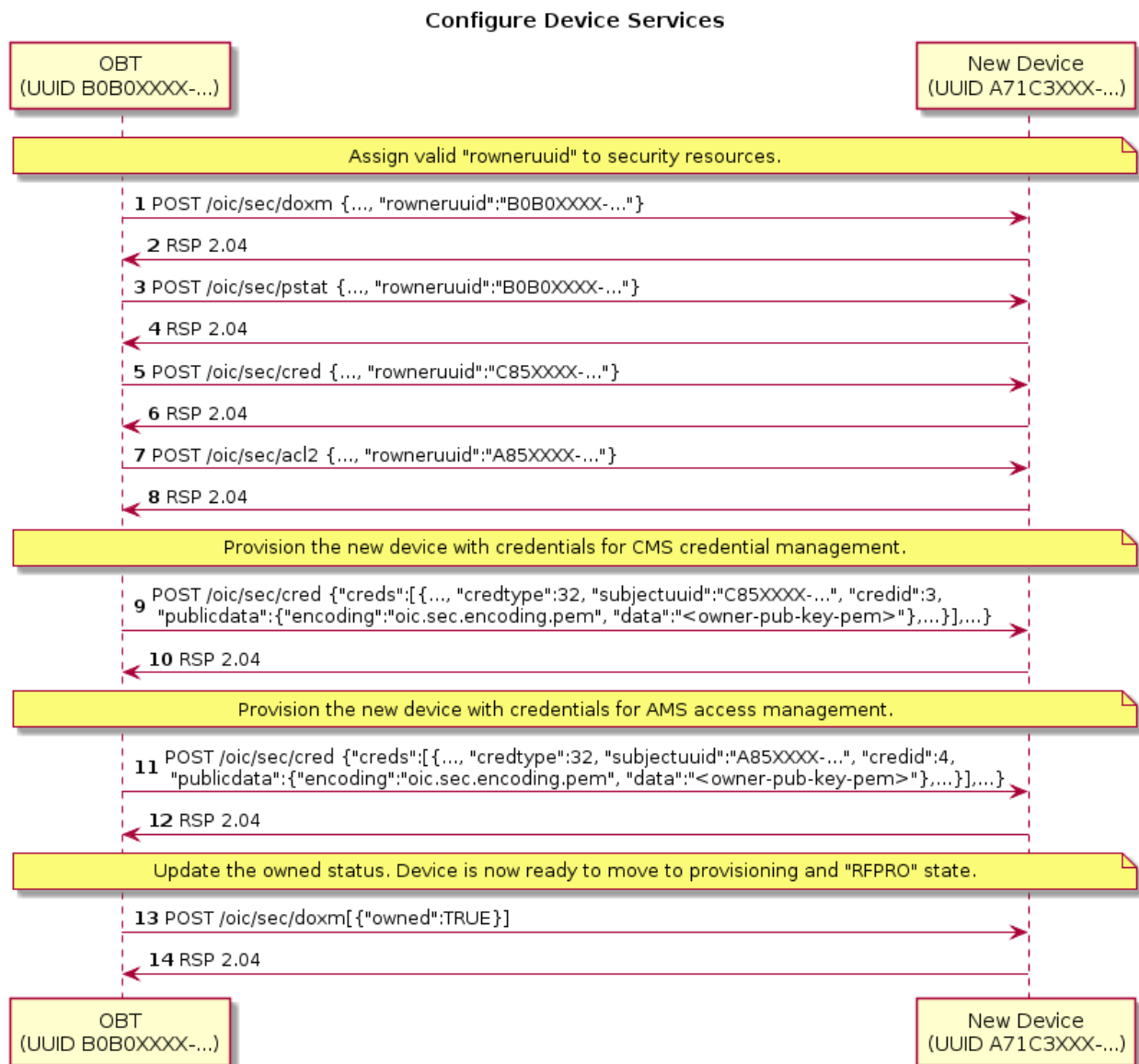
	Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric 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.

1807 If the OBT selects asymmetric owner credentials:

- 1808 – The OBT shall add its public key to the new Device's "/oic/sec/cred" Resource, identified as  
1809 an Asymmetric Encryption Key.
- 1810 – The OBT shall query the "/oic/sec/cred" Resource from the new Device, supplying the new  
1811 Device's UUID via the SubjectID query parameter. In response, the new Device shall return  
1812 the public Asymmetric Encryption Key, which the OBT shall retain for future owner  
1813 authentication of the new Device.

1814 If the OBT selects certificate owner credentials:

- 1815 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the  
1816 public key returned by the new Device, signed by a mutually-trusted CA, and complying with  
1817 the Certificate Credential Generation requirements defined in 7.3.3.
- 1818 – The OBT shall add the newly-created certificate chain to the "/oic/sec/cred" Resource,  
1819 identified as an Asymmetric Signing Key with Certificate.



1820

1821

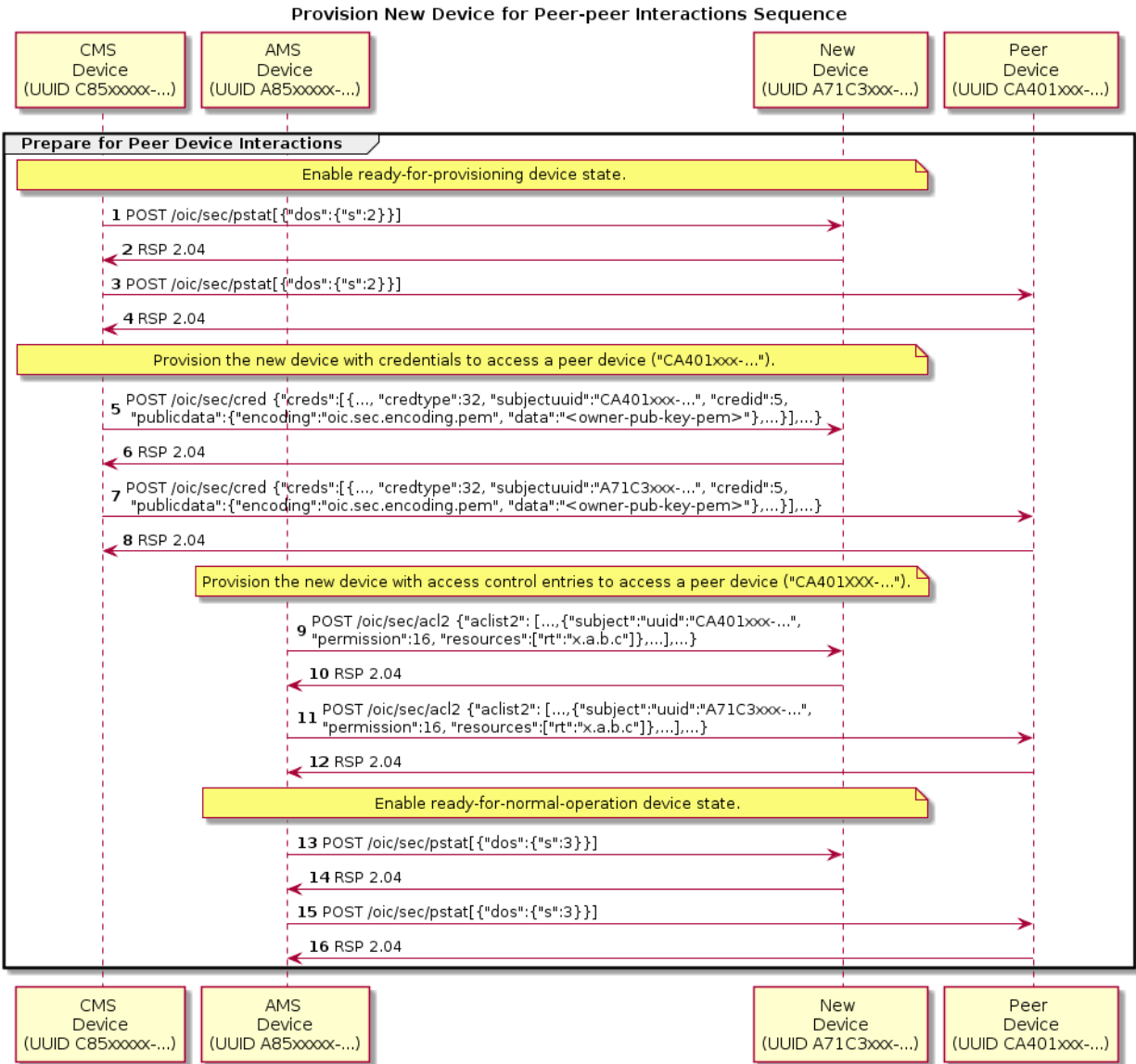
1822

1823

**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.



1824

1825

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

1826

1827

**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.



1828 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1829 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are  
1830 acceptable when transferring ownership of a new Device. Some of the factors to be considered  
1831 when determining those requirements are:

- 1832 – The security considerations described for each of the OTMs
- 1833 – The probability that a man-in-the-middle attacker might be present in the environment used to  
1834 perform the ownership transfer

1835 For example, the operator of an OBT might require that all of the Devices being onboarded  
1836 support either the Random PIN or the Manufacturer Certificate OTM.

1837 When such a local OTM policy exists, the OBT should try to use just the OTMs that are  
1838 acceptable according to that policy, regardless of the doxm contents obtained during step 1 from  
1839 the sequence diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an  
1840 unauthenticated and/or unencrypted connection between the OBT and the Device, the contents of  
1841 the response to the GET request might have been tampered by a man-in-the-middle attacker. For  
1842 example, the list of OTMs supported by the new Device might have been altered by the attacker.

1843 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new  
1844 Device to fail. In such cases, the OBT has no way of determining if the session failed because  
1845 the new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle  
1846 injected such a failure into the communication between the OBT and the new Device.

1847 The current version of this document leaves the design and user experience related to the OTM  
1848 policy as OBT implementation details.

1849 **7.3.10 Security Profile Assignment**

1850 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results  
1851 could be accessed from a manufacturer's certificate, OCF web server or other public repository.  
1852 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device  
1853 is authorized to possess and configures the Device with the subset of evaluated security profiles  
1854 best suited for the OCF Security Domain owner's intended segmentation strategy.

1855 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles"  
1856 Property of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification  
1857 process. The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the  
1858 values contained in the "supportedprofiles". The manufacturer default value shall be re-asserted  
1859 when the Device transitions to RESET Device State.

1860 The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in  
1861 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as  
1862 directed by a Security Profile.

1863 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a  
1864 subset of the OCF Security Profiles values the Device achieved as part of OCF Conformance  
1865 testing. The DOTS may locate conformance results by inspecting manufacturer certificates  
1866 supplied with the OCF Device by selecting the "credusage" Property of the "/oic/sec/cred"  
1867 Resource having the value of "oic.sec.cred.mfgcert". The DOTS may further locate conformance  
1868 results by visiting a well-known OCF web site URI corresponding to the ocCPLAttributes  
1869 extension fields (clause 9.4.2.2.7). The DOTS may select a subset of Security Profiles (from  
1870 those evaluated by OCF conformance testing) based on a local policy.

1871 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries  
1872 to allow DOTS access subsequent to onboarding.

1873 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the  
1874 value that most correctly depicts the OCF Security Domain owner's intended Device deployment  
1875 strategy.

1876 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID")  
1877 to indicate the OCF Security Domain owner's intention to segment the OCF Security Domain  
1878 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the  
1879 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security  
1880 Profiles when issuing role credentials.

1881 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control  
1882 entries that include the role designation(s).

## 1883 **7.4 Provisioning**

### 1884 **7.4.1 Provisioning Flows**

#### 1885 **7.4.1.1 Provisioning Flows General**

1886 As part of onboarding a new Device a secure channel is formed between the new Device and the  
1887 OBT. Subsequent to the Device ownership status being changed to "owned", there is an  
1888 opportunity to begin provisioning. The OBT decides how the new Device will be managed going  
1889 forward and provisions the support services that should be subsequently used to complete  
1890 Device provisioning and on-going Device management.

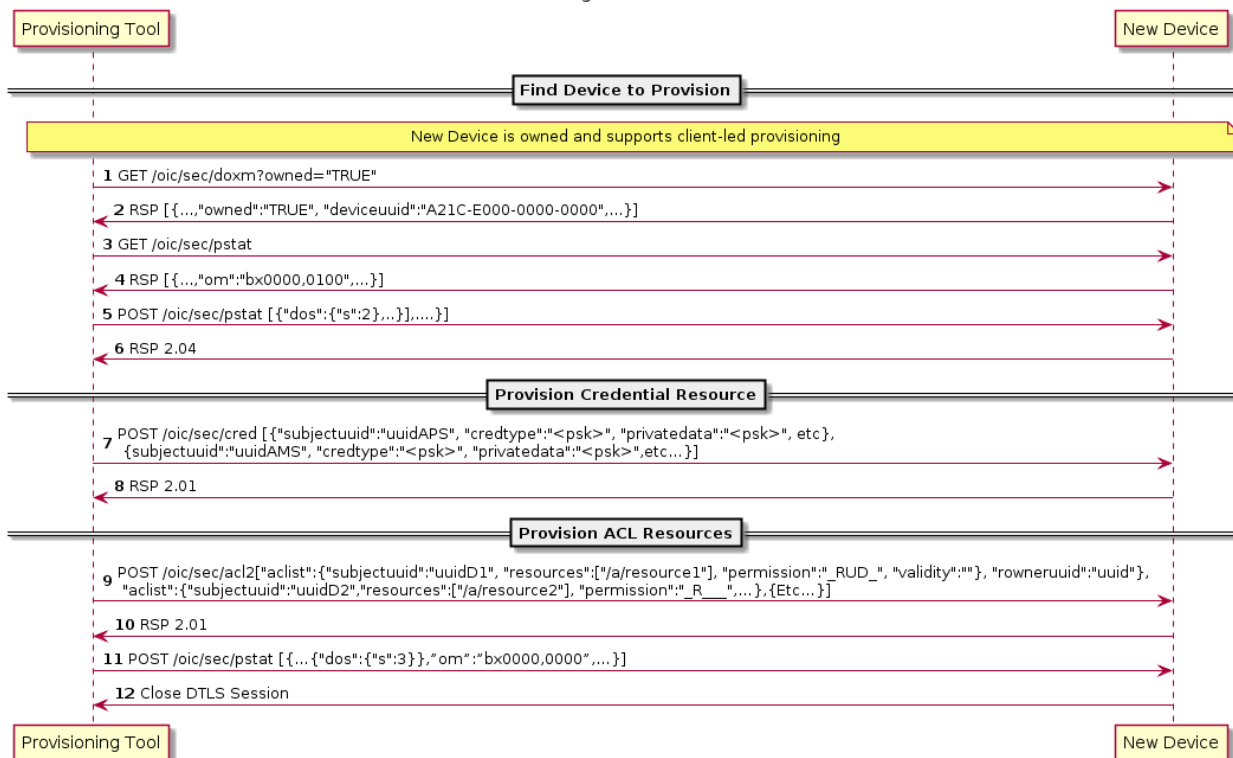
1891 The Device employs a Server-directed or Client-directed provisioning strategy. The  
1892 "/oic/sec/pstat" Resource identifies the provisioning strategy and current provisioning status. The  
1893 provisioning service should determine which provisioning strategy is most appropriate for the  
1894 OCF Security Domain. See 13.8 for additional detail.

#### 1895 **7.4.1.2 Client-directed Provisioning**

1896 Client-directed provisioning relies on a provisioning service that identifies Servers in need of  
1897 provisioning then performs all necessary provisioning duties.

1898 An example of Client-directed provisioning is shown in Figure 24 and steps described in Table 12.

OCF Client-directed Provisioning  
with a Single Service Provider



1899

1900

1901

1902

Figure 24 – Example of Client-directed provisioning

Table 12 – Steps describing Client -directed provisioning

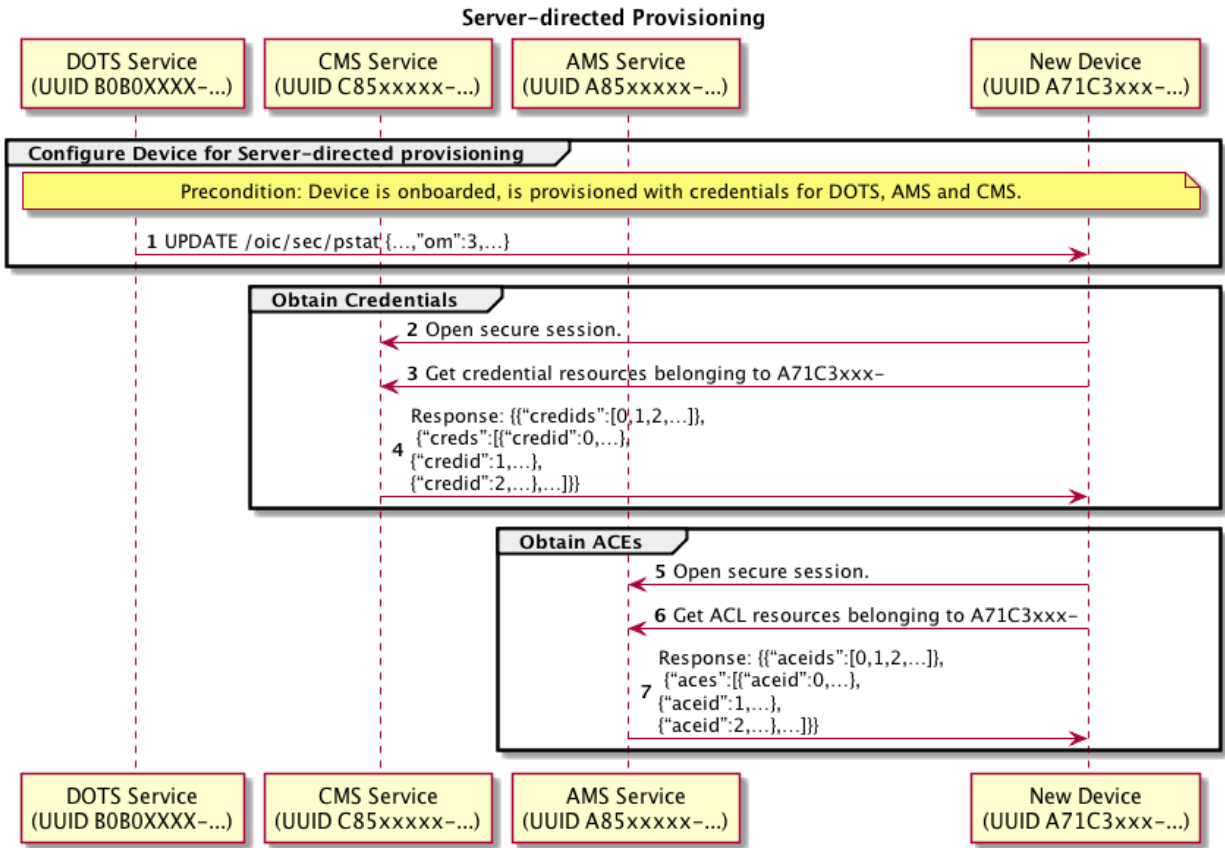
Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The "/oic/sec/doxm" Resource identifies the Device and it's owned status.
3	Provisioning Tool (PT) obtains the new Device's provisioning status found in "/oic/sec/pstat" Resource
4	The "pstat" Resource describes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

1903 **7.4.1.3 Server-directed Provisioning**

1904 Server-directed provisioning relies on the Server (i.e. new Device) for directing much of the  
 1905 provisioning work. As part of the onboarding process the support services used by the Server to  
 1906 seek additional provisioning are provisioned. The new Device uses a self-directed, state-driven  
 1907 approach to analyse current provisioning state, and tries to drive toward target state. This  
 1908 example assumes a single support service is used to provision the new Device.

1909 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1910  
 1911 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**

1912 **Table 13 – Steps for Server-directed provisioning using a single provisioning service**

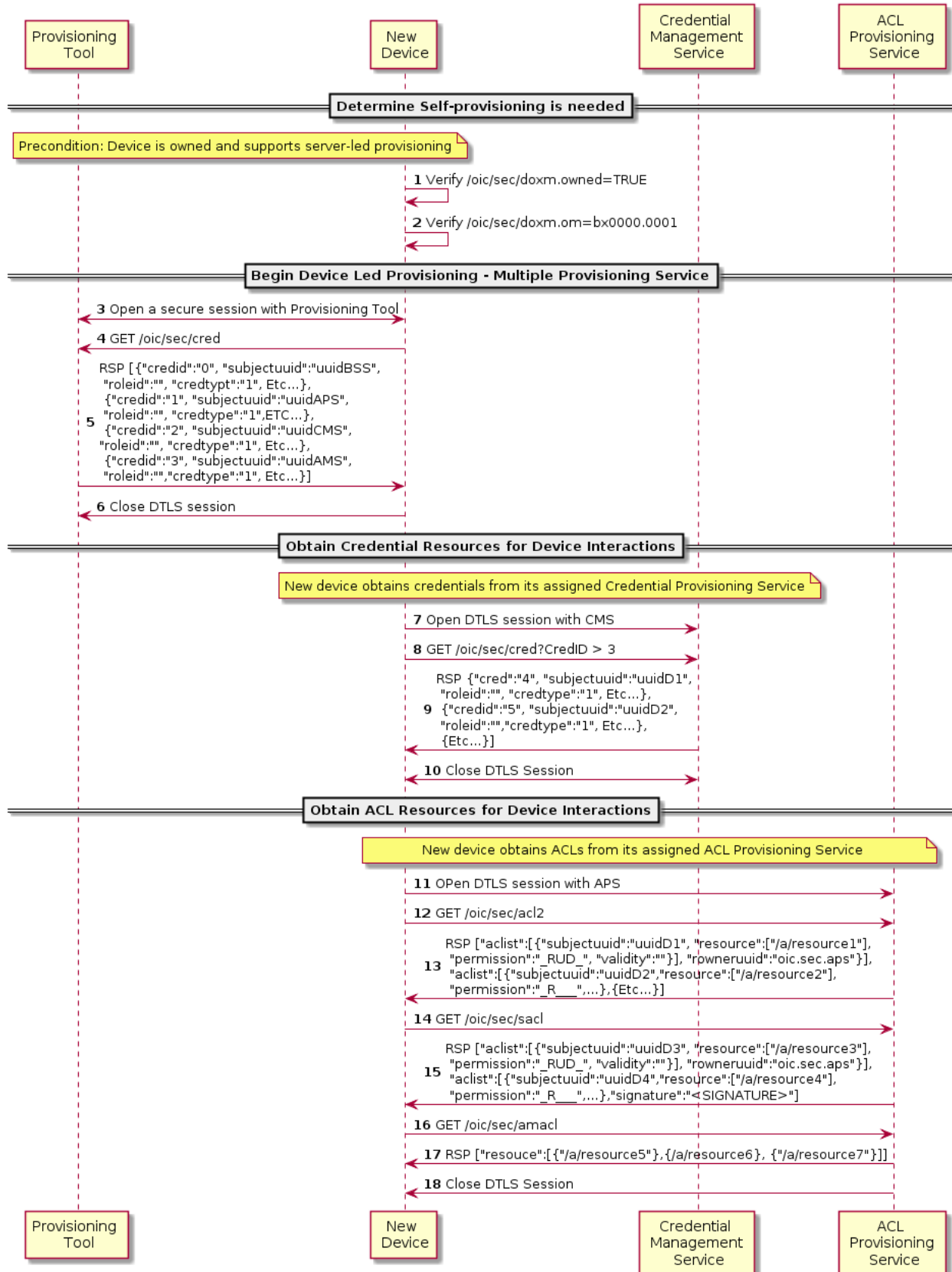
Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a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

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

1913 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1914 A Server-directed provisioning flow, involving multiple support services distributes the  
 1915 provisioning work across multiple support services. Employing multiple support services is an  
 1916 effective way to distribute provisioning workload or to deploy specialized support. The example in  
 1917 Figure 26 demonstrates using a provisioning tool to configure two support services, a CMS and  
 1918 an AMS. Steps for the example are described in Table 14.

### OCF Server Led Provisioning with Multiple Service Providers



1920 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1921 **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.

1922 **7.5 Device Provisioning for OCF Cloud**

1923 **7.5.1 Cloud Provisioning General**

1924 The Device that connects to the OCF Cloud shall support the "oic.r.coapcloudconf" Resource on  
 1925 Device and following SVRs on the OCF Cloud: "/oic/sec/account", "/oic/sec/session",  
 1926 "/oic/sec/tokenrefresh".

1927 The OCF Cloud is expected to use a secure mechanism for associating a Mediator with an OCF  
 1928 Cloud User. The choice of mechanism is up to the OCF Cloud. Example, mechanisms include  
 1929 HTTP authentication (with username and password) or OAuth 2.0 (using an Authorization Server  
 1930 which could be operated by the OCF Cloud provider or a third party). OCF Cloud is expected to  
 1931 ensure that the suitable authentication mechanism is used to authenticate the OCF Cloud User.

1932 **7.5.2 Device Provisioning by Mediator**

1933 The Mediator and the Device shall use the secure session to provision the Device to connect with  
 1934 the OCF Cloud.

1935 The Mediator obtains an Access Token from the OCF Cloud as described in OCF Cloud  
 1936 Specification. This Access Token is then used by the Device for registering with the OCF Cloud

1937 as described in 10.5. The OCF Cloud maintains a map where Access Token and Mediator  
 1938 provided Device ID are stored. At the time of Device Registration OCF Cloud validates the  
 1939 Access Token and associates the TLS session with corresponding Device ID.

1940 The Mediator provisions the Device, as described in OCF Cloud Specification. The Mediator  
 1941 provisions OCF Cloud URI to the "cis" Property of "oic.r.coapcloudconf" Resource, OCF Cloud  
 1942 UUID to the "sid" Property of "oic.r.coapcloudconf" Resource and per-device Access Token to the  
 1943 "at" Property of "oic.r.coapcloudconf" Resource on Device. Provisioned "at" is to be treated by  
 1944 Device as an Access Token with "Bearer" token type as defined in IETF RFC 6750.

1945 For the purposes of access control, the Device shall identify the OCF Cloud using the OCF Cloud  
 1946 UUID in the Common Name field of the End-Entity certificate used to authenticate the OCF Cloud.

1947 AMS should configure the ACE2 entries on a Device so that the Mediator(s) is the only Device(s)  
 1948 with UPDATE permission for the "oic.r.coapcloudconf" Resource.

1949 The AMS should configure the ACE2 entries on the Device to allow request from the OCF Cloud.  
 1950 By request from the Mediator, the AMS removes old ACL2 entries with previous OCF Cloud UUID.  
 1951 This request happens before "oic.r.coapcloudconf" is configured by the Mediator for the new OCF  
 1952 Cloud. The Mediator also requests AMS to set the OCF Cloud UUID as the "subject" Property for  
 1953 the new ACL2 entries. AMS may use "sid" Property of "oic.r.coapcloudconf" Resource as the  
 1954 current OCF Cloud UUID. AMS could either provision a wildcard entry for the OCF Cloud or  
 1955 provision an entry listing each Resource published on the Device.

1956 If OCF Cloud provides "redirecturi" Value as response during Device Registration, the redirected-  
 1957 to OCF Cloud is assumed to have the same OCF Cloud UUID and to use the same trust anchor.  
 1958 Otherwise, presented OCF Cloud UUID wouldn't match the provisioned ACL2 entries.

1959 The Mediator should provision the "oic.r.coapcloudconf" Resource with the Properties in Table 15.  
 1960 These details once provisioned are used by the Device to perform Device Registration to the  
 1961 OCF Cloud. After the initial registration, the Device should use updated values received from the  
 1962 OCF Cloud instead. If OCF Cloud User wants the Device to re-register with the OCF Cloud, they  
 1963 can use the Mediator to re-provision the "oic.r.coapcloudconf" Resource with the new values.

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

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

## 1966 **8 Device Onboarding State Definitions**

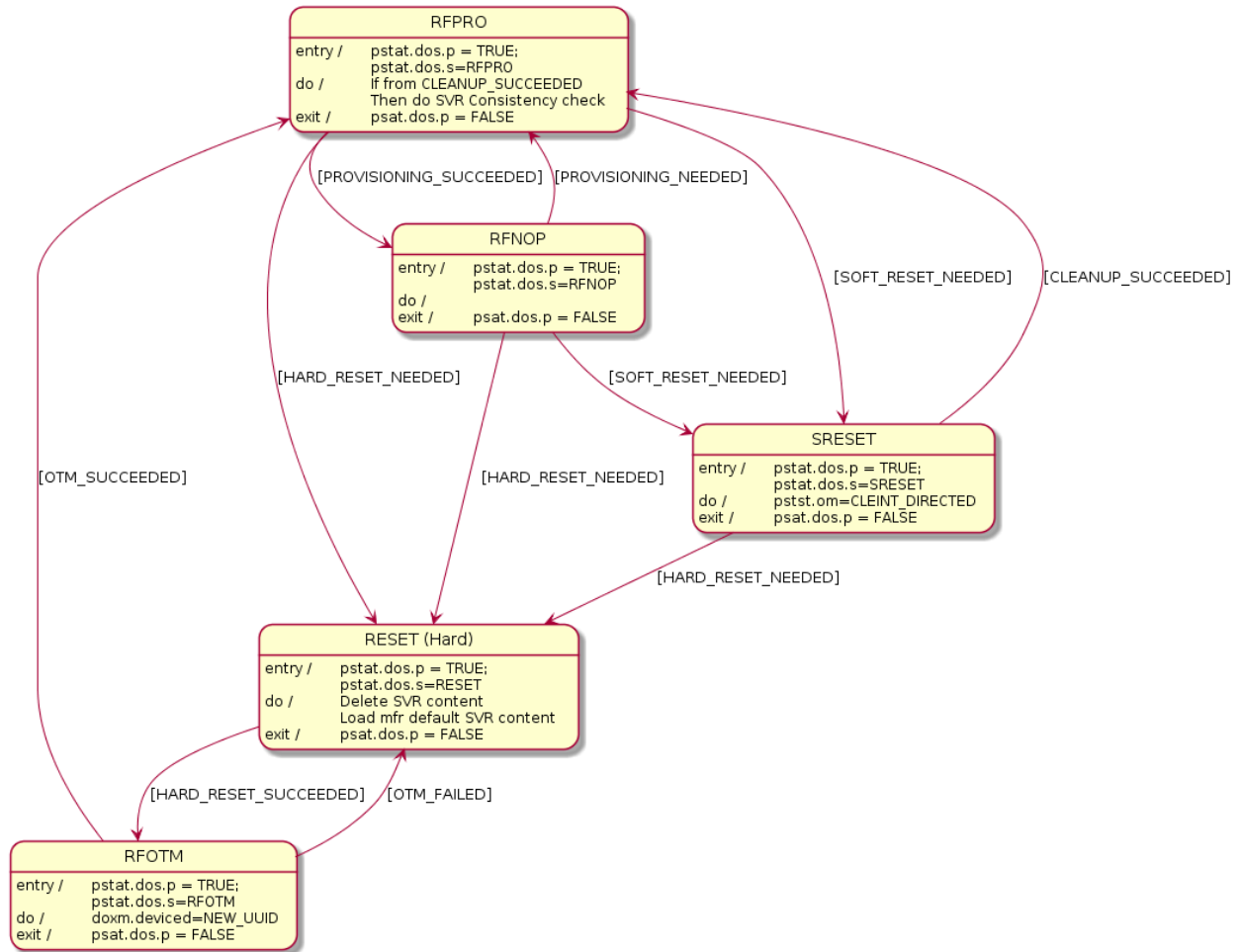
### 1967 **8.1 Device Onboarding General**

1968 As explained in 5.3, the process of onboarding completes after the ownership of the Device has  
 1969 been transferred and the Device has been provisioned with relevant configuration/services as



1970 explained in 5.4. The Figure 27 shows the various states a Device can be in during the Device  
 1971 lifecycle.

1972 The "/pstat.dos.s" Property is RW by the "/oic/sec/pstat" resource owner (e.g. "doxs" service) so  
 1973 that the resource owner can remotely update the Device state. When the Device is in RFNOP or  
 1974 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the  
 1975 Device state is SRESET the Device OC may be the only indication of authorization to access the  
 1976 Device. The Device owner may perform low-level consistency checks and re-provisioning to get  
 1977 the Device suitable for a transition to RFPRO.



1978 **Figure 27 – Device state model**

1980 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the  
 1981 "Ready for Normal Operation" state where it has all it needs in order to start interoperating with  
 1982 other Devices. Clause 8.5 specifies the minimum mandatory configuration that a Device shall  
 1983 hold in order to be considered as "Ready for Normal Operation".

1984 In the event of power loss or Device failure, the Device should remain in the same state that it  
 1985 was in prior to the power loss / failure

1986 If a Device or resource owner OBSERVEs "/pstat.dos.s", then transitions to SRESET will give  
 1987 early warning notification of Devices that may require SVR consistency checking.

1988 In order for onboarding to function, the Device shall have the following Resources installed:

1989 1) "/oic/sec/doxm" Resource

1990 2) "/oic/sec/pstat" Resource

1991 3) "/oic/sec/cred" Resource

1992 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5  
1993 and 8.6.

## 1994 **8.2 Device Onboarding-Reset State Definition**

1995 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset  
1996 also defines a state where the Device asset is ready to be transferred to another party.

1997 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces  
1998 Platform reset. All Devices hosted on the same Platform transition their Device states to RESET  
1999 when the Platform reset is asserted.

2000 The following Resources and their specific properties shall have the value as specified:

2001 1) The "owned" Property of the "/oic/sec/doxm" Resource shall transition to FALSE.

2002 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.

2003 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2004 implemented.

2005 4) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2006 default value.

2007 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2008 default value, if this Property is implemented.

2009 6) The "sct" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2010 default value.

2011 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2012 default value.

2013 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.

2014 9) The "dos" Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal  
2015 "RESET" state and dos.p shall equal "FALSE".

2016 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the  
2017 manufacturer default value.

2018 11) The "sm" (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set  
2019 to the manufacturer default value.

2020 12) The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and  
2021 "/oic/sec/cred" Resources shall be nil UUID.

2022 13) The "supportedprofiles" Property of the "/oic/sec/sp" Resource shall be set to the  
2023 manufacturer default value.

2024 14) The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer  
2025 default value.

## 2026 **8.3 Device Ready-for-OTM State Definition**

2027 The following Resources and their specific properties shall have the value as specified when the  
2028 Device enters ready for ownership transfer:

2029 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to  
2030 TRUE.

- 2031 2) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2032 implemented.
- 2033 3) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2034 4) The "deviceid" Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is  
2035 implemented. The value of the di Property in "/oic/d" is undefined.
- 2036 5) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2037 default value.
- 2038 6) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2039 7) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFOTM"  
2040 state and dos.p shall equal "FALSE".
- 2041 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

2042 **8.4 Device Ready-for-Provisioning State Definition**

2043 The following Resources and their specific properties shall have the value as specified when the  
2044 Device enters ready for provisioning:

- 2045 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2046 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2047 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
2048 set to the value that was determined during RFOTM processing. Also the value of the "di"  
2049 Property in "/oic/d" Resource shall be the same as the "deviceid" Property in the  
2050 "/oic/sec/doxm" Resource.
- 2051 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual  
2052 OTM used during ownership transfer.
- 2053 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2054 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFPRO"  
2055 state and "dos.p" shall equal "FALSE".
- 2056 7) The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner  
2057 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2058 "rowneruuid" may result in an orphan Resource.
- 2059 8) The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by  
2060 "rowneruuid" and "devowneruuid" Properties.

2061 **8.5 Device Ready-for-Normal-Operation State Definition**

2062 The following Resources and their specific properties shall have the value as specified when the  
2063 Device enters ready for normal operation:

- 2064 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2065 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2066 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
2067 set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d"  
2068 shall be the same as the deviceuuid.
- 2069 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual  
2070 OTM used during ownership transfer.
- 2071 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once  
2072 transition to RFNOP is otherwise complete.
- 2073 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFNOP"  
2074 state and dos.p shall equal "FALSE".

2075 7) The "rowneruuid" Property of every installed Resource shall be set to a valid resource owner  
2076 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2077 "rowneruuid" results in an orphan Resource.

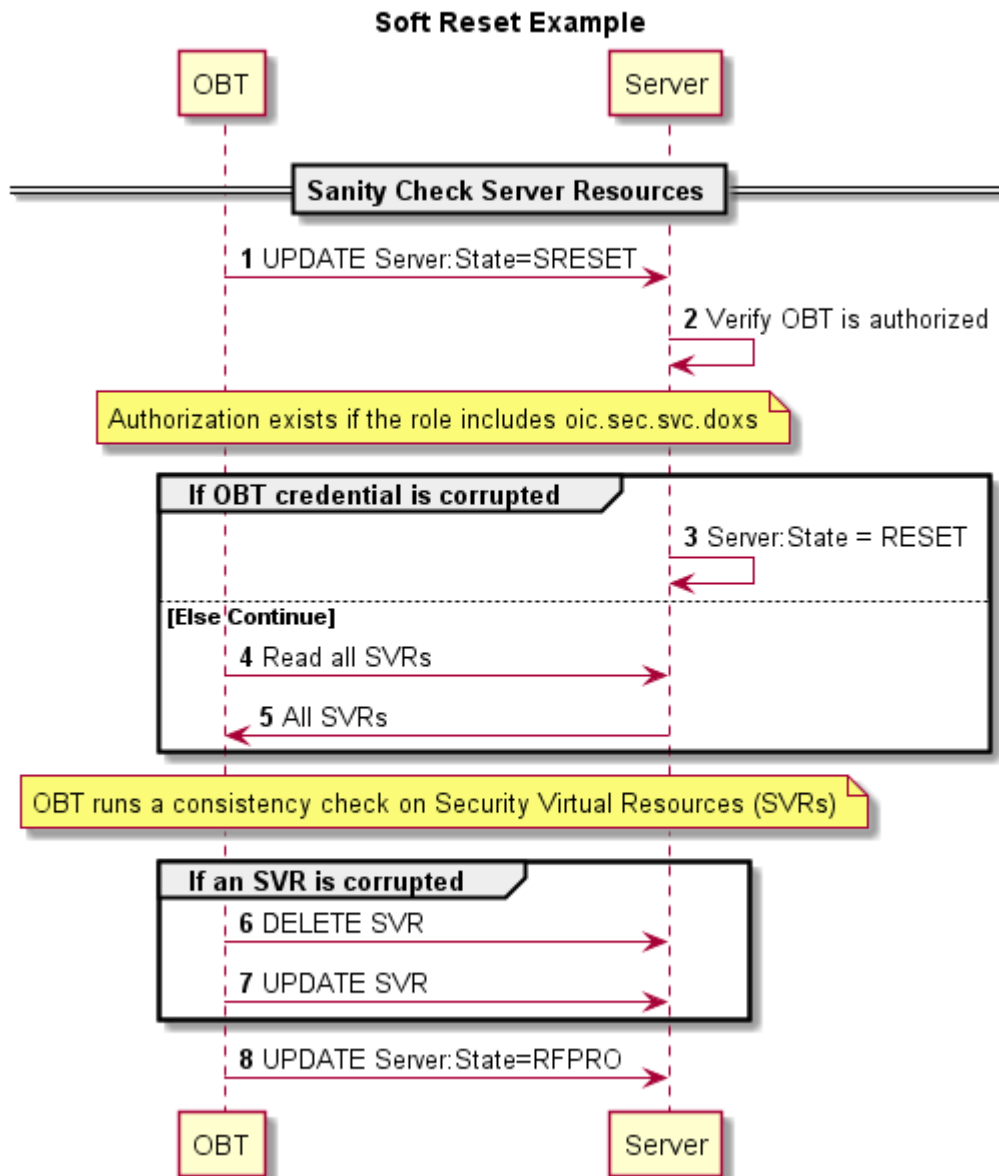
2078 8) The "/oic/sec/cred" Resource shall contain credentials for each service referenced by  
2079 "rowneruuid" and "devowneruuid" Properties.

## 2080 **8.6 Device Soft Reset State Definition**

2081 The soft reset state is defined (e.g. "/pstat.dos.s" = SRESET) where entrance into this state  
2082 means the Device is not operational but remains owned by the current owner. The Device may  
2083 exit SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during  
2084 original onboarding (but should not require use of an OTM /doxm.oxms).

2085 The DOTS should perform a consistency check of the SVR and if necessary, re-provision them  
2086 sufficiently to allow the Device to transition to RFPRO.

2087 Figure 28 depicts OBT Sanity Check Sequence in SRESET.



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

2088  
 2089  
 2090 The DOTS should perform a sanity check of SVRs before final transition to RFPRO Device state.  
 2091 If the DOTS credential cannot be found or is determined to be corrupted, the Device state  
 2092 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to  
 2093 validate the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS  
 2094 Devices.

2095 When in SRESET, the following Resources and their specific Properties shall have the values as  
 2096 specified.

- 2097 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.  
 2098 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.  
 2099 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be non-null, if this Property is  
 2100 implemented.

- 2101 4) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2102 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2103 6) The "sct" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 2104 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 2105 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2106 9) The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- 2107 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-
- 2108 directed mode".
- 2109 11) The "sm" (supported operational modes) Property of "/oic/sec/pstat" Resource may be
- 2110 updated by the Device owner (aka DOTS).
- 2111 12) The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2",
- 2112 "/oic/sec/amacl", "/oic/sec/sacl", and "/oic/sec/cred" Resources may be reset by the Device
- 2113 owner (aka DOTS) and re-provisioned.
- 2114

## 2115 **9 Security Credential Management**

### 2116 **9.1 Preamble**

2117 This clause provides an overview of the credential types in OCF, along with details of credential  
2118 use, provisioning and ongoing management.

### 2119 **9.2 Credential Lifecycle**

#### 2120 **9.2.1 Credential Lifecycle General**

2121 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)  
2122 issuance and (5) revocation.

#### 2123 **9.2.2 Creation**

2124 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is  
2125 authorized by matching the rowneruuid Property of the "/oic/sec/cred" resource to the DeviceID of  
2126 the credential the CMS used to establish the secure connection.

2127 Credential Resources created using a CMS may involve specialized credential issuance protocols  
2128 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate  
2129 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of  
2130 a provisioning action by a DOTS, CMS or AMS.

#### 2131 **9.2.3 Deletion**

2132 The CMS should delete known compromised credential Resources. The Device (e.g. the Device  
2133 where the credential Resource is hosted) should delete credential Resources that have expired.

2134 An expired credential Resource may be deleted to manage memory and storage space.

2135 Deletion in OCF key management is equivalent to credential suspension.

#### 2136 **9.2.4 Refresh**

2137 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2138 The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may  
2139 be applied when a credential is about to expire or is about to exceed a maximum threshold for  
2140 bytes encrypted.

2141 A credential refresh method specifies the options available when performing key refresh. The  
2142 Period Property informs when the credential should expire. The Device may proactively obtain a  
2143 new credential using a credential refresh method using current unexpired credentials to refresh  
2144 the existing credential. If the Device does not have an internal time source, the current time  
2145 should be obtained from a CMS at regular intervals.

2146 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the  
2147 CMS credentials to the Device. If the onboarding established credentials are allowed to expire  
2148 the DOTS shall re-onboard the Device to re-apply device owner transfer steps.

2149 All Devices shall support at least one credential refresh method.

#### 2150 **9.2.5 Revocation**

2151 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where  
2152 the revocation method involves provisioning of a revocation object that identifies a credential that  
2153 is to be revoked prior to its normal expiration period, a credential Resource is created containing  
2154 the revocation information that supersedes the originally issued credential. The revocation object

2155 expiration should match that of the revoked credential so that the revocation object is cleaned up  
2156 upon expiry.

2157 It is conceptually reasonable to consider revocation applying to a credential or to a Device.  
2158 Device revocation asserts all credentials associated with the revoked Device should be  
2159 considered for revocation. Device revocation is necessary when a Device is lost, stolen or  
2160 compromised. Deletion of credentials on a revoked Device might not be possible or reliable.

## 2161 **9.3 Credential Types**

### 2162 **9.3.1 Preamble**

2163 The "/oic/sec/cred" Resource maintains a credential type Property that supports several  
2164 cryptographic keys and other information used for authentication and data protection. The  
2165 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric  
2166 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.  
2167 PIN/password).

### 2168 **9.3.2 Pair-wise Symmetric Key Credentials**

2169 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The  
2170 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2171 Pair-wise keys could be established through ad-hoc key agreement protocols.

2172 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2173 The PublicData Property may contain a token encrypted to the peer Device containing the pair-  
2174 wise key.

2175 The OptionalData Property may contain revocation status.

2176 The Device implementer should apply hardened key storage techniques that ensure the  
2177 PrivateData remains private.

2178 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2179 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2180 unauthorized modifications.

### 2181 **9.3.3 Group Symmetric Key Credentials**

2182 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are  
2183 used for efficient sharing of data among group participants.

2184 Group keys do not provide authentication of Devices but only establish membership in a group.

2185 The CMS shall provision group symmetric key credentials to the group members. The CMS  
2186 maintains the group memberships.

2187 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2188 The PublicData Property may contain the group name.

2189 The OptionalData Property may contain revocation status.

2190 The Device implementer should apply hardened key storage techniques that ensure the  
2191 PrivateData remains private.



2192 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2193 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2194 unauthorized modifications.

### 2195 **9.3.4 Asymmetric Authentication Key Credentials**

#### 2196 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2197 Asymmetric authentication key credentials contain either a public and private key pair or only a  
2198 public key. The private key is used to sign Device authentication challenges. The public key is  
2199 used to verify a device authentication challenge-response.

2200 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2201 The PublicData Property contains the public key.

2202 The OptionalData Property may contain revocation status.

2203 The Device implementer should apply hardened key storage techniques that ensure the  
2204 PrivateData remains private.

2205 Devices should generate asymmetric authentication key pairs internally to ensure the private key  
2206 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material  
2207 between Devices.

2208 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2209 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2210 unauthorized modifications.

#### 2211 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2212 Devices should employ industry-standard high-assurance techniques when allowing off-device  
2213 key pair creation and provisioning. Use of such key pairs should be minimized, particularly if the  
2214 key pair is immutable and cannot be changed or replaced after provisioning.

2215 When used as part of onboarding, these key pairs can be used to prove the Device possesses  
2216 the manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept  
2217 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the  
2218 Device, and then provisions new OCF Security Domain credentials for use.

### 2219 **9.3.5 Asymmetric Key Encryption Key Credentials**

2220 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when  
2221 distributing or storing the key.

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

2223 The PublicData Property contains the public key.

2224 The OptionalData Property may contain revocation status.

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

2227 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2228 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2229 unauthorized modifications.

### 2230 **9.3.6 Certificate Credentials**

2231 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a  
2232 CMS or an external certificate authority (CA).

2233 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2234 The issued certificate is stored with the asymmetric key credential Resource.

2235 Other objects useful in managing certificate lifecycle such as certificate revocation status are  
2236 associated with the credential Resource.

2237 Either an asymmetric key credential Resource or a self-signed certificate credential is used to  
2238 terminate a path validation.

2239 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2240 The PublicData Property contains the issued certificate.

2241 The OptionalData Property may contain revocation status.

2242 The Device implementer should apply hardened key storage techniques that ensure the  
2243 PrivateData remains private.

2244 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2245 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2246 unauthorized modifications.

### 2247 **9.3.7 Password Credentials**

2248 Shared secret credentials are used to maintain a PIN or password that authorizes Device access  
2249 to a foreign system or Device that doesn't support any other OCF credential types.

2250 The PrivateData Property in the "/oic/sec/cred" Resource contains the PIN, password and other  
2251 values useful for changing and verifying the password.

2252 The PublicData Property may contain the user or account name if applicable.

2253 The OptionalData Property may contain revocation status.

2254 The Device implementer should apply hardened key storage techniques that ensure the  
2255 PrivateData remains private.

2256 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2257 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2258 unauthorized modifications.

## 2259 **9.4 Certificate Based Key Management**

### 2260 **9.4.1 Overview**

2261 To achieve authentication and transport security during communications in OCF Security Domain,  
2262 certificates containing public keys of communicating parties and private keys can be used.

2263 The certificate and private key may be issued by a local or remote certificate authority (CA). For  
2264 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of  
2265 identity. In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to  
2266 validate proof of identity and validity.

2267 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,  
 2268 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in  
 2269 X.509 are not supported so that the format intends to meet the constrained Device's requirement.

2270 As for the certificate and CRL management in the Server, the process of storing, retrieving and  
 2271 parsing Resources of the certificates and CRL will be performed at the security resource  
 2272 manager layer; the relevant interfaces may be exposed to the upper layer.

2273 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of  
 2274 certificates and CRL will be stored and managed in SVR database.

2275 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a  
 2276 CMS to a Device when it is newly onboarded. The issuing CMS should process certificate  
 2277 revocations for certificates it issues. If a certificate private key is compromised, the CMS should  
 2278 revoke the certificate. If CRLs are used by a Device, the CMS should regularly (for example;  
 2279 every 3 months) update the "/oic/sec/crl" resource for the Devices it manages.

2280 **9.4.2 X.509 Digital Certificate Profiles**

2281 **9.4.2.1 Digital Certificate Profile General**

2282 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in  
 2283 IETF RFC 5280.

2284 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications  
 2285 for those communities wishing to make use of X.509 technology. The X.509 v3 certificate format  
 2286 is described in detail, with additional information regarding the format and semantics of OCF  
 2287 specific extension(s). The supported standard certificate extensions are also listed.

2288 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this  
 2289 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are  
 2290 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,  
 2291 compliant entities shall ignore their contents.

2292 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as  
 2293 defined in ISO/IEC 8825-1 to encode certificates.

2294 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public  
 2295 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs  
 2296 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and  
 2297 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2298 The following clauses specify the supported standard and custom extensions for the OCF  
 2299 certificates profile.

2300 **9.4.2.2 Certificate Profile and Fields**

2301 **9.4.2.2.1 Root CA Certificate Profile**

2302 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2303 **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)

2304 Table 17 describes X.509 v3 extensions required for Root CA Certificates.

2305 **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)

2306 **9.4.2.2.2 Intermediate CA Certificate Profile**

2307 Table 18 describes X.509 v1 fields required for Intermediate CA Certificates.

2308 **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)

2309 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

2311 **9.4.2.2.3 End-Entity Black Certificate Profile**

2312 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security  
2313 profile.

2314 **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)

2315 Table 21 describes X.509 v3 extensions required for End-Entity Certificates.

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

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

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

2317 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2318 The OCF Compliance Extension defines required parameters to correctly identify the type of  
2319 Device, its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2320 The extension carries an "ocfVersion" field which provides the specific base version of the OCF  
2321 documents the device implements. The "ocfVersion" field shall contain a sequence of three  
2322 integers ("major", "minor", and "build"). For example, if an entity is certified to be compliant with

2323 OCF specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be  
2324 set to "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote  
2325 compliance to a specified base version of the OCF documents.

2326 The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more  
2327 supported Security Profiles associated with the certificate in string form (UTF-8). All Security  
2328 Profiles associated with the certificate should be identified by this field.

2329 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".  
2330 The fields carry human-readable descriptions of the Device's name and manufacturer,  
2331 respectively.

2332 The ASN.1 definition of the OCFCCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined  
2333 as follows:

```
2334 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2335                               private(4) enterprise(1) OCF(51414) }
2336
2337 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2338
2339 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2340
2341 ocfVersion ::= SEQUENCE {
2342     major INTEGER,
2343     --Major version number
2344     minor INTEGER,
2345     --Minor version number
2346     build INTEGER,
2347     --Build/Micro version number
2348 }
2349
2350 ocfCompliance ::= SEQUENCE {
2351     version ocfVersion,
2352     --Device/OCF version
2353     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2354     --Sequence of OCF Security Profile OID strings
2355     --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2356     deviceName UTF8String,
2357     --Name of the device
2358     deviceManufacturer UTF8String,
2359     --Human-Readable Manufacturer
2360     --of the device
2361 }
```

#### 2362 **9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension**

2363 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for  
2364 devices to signal to the network the access and network functionality they require to properly  
2365 function. Access controls can be more easily achieved and deployed at scale when the MUD  
2366 extension is used. The current draft of the MUD v3 extension at this time of writing is:

2367 <https://tools.ietf.org/html/rfc8520#section-11>

2368 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2369 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2370                       internet(1) security(5) mechanisms(5) pkix(7)
2371                       id-mod(0) id-mod-mudURLExtn2016(88) }
2372
2373 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2374 -- EXPORTS ALL --
2375 IMPORTS
```



```

2376     EXTENSION
2377     FROM PKIX-CommonTypes-2009
2378         { iso(1) identified-organization(3) dod(6) internet(1)
2379           security(5) mechanisms(5) pkix(7) id-mod(0)
2380           id-mod-pkixCommon-02(57) }
2381     id-pe
2382     FROM PKIX1Explicit-2009
2383         { iso(1) identified-organization(3) dod(6) internet(1)
2384           security(5) mechanisms(5) pkix(7) id-mod(0)
2385           id-mod-pkix1-explicit-02(51) } ;
2386     MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2387     ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2388                               IDENTIFIED BY id-pe-mud-url }
2389
2390     id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2391
2392     MUDURLSyntax ::= IA5String
2393
2394     END

```

#### 2395 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2396 The OCF Security Claims Extension defines a list of OIDs representing security claims that the  
2397 manufacturer/integrator is making as to the security posture of the device above those required  
2398 by the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2399 The purpose of this extension is to allow for programmatic evaluation of assertions made about  
2400 security to enable some platforms/policies/administrators to better understand what is being  
2401 onboarded or challenged.

2402 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is  
2403 defined as follows:

```

2404 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2405                               private(4) enterprise(1) OCF(51414) }
2406
2407 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2408
2409 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2410
2411     claim-secure-boot           ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2412     --Device claims that the boot process follows a procedure trusted
2413     --by the firmware and the BIOS
2414
2415     claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2416     --Device claims that credentials are stored in a specialized hardware
2417     --protection environment such as a Trusted Platform Module (TPM) or
2418     --similar mechanism.
2419
2420     ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2421
2422     ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

#### 2423 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2424 The OCF Certified Product List Extension defines required parameters to utilize the OCF  
2425 Compliance Management System Certified Product List (OCMS-CPL). This clause is only  
2426 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to  
2427 verify the compliance level of a device.

2428 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model  
2429 and Version.

2430 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique  
2431 PEN established in the IANA PEN list located at: [https://www.iana.org/enterprise-](https://www.iana.org/assignments/enterprise-)  
2432 numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as  
2433 reported during OCF Certification.

2434 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field  
2435 found in end-products shall be the same information as reported during OCF Certification.

2436 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found  
2437 in end-products shall be the same information as reported during OCF Certification.

2438 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is  
2439 defined as follows:

```
2440 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2441                               private(4) enterprise(1) OCF(51414) }
2442
2443 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2444
2445     id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2446
2447         cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2448         cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2449         cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2450
2451
2452     ocfCPLAttributes ::= SEQUENCE {
2453         cpl-at-IANAPen      UTF8String,
2454                             --Manufacturer's registered IANA Private Enterprise Number
2455         cpl-at-model      UTF8String,
2456                             --Device OCF Security Profile
2457         cpl-at-version    UTF8String
2458                             --Name of the device
2459     }
```

### 2460 9.4.2.3 Supported Certificate Extensions

2461 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the  
2462 clause number from that RFC to include it by reference. Each extension is summarized here, and  
2463 any modifications to the RFC definition are listed. Devices MUST implement and understand the  
2464 extensions listed here; other extensions from the RFC are not included in this document and  
2465 therefore are not required. 10.4 describes what Devices must implement when validating  
2466 certificate chains, including processing of extensions, and actions to take when certain  
2467 extensions are absent.

#### 2468 – Authority Key Identifier (4.2.1.1)

2469 The Authority Key Identifier (AKI) extension provides a means of identifying the public key  
2470 corresponding to the private key used to sign a certificate. This document makes the following  
2471 modifications to the referenced definition of this extension:

2472 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier  
2473 sequence are not permitted; only keyIdentifier is allowed. This results in the following  
2474 grammar definition:

```
2475 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2476
2477 AuthorityKeyIdentifier ::= SEQUENCE {
2478     keyIdentifier          [0] KeyIdentifier
2479 }
2480 KeyIdentifier ::= OCTET STRING
```

#### 2481 – Subject Key Identifier (4.2.1.2)

2482 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that  
2483 contain a particular public key.

2484 This document makes the following modification to the referenced definition of this extension:

2485 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's  
2486 SubjectPublicKeyInfo field or a method that generates unique values. This document  
2487 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey  
2488 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains  
2489 must not assume any particular method of computing key identifiers, however, and must only  
2490 base matching AKI's and SKI's in certification path constructions on key identifiers seen in  
2491 certificates.

2492 – Subject Alternative Name

2493 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role  
2494 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and  
2495 interpreted as described below. When no EKU is present, or has another value, the  
2496 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to  
2497 encode one or more Role ID values in role certificates, binding the roles to the subject public  
2498 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2499 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2500
2501 SubjectAltName ::= GeneralNames
2502
2503 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2504
2505 GeneralName ::= CHOICE {
2506     otherName                [0]     OtherName,
2507     rfc5322Name              [1]     IA5String,
2508     dNSName                  [2]     IA5String,
2509     x400Address              [3]     ORAddress,
2510     directoryName            [4]     Name,
2511     ediPartyName             [5]     EDIPartyName,
2512     uniformResourceIdentifier [6]     IA5String,
2513     iPAddress                [7]     OCTET STRING,
2514     registeredID             [8]     OBJECT IDENTIFIER }
2515
2516     EDIPartyName ::= SEQUENCE {
2517         nameAssigner          [0]     DirectoryString OPTIONAL,
2518         partyName             [1]     DirectoryString }
```

2519  
2520 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a  
2521 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name  
2522 shall contain exactly one CN (Common Name) component, and zero or one OU  
2523 (Organizational Unit) components. The OU component, if present, shall specify the authority  
2524 that defined the semantics of the role. If the OU component is absent, the certificate issuer  
2525 has defined the role. The CN component shall encode the role ID. Other GeneralName types  
2526 in the SEQUENCE may be present, but shall not be interpreted as roles. Therefore, if the  
2527 certificate issuer includes non-role names in the subjectAltName extension, the extension  
2528 should not be marked critical.

2529 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted  
2530 character set [0-9a-z-A-z '()+,.-/:=?].

2531 – Key Usage (4.2.1.3)

2532 The key usage extension defines the purpose (e.g., encipherment, signature, certificate  
2533 signing) of the key contained in the certificate. The usage restriction might be employed when  
2534 a key that could be used for more than one operation is to be restricted.

2535 This document does not modify the referenced definition of this extension.

2536 – Basic Constraints (4.2.1.9)

2537 The basic constraints extension identifies whether the subject of the certificate is a CA and  
2538 the maximum depth of valid certification paths that include this certificate. Without this  
2539 extension, a certificate cannot be an issuer of other certificates.

2540 This document does not modify the referenced definition of this extension.

2541 – Extended Key Usage (4.2.1.12)

2542

2543 Extended Key Usage describes allowed purposes for which the certified public key may can  
2544 be used. When a Device receives a certificate, it determines the purpose based on the  
2545 context of the interaction in which the certificate is presented, and verifies the certificate can  
2546 be used for that purpose.

2547 This document makes the following modifications to the referenced definition of this extension:  
2548 CAs SHOULD mark this extension as critical.

2549 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).  
2550

2551 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2552 – Identity certificate 1.3.6.1.4.1.44924.1.6  
2553 – Role certificate 1.3.6.1.4.1.44924.1.7

2554 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2555 See 9.4.3.5 for details.

2556 **9.4.2.5 Encoding of Certificate**

2557 See 9.4.2 for details.

2558 **9.4.3 Certificate Revocation List (CRL) Profile**

2559 **9.4.3.1 CRL General**

2560 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use  
2561 within OCF applications for those communities wishing to support revocation features in their  
2562 PKIs.

2563 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in  
2564 IETF RFC 5280 – Clause 5.1

2565 **9.4.3.2 CRL Profile and Fields**

2566 This clause intentionally left empty.

2567 **9.4.3.3 Encoding of CRL**

2568 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]  
2569 should be used to encode CRL.

2570 **9.4.3.4 CRLs Supported Standard Extensions**

2571 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55]  
2572 provide methods for associating additional attributes with CRLs. The following list of X.509  
2573 extensions should be supported in this certificate profile:

2574 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides  
2575 a means of identifying the public key corresponding to the private key used to sign a CRL.  
2576 Conforming CRL issuers should use the key identifier method, and shall include this extension  
2577 in all CRLs issued

2578 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that  
2579 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer  
2580 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for  
2581 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]  
2582 [X9.55]. Although this document does not provide any recommendation about the use of specific  
2583 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not  
2584 marked critical.

#### 2585 **9.4.3.5 Encryption Ciphers and TLS support**

2586 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support  
2587 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 cipher suite as defined in IETF RFC 7251 and  
2588 may support additional ciphers as defined in the TLS v1.2 specifications.

#### 2589 **9.4.4 Resource Model**

2590 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated  
2591 with a separate crl Resource that is defined for maintaining the revocation list.

2592 The cred Resource contains the certificate information pertaining to the Device. The publicData  
2593 Property holds the device certificate and CA certificate chain. PrivateData Property holds the  
2594 Device private key paired to the certificate. (See 13.3 for additional detail regarding the  
2595 "/oic/sec/cred" Resource).

2596 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained  
2597 through the CMS. The Device must consider revoked certificates as part of certificate path  
2598 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain  
2599 a full list, the Device must query the CMS for current revocation status. (See 13.4 for additional  
2600 detail regarding the "/oic/sec/crl" Resource).

#### 2601 **9.4.5 Certificate Provisioning**

2602 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have  
2603 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)  
2604 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate  
2605 shall have the format described in 9.4.2.

2606 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,  
2607 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer  
2608 them to the Device including its CA certificate chain. Optionally, the CMS may also transfer one  
2609 or more role certificates, which shall have the format described in clause 9.4.2. . The  
2610 subjectPublicKey of each role certificate shall match the subjectPublicKey in the Device  
2611 certificate.

2612 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in  
2613 IETF RFC 2986, and is included here by reference.

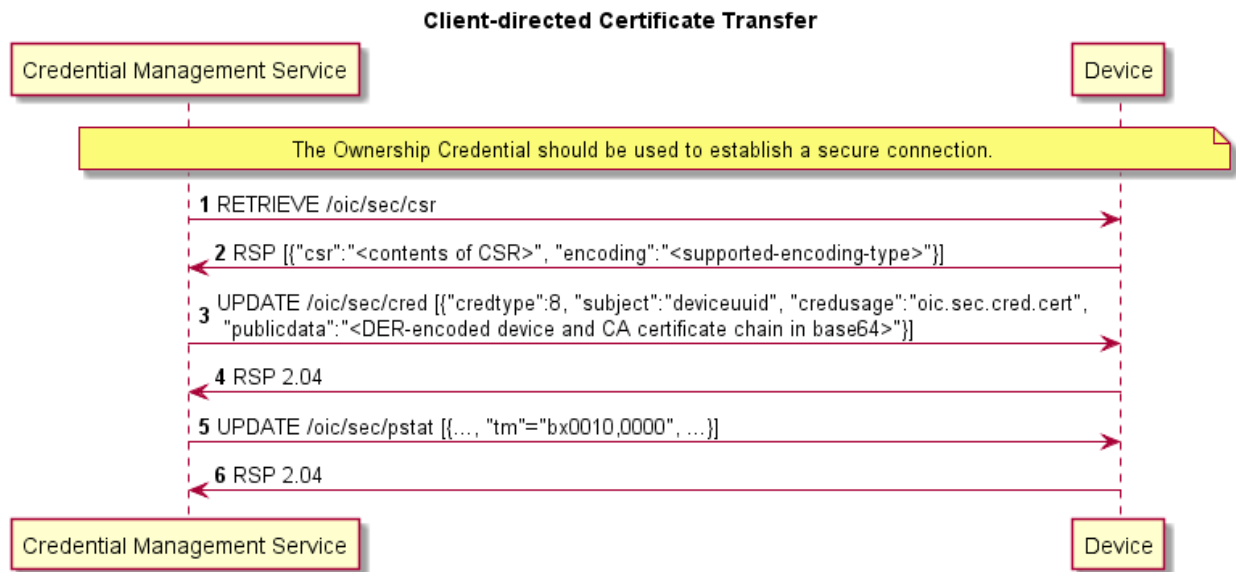
2614 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2615 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device  
2616 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.  
2617 The Device determines the public key to present; this may be an already-provisioned key it  
2618 has selected for use with authentication, or if none is present, it may generate a new key pair  
2619 internally and provide the public part. The key pair shall be compatible with the allowed  
2620 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF  
2621 authentication.

2622 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on  
2623 its own, then it is not capable of using certificates. The Device shall advertise this fact both by

2624 setting the 0x8 bit position in the sct Property of "/oic/sec/doxm" to 0, and return an error that  
2625 the "/oic/sec/csr" resource does not exist.

2626 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using  
2627 the same credid, to maintain the association with the private key. The credential type  
2628 ("oic.sec.cred") used to transfer certificates in Figure 29 is also used to transfer role  
2629 certificates, by including multiple credentials in the POST from CMS to Device. Identity  
2630 certificates shall be stored with the credusage Property set to "oic.sec.cred.cert" and role  
2631 certificates shall be stored with the credusage Property set to "oic.sec.cred.rolecert".



2632

2633 **Figure 29 – Client-directed Certificate Transfer**

#### 2634 9.4.6 CRL Provisioning

2635 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the  
2636 function to register revocation certificates, to sign CRL and to transfer it to Devices.

2637 The CMS sends the CRL to the Device.

2638 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2639 – change of issuer name
- 2640 – change of association between Devices and CA
- 2641 – certificate compromise
- 2642 – suspected compromise of the corresponding private key

2643 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In  
2644 some special cases, Devices may request CRL to a given CMS.

2645 There are two options to update and deliver CRL;

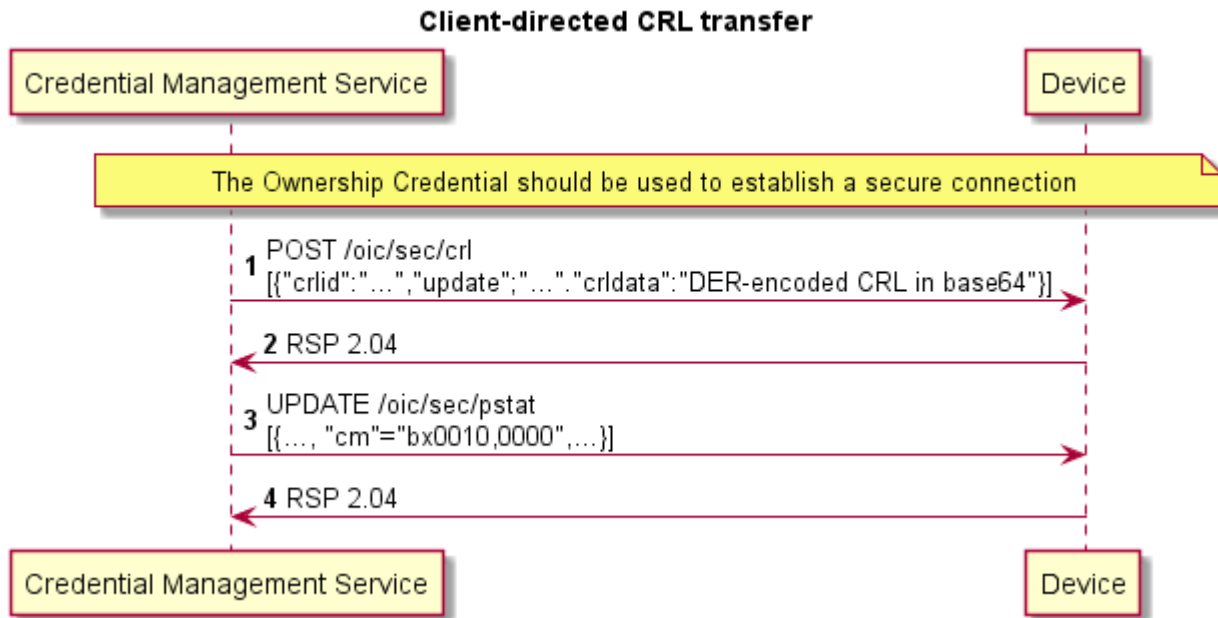
- 2646 – CMS pushes CRL to each Device
- 2647 – each Device periodically requests to update CRL

2648 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

2649 1) The CMS may retrieve the CRL Resource Property.

2650 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.

2651  
2652



2653

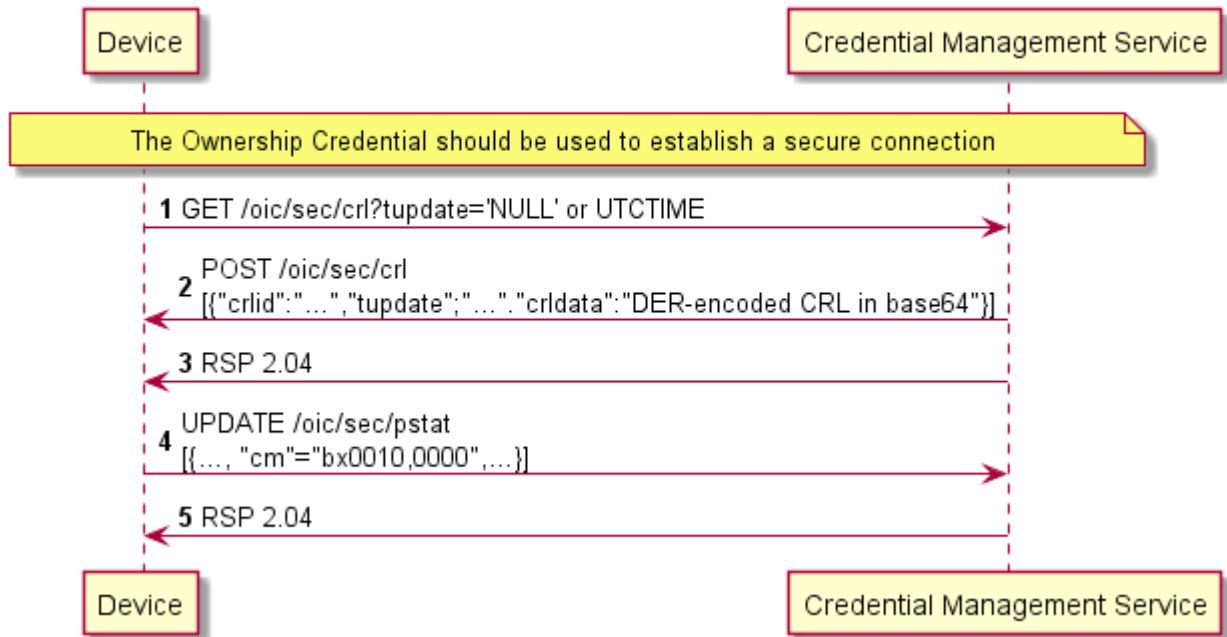
**Figure 30 – Client-directed CRL Transfer**

2654 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.

2655 1) The Device retrieves the CRL Resource Property "tupdate" to the CMS.

2656 2) If the CMS recognizes the updated CRL information after the designated "tupdate" time, it  
2657 may transfer its CRL to the Device.

### Server-directed CRL transfer



2658  
2659  
2660

Figure 31 – Server-directed CRL Transfer



## 2661 **10 Device Authentication**

### 2662 **10.1 Device Authentication General**

2663 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the  
2664 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or  
2665 more roles that the server can use in access control decisions. Roles may be asserted when the  
2666 Device authentication is done with certificates.

### 2667 **10.2 Device Authentication with Symmetric Key Credentials**

2668 When using symmetric keys to authenticate, the Server Device shall include the  
2669 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client  
2670 shall validate that it has a credential with the Subject ID set to the Server's Device ID, and a  
2671 credential type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error  
2672 or other suitable error.

2673 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that  
2674 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a  
2675 credential with the matching Subject ID and type. If it does not, the Server shall respond with an  
2676 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS  
2677 protocol, and both Client and Server shall compute the resulting premaster secret.

### 2678 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2679 When using raw asymmetric keys to authenticate, the Client and the Server shall include a  
2680 suitable public key from a credential that is bound to their Device. Each Device shall verify that  
2681 the provided public key matches the `PublicData` field of a credential they have, and use the  
2682 corresponding Subject ID of the credential to identify the peer Device.

### 2683 **10.4 Device Authentication with Certificates**

#### 2684 **10.4.1 Device Authentication with Certificates General**

2685 When using certificates to authenticate, the Client and Server shall each include their certificate  
2686 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.  
2687 Each Device shall validate the certificate chain presented by the peer Device. Each certificate  
2688 signature shall be verified until a public key is found within the `"/oic/sec/cred"` Resource with the  
2689 `"oic.sec.cred.trustca"` credusage. Credential Resource found in `"/oic/sec/cred"` is used to  
2690 terminate certificate path validation. Also, the validity period and revocation status should be  
2691 checked for all above certificates, but at this time a failure to obtain a certificate's revocation  
2692 status (CRL or OCSP response) MAY continue to allow the use of the certificate if all other  
2693 verification checks succeed.

2694 If available, revocation information should be used to verify the revocation status of the certificate.  
2695 The URL referencing the revocation information should be retrieved from the certificate (via the  
2696 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used  
2697 to gather relevant revocation information like CRLs or OCSP responses.

2698 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2699 Devices must follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2700 particular:

- 2701 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is  
2702 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate  
2703 chain **MUST** be rejected. If the `pathLenConstraint` field is present, Devices will confirm the  
2704 number of certificates between this certificate and the End-Entity certificate is less than or  
2705 equal to `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity

- 2706 certificate can be issued by this certificate. If the pathLenConstraint field is absent, there is no  
2707 limit to the chain length.
- 2708 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is  
2709 present, and that the keyCertSign bit is asserted.
- 2710 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.  
2711 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt  
2712 validation with the public keys of possible issuer certificates whose subject name equals the  
2713 issuer name of this certificate.
- 2714 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage  
2715 (EKU) suitable to the purpose for which it is being presented. An End-Entity certificate which  
2716 contains no ECU extension is not valid for any purpose and must be rejected. Any certificate  
2717 which contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other  
2718 valid EKUs are also present.
- 2719 – Devices MUST verify "transitive ECU" for certificate chains. Issuer certificates (any certificate  
2720 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the  
2721 certificate chain is being presented. An issuer certificate is valid for a purpose if it contains an  
2722 ECU extension and the ECU OID for that purpose is listed in the extension, OR it does not  
2723 have an ECU extension. An issuer certificate SHOULD contain an ECU extension and a  
2724 complete list of EKUs for the purposes for which it is authorized to issue certificates. An  
2725 issuer certificate without an ECU extension is valid for all purposes; this differs from End-  
2726 Entity certificates without an ECU extension.
- 2727 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2728 If the Device does not recognize an extension, it must examine the `critical` field. If the field is  
2729 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the  
2730 certificate as if the extension were absent and proceed accordingly. This applies to all certificates  
2731 in a chain.

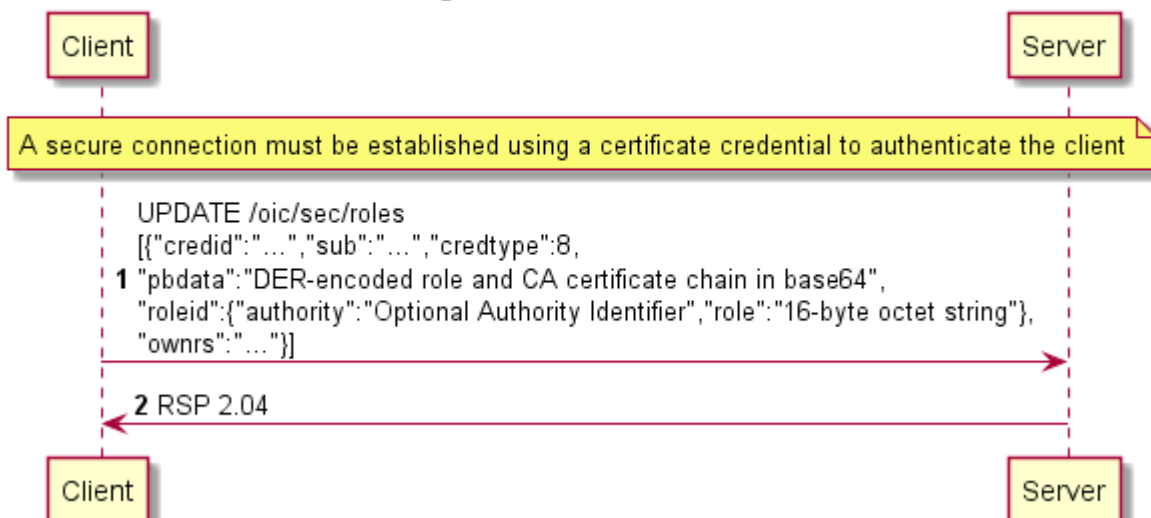
2732 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

#### 2733 **10.4.2 Role Assertion with Certificates**

2734 This clause describes role assertion by a client to a server using a certificate role credential. If a  
2735 server does not support the certificate credential type, clients should not attempt to assert roles  
2736 with certificates.

2737 Following authentication with a certificate, a client may assert one or more roles by updating the  
2738 server's roles resource with the role certificates it wants to use. The role credentials must be  
2739 certificate credentials and shall include a certificate chain. The server shall validate each  
2740 certificate chain as specified in clause 10.3. Additionally, the public key in the End-Entity  
2741 certificate used for Device authentication must be identical to the public key in all role (End-Entity)  
2742 certificates. Also, the subject distinguished name in the End-Entity authentication and role  
2743 certificates must match. The roles asserted are encoded in the subjectAltName extension in the  
2744 certificate. The subjectAltName field can have multiple values, allowing a single certificate to  
2745 encode multiple roles that apply to the client. The server shall also check that the ECU extension  
2746 of the role certificate(s) contains the value 1.3.6.1.4.1.44924.1.7 (see clause 9.4.2.2) indicating  
2747 the certificate may be used to assert roles. Figure 32 describes how a client Device asserts roles  
2748 to a server.

## Asserting Certificate Role Credentials



2749

2750

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

2751 Additional comments for Figure 32

- 2752 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If  
 2753 the server does not support certificate credentials, it should return "501 Not Implemented"
- 2754 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration  
 2755 shall not exceed the validity period of the role certificate. When fresh CRL information is  
 2756 obtained, the certificates in "/oic/sec/roles" should be checked, and the role removed if the  
 2757 certificate is revoked or expired.
- 2758 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role  
 2759 by a client. It is recommended that servers use the validity period of the certificate as a  
 2760 duration, effectively allowing the CMS to decide the duration.
- 2761 4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see  
 2762 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not  
 2763 be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the  
 2764 value in the certificate shall be used for validation. For example, if the Period field is set in  
 2765 the credential, the server shall treat the validity period in the certificate as authoritative.  
 2766 Similar for the roleid data (authority, role).
- 2767 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)
- 2768 6) Clients may GET the "/oic/sec/roles" resource to determine the roles that have been  
 2769 previously asserted. An array of credential objects shall be returned. If there are no valid  
 2770 certificates corresponding to the currently connected and authenticated Client's identity, then  
 2771 an empty array (i.e. []) shall be returned.

### 2772 10.4.3 OCF PKI Roots

2773 This clause intentionally left empty.

### 2774 10.4.4 PKI Trust Store

2775 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store  
 2776 the OCF Root CA certificates in the "oic/sec/cred" resource and SHOULD physically store this  
 2777 resource in a hardened memory location where the certificates cannot be tampered with.

2778 **10.4.5 Path Validation and extension processing**

2779 Devices SHALL follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2780 addition, the following are best practices and SHALL be adhered to by any OCF-compliant  
2781 application handling digital certificates

2782 – Validity Period checking

2783 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and  
2784 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,  
2785 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2786 – Revocation checking

2787 Relying applications SHOULD check the revocation status for all certificates, but at this time,  
2788 an application MAY continue to allow the use of the certificate upon a failure to obtain a  
2789 certificate's revocation status (CRL or OCSP response), if all other verification checks  
2790 succeed.

2791 – basicConstraints

2792 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify  
2793 that the basicConstraints extension is present, flagged critical, and that the cA boolean value  
2794 in the extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2795 If the pathLenConstraint field is present, Devices will confirm the number of certificates  
2796 between this certificate and the End-Entity certificate is less than or equal to  
2797 pathLenConstraint. In particular, if pathLenConstraint is zero, only an End-Entity certificate  
2798 can be issued by this certificate. If the pathLenConstraint field is absent, there is no limit to  
2799 the chain length.

2800 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged  
2801 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a  
2802 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a  
2803 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a  
2804 null value.

2805 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-  
2806 compliant Root CA certificates SHALL NOT contain a pathLenConstraint. This allows  
2807 additional tiers of Intermediate CAs to be implemented in the future without changing the Root  
2808 CA trust anchors, should such a requirement emerge.

2809 – keyUsage

2810 For all certificates, Devices shall verify that the key usage extension is present and flagged  
2811 critical.

2812 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits  
2813 SHALL be asserted.

2814 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be  
2815 asserted.

2816 – extendedKeyUsage:

2817 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be  
2818 rejected.

2819 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)  
2820 are required for compatibility with various TLS implementations.

2821 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)  
2822 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be  
2823 present and End-Entity certificates with EKU extensions containing both OIDs SHALL be  
2824 rejected.

2825 – certificatePolicies  
2826 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one  
2827 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)  
2828 corresponding to the version of the OCF Certificate Policy under which it was issued.  
2829 Additional manufacturer-specific CP OIDs may also be populated.

## 2830 **10.5 Device Authentication with OCF Cloud**

### 2831 **10.5.1 Device Authentication with OCF Cloud General**

2832 The mechanisms for Device Authentication in clauses 10.2, 10.3 and 10.4 imply that a Device is  
2833 authorized to communicate with any other Device meeting the criteria provisioned in  
2834 "/oic/sec/cred"; the "/oic/sec/acl2" Resource is additionally used to restrict access to specific  
2835 Resources. The present clause describes Device authentication for OCF Cloud, which uses  
2836 slightly different criteria as described in clause 5. A Device accessing an OCF Cloud shall  
2837 establish a TLS session. The mutual authenticated TLS session is established using Server  
2838 certificate and Client certificate.

2839 Each Device is identified based on the Access Token it is assigned during Device Registration.  
2840 The OCF Cloud holds an OCF Cloud association table that maps Access Token, User ID and  
2841 Device ID. The Device Registration shall happen while the Device is in RFNOP state. After  
2842 Device Registration, the updated Access Token, Device ID and User ID are used by the Device  
2843 for the subsequent connection with the OCF Cloud.

### 2844 **10.5.2 Device Connection with the OCF Cloud**

2845 The Device should establish the TLS connection using the certificate based credential. The  
2846 connection should be established after Device is provisioned by Mediator.

2847 The TLS session is established between Device and the OCF Cloud as specified in IETF RFC  
2848 8323. The OCF Cloud is expected to provide certificate signed by trust anchor that is present in  
2849 cred entries of the Device. These cred entries are expected to be configured by the Mediator.

2850 The Device shall validate the OCF Cloud's identity based on the credentials that are contained in  
2851 "/oic/sec/cred" Resource entries of the Device.

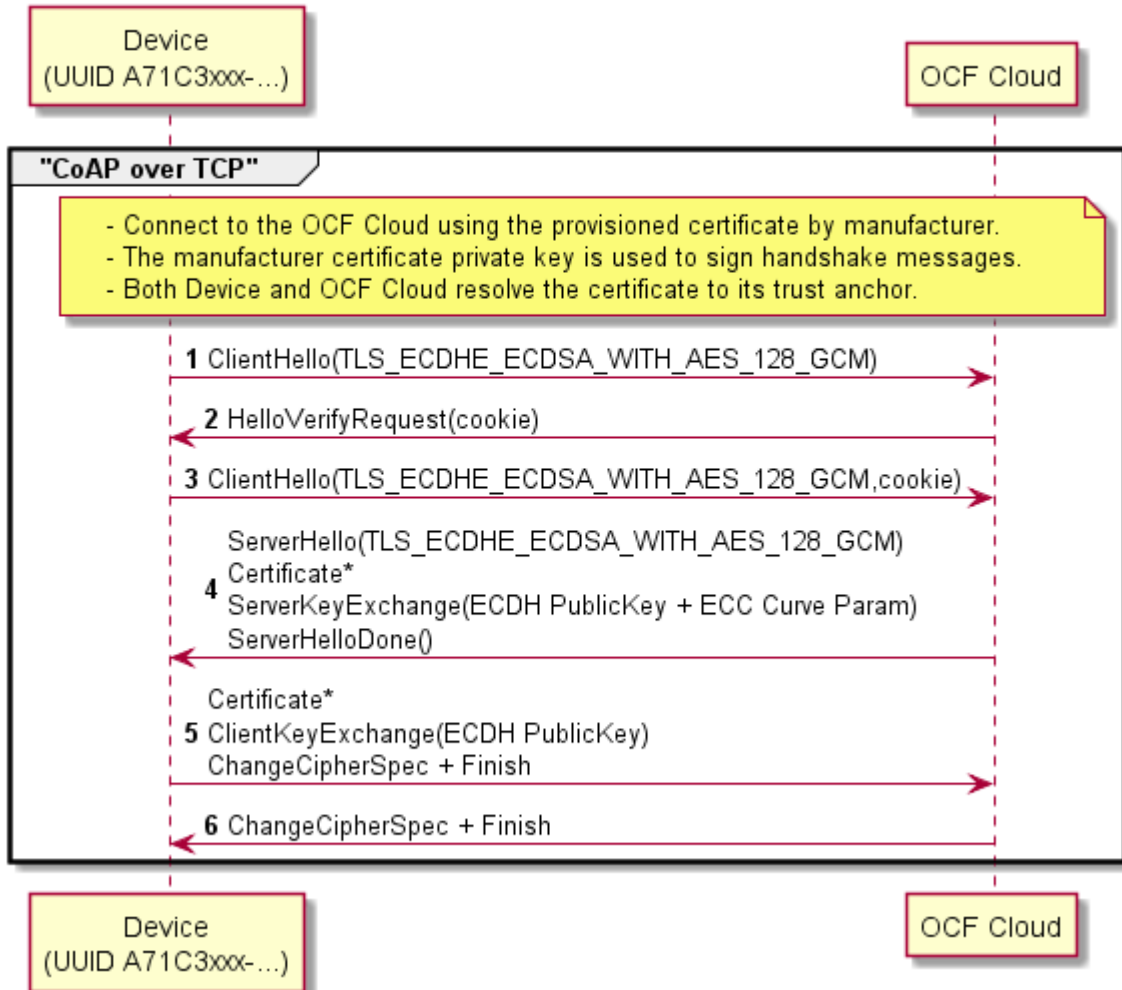
2852 The OCF Cloud is expected to validate the manufacturer certificate provided by the Device.

2853 The assumption is that the OCF Cloud User trusts the OCF Cloud that the Device connects. The  
2854 OCF Cloud connection should not happen without the consent of the OCF Cloud User. The  
2855 assumption is that the OCF Cloud User has either service agreement with the OCF Cloud  
2856 provider or uses manufacturer provided OCF Cloud.

2857 If authentication fails, the "clec" Property of "oic.r.coapcloudconf" Resource on the Device shall  
2858 be updated about the failed state, if it is supported by the Device. If authentication succeeds, the  
2859 Device and OCF Cloud should establish an encrypted link in accordance with the negotiated  
2860 cipher suite.

2861 Figure 33 depicts sequence for Device connection with OCF Cloud and steps described in  
2862 Table 22.

### Device Connection with OCF Cloud



2863

2864

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

2865

**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

2866

#### 10.5.3 Security Considerations

2867

2868

2869

2870

2871

2872

2873

2874

2875

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.

2876 The security considerations about the manufacturer certificate as described in 7.3.6.5 are also  
2877 applicable in the Device authentication with the OCF Cloud.

2878 The Device should validate the OCF Cloud's TLS certificate as defined by IETF RFC 6125 and in  
2879 accordance with its requirements for Server identity authentication.

2880 The "uid" and "di" Property Value of "/oic/d" Resource may be considered personally identifiable  
2881 information in some regulatory regions, and the OCF Cloud is expected to provide protections  
2882 appropriate to its governing regulatory bodies.

2883

2884 **11 Message Integrity and Confidentiality**

2885 **11.1 Preamble**

2886 Secured communications between Clients and Servers are protected against eavesdropping,  
2887 tampering, or message replay, using security mechanisms that provide message confidentiality  
2888 and integrity.

2889 **11.2 Session Protection with DTLS**

2890 **11.2.1 DTLS Protection General**

2891 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices  
2892 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See  
2893 11.3 for a list of required and optional cipher suites for message communication.

2894 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1  
2895 or lower.

2896 Multicast session semantics are not yet defined in this version of the security document.

2897 **11.2.2 Unicast Session Semantics**

2898 For unicast messages between a Client and a Server, both Devices shall authenticate each other.  
2899 See clause 10 for details on Device Authentication.

2900 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.  
2901 The sending Device shall encrypt and authenticate messages as defined by the selected cipher  
2902 suite and the receiving Device shall verify and decrypt the messages before processing them.

2903 **11.2.3 Cloud Session Semantics**

2904 The messages between the OCF Cloud and Device shall be exchanged only if the Device and  
2905 OCF Cloud authenticate each other as described in 10.4.3. The asymmetric cipher suites as  
2906 described in 11.3.5 shall be employed for establishing a secured session and for  
2907 encrypting/decrypting between the OCF Cloud and the Device. The OCF Endpoint sending the  
2908 message shall encrypt and authenticate the message using the cipher suite as described in  
2909 11.3.5 and the OCF Endpoint shall verify and decrypt the message before processing it.

2910 **11.3 Cipher Suites**

2911 **11.3.1 Cipher Suites General**

2912 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher  
2913 suites allowed during ownership transfer and normal operation. The following RFCs provide  
2914 additional information about the cipher suites used in OCF.

2915 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2916 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2917 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and  
2918 PSKs

2919 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2920 **11.3.2 Cipher Suites for Device Ownership Transfer**

2921 **11.3.2.1 Just Works Method Cipher Suites**

2922 The Just Works OTM may use the following (D)TLS cipher suites.

2923 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,



2924 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

2925 All Devices supporting Just Works OTM shall implement:

2926 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256 (with the value 0xFF00)

2927 All Devices supporting Just Works OTM should implement:

2928 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256 (with the value 0xFF01)

2929 **11.3.2.2 Random PIN Method Cipher Suites**

2930 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2931 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2932 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2933 All Devices supporting Random Pin Based OTM shall implement:

2934 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256

2935 **11.3.2.3 Certificate Method Cipher Suites**

2936 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2937 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

2938 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2939 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2940 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2941 Using the following curve:

2942 secp256r1 (See IETF RFC 4492)

2943 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2944 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2945 Devices supporting Manufacturer Certificate Based OTM should implement:

2946 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2947 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2948 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2949 **11.3.3 Cipher Suites for Symmetric Keys**

2950 The following cipher suites are defined for (D)TLS communication using PSKs:

2951 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2952 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2953 TLS\_PSK\_WITH\_AES\_128\_CCM\_8, (\* 8 OCTET Authentication tag \*)

2954 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

2955 TLS\_PSK\_WITH\_AES\_128\_CCM, (\* 16 OCTET Authentication tag \*)

2956 TLS\_PSK\_WITH\_AES\_256\_CCM,

2957 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2958 All Devices shall implement the following:

2959 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,  
2960

2961 Devices should implement the following:

2962 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,  
2963 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,  
2964 TLS\_PSK\_WITH\_AES\_128\_CCM\_8,  
2965 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,  
2966 TLS\_PSK\_WITH\_AES\_128\_CCM,  
2967 TLS\_PSK\_WITH\_AES\_256\_CCM

#### 2968 **11.3.4 Cipher Suites for Asymmetric Credentials**

2969 The following cipher suites are defined for (D)TLS communication with asymmetric keys or  
2970 certificates:

2971 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,  
2972 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,  
2973 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,  
2974 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2975 Using the following curve:

2976 secp256r1 (See IETF RFC 4492)

2977 All Devices supporting Asymmetric Credentials shall implement:

2978 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2979 All Devices supporting Asymmetric Credentials should implement:

2980 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,  
2981 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,  
2982 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

#### 2983 **11.3.5 Cipher suites for OCF Cloud Credentials**

2984 The following cipher suites are defined for TLS communication with certificates:

2985 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,  
2986 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,  
2987 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,  
2988 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,  
2989 TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,  
2990 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

2991 All Devices supporting OCF Cloud Certificate Credentials shall implement:

2992 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256

2993 All Devices supporting OCF Cloud Certificate Credentials should implement:

2994 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,  
2995 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,

2996 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,  
2997 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,  
2998 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  
2999

3000 **12 Access Control**

3001 **12.1 ACL Generation and Management**

3002 This clause will be expanded in a future version of the document.

3003 **12.2 ACL Evaluation and Enforcement**

3004 **12.2.1 ACL Evaluation and Enforcement General**

3005 The Server enforces access control over application Resources before exposing them to the  
3006 requestor. The Security Layer in the Server authenticates the requestor when access is received  
3007 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL  
3008 entries that specify the requestor's identity, role or may match authenticated requestors using a  
3009 subject wildcard.

3010 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a  
3011 subject wildcard match of anonymous requestors.

3012 Access is denied if a requested Resource is not matched by an ACL entry.

3013 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources  
3014 may be granted prior to provisioning of ACL Resources.

3015 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries  
3016 (ACE2) that employ a Resource matching algorithm that uses an array of Resource references to  
3017 match Resources to which the ACE2 access policy applies. Matching consists of comparing the  
3018 values of the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources  
3019 are matched in two ways:

- 3020 1) host reference ("href")
- 3021 2) resource wildcard ("wc").

3022 **12.2.2 Host Reference Matching**

3023 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for  
3024 Resource matching.

3025 – The href Property shall be used to find an exact match of the Resource name if present.

3026 **12.2.3 Resource Wildcard Matching**

3027 When present, a wildcard (wc) expression shall be used to match multiple Resources using a  
3028 wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

3029 A wildcard expression may be used to match multiple Resources using a wildcard Property  
3030 contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined  
3031 in Table 23.

3032 **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.

3033 NOTE Discoverable resources appear in the "/oic/res" Resource, while non-discoverable resources may appear in  
3034 other collection resources but do not appear in the /res collection.

3035 **12.2.4 Multiple Criteria Matching**

3036 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for  
3037 each array element. For example, if a first array element of the "resources" Property contains  
3038 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",  
3039 then Resources that match either of the two "href" criteria shall be included in the set of matched  
3040 Resources.

3041 Example 1 JSON for Resource matching

```
3042 {  
3043 //Matches Resources named "/x/door1" or "/x/door2"  
3044 "resources": [  
3045   {  
3046     "href": "/x/door1"  
3047   },  
3048   {  
3049     "href": "/x/door2"  
3050   },  
3051 ]  
3052 }
```

3053 Example 2 JSON for Resource matching

```
3054 {  
3055 // Matches all Resources  
3056 "resources": [  
3057   {  
3058     "wc": "*"   
3059   }  
3060 ]  
3061 }
```

3062 **12.2.5 Subject Matching using Wildcards**

3063 When the ACE subject is specified as the wildcard string "\*" any requestor is matched. The OCF  
3064 server may authenticate the OCF client, but is not required to.

3065 Examples: JSON for subject wildcard matching

```
3066 //matches all subjects that have authenticated and confidentiality protections in place.  
3067 "subject" : {  
3068   "conntype" : "auth-crypt"  
3069 }  
3070 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
3071 "subject" : {  
3072   "conntype" : "anon-clear"  
3073 }
```

3074 **12.2.6 Subject Matching using Roles**

3075 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 3076 1) The requestor authenticated with a symmetric key credential, and the role is present in the  
3077 roleid Property of the credential's entry in the credential resource, or

3078 2) The requestor authenticated with a certificate, and a valid role certificate is present in the  
3079 roles resource with the requestor's certificate's public key at the time of evaluation. Validating  
3080 role certificates is defined in 10.3.1.

## 3081 **12.2.7 ACL Evaluation**

### 3082 **12.2.7.1 ACE2 matching algorithm**

3083 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

3084 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these  
3085 ACE2 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the  
3086 signature, at least once, following update of the "/oic/sec/sacl" Resource.

3087 2) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.

3088 3) Access shall be granted when all these criteria are met:

3089 a) The requestor is matched by the ACE2 "subject" Property.

3090 b) The requested Resource is matched by the ACE2 resources Property and the requested  
3091 Resource shall exist on the local Server.

3092 c) The "period" Property constraint shall be satisfied.

3093 d) The "permission" Property constraint shall be applied.

3094 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching  
3095 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then  
3096 UNION (Perm1, Perm2)=CRUDN.

3097 The Server shall enforce access based on the effective permissions granted.

3098 Batch requests to Resource containing Links require additional considerations when accessing  
3099 the linked Resources. ACL considerations for batch request to the Atomic Measurement  
3100 Resource Type are provided in clause 12.2.7.2. ACL considerations for batch request to the  
3101 Collection Resource Type are provided in 12.2.7.3.

### 3102 **12.2.7.2 (Currently blank)**

3103 This clause intentionally left empty.

### 3104 **12.2.7.3 ACL considerations for a batch OCF Interface request to a Collection**

3105 This clause addresses the additional authorization processes which take place when a Server  
3106 receives a batch OCF Interface request from a Client to a Collection hosted on that Server,  
3107 assuming there is an ACE matching the Collection which permits the original Client request. For  
3108 the purposes of this clause, the Server hosting this Collection is called the "Collection host". The  
3109 additional authorization process is dependent on whether the linked Resource is hosted on the  
3110 Collection host or the linked Resource is hosted on another Server:

3111 – For each generated request to a linked Resource hosted on the Collection host, the Collection  
3112 host shall apply the ACE2 matching algorithm in clause 12.2.7.1 to determine whether the  
3113 linked Resource is permitted to process the generated request, with the following  
3114 clarifications:

3115 – The requestor in clause 12.2.7.1 shall be the Client which sent the original Client request.

3116 – The requested Resource in clause 12.2.7.1 shall be the linked Resource, which shall be  
3117 matched using at least one of:

3118 – a Resource Wildcard matching the linked Resource, or

3119 – an exact match of the local path of the linked Resource with a "href" Property in the  
3120 "resources" array in the ACE2.

3121           – an exact match of the full URI of the linked Resource with a "href" Property in the  
3122           "resources" array in the ACE2.

3123 NOTE The full URI of a linked Resource is obtained by concatenating the "anchor" Property of the Link, if present,  
3124 and the "href" Property of the Link. The local path can then be determined from the full URI.

3125 If the linked Resource is not permitted to process the generated request, then the Collection host  
3126 shall treat such cases as a linked Resource which cannot process the request when composing  
3127 the aggregated response to the original Client Request, as specified for the batch OCF Interface  
3128 in the ISO/IEC 30118-1:2018.

3129 **13 Security Resources**

3130 **13.1 Security Resources General**

3131 OCF Security Resources are shown in Figure 34.

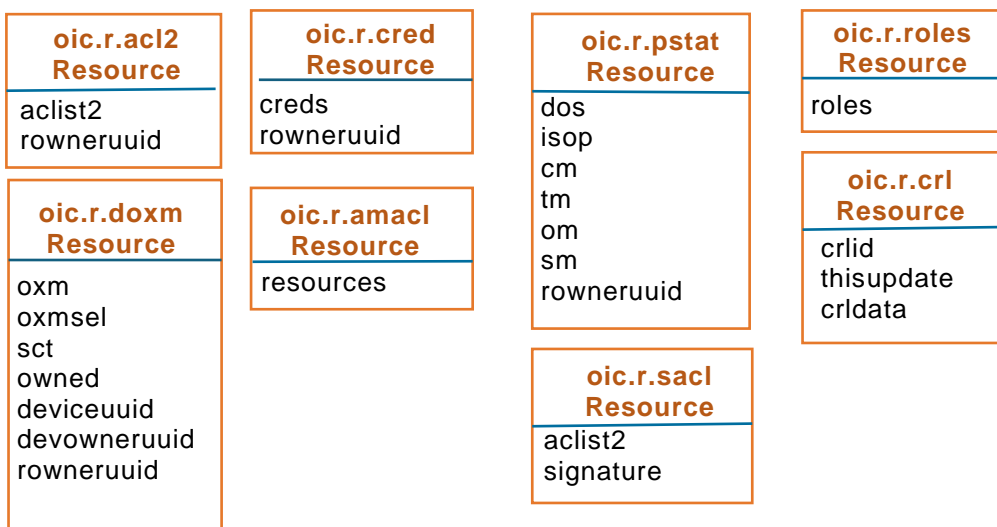
3132 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

3133 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

3134 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

3135 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

3136



3137

**Figure 34 – OCF Security Resources**



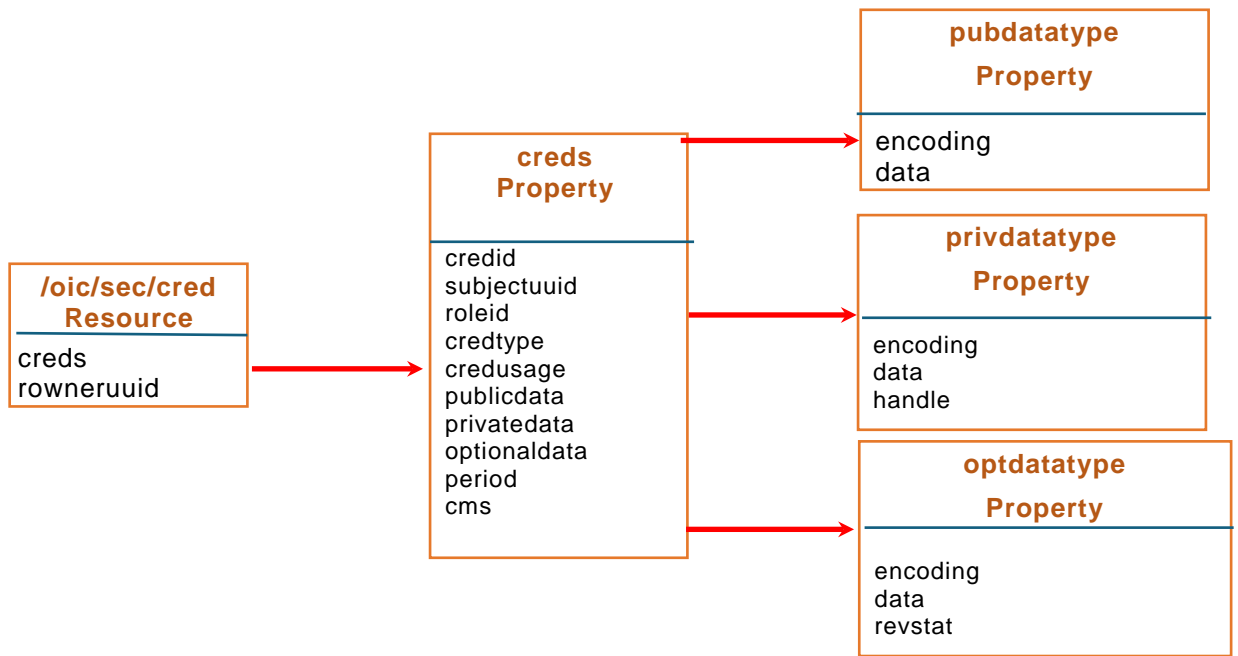


Figure 35 – "/oic/sec/cred" Resource and Properties

3138

3139

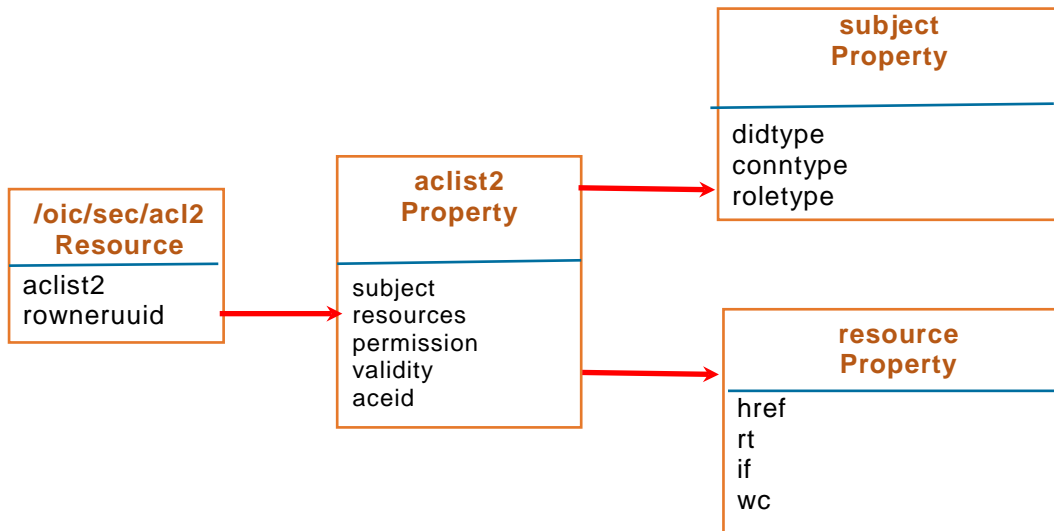
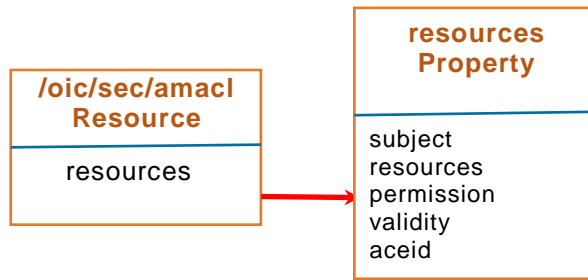


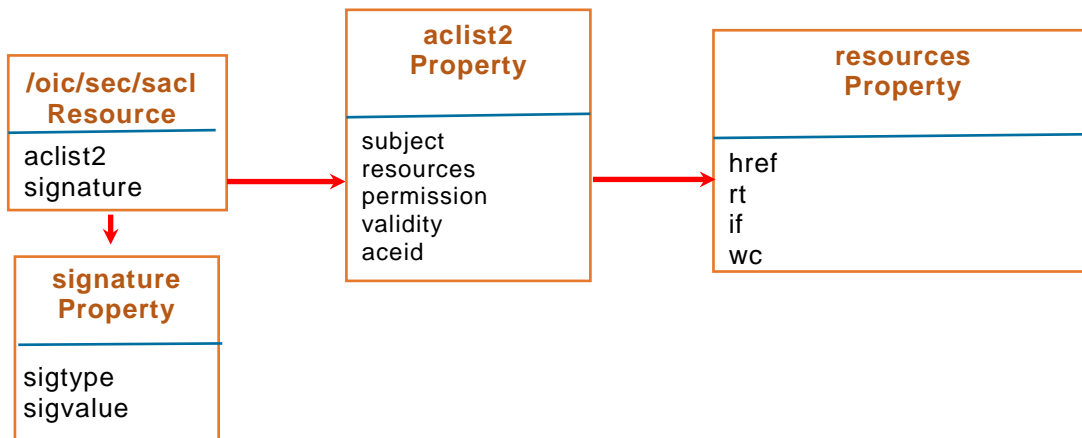
Figure 36 – "/oic/sec/acl2" Resource and Properties

3140



3141

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



3142

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

## 3143 13.2 Device Owner Transfer Resource

### 3144 13.2.1 Device Owner Transfer Resource General

3145 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

3146 Resource discovery processing respects the CRUDN constraints supplied as part of the security  
 3147 Resource definitions contained in this document.

3148 "/oic/sec/doxm" Resource is defined in Table 24.

**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

3150 Table 25 defines the Properties of the "/oic/sec/doxm" Resource.

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

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
OTM	oxms	oic.sec.doxmtype	array	Yes		R	Value identifying the owner-transfer-method and the organization that defined the method.
OTM Selection	oxmsel	oic.sec.doxmtype	UINT16	Yes	RESET	R	Server shall set to (4) "oic.sec.oxm.self"
					RFOTM	RW	DOTS shall set to its selected DOTS and both parties execute the DOTS. After secure owner transfer session is established DOTS shall update the oxmsel again making it permanent. If the DOTS fails the Server shall transition device state to RESET.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a
Supported Credential Types	sct	oic.sec.credtype	bitmask	Yes		R	Identifies the types of credentials the Device supports. The Server sets this value at framework initialization after determining security capabilities.
Device Ownership Status	owned	Boolean	T F	Yes	RESET	R	Server shall set to FALSE.
					RFOTM	RW	DOTS shall set to TRUE after secure owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	TRUE.n/a
					SRESET	R	TRUE.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.

3152 Table 26 defines the Properties of the "oic.sec.didtype".

3153 **Table 26 – Properties of the "oic.sec.didtype" type**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Device ID	uuid	String	uuid	Yes	RW	-	A uuid value

3154 The oxms Property contains a list of OTM where the entries appear in the order of preference.  
3155 This Property contains the higher priority methods appearing before the lower priority methods.  
3156 The DOTS queries this list at the time of onboarding and selects the most appropriate method.

3157 The DOTS shall update the oxmsel Property of the "/oic/sec/doxm" Resource with the OTM that  
3158 was used to onboard the Device.

3159 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3160 <DoxmType> ::= <NSS>
3161 <NSS> ::= <Identifier> | { {<NID> "."} <NameSpaceQualifier> "."} <Method>
3162 <NID> ::= <Vendor-or-Organization>
3163 <Identifier> ::= INTEGER
3164 <NameSpaceQualifier> ::= String
3165 <Method> ::= String
3166 <Vendor-Organization> ::= String

```

3167 When an OTM successfully completes, the "owned" Property is set to "1" (TRUE). Consequently,  
3168 subsequent attempts to take ownership of the Device will fail.

3169 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in  
3170 the "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource  
3171 is UPDATED to non-nil UUID value.

3172 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource  
3173 after it has updated the devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-  
3174 nil-UUID value.

3175 The Device vendor shall determine that the Device identifier ("deviceuuid") is persistent (not  
3176 updatable) or that it is non-persistent (updatable by the owner transfer service – aka. DOTS).

3177 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall  
3178 fail with the error PROPERTY\_NOT\_FOUND.

3179 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is non-persistent, the request to  
3180 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is  
3181 RESET. If the UPDATE to deviceuuid Property of the "/oic/sec/doxm" Resource fails while in the  
3182 RFOTM Device state the device state shall transition to RESET where the Server shall set the  
3183 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g.  
3184 "00000000-0000-0000-0000-000000000000").

3185 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the  
3186 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the  
3187 "deviceuuid" Property of the "/oic/sec/doxm" Resource each time the device enters RESET  
3188 Device state. The temporary deviceuuid value is used while the device state is in the RESET  
3189 state and while in the RFOTM device state until the DOTS establishes a secure OTM connection.  
3190 The DOTS should RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm"  
3191 Resource after it has updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a  
3192 non-nil-UUID value.

3193 The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall expose a persistent value (i.e. is  
3194 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTS via  
3195 an OCF Interface to the deviceuuid Property of the "/oic/sec/doxm" Resource during RFOTM  
3196 Device state.).

3197 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a  
3198 secure OTM connection and UPDATES the "devowneruuid" Property to a non-nil UUID value.  
3199 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the "deviceuuid" Property  
3200 of the "/oic/sec/doxm" Resource shall reveal the persistent or semi-persistent value to  
3201 authenticated requestors and shall reveal the temporary non-repeated value to unauthenticated  
3202 requestors.

3203 See 13.16 for additional details related to privacy sensitive considerations.

### 3204 **13.2.2 Persistent and Semi-Persistent Device Identifiers**

3205 The Device vendor determines whether a device identifier can be set by a configuration tool or  
3206 whether it is immutable. If it is an immutable value this document refers to it as a persistent  
3207 device identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four  
3208 device identifiers that could be considered persistent or semi-persistent:

- 3209 1) "deviceuuid" Property of "/oic/sec/doxm"
- 3210 2) "di" Property of "/oic/d"
- 3211 3) "piid" Property of "/oic/d"
- 3212 4) "pi" Property of "/oic/p"

### 3213 **13.2.3 Onboarding Considerations for Device Identifier**

3214 The "deviceuuid" is used to onboard the Device. The other identifiers ("di", "piid" and "pi") are not  
3215 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the  
3216 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security  
3217 Domain owner may have a preference for persistent or semi-persistent device identifiers.

3218 Detecting whether the Device is using persistent or semi-persistent deviceuuid can be achieved  
 3219 by attempting to update it.

3220 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is persistent, then an UPDATE  
 3221 request, at the appropriate time during onboarding shall fail with an appropriate error response.

3222 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device  
 3223 state is RFOTM and when devowneruid Property value of the "/oic/sec/doxm" Resource has a  
 3224 non-nil UUID value.

3225 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is semi-persistent, subsequent to a  
 3226 successful UPDATE request to change it; the Device shall remember the semi-persistent value  
 3227 until the next successful UPDATE request or until the Device state transitions to RESET.

3228 See 13.16 for addition behaviour regarding "deviceuuid".

3229

### 3230 13.2.4 OCF defined OTMs

3231 Table 27 defines the Properties of the "oic.sec.doxmtype".

3232 **Table 27 – Properties of the "oic.sec.doxmtype" type**

Value Type Name	Value Type URN (optional)	Enumeration Value (mandatory)	Description
OCFJustWorks	oic.sec.doxm.jw	0	The just-works method relies on anonymous Diffie-Hellman key agreement protocol to allow an DOTS to assert ownership of the new Device. The first DOTS to make the assertion is accepted as the Device owner. The just-works method results in a shared secret that is used to authenticate the Device to the DOTS and likewise authenticates the DOTS to the Device. The Device allows the DOTS to take ownership of the Device, after which a second attempt to take ownership by a different DOTS will fail <sup>a</sup> .
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a DOTS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTS signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec.doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.oxm.self	4	The manufacturer shall set the "/doxm.oxmself" value to (4). The Server shall reset this value to (4) upon entering RESET Device state.
OCF Reserved	<Reserved>	5~0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	Reserved for vendor-specific OTM use

<sup>a</sup> The just-works method is subject to a man-in-the-middle attacker. Precautions should be taken to provide physical security when this method is used.

3233 **13.3 Credential Resource**

3234 **13.3.1 Credential Resource General**

3235 The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and  
3236 support services as well as credentials used to verify Clients and support services.

3237 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared  
3238 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to  
3239 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3240 In order to provide an interface which allows management of the "creds" Array Property, the  
3241 RETRIEVE, UPDATE and DELETE operations on the "oic.r.cred" Resource shall behave as  
3242 follows:

3243 1) A RETRIEVE shall return the full Resource representation, except that any write-only  
3244 Properties shall be omitted (e.g. private key data).

3245 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3246 UPDATE request, as follows:

3247 a) If an UPDATE representation includes the "creds" array Property, then:

3248 i) Supplied "creds" with a "credid" that matches an existing "credid" shall replace  
3249 completely the corresponding "cred" in the existing "creds" array.

3250 ii) Supplied "creds" without a "credid" shall be appended to the existing "creds" array,  
3251 and a unique (to the cred Resource) "credid" shall be created and assigned to the new  
3252 "cred" by the Server. The "credid" of a deleted "cred" should not be reused, to  
3253 improve the determinism of the interface and reduce opportunity for race conditions.

3254 iii) Supplied "creds" with a "credid" that does not match an existing "credid" shall be  
3255 appended to the existing "creds" array, using the supplied "credid".

3256 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device  
3257 States in which an UPDATE of the "creds" array Property is always rejected. If OCF  
3258 Device is in a Device State where the Access Mode in this row contains "R", then the  
3259 OCF Device shall reject all UPDATES of the "creds" array Property.

3260 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not  
3261 remove the "oic.r.cred" Resource.

3262 4) A DELETE with one or more "credid" query parameters shall remove the "cred"(s) with the  
3263 corresponding "credid"(s) from the "creds" array.

3264 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in  
3265 which a DELETE is always rejected. If OCF Device is in a Device State where the Access  
3266 Mode in this row contains "R", then the OCF Device shall reject all DELETES.

3267 NOTE The "oic.r.cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3268 in ISO/IEC 30118-1:2018.

3269 "oic.r.cred" Resource is defined in Table 28.

3270 **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

3271 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.

3273 All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end  
3274 interaction.

3275 The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid  
3276 Property.

3277 ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access  
3278 modes.

3279 Table 30 defines the Properties of "oic.sec.cred".



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

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Credential ID	credid	UINT16	0 – 64K-1	Yes	RW		Short credential ID for local references from other Resource
Subject UUID	subjectuuid	String	uuid	Yes	RW		A uuid that identifies the subject to which this credential applies or "*" if any identity is acceptable
Role ID	roleid	oic.sec.roletype	-	No	RW		Identifies the role(s) the subject is authorized to assert.
Credential Type	credtype	oic.sec.credtype	bitmask	Yes	RW		Represents this credential's type. 0 – Used for testing 1 – Symmetric pair-wise key 2 – Symmetric group key 4 – Asymmetric signing key 8 – Asymmetric signing key with certificate 16 – PIN or password 32 – Asymmetric encryption key
Credential Usage	credusage	oic.sec.credusage	String	No	RW		Used to resolve undecidability of the credential. Provides indication for how/where the cred is used "oic.sec.cred.trustca": certificate trust anchor "oic.sec.cred.cert": identity certificate "oic.sec.cred.rolecert": role certificate "oic.sec.cred.mfgtrustca": manufacturer certificate trust anchor "oic.sec.cred.mfgcert": manufacturer certificate
Public Data	publicdata	oic.sec.pubdatatype	-	No	RW		Public credential information 1:2: ticket, public SKDC values 4, 32: Public key value 8: A chain of one or more certificate
Private Data	privatedata	oic.sec.privdatatype	-	No	-	RESET	Server shall set to manufacturer default
					RW	RFOTM	Set by DOTS after successful OTM
					W	RFPRO	Set by authenticated DOTS or CMS
					-	RFNOP	Not writable during normal operation.
					W	SRESET	DOTS may modify to enable transition to RFPRO.
Optional Data	optionaldata	oic.sec.optdatatype	-	No	RW		Credential revocation status information 1, 2, 4, 32: revocation status information 8: Revocation information
Period	period	String	-	No	RW		Period as defined by IETF RFC 5545. The credential should not be used if the current time is outside the Period window.
Credential Refresh Method	crms	oic.sec.crmtype	array	No	RW		Credentials with a Period Property are refreshed using the credential refresh method (crm) according to the type definitions for "oic.sec.crm".

3281 Table 31 defines the Properties of "oic.sec.credusagetype".

3282 **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

3283 Table 32 defines the Properties of "oic.sec.pubdatatype".

3284 **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

3285 Table 33 defines the Properties of "oic.sec.privdatatype".

3286 **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

3287 Table 34 defines the Properties of "oic.sec.optdatatype".

3288

**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

3289 Table 35 defines the Properties of "oic.sec.roletype".

3290

**Table 35 – Definition of the "oic.sec.roletype" type.**

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.

3291 **13.3.2 Properties of the Credential Resource**

3292 **13.3.2.1 Credential ID**

3293 Credential ID ("credid") is a local reference to an entry in a "creds" Property array of the  
3294 "/oic/sec/cred" Resource. The SRM generates it. The "credid" Property shall be used to  
3295 disambiguate array elements of the "creds" Property.

3296 **13.3.2.2 Subject UUID**

3297 The "subjectuid" Property identifies the Device to which an entry in a "creds" Property array of  
3298 the "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication  
3299 challenge-response or to authenticate an authentication challenge.

3300 A "subjectuid" Property that matches the Server's own "deviceuid" Property, distinguishes the  
3301 array entries in the "creds" Property that pertain to this Device.

3302 The "subjectuid" Property shall be used to identify a group to which a group key is used to  
3303 protect shared data.

3304 When certificate chain is used during secure connection establishment, the "subjectuuid"  
3305 Property shall also be used to verify the identity of the responder. The presented certificate chain  
3306 shall be accepted, if there is a matching Credential entry on the Device that satisfies all of the  
3307 following:

- 3308 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3309 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity  
3310 certificate in the presented chain. If Subject UUID of the entry is set as a wildcard "\*", this  
3311 condition is automatically satisfied.
- 3312 – Credential Usage of the entry is "oic.sec.cred.trustca".

#### 3313 **13.3.2.3 Role ID**

3314 The roleid Property identifies a role that has been granted to the credential.

#### 3315 **13.3.2.4 Credential Type**

3316 The "credtype" Property is used to interpret several of the other Property values whose contents  
3317 can differ depending on credential type. These Properties include "publicdata", "privatedata" and  
3318 "optionaldata". The "credtype" Property value of "0" ("no security mode") is reserved for testing  
3319 and debugging circumstances. Production deployments shall not allow provisioning of credentials  
3320 of type "0". The SRM should introduce checking code that prevents its use in production  
3321 deployments.

#### 3322 **13.3.2.5 Public Data**

3323 The "publicdata" Property contains information that provides additional context surrounding the  
3324 issuance of the credential. For example, it might contain information included in a certificate or  
3325 response data from a CMS. It might contain wrapped data.

#### 3326 **13.3.2.6 Private Data**

3327 The "privatedata" Property contains secret information that is used to authenticate a Device,  
3328 protect data or verify an authentication challenge-response.

3329 The "privatedata" Property shall not be disclosed outside of the SRM's trusted computing  
3330 perimeter. A secure element (SE) or trusted execution environment (TEE) should be used to  
3331 implement the SRM's trusted computing perimeter. The privatedata contents may be referenced  
3332 using a handle; for example, if used with a secure storage sub-system.

#### 3333 **13.3.2.7 Optional Data**

3334 The "optionaldata" Property contains information that is optionally supplied, but facilitates key  
3335 management, scalability or performance optimization.

#### 3336 **13.3.2.8 Period**

3337 The "period" Property identifies the validity period for the credential. If no validity period is  
3338 specified, the credential lifetime is undetermined. Constrained devices that do not implement a  
3339 date-time capability shall obtain current date-time information from its CMS.

#### 3340 **13.3.2.9 Credential Refresh Method Type Definition**

3341 The CMS shall implement the credential refresh methods specified in the "crms" Property of the  
3342 "oic.sec.creds" array in the "/oic/sec/cred" Resource.

3343 Table 36 defines the values of "oic.sec.crmttype".

**Table 36 – Value Definition of the "oic.sec.crmtyp" Property**

Value Type Name	Value Type URN	Applicable Credential Type	Description
Provisioning Service	oic.sec.crm.pro	All	A CMS initiates re-issuance of credentials nearing expiration. The Server should delete expired credentials to manage storage resources. The Resource Owner Property references the provisioning service. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify additional key management service that supports this credential refresh method.
Pre-shared Key	oic.sec.crm.psk	[1]	The Server performs ad-hoc key refresh by initiating a DTLS connection with the Device prior to credential expiration using a Diffie-Hellman based ciphersuite and the current PSK. The new DTLS MasterSecret value becomes the new PSK. The Server selects the new validity period. The new validity period value is sent to the Device who updates the validity period for the current credential. The Device acknowledges this update by returning a successful response or denies the update by returning a failure response. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify a key management service that supports this credential refresh method.
Random PIN	oic.sec.crm.rdp	[16]	The Server performs ad-hoc key refresh following the "oic.sec.crm.psk" approach, but in addition generates a random PIN value that is communicated out-of-band to the remote Device. The current PSK + PIN are hashed to form a new PSK' that is used with the DTLS ciphersuite. I.e. PSK' = SHA256(PSK, PIN). The Server uses its "/oic/sec/cred.rownruuid" Resource to identify a key management service that supports this credential refresh method.
SKDC	oic.sec.crm.skdc	[1, 2, 4, 32]	The Server issues a request to obtain a ticket for the Device. The Server updates the credential using the information contained in the response to the ticket request. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify the key management service that supports this credential refresh method.
PKCS10	oic.sec.crm.pk10	[8]	The Server issues a PKCS#10 certificate request message to obtain a new certificate. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify the key management service that supports this credential refresh method.

### 3345 13.3.2.10 Credential Usage

3346 Credential Usage indicates to the Device the circumstances in which a credential should be used.  
3347 Five values are defined:

- 3348 – "oic.sec.cred.trustca": This certificate is a trust anchor for the purposes of certificate chain  
3349 validation, as defined in 10.4. OCF Server SHALL remove any "/oic/sec/cred" entries with an  
3350 "oic.sec.cred.trustca" credusage upon transitioning to RFOTM. OCF Servers SHALL use  
3351 "/oic/sec/cred" entries that have an "oic.sec.cred.trustca" Value of "credusage" Property only  
3352 as trust anchors for post-onboarding (D)TLS session establishment in RFNOP state; these  
3353 entries are not to be used for onboarding (D)TLS sessions.
- 3354 – "oic.sec.cred.cert": This "credusage" is used for certificates for which the Device possesses  
3355 the private key and uses it for identity authentication in a secure session, as defined in clause  
3356 10.4.
- 3357 – "oic.sec.cred.rolecert": This "credusage" is used for certificates for which the Device  
3358 possesses the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3359 – "oic.sec.cred.mfgtrustca": This certificate is a trust anchor for the purposes of the  
3360 Manufacturer Certificate Based OTM as defined in clause 7.3.6. OCF Servers SHALL use  
3361 "/oic/sec/cred" entries that have an "oic.sec.cred.mfgtrustca" Value of "credusage" Property  
3362 only as trust anchors for onboarding (D)TLS session establishment; these entries are not to  
3363 be used for post-onboarding (D)TLS sessions.

3364 – "oic.sec.cred.mfgcert": This certificate is used for certificates for which the Device possesses  
3365 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as  
3366 defined in clause 7.3.6.

### 3367 **13.3.3 Key Formatting**

#### 3368 **13.3.3.1 Symmetric Key Formatting**

3369 Symmetric keys shall have the format described in Table 37 and Table 38.

3370 **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.

3371

3372 **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.

#### 3373 **13.3.3.2 Asymmetric Keys**

3374 Asymmetric key formatting is not available in this revision of the document.

#### 3375 **13.3.3.3 Asymmetric Keys with Certificate**

3376 Key formatting is defined by certificate definition.

#### 3377 **13.3.3.4 Passwords**

3378 Password formatting is not available in this revision of the document.

### 3379 **13.3.4 Credential Refresh Method Details**

#### 3380 **13.3.4.1 Provisioning Service**

3381 The resource owner identifies the provisioning service. If the Server determines a credential  
3382 requires refresh and the other methods do not apply or fail, the Server will request re-  
3383 provisioning of the credential before expiration. If the credential is allowed to expire, the Server  
3384 should delete the Resource.

#### 3385 **13.3.4.2 Pre-Shared Key**

##### 3386 **13.3.4.2.1 Pre-Shared Key General**

3387 Using this mode, the current PSK is used to establish a Diffie-Hellman session key in DTLS. The  
3388 TLS\_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3389  $PSK = TLS\_PRF(\text{MasterSecret}, \text{Message}, \text{length});$

3390 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of  
3391 the above ciphersuites.

3392 – Message is the concatenation of the following values:

3393 – RM - Refresh method – I.e. "oic.sec.crm.psk"

3394 – Device ID\_A is the string representation of the Device ID that supplied the DTLS  
3395 ClientHello.

3396 – Device ID\_B is the Device responding to the DTLS ClientHello message

3397 – Length of Message in bytes.

3398 Both Server and Client use the PSK to update the "/oic/sec/cred" Resource's "privatedata"  
3399 Property. If Server initiated the credential refresh, it selects the new validity period. The Server  
3400 sends the chosen validity period to the Client over the newly established DTLS session so it can  
3401 update the corresponding credential Resource for the Server.

#### 3402 **13.3.4.2.2 Random PIN**

3403 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.  
3404 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session  
3405 key should be used to switch from PIN to PSK mode.

3406 The PIN is randomly generated by the Server and communicated to the Client through an out-of-  
3407 band method. The OOB method used is out-of-scope.

3408 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used  
3409 to generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a  
3410 DTLS ciphersuite that accepts a PSK.

3411  $PPSK = PBKDF2(PRF, PIN, RM, Device\ ID, c, dkLen)$

3412 The PBKDF2 function has the following parameters:

3413 – PRF – Uses the DTLS PRF.

3414 – PIN – Shared between Devices.

3415 – RM - Refresh method – I.e. "oic.sec.crm.rdp"

3416 – Device ID – UUID of the new Device.

3417 – c – Iteration count initialized to 1000, incremented upon each use.

3418 – dkLen – Desired length of the derived PSK in octets.

3419 Both Server and Client use the PPSK to update the "/oic/sec/cred" Resource's PrivateData  
3420 Property. If Server initiated the credential refresh, it selects the new validity period. The Server  
3421 sends the chosen validity period to the Client over the newly established DTLS session so it can  
3422 update its corresponding credential Resource for the Server.

#### 3423 **13.3.4.2.3 SKDC**

3424 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid  
3425 Property value that matches a CMS that implements SKDC functionality and where the Client  
3426 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request  
3427 message is delivered to the CMS and in response returns the ticket request. The Server updates  
3428 or instantiates a "/oic/sec/cred" Resource guided by the ticket response contents.

#### 3429 **13.3.4.2.4 PKCS10**

3430 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid  
3431 Property value that matches a CMS that supports the "oic.sec.crm.pk10" credential refresh  
3432 method. A PKCS10 formatted message is delivered to the service. After the refreshed certificate  
3433 is issued, the CMS pushes the certificate to the Server. The Server updates or instantiates an  
3434 "/oic/sec/cred" Resource guided by the certificate contents.

3435 **13.3.4.3 Resource Owner**

3436 The Resource Owner Property allows credential provisioning to occur soon after Device  
 3437 onboarding before access to support services has been established. It identifies the entity  
 3438 authorized to manage the "/oic/sec/cred" Resource in response to Device recovery situations.

3439 **13.4 Certificate Revocation List**

3440 **13.4.1 CRL Resource Definition**

3441 Device certificates and private keys are kept in "cred" Resource. CRL is maintained and updated  
 3442 with a separate "crl" Resource that is newly defined for maintaining the revocation list.

3443 "oic.r.crl" Resource is defined in Table 39.

3444 **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

3445 Table 40 defines the Properties of "oic.r.crl".

3446 **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

3447 **13.5 ACL Resources**

3448 **13.5.1 ACL Resources General**

3449 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be  
 3450 expressed using three ACL Resource Types: "/oic/sec/acl2", "/oic/sec/amacl" and "/oic/sec/sacl".  
 3451 The subject (e.g. "deviceuuid" of the Client) requesting access to a Resource shall be  
 3452 authenticated prior to applying the ACL check. Resources that are available to multiple Clients  
 3453 can be matched using a wildcard subject. All Resources accessible via the unsecured  
 3454 communication endpoint shall be matched using a wildcard subject.

3455 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3456 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3457 **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>

3458 The <Deviceld> token means the requestor must possess a credential that uses <UUID> as its  
3459 identity in order to match the requestor to the <ACE> policy.

3460 The <RoleID> token means the requestor must possess a role credential with <Character> as its  
3461 role in order to match the requestor to the <ACE> policy.

3462 The <Wildcard> token "\*" means any requestor is matched to the <ACE> policy, with or without  
3463 authentication.

3464 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the  
3465 <ACE> policy to Resources.

3466 The <OIC\_LINK> token contains values used to query existence of hosted Resources.

3467 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>  
3468 and <ResourceRef> matching does not produce the empty set match.

3469 Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE  
3470 ("D"), NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the  
3471 respective permission is not granted.

3472 The empty set match result defaults to a condition where no access rights are granted.

3473 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.  
3474 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively  
3475 be granted and rescinded according to the pattern.

### 3476 13.5.3 ACL Resource

3477 An "acl2" is a list of type "ace2".

3478 In order to provide an interface which allows management of array elements of the "aclist2"  
3479 Property associated with a "/oic/sec/acl2" Resource. The RETRIEVE, UPDATE and DELETE  
3480 operations on the "/oic/sec/acl2" Resource SHALL behave as follows:

- 3481 1) A RETRIEVE shall return the full Resource representation.
- 3482 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3483 UPDATE request, as follows:

3484 a) If an UPDATE representation includes the array Property, then:

- 3485 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace  
3486 completely the corresponding ACE in the existing "aces2" array.

3487 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array,  
3488 and a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new  
3489 ACE by the Server. The "aceid" of a deleted ACE should not be reused, to improve  
3490 the determinism of the interface and reduce opportunity for race conditions.

3491 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be  
3492 appended to the existing "aces2" array, using the supplied "aceid".

3493 The rows in Table 44 defines the Properties of "oic.sec.acl2".

3494 iv) Table 44 corresponding to the "aclist2" array Property dictate the Device States in  
3495 which an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is  
3496 in a Device State where the Access Mode in this row contains "R", then the OCF  
3497 Device shall reject all UPDATES of the "aclist2" array Property.

3498 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not  
3499 remove the "oic.r.ace2" Resource.

3500 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the  
3501 corresponding "aceid"(s) from the "aces2" array.

3502 The rows in Table 44 defines the Properties of "oic.sec.acl2".

3503 5) Table 44 corresponding to the "aclist2" array Property dictate the Device States in which a  
3504 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this  
3505 row contains "R", then the OCF Device shall reject all DELETES.

3506 NOTE The "oic.r.acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces  
3507 defined in ISO/IEC 30118-1:2018.

3508 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no  
3509 entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl2",  
3510 "/oic/sec/sacl" and "/oic/sec/amacl" do not match the subject and the requested Resource.

3511 It is possible the AMS has an ACL policy that satisfies a resource access request, but the  
3512 necessary ACE has not been provisioned to Server. The Server may open a secure connection to  
3513 the AMS to request ACL provisioning. The Server may use filter criteria that returns a subset of  
3514 the AMS ACL policy. The AMS shall obtain the Server Device ID using the secure connection  
3515 context.

3516 The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS  
3517 to process an "/oic/sec/amacl" Resource. The AMS shall match the AMACL policy and return the  
3518 Permission Property or an error if no match is found.

3519 If the requested Resource is still not matched, the Server returns an error. The requester should  
3520 query the Server to discover the configured AMS services. The Client should contact the AMS to  
3521 request a sacl ("/oic/sec/sacl") Resource. Performing the following operations implement this type  
3522 of request:

3523 1) Client: Open secure connection to AMS.

3524 2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3525 3) AMS: constructs a "/oic/sec/sacl" Resource that is signed by the AMS and returns it in  
3526 response to the RETRIEVE command.

3527 4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3528 5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3529 6) Client: retries original Resource access request. This time the new ACL is included in the  
3530 local ACL evaluation.

3531 The ACL contained in the "/oic/sec/sacl" Resource should grant longer term access that satisfies  
 3532 repeated Resource requests.

3533 Table 42 defines the values of "oic.sec.crudntype".

3534 **Table 42 – Value Definition of the "oic.sec.crudntype" Property**

Value	Access Policy	Description	RemarksNotes
bx0000,0000 (0)	No permissions	No permissions	N/A
bx0000,0001 (1)	C	CREATE	N/A
bx0000,0010 (2)	R	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

3535 "oic.sec.acl2" Resource is defined in Table 28.

3536 **Table 43 – Definition of the "oic.sec.acl2" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/acl2	ACL2	oic.r.acl2	baseline	Resource for managing access	Security

3537 Table 44 defines the Properties of "oic.sec.acl2".

3538 **Table 44 – Properties of the "oic.sec.acl2" Resource**

Property Name	Value Type	Mandatory	Device State	Access Mode	Description
aclist2	array of oic.sec.ace2	Yes	N/A		The aclist2 Property is an array of ACE records of type "oic.sec.ace2". The Server uses this list to apply access control to its local resources.
N/A	N/A	N/A	RESET	R	Server shall set to manufacturer defaults.
			RFOTM	RW	Set by DOTS after successful OTM
			RFPRO	RW	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
			RFNOP	R	Access to NCRs is permitted after a matching ACE2 is found.
			SRESET	RW	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm Resource") should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.

rowneruid	uuid	Yes	N/A		The resource owner Property (rowneruid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
			RFOTM	RW	The DOTS should configure the rowneruid Property of "/oic/sec/acl2" Resource when a successful owner transfer session is established.
			RFPRO	R	n/a
			RFNOP	R	n/a
			SRESET	RW	The DOTS (referenced via devowneruid Property or rowneruid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruid Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3539

3540 Table 45 defines the Properties of "oic.sec.ace2".

3541

**Table 45 – "oic.sec.ace2" data type definition.**

Property Name	Value Type	Mandatory	Description
subject	oic.sec.roletype, oic.sec.didtype, oic.sec.conntype	Yes	The Client is the subject of the ACE when the roles, Device ID, or connection type matches.
resources	array of oic.sec.ace2.resource-ref	Yes	The application's resources to which a security policy applies
permission	oic.sec.crudntype.bitmask	Yes	Bitmask encoding of CRUDN permission
validity	array of oic.sec.time-pattern	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the IETF RFC 5545 Period, and a string array representing a recurrence rule using the IETF RFC 5545 Recurrence.
aceid	integer	Yes	An aceid is unique with respect to the array entries in the aclist2 Property.

3542 Table 46 defines the Properties of "oic.sec.ace2.resource-ref".

3543

**Table 46 – "oic.sec.ace2.resource-ref" data type definition.**

Property Name	Value Type	Mandatory	Description
href	uri	No	A URI referring to a resource to which the containing ACE applies
wc	string	No	Refer to Table 23.

3544 Table 47 defines the values of "oic.sec.ace2.resource-ref".

3545

**Table 47 – Value definition "oic.sec.conntype" Property**

Property Name	Value Type	Value Rule	Description
conntype	string	enum [ "auth-crypt", "anon-clear" ]	This Property allows an ACE to be matched based on the connection or message protection type
		auth-crypt	ACE applies if the Client is authenticated and the data channel or message is encrypted and integrity protected
		anon-clear	ACE applies if the Client is not authenticated and the data channel or message is not encrypted but may be integrity protected

3546 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack  
3547 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's  
3548 request using policies contained in ACL resources.

3549 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified  
3550 Resource references include the device identifier in the href Property that identifies the remote  
3551 Resource Server that hosts the Resource. Partially qualified references mean that the local  
3552 Resource Server hosts the Resource. If a fully qualified resource reference is given, the  
3553 Intermediary enforcing access shall have a secure channel to the Resource Server and the  
3554 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource  
3555 access enforcement point.

3556 Resource Servers should include references to Device and ACL Resources where access  
3557 enforcement is to be applied. However, access enforcement logic shall not depend on these  
3558 references for access control processing as access to Server Resources will have already been  
3559 granted.

3560 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and  
3561 modify this Resource. This prevents non-terminating dependency on some other ACL Resource.  
3562 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL  
3563 Resource.

3564 An ACE2 entry is considered "currently valid" if the validity period of the ACE2 entry includes the  
3565 time of the request. The validity period in the ACE2 may be a recurring time period (e.g., daily  
3566 from 1:00-2:00). Matching the resource(s) specified in a request to the resource Property of the  
3567 ACE2 is defined in clause 12.2. For example, one way they can match is if the Resource URI in  
3568 the request exactly matches one of the resource references in the ACE2 entries.

3569 A request will match an ACE2 if any of the following are true:

- 3570 1) The ACE2 "subject" Property is of type "oic.sec.didtype" has a UUID value that matches the  
3571 "deviceuuid" Property associated with the secure session;  
3572 AND the Resource of the request matches one of the resources Property of the ACE2  
3573 "oic.sec.ace2.resource-ref";  
3574 AND the ACE2 is currently valid.
- 3575 2) The ACE2 "subject" Property is of type "oic.sec.conntype" and has the wildcard value that  
3576 matches the currently established connection type;  
3577 AND the resource of the request matches one of the resources Property of the ACE2  
3578 "oic.sec.ace2.resource-ref";  
3579 AND the ACE2 is currently valid.

3580 3) When Client authentication uses a certificate credential;  
3581 AND one of the "roleid" values contained in the role certificate matches the "roleid" Property  
3582 of the ACE2 "oic.sec.roletype";  
3583 AND the role certificate public key matches the public key of the certificate used to establish  
3584 the current secure session;  
3585 AND the resource of the request matches one of the array elements of the "resources"  
3586 Property of the ACE2 "oic.sec.ace2.resource-ref";  
3587 AND the ACE2 is currently valid.

3588 4) When Client authentication uses a certificate credential;  
3589 AND the CoAP payload query string of the request specifies a role, which is member of the  
3590 set of roles contained in the role certificate;  
3591 AND the roleid values contained in the role certificate matches the "roleid" Property of the  
3592 ACE2 "oic.sec.roletype";  
3593 AND the role certificate public key matches the public key of the certificate used to establish  
3594 the current secure session;  
3595 AND the resource of the request matches one of the resources Property of the ACE2  
3596 "oic.sec.ace2.resource-ref";  
3597 AND the ACE2 is currently valid.

3598 5) When Client authentication uses a symmetric key credential;  
3599 AND one of the "roleid" values associated with the symmetric key credential used in the  
3600 secure session, matches the "roleid" Property of the ACE2 "oic.sec.roletype";  
3601 AND the resource of the request matches one of the array elements of the "resources"  
3602 Property of the ACE2 "oic.sec.ace2.resource-ref";  
3603 AND the ACE2 is currently valid.

3604 6) When Client authentication uses a symmetric key credential;  
3605 AND the CoAP payload query string of the request specifies a role, which is contained in the  
3606 "oic.r.cred.creds.roleid" Property of the current secure session;  
3607 AND CoAP payload query string of the request specifies a role that matches the "roleid"  
3608 Property of the ACE2 "oic.sec.roletype";  
3609 AND the resource of the request matches one of the array elements of the "resources"  
3610 Property of the ACE2 "oic.sec.ace2.resource-ref";  
3611 AND the ACE2 is currently valid.

3612 A request is granted if ANY of the 'matching' ACE2 entries contain the permission to allow the  
3613 request. Otherwise, the request is denied.

3614 There is no way for an ACE2 entry to explicitly deny permission to a resource. Therefore, if one  
3615 Device with a given role should have slightly different permissions than another Device with the  
3616 same role, they must be provisioned with different roles.

3617 The Server is required to verify that any hosted Resource has authorized access by the Client  
3618 requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the  
3619 Resource request processing should be applied securely and efficiently. See Annex A for  
3620 example.

### 3621 **13.6 Access Manager ACL Resource**

3622 "oic.r.amacl" Resource is defined in Table 48.

3623

**Table 48 – Definition of the "oic.r.amacl" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/amacl	Managed ACL	oic.r.amacl	baseline	Resource for managing access	Security

3624 Table 49 defines the Properties of "oic.r.amacl".

3625

**Table 49 – Properties of the "oic.r.amacl" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Resources	resources	oic.sec.ace2.resource-ref	array	RW	Yes	Multiple links to this host's Resources

3626 The AMS should be used to centralize management of access policy, but requires Servers to  
 3627 open a connection to the AMS whenever the named Resources are accessed. See A.2 for  
 3628 example.

3629 **13.7 Signed ACL Resource**

3630 "oic.r.sacl" Resource is defined in Table 50.

3631

**Table 50 – Definition of the "oic.r.sacl" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/sacl	Signed ACL	oic.r.sacl	baseline	Resource for managing access	Security

3632 Table 51 defines the Properties of "oic.r.sacl".

3633

**Table 51 – Properties of the "oic.r.sacl" Resource**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	State	Description
ACE List	aclist2	oic.sec.ace2	array	Yes	N/A	N/A	Access Control Entries in the ACL Resource
					N/A	RESET	Server shall set to manufacturer defaults.
					N/A	RFOTM	Set by DOTS after successful OTM
					N/A	RFPRO	The AMS (referenced via 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

3634 Table 52 defines the Properties of "oic.sec.sigtype".

3635 **Table 52 – Properties of the "oic.sec.sigtype" Property**

Property Title	Property Name	Value Type	Value Rule	Unit	Access Mode	Mandatory	Description
Signature Type	sigtype	String	N/A	N/A	RW	Yes	The string specifying the predefined signature format. "oic.sec.sigtype.jws" – IETF RFC 7515 JSON web signature (JWS) object "oic.sec.sigtype.pk7" – IETF RFC 2315 base64-encoded object "oic.sec.sigtype.cws" – CBOR-encoded JWS object
Signature Value	sigvalue	String	N/A	N/A	RW	Yes	The encoded signature

3636 **13.8 Provisioning Status Resource**

3637 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning  
3638 should be Client-directed or Server-directed. Client-directed provisioning relies on a Client device  
3639 to determine what, how and when Server Resources should be instantiated and updated. Server-  
3640 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-  
3641 directed provisioning depends on configuration of the rowneruuid Property of the "/oic/sec/doxm",  
3642 "/oic/sec/cred" and "/oic/sec/acl2" Resources to identify the device ID of the trusted DOTS, CMS  
3643 and AMS services respectively. Furthermore, the "/oic/sec/cred" Resource should be provisioned  
3644 at ownership transfer with credentials necessary to open a secure connection with appropriate  
3645 support service.

3646 "oic.r.pstat" Resource is defined in Table 53.

3647 **Table 53 – Definition of the "oic.r.pstat" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/pstat	Provisioning Status	oic.r.pstat	baseline	Resource for managing Device provisioning status	Configuration

3648 Table 54 defines the Properties of "oic.r.pstat".



Table 54 – Properties of the "oic.r.pstat" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	N/A	Yes	RW		Device Onboarding State
Is Device Operational	isop	Boolean	T F	Yes	R	RESET	Server shall set to FALSE
					R	RFOTM	Server shall set to FALSE
					R	RFPRO	Server shall set to FALSE
					R	RFNOP	Server shall set to TRUE
					R	SRESET	Server shall set to FALSE
Current Mode	cm	oic.sec.dpmttype	bitmask	Yes	R		Current Mode
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes	RW		Target Mode
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	rowneruuid	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 rowneruuid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruuid 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 does not refer to a valid DOTS the Server shall transition to RESET Device state.

3650 The provisioning status Resource "/oic/sec/pstat" is used to enable Devices to perform self-  
3651 directed provisioning. Devices are aware of their current configuration status and a target  
3652 configuration objective. When there is a difference between current and target status, the Device

3653 should consult the rowneruid Property of "/oic/sec/cred" Resource to discover whether any  
 3654 suitable provisioning services exist. The Device should request provisioning if configured to do so.  
 3655 The om Property of "/oic/sec/pstat" Resource will specify expected Device behaviour under these  
 3656 circumstances.

3657 Self-directed provisioning enables Devices to function with greater autonomy to minimize  
 3658 dependence on a central provisioning authority that should be a single point of failure in the OCF  
 3659 Security Domain.

3660 Table 55 defines the Properties of "/oic/sec/dostype".

3661 **Table 55 – 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

3662 In all Device states:

3663 – An authenticated and authorised Client may change the Device state of a Device by updating  
 3664 pstat.dos.s to the desired value. The allowed Device state transitions are defined in  
 3665 Figure 27.

3666 – Prior to updating "pstat.dos.s", the Client configures the Device to meet entry conditions for  
 3667 the new Device state. The SVR definitions define the entity (Client or Server) expected to  
 3668 perform the specific SVR configuration change to meet the entry conditions. Once the Client  
 3669 has configured the aspects for which the Client is responsible, it may update "pstat.dos.s".  
 3670 The Server then makes any changes for which the Server is responsible, including updating  
 3671 required SVR values, and set pstat.dos.s to the new value.

3672 – The "pstat.dos.p" Property is read-only by all Clients.

3673 – The Server sets "pstat.dos.p" to TRUE before beginning the process of updating "pstat.dos.s",  
 3674 and sets it back to FALSE when the "pstat.dos.s" change is completed.

3675 Any requests to update "pstat.dos.s" while "pstat.dos.p" is TRUE are denied.

3676 When Device state is RESET:

3677 – All SVR content is removed and reset to manufacturer default values.

3678 – The default manufacturer Device state is RESET.

3679 – NCRs are reset to manufacturer default values.

3680 – NCRs shall not be accessible.

3681 – After successfully processing RESET the SRM transitions to RFOTM by setting "s" Property of  
3682 "/oic/sec/dostype" Resource to RFOTM.

3683 When Device state is RFOTM:

3684 – NCRs shall not be accessible.

3685 – Before OTM is successful, the deviceuuid Property of "/oic/sec/doxm" Resource shall be set  
3686 to a temporary non-repeated value as defined in clauses 13.2 and 13.16.

3687 – Before OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-only by  
3688 unauthenticated requestors

3689 – After the OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-write by  
3690 authorized requestors.

3691 – The negotiated Device OC is used to create an authenticated session over which the DOTS  
3692 directs the Device state to transition to RFPRO.

3693 – If an authenticated session cannot be established the ownership transfer session should be  
3694 disconnected and SRM sets back the Device state to RESET state.

3695 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the  
3696 SRM asserts the OTM failed, should be disconnected, and transitions to RESET  
3697 ("/pstat.dos.s"=RESET).

3698 – The DOTS UPDATES the "devowneruuid" Property in the "/doxm" Resource to a non-nil UUID  
3699 value. The DOTS (or other authorized client) may update it multiple times while in RFOTM. It  
3700 is not updatable while in other device states except when the Device state returns to RFOTM  
3701 through RESET.

3702 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done,  
3703 the DOTS UPDATES the "owned" Property in the "/doxm" Resource to "true".

3704 When Device state is RFPRO:

3705 – The s Property of "/oic/sec/dostype" Resource is read-only by unauthorized requestors and  
3706 read-write by authorized requestors.

3707 – NCRs shall not be accessible, except for Easy Setup Resources, if supported.

3708 – The OCF Server may re-create NCRs.

3709 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.

3710 – An authorized Client may perform consistency checks on SVRs to determine which shall be  
3711 re-provisioned.

3712 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if  
3713 the Device has already transitioned from SRESET but consistency checks continue to fail.

3714 – The authorized Client sets the "/pstat.dos.s"=RFNOP.

3715 When Device state is RFNOP:

3716 – The "/pstat.dos.s" Property is read-only by unauthorized requestors and read-write by  
3717 authorized requestors.

3718 – NCRs, SVRs and core Resources are accessible following normal access processing.

3719 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or  
3720 RESET.

3721 When Device state is SRESET:

3722 – NCRs shall not be accessible. The integrity of NCRs may be suspect but the SRM doesn't  
3723 attempt to access or reference them.

- 3724 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These  
3725 include devowneruuid Property of the "/oic/sec/doxm" Resource,  
3726 "creds":[{"subjectuuid":<devowneruuid>},...] Property of the "/oic/sec/cred" Resource and  
3727 s Property of the "/oic/sec/dostype" Resource of "/oic/sec/pstat" Resource.
- 3728 – The certificates that identify and authorize the Device owner are sufficient to re-create  
3729 minimalist "/cred" and "/doxm" resources enabling Device owner control of SRESET. If the  
3730 SRM can't establish these Resources, then it will transition to RESET state.
- 3731 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as  
3732 needed to ensure they are available for continued provisioning in RFPRO or for normal  
3733 functioning in RFNOP.
- 3734 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING  
3735 "dos.s" Property of the "/pstat" Resource with RFPRO or RFNOP values
- 3736 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to  
3737 Device owner privileges.
- 3738 – The SRM asserts a Client-directed operational mode (e.g. "/pstat.om"=CLIENT\_DIRECTED).  
3739 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.  
3740 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning  
3741 mode without selecting any particular value.
- 3742 "oic.sec.dpmttype" is defined in Table 56.

3743 **Table 56 – Definition of the "oic.sec.dpmttype" Property**

Type Name	Type URN	Description
Device Provisioning Mode	oic.sec.dpmttype	Device provisioning mode is a 16-bit bitmask describing various provisioning modes

3744 Table 57 and Table 58 define the values of "oic.sec.dpmttype".

3745 **Table 57 – Value Definition of the "oic.sec.dpmttype" Property (Low-Byte)**

Value	Device Mode	Description
bx0000,0001 (1)	Deprecated	
bx0000,0010 (2)	Deprecated	
bx0000,0100 (4)	Deprecated	
bx0000,1000 (8)	Deprecated	
bx0001,0000 (16)	Deprecated	
bx0010,0000 (32)	Deprecated	
bx0100,0000 (64)	Initiate Software Version Validation	Software version validation requested/pending (1) Software version validation complete (0) Requires software download to verify integrity of software package
bx1000,0000 (128)	Initiate Secure Software Update	Secure software update requested/pending (1) Secure software update complete (0)

3746 **Table 58 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte)**

Value	Device Mode	Description
bx0000,0001 (1)	Initiate Software Availability Check	Checks if new software is available on remote endpoint. Does not require to download software. Methods used are out of bound.

Bits 2-8	<Reserved>	Reserved for later use
----------	------------	------------------------

3747 The *provisioning operation mode* type is an 8-bit mask enumerating the various provisioning  
3748 operation modes.

3749 "oic.sec.pomtype" is defined in Table 59.

3750 **Table 59 – Definition of the "oic.sec.pomtype" Property**

Type Name	Type URN	Description
Device Provisioning OperationMode	oic.sec.pomtype	Device provisioning operation mode is a 8-bit bitmask describing various provisioning operation modes

3751 Table 60 defines the values of "oic.sec.pomtype".

3752 **Table 60 – Value Definition of the "oic.sec.pomtype" Property**

Value	Operation Mode	Description
bx0000,0001 (1)	Server-directed utilizing multiple provisioning services	Provisioning related services are placed in different Devices. Hence, a provisioned Device should establish multiple DTLS sessions for each service. This condition exists when bit 0 is FALSE.
bx0000,0010 (2)	Server-directed utilizing a single provisioning service	All provisioning related services are in the same Device. Hence, instead of establishing multiple DTLS sessions with provisioning services, a provisioned Device establishes only one DTLS session with the Device. This condition exists when bit 0 is TRUE.
bx0000,0100 (4)	Client-directed provisioning	Device supports provisioning service control of this Device's provisioning operations. This condition exists when bit 1 is TRUE. When this bit is FALSE this Device controls provisioning steps.
bx0000,1000(8) – bx1000,0000(128)	<Reserved>	Reserved for later use
bx1111,11xx	<Reserved>	Reserved for later use

### 3753 13.9 Certificate Signing Request Resource

3754 The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be  
3755 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC  
3756 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of  
3757 "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr"  
3758 Resource.

3759 "oic.r.csr" Resource is defined in Table 61.

3760 **Table 61 – Definition of the "oic.r.csr" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/csr	Certificate Signing Request	oic.r.csr	baseline	The CSR resource contains a Certificate Signing Request for the Device's public key.	Configuration

3761 Table 62 defines the Properties of "oic.r.csr".

**Table 62 – Properties of the "oic.r.csr" Resource**

Property Title	Property Name	Value Type	Access Mode	Mandatory	Description
Certificate Signing Request	csr	String	R	Yes	Contains the signed CSR encoded according to the encoding Property
Encoding	encoding	String	R	Yes	A string specifying the encoding format of the data contained in the csr Property "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate signing request "oic.sec.encoding.der" – Encoding for DER-encoded certificate signing request

3763 The Device chooses which public key to use, and may optionally generate a new key pair for this  
3764 purpose.

3765 In the CSR, the Common Name component of the Subject Name shall contain a string of the  
3766 format "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122.  
3767 The Common Name, and other components of the Subject Name, may contain other data. If the  
3768 Device chooses to include additional information in the Common Name component, it shall delimit  
3769 it from the UUID field by white space, a comma, or a semicolon.

3770 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to  
3771 generate a new key pair, the Device may begin generation of a key pair as a result of a  
3772 RETRIEVE of this resource. If the Device cannot immediately respond to the RETRIEVE request  
3773 due to time required to generate a key pair, the Device shall return an "operation pending" error.  
3774 This indicates to the Client that the Device is not yet ready to respond, but will be able at a later  
3775 time. The Client should retry the request after a short delay.

### 3776 13.10 Roles Resource

3777 The roles Resource maintains roles that have been asserted with role certificates, as described in  
3778 clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role  
3779 certificate. Servers shall only grant access to the roles information associated with the public key  
3780 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.  
3781 See 10.4.2 for how role certificates are validated.

3782 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS  
3783 session with a Client, if is not already created. The roles Resource shall only expose a secured  
3784 OCF Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long  
3785 as the (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least  
3786 until the certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of  
3787 clause 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A  
3788 Server should regularly inspect the contents of the roles resource and purge contents based on a  
3789 policy it determines based on its resource constraints. For example, expired certificates, and  
3790 certificates from Clients that have not been heard from for some arbitrary period of time could be  
3791 candidates for purging.

3792 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In  
3793 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall  
3794 behave as follows. Unlisted operations are implementation specific and not reliable.

3795 1) A RETRIEVE request shall return all previously asserted roles associated with the currently  
3796 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query  
3797 parameter is not supported; all previously asserted roles associated with the currently  
3798 connected and authenticated Client's identity are returned.

- 3799 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties  
3800 included in the array as follows:
- 3801 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the  
3802 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid"  
3803 value.
- 3804 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,  
3805 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed  
3806 response and a duplicate entry shall not be added to the array.
- 3807 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored  
3808 by the Server. The Server shall assign a unique "credid" value for every entry of the  
3809 "roles" array.
- 3810 3) A DELETE request without a "credid" query parameter shall remove all entries from the  
3811 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
3812 Client's identity.
- 3813 4) A DELETE request with a "credid" query parameter shall remove only the entries of the  
3814 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
3815 Client's identity and where the corresponding "credid" matches the entry.
- 3816 NOTE The "oic.r.roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3817 in ISO/IEC 30118-1:2018.
- 3818 "oic.r.roles" Resource is defined in Table 63.

3819 **Table 63 – Definition of the "oic.r.roles" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/roles	Roles	oic.r.roles	baseline	Resource containing roles that have previously been asserted to this Server	Security

3820 Table 64 defines the Properties of "oic.r.roles".

3821 **Table 64 – Properties of the "oic.r.roles" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Roles	roles	oic.sec.cred	array	RW	Yes	List of roles previously asserted to this Server

3822 Because "oic.r.roles" shares the "oic.sec.cred" schema with "oic.r.cred", "subjectuud" is a required Property. However,  
3823 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property  
3824 is contained in an UPDATE request to the "/oic/sec/roles" Resource.

### 3825 13.11 Account Resource

3826 The Account Resource specifies the Properties based on IETF RFC 6749 Access Token based  
3827 account creation. The mechanism to obtain credentials is described in clause 7.5. The Account  
3828 Resource is used for Device Registration. The Account Resource is instantiated on the OCF  
3829 Cloud as "oic/sec/account" SVR and is used by cloud-enabled Devices to register with the OCF  
3830 Cloud. It should be only accessible on a secure channel; non-secure channel should not be able  
3831 access this Resource.

3832 During the Device Registration process, an OCF Cloud can provide a distinct URI of another OCF  
3833 Cloud ("redirected-to" OCF Cloud). Both initial and redirected-to OCF Clouds are expected to  
3834 belong to the same Vendor; they are assumed to have the same UUID and are assumed to have  
3835 an out-of-band communication mechanism established. Device does not have to perform the  
3836 Device Registration on the redirected-to OCF Cloud and the OCF Cloud may ignore such

3837 attempts. Redirected-to OCF Cloud is expected to accept the Access Token, provided to the  
 3838 Device by the initial OCF Cloud.

3839 The "di", "uid", "refresh token" and "access token" Properties of the Account Resource should be  
 3840 securely stored as described in clause 15.

3841 The RETRIEVE operation on OCF Cloud's "/oic/sec/account" Resource is not allowed and the  
 3842 OCF Cloud is expected to reject all attempts to perform such operation.

3843 The UPDATE operation on the OCF Cloud's "/oic/sec/account" Resource behaves as follows:

3844 – A Device intending to register with the OCF Cloud shall send UPDATE with following  
 3845 Properties "di" ("di" Property Value of "/oic/d" Resource), and "access token" as configured by  
 3846 the Mediator ("at" Property Value of "oic.r.coapcloudconf" Resource). The OCF Cloud verifies  
 3847 it is the same "access token" which was assigned to the Mediator for the corresponding "di"  
 3848 Property Value. The "access token" is the permission for the Device to access the OCF Cloud.  
 3849 If the "apn" was included when the Mediator UPDATED the "oic.r.coapcloudconf" Resource,  
 3850 the Device shall also include "auth provider" Property when registering with the OCF Cloud. If  
 3851 no "apn" is specified, then the "auth provider" Property shall not be included in the UPDATE  
 3852 request.

3853 – OCF Cloud returns "access token", "uid", "refresh token", "expires in" It may also return  
 3854 "redirect uri". Received "access token" is to be treated by Device as an Access Token with  
 3855 "Bearer" token type as defined in IETF RFC 6750. This "access token" shall be used for the  
 3856 following Account Session start using "oic/sec/session" SVR. Received "refresh token" is to be  
 3857 treated by Device as a Refresh Token as defined in IETF RFC 6749. The Device stores the  
 3858 OCF Cloud's Response values. If "redirect uri" is received, Device shall use received value as  
 3859 a new OCF Cloud URI instead of "cis" Property Value of "oic.r.coapcloudconf" Resource for  
 3860 further connections.

3861 The DELETE operation on the OCF Cloud's "/oic/sec/account" Resource should behave as  
 3862 follows:

3863 – To deregister with the OCF Cloud, a DELETE operation shall be sent with the "access token"  
 3864 and either "uid", or "di" to be deregistered with the OCF Cloud. On DELETE with the OCF  
 3865 Cloud, the Device should also delete values internally stored. Once deregister with an OCF  
 3866 Cloud, Device can connect to any other OCF Cloud. Device deregistered need to go through  
 3867 the steps in 7.5 again to be registered with the OCF Cloud.

3868 "oic.r.account" Resource is defined in Table 65.

3869 **Table 65 – Definition of the "oic.r.account" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/account	Account	oic.r.account	oic.if.baseline	Resource used for a device to add itself under a given credential	N/A

3870 Table 66 defines the Properties of "oic.r.account".

3871 **Table 66 – Properties of the "oic.r.account" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Device ID	di	string	uuid	W	Yes	Unique Device identifier
Auth Provider	authprovider	string	N/A	W	No	The name of Authorization Provider through which Access Token was obtained.



Access-Token	accesstoken	string	Non-empty string	RW	Yes	Access-Token used for communication with OCF Cloud after account creation
Refresh Token	refreshtoken	string	Non-empty string	R	Yes	Refresh token can be used to refresh the Access Token before getting expired
Token Expiration	expiresin	integer	-	R	Yes	Access-Token life time in seconds (-1 if permanent)
User ID	uid	string	uuid	R	Yes	Unique OCF Cloud User identifier
Redirect URI	redirecturi	string	-	R	No	Using this URI, the Client needs to reconnect to a redirected OCF Cloud. If provided, this value shall be used by the Device instead of Mediator-provided URI during the Device Registration.

3872 **13.12 Account Session Resource**

3873 The "/oic/sec/session" Resource hosted on the OCF Cloud is used for creating connections with  
 3874 the OCF Cloud subsequent to Device registration though "/oic/sec/account" Resource. The  
 3875 "/oic/sec/session" Resource requires the device ID, User ID and Access Token which are stored  
 3876 securely on the Device.

3877 The "/oic/sec/session" Resource is exposed by the OCF Cloud. It should be only accessible on a  
 3878 secure channel; non-secure channel cannot access this Resource.

3879 The RETRIEVE operation on OCF Cloud's "/oic/sec/session" Resource is not allowed and the  
 3880 OCF Cloud is expected to reject all attempts to perform such operation.

3881 The UPDATE operation is defined as follows for OCF Cloud's "/oic/sec/session" Resource:

- 3882 – The Device connecting to the OCF Cloud shall send an UPDATE request message to the OCF  
 3883 Cloud's "/oic/sec/session" Resource. The message shall include the "di" Property Value of  
 3884 "/oic/d" Resource and "uid", "login" Value ("true" to establish connection; "false" to disconnect)  
 3885 and "accesstoken" as returned by OCF Cloud during Device Registration. The OCF Cloud  
 3886 verifies it is the same Access Token which was returned to the Device during Device  
 3887 Registration process. If Device was attempting to establish the connection and provided  
 3888 values were verified as correct by the OCF Cloud, OCF Cloud sends a response with  
 3889 remaining lifetime of the associated Access Token ("expiresin" Property Value).

3890 "/oic.r.session" Resource is defined in Table 67.

3891 **Table 67 – Definition of the "oic.r.session" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/session	Account Session	oic.r.session	oic.if.baseline	Resource that enables a device to manage its session using login or logout	N/A

3892 Table 68 defines the Properties of "oic.r.session".

3893 **Table 68 – Properties of the "oic.r.session" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
User ID	uid	string	uuid	W	Yes	User ID which provided by Device Registration process

Device ID	di	string	uuid	W	Yes	Unique device id registered for a Device
Access Token	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

3894 **13.13 Account Token Refresh Resource**

3895 The "/oic/sec/tokenrefresh" Resource is used by the Device for refreshing the Access Token.

3896 The "/oic/sec/tokenrefresh" Resource is hosted by the OCF Cloud. It should be only accessible  
3897 on a secure channel; non-secure channel cannot access this Resource.

3898 The Device should use "/oic/sec/tokenrefresh" to refresh the Access Token with the OCF Cloud,  
3899 when the time specified in "expiresin" is near.

3900 The RETRIEVE operation on OCF Cloud's "/oic/sec/ tokenrefresh" Resource is not allowed and  
3901 the OCF Cloud is expected to reject all attempts to perform such operation.

3902 The UPDATE operation is defined as follows for "/oic/sec/tokenrefresh" Resource

3903 – The Device attempting to refresh the Access Token shall send an UPDATE request message  
3904 to the OCF Cloud's "/oic/sec/tokenrefresh" Resource. The message shall include the "di"  
3905 Property Value of "/oic/d" Resource, "uid" and "refreshtoken", as returned by OCF Cloud.

3906 – OCF Cloud response is expected to include a "refreshtoken", new "acesstoken", and  
3907 "expiresin". Received "acesstoken" is to be treated by Device as an Access Token with  
3908 "Bearer" token type as defined in IETF RFC 6750. This Access Token is the permission for  
3909 the Device to access the OCF Cloud. Received "refreshtoken" is to be treated by Device as a  
3910 Refresh Token as defined in IETF RFC 6749. Received "refreshtoken" may be the new  
3911 Refresh Token or the same one as provided by the Device in the UPDATE request. In case  
3912 when new distinct "refreshtoken" is provided by the OCF Cloud, the Device shall discard the  
3913 old value. The OCF Cloud's response values "refreshtoken", "acesstoken" and "expiresin" are  
3914 securely stored on the Device.

3915 "/oic.r.tokenrefresh" Resource is defined in Table 69.

3916 **Table 69 – Definition of the "oic.r.tokenrefresh" Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/tokenrefresh	Token Refresh	oic.r.tokenrefresh	oic.if.baseline	Resource to manage the access-token using refresh token	N/A

3917 Table 70 defines the Properties of "oic.r.tokenrefresh".

**Table 70 – Properties of the "oic.r.tokenrefresh" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
User ID	uid	string	uuid	W	Yes	User ID which provided by Sign-up process
Device ID	di	string	uuid	W	Yes	Unique device id registered for an OCF Cloud User account
Refresh Token	refreshtoken	string	A string of at least one character	RW	Yes	Refresh token received by account management or during token refresh procedure
Access Token	accesstoken	string	A string of at least one character	R	Yes	Granted Access-Token
Token Expiration	expiresin	integer	-	R	Yes	Access-Token life time in seconds (-1 if permanent)

### 3919 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3920 The SVRs expose the security-related Properties of the Device.

3921 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to  
3922 unauthenticated (anonymous) Clients could create privacy or security concerns.

3923 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for  
3924 the "oic.r.doxm" Resource to anonymous requesters, so that the Device can be discovered and  
3925 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the "oic.r.doxm"  
3926 Resource to anonymous requesters, to preserve privacy.

### 3927 **13.15 SVRs, Discoverability and OCF Endpoints**

3928 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy  
3929 Parameter clause 7.8.2.1.2).

3930 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS)  
3931 (reference ISO/IEC 30118-1:2018, clause 10).

3932 The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM  
3933 (reference ISO/IEC 30118-1:2018, clause 10).

### 3934 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

#### 3935 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

3936 Unique identifiers are a privacy consideration due to their potential for being used as a tracking  
3937 mechanism. These include the following Resources and Properties:

- 3938 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3939 – "/oic/p" Resource containing the "pi" Property.
- 3940 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3941 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous  
3942 requestors. This implies any Client Device, including those with malicious intent, are able to  
3943 reliably obtain identifiers useful for building a log of activity correlated with a specific Platform  
3944 and Device.

3945 There are two strategies for privacy protection of Devices:

- 3946 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers  
3947 2) Limit identifier persistence to make it impractical for tracking use.  
3948 Both techniques can be used effectively together to limit exposure to privacy attacks.

- 3949 1) A Platform / Device manufacturer should specify a default ACL policy that restricts  
3950 anonymous requestors from accessing unique identifiers. An OCF Security Domain owner  
3951 should modify the ACL policy to grant access to authenticated Devices who, presumably, do  
3952 not present a privacy threat.
- 3953 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the  
3954 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and  
3955 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated  
3956 identifiers shall be:
- 3957 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
  - 3958 b) Generated by a function that is pre-image resistant, second pre-image resistant and  
3959 collision resistant

3960 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier  
3961 used to begin the onboarding process. However, attackers could obtain the value too and use it  
3962 for Device tracking throughout the Device's lifetime.

3963 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the  
3964 deviceuuid Property of the "/oic/sec/doxm" Resource to unauthenticated "/oic/res" and  
3965 "/oic/sec/doxm" Resource RETRIEVE requests when the devowneruuid Property of  
3966 "/oic/sec/doxm" Resource is the nil-UUID. The Server shall expose a new temporary non-  
3967 repeated deviceuuid Property of the "/oic/sec/doxm" Resource when the device state transitions  
3968 to RESET. This ensures the deviceuuid Property of the "/oic/sec/doxm" cannot be used to track  
3969 across multiple owners.

3970 The devowneruuid Property of "/oic/sec/doxm" Resource is initialized to the nil-UUID upon  
3971 entering RESET; which is retained until being set to a non-nil-UUID value during RFOTM device  
3972 state. The device shall supply a temporary, non-repeated deviceuuid Property of "/oic/sec/doxm"  
3973 Resource to RETRIEVE requests on "/oic/sec/doxm" and "/oic/res" Resources while  
3974 devowneruuid Property of "/oic/sec/doxm" Resource is the nil-UUID. During the OTM process the  
3975 DOTS shall UPDATE devowneruuid Property of the "/oic/sec/doxm" Resource to a non-nil UUID  
3976 value which is the trigger for the Device to expose its persistent or semi-persistent device  
3977 identifier. Therefore, the Device shall supply deviceuuid Property of "/oic/sec/doxm" Resource in  
3978 response to RETRIEVE requests while the devowneruuid Property of the "/oic/sec/doxm"  
3979 Resource is a non-nil-UUID value.

3980 The DOTS or AMS may also provision an ACL policy that restricts access to the "/oic/sec/doxm"  
3981 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent  
3982 device identifier via the deviceuuid Property value of the "/oic/sec/doxm" Resource.

3983 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent  
3984 or semi-persistent identifier using the "/oic/sec/cred" Resource to first establish an authenticated  
3985 connection. This is achieved by first provisioning a "/oic/sec/cred" Resource entry that contains  
3986 the Server's deviceuuid Property value of the "/oic/sec/doxm" Resource.

3987 The "di" Property in the "/oic/d" Resource shall mirror that of the deviceuuid Property of the  
3988 "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the  
3989 "/oic/d" resource such that only authenticated Clients are able to obtain the "di" Property of  
3990 "/oic/d" Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

3991 Servers should expose a temporary, non-repeated, piid Property of "/oic/p" Resource Value upon  
3992 entering RESET Device state. Servers shall expose a persistent value via the "piid" Property of

3993 "/oic/p" Property when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. An ACL  
 3994 policy on the "/oic/d" Resource should protect the "piid" Property of "/oic/p" Resource from being  
 3995 disclosed to unauthenticated requestors.

3996 Servers shall expose a temporary, non-repeated, "pi" Property value upon entering RESET  
 3997 Device state. Servers shall expose a persistent or semi-persistent platform identifier value via the  
 3998 "pi" Property of the "/oic/p" Resource when onboarding sets "devowneruuid" Property to a non-  
 3999 nil-UUID value. An ACL policy on the "/oic/p" Resource should protect the "pi" Property from  
 4000 being disclosed to unauthenticated requestors.

4001 Table 71 depicts Core Resource Properties Access Modes given various Device States.

4002 **Table 71 – Core Resource Properties Access Modes given various Device States**

Resource Type	Property title	Property name	Value type	Access Mode		Behaviour
oic.wk.p	Platform ID	pi	oic.types-schema.uuid	All States	R	Server shall construct a temporary random UUID (The temporary value shall not overwrite the persistent pi internally). Server sets to its persistent value after secure Owner Transfer session is established.
oic.wk.d	Protocol Independent Identifier	piid	oic.types-schema.uuid	All States	R	Server should construct a temporary random UUID when entering RESET state.
oic.wk.d	Device Identifier	di	oic.types-schema.uuid	All states	R	/d di shall mirror the value contained in "/doxm" deviceuuid in all device states.

4003 Four identifiers are thought to be privacy sensitive:

- 4004 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 4005 – "/oic/p" Resource containing the "pi" Property.
- 4006 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

4007 There are three strategies for privacy protection of Devices:

- 4008 1) Apply access control to restrict read access to Resources containing unique identifiers. This  
 4009 ensures privacy sensitive identifiers do not leave the Device.
- 4010 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy  
 4011 sensitive identifiers are less effective for tracking and correlation.
- 4012 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can  
 4013 do so.

4014 These techniques can be used to limit exposure to privacy attacks. For example:

- 4015 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent  
 4016 identifiers can be created.
- 4017 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to  
 4018 facilitate onboarding.
- 4019 – Persistent and semi-persistent identifiers can be encrypted before sending them to another  
 4020 Device.

4021 A temporary, non-repeated identifier shall be:

- 4022 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 4023 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
4024 resistant

4025 NOTE This requirement is met through a vendor attestation certification mechanism.

### 4026 **13.16.2 Privacy Protecting the Device Identifiers**

4027 The "di" Property Value of the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of  
4028 the "/oic/sec/doxm" Resource. The Device should use a new, temporary non-repeated identifier in  
4029 place of the "deviceuuid" Property Value of "/oic/sec/doxm" Resource upon entering the RESET  
4030 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID  
4031 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value  
4032 of the "/oic/sec/doxm" Resource after the DOTS sets the "devowneruuid" Property to a non-nil-  
4033 UUID value. The temporary identifier should not change more frequently than once per Device  
4034 state transition to RESET.

4035 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4036 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"  
4037 Property values:
  - 4038 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di")  
4039 Property value only if responding to an authenticated requestor and the "deviceuuid" (or  
4040 "di") value is confidentiality protected .
  - 4041 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if  
4042 responding to an unauthenticated requestor.
- 4043 – The AMS should provision an ACL policy on the "/oic/sec/doxm" and "/oic/d" resources to  
4044 further protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

4045 See 13.2 for deviceuuid Property lifecycle requirements.

4046 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary  
4047 discovery requests. This is achieved by provisioning a "/oic/sec/cred" Resource entry that contains the Server's  
4048 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

### 4049 **13.16.3 Privacy Protecting the Protocol Independent Device Identifier**

4050 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property  
4051 Value of "/oic/d" Resource upon entering the RESET Device state. If a temporary, non-repeated  
4052 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID  
4053 value. The Device should use its persistent "piid" Property value after the DOTS sets the  
4054 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change  
4055 more frequently than once per Device state transition to RESET.

4056 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4057 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
  - 4058 – The Device should include its persistent "piid" Property value only if responding to an  
4059 authenticated requestor and the "piid" value is confidentiality protected.
  - 4060 – The Device should include a temporary non-repeated "piid" Property value if responding to  
4061 an unauthenticated requestor.
- 4062 – The AMS should provision an ACL policy on the "/oic/d" Resource to further protect the piid  
4063 Property of "/oic/p" Resource from being disclosed unnecessarily.

4064 **13.16.4 Privacy Protecting the Platform Identifier**

4065 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property  
4066 Value of the "/oic/p" Resource upon entering the RESET Device state. This value should be  
4067 exposed while the "devowneruuid" Property has a nil UUID value. The Device should use its  
4068 persistent (or semi-persistent) "pi" Property value after the DOTS sets the "devowneruuid"  
4069 Property to a non-nil-UUID value. The temporary identifier should not change more frequently  
4070 than once per Device state transition to RESET.

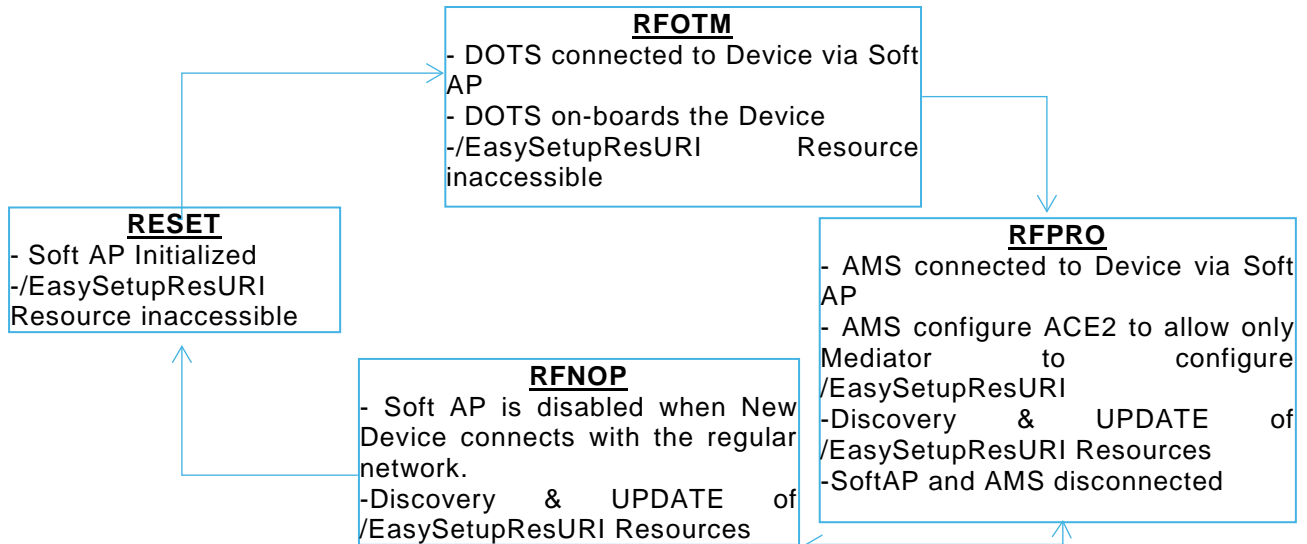
4071 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4072 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
  - 4073 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
  - 4074 responding to an authenticated requestor and the "pi" value is confidentiality protected.
  - 4075 – The Device should include a temporary non-repeated "pi" Property value if responding to
  - 4076 an unauthenticated requestor.
- 4077 – The AMS should provision an ACL policy on the "/oic/p" Resource to protect the pi Property
- 4078 from being disclosed unnecessarily.

4079 **13.17 Easy Setup Resource Device State**

4080 This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined  
4081 in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of  
4082 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup  
4083 Devices.

4084 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



4102 **Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states**

4103 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's  
4104 state.

4105 While it is reasonable for a user to expect that power cycling a new Device will turn on the Soft  
4106 AP for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it  
4107 is a security risk to make this the default behaviour of a device that remains unenrolled beyond a  
4108 reasonable period after first boot.

4109 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

- 4110 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes  
4111 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for  
4112 Easy Setup.
- 4113 – If a new Device tried and failed to complete Easy Setup Enrolment immediately following the  
4114 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically  
4115 for another 30 minutes upon being power cycled, provided that the power cycle occurs within  
4116 3 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup  
4117 Soft AP directly without a factory reset, it is not necessary to turn it back on if it was on  
4118 immediately prior to power cycle, because the user obviously knows how to initiate the  
4119 process manually.
- 4120 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft  
4121 AP should not turn back on for Easy Setup until another factory reset occurs, or the user  
4122 initiates the Easy Setup Soft AP directly.
- 4123 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new  
4124 Device to connect to the Enroller.
- 4125 – The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the  
4126 Enroller.
- 4127 – Once a new Device has successfully connected to the Enroller, it shall not turn the Easy  
4128 Setup Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or  
4129 the user initiates the Easy Setup Soft AP directly.
- 4130 – Just Works OTM shall not be enabled on Devices which support Easy Setup.
- 4131 – The Soft AP shall be secured (e.g. shall not expose an open AP).
- 4132 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase  
4133 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on  
4134 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.
- 4135 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the  
4136 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by  
4137 an attacker with knowledge of the Device type, model, manufacturer, or any other information  
4138 discoverable through Device's exposed interfaces.
- 4139 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the  
4140 "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the  
4141 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller,  
4142 for use in connecting the Enrollee to the Enroller.
- 4143 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the  
4144 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.
- 4145 The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET  
4146 state. After ownership transfer process is completed with the DOTS, and the Device enters in  
4147 RFPRO Device state, the "/example/EasySetupResURI" may be Discoverable. The DOTS may be  
4148 hosted on the Mediator Device.
- 4149 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership  
4150 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used  
4151 by AMS for "/oic/sec/acl2" Resource provisioning in RFPRO state. The CoAPS session  
4152 authentication and encryption is already defined in the Security spec.
- 4153 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following  
4154 Resources to be only configurable by the Mediator Device with permission to UPDATE or  
4155 RETRIEVE access:
- 4156 – "/example/EasySetupResURI"



4157 – "/example/WifiConfResURI"  
4158 – "/example/DevConfResURI"  
4159 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
4160 {  
4161     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
4162     "resources": [  
4163         { "href": "/example/EasySetupResURI" },  
4164         { "href": "/example/WiFiConfResURI" },  
4165         { "href": "/example/DevConfResURI" },  
4166     ],  
4167     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
4168 }
```

4169 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to  
4170 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

4171 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE  
4172 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

## 4173 **14 Security Hardening Guidelines/ Execution Environment Security**

### 4174 **14.1 Preamble**

4175 This is an informative clause. Many TGs in OCF have security considerations for their protocols  
4176 and environments. These security considerations are addressed through security mechanisms  
4177 specified in the security documents for OCF. However, effectiveness of these mechanisms  
4178 depends on security robustness of the underlying hardware and software Platform. This clause  
4179 defines the components required for execution environment security.

### 4180 **14.2 Execution Environment Elements**

#### 4181 **14.2.1 Execution Environment Elements General**

4182 Execution environment within a computing Device has many components. To perform security  
4183 functions in a robustness manner, each of these components has to be secured as a separate  
4184 dimension. For instance, an execution environment performing AES cannot be considered secure  
4185 if the input path entering keys into the execution engine is not secured, even though the  
4186 partitions of the CPU, performing the AES encryption, operate in isolation from other processes.  
4187 Different dimensions referred to as elements of the execution environment are listed below. To  
4188 qualify as a secure execution environment (SEE), the corresponding SEE element must qualify as  
4189 secure.

- 4190 – (Secure) Storage
- 4191 – (Secure) Execution engine
- 4192 – (Trusted) Input/output paths
- 4193 – (Secure) Time Source/clock
- 4194 – (Random) number generator
- 4195 – (Approved) cryptographic algorithms
- 4196 – Hardware Tamper (protection)

4197 **NOTE** Software security practices (such as those covered by OWASP) are outside scope of this document, as  
4198 development of secure code is a practice to be followed by the open source development community. This document  
4199 will however address the underlying Platform assistance required for executing software. Examples are secure boot  
4200 and secure software upgrade.

4201 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,  
4202 14.2.7.

#### 4203 **14.2.2 Secure Storage**

##### 4204 **14.2.2.1 Secure Storage General**

4205 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive  
4206 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,  
4207 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive  
4208 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both  
4209 its integrity and confidentiality be maintained.

4210 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive  
4211 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either  
4212 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication  
4213 and encryption, it must maintain its integrity against intentional or accidental alteration.

4214 A partial list of Sensitive Data is outlined in Table 72:

**Table 72 – Examples of Sensitive Data**

<b>Data</b>	<b>Integrity protection</b>	<b>Confidentiality protection</b>
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
OCF Cloud URL	Yes	Not required
OCF Cloud Identity	Yes	Not required
Access Token	Yes	Yes

4216 Exact method of protection for secure storage is implementation specific, but typically  
4217 combinations of hardware and software methods are used.

#### 4218 **14.2.2.2 Hardware Secure Storage**

4219 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric  
4220 and asymmetric private keys, access credentials, and personal private data. Hardware secure  
4221 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes  
4222 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

4223 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides  
4224 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or  
4225 electronic attacks. It is not necessary to prevent the attacks themselves, but an attempted attack  
4226 should not result in an unauthorized entity successfully retrieving Sensitive Data.

4227 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data  
4228 from attacks that include but are not limited to:

- 4229 1) Physical decapping of chip packages to optically read NVRAM contents
- 4230 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 4231 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit  
4232 patterns of Critical Sensitive Data
- 4233 4) Use of malicious software or firmware to read memory contents at rest or in transit within a  
4234 microcontroller
- 4235 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

#### 4236 **14.2.2.3 Software Storage**

4237 It is generally NOT recommended to rely solely on software and unsecured memory to store  
4238 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and  
4239 encryption keys should be housed in hardware secure storage whenever possible.

4240 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable  
4241 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

#### 4242 **14.2.2.4 Additional Security Guidelines and Best Practices**

4243 Some general practices that can help ensure that Sensitive Data is not compromised by various  
4244 forms of security attacks:

- 4245 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG  
4246 used for authentication challenges can substantially degrade security strength. For this  
4247 reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source  
4248 be used for all authentication challenges.
- 4249 2) Secure download and boot – To prevent the loading and execution of malicious software,  
4250 where it is practical, it is recommended that Secure Download and Secure Boot methods that  
4251 authenticate a binary's source as well as its contents be used.
- 4252 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered  
4253 unsecure and shall not be used for any security-related function:
  - 4254 a) SHA-1
  - 4255 b) MD5
  - 4256 c) RC4
  - 4257 d) RSA 1024
- 4258 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is  
4259 stored in Secure Storage, any use of that data that requires its transmission out of that  
4260 Secure Storage should be encrypted to prevent eavesdropping by malicious software within  
4261 an MCU/MPU.
- 4262 5) It is recommended to avoid using wildcard in Subject Id ("\*"), when setting up "oic.r.cred"  
4263 Resource entries, since this opens up an identity spoofing opportunity.
- 4264 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device  
4265 meets security requirements for its intended uses. As an example, IoTivity is a reference  
4266 implementation intended to be used as a basis for a product, but IoTivity has not undergone  
4267 3rd party security review, penetration testing, etc. Any Device based on IoTivity should  
4268 undergo appropriate penetration testing and security review prior to sale or deployment.
- 4269 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to  
4270 consumers. Changes should be made to a public and accessible website. Expectations  
4271 should be clear as to what will be supported and for how long the Device vendor expects to  
4272 support security updates to the software, operating system, drivers, networking, firmware and  
4273 hardware of the device.
- 4274 8) Device vendor has not implemented test or debug interfaces on the Device which are  
4275 operable or which can be enabled which might present an attack vector on the Device which  
4276 circumvents the interface-level security or access policies of the Device.
- 4277 9) Device vendor understands that if an application running on the Device has access to  
4278 cryptographic elements such as the private keys or Ownership Credential, then those  
4279 elements have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or  
4280 a Device with access to the Internet beyond the local network, the execution of critical  
4281 functions should take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4282 10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management  
4283 or access, or other security-critical function, should be sufficiently unique (do not duplicate  
4284 passphrases. The creation of these passphrases or PINS should not be algorithmically  
4285 deterministic nor should they use insufficient entropy in their creation.
- 4286 11) Ensure that there are no remaining "VENDOR\_TODO" items in the source code.

4287 12) If the implementation of this document uses the "Just Works" onboarding method, understand  
4288 that there is a man-in-the-middle vulnerability during the onboarding process where a  
4289 malicious party could intercept messages between the device being onboarded and the OBT  
4290 and could persist, acting as an intermediary with access to message traffic, during the lifetime  
4291 of that onboarded device. The recommended best practice would be to use an alternate  
4292 ownership transfer method (OTM) instead of "Just Works".

4293 13) It is recommended that at least one static and dynamic analysis tool<sup>1</sup> be applied to any  
4294 proposed major production release of the software before its release, and any vulnerabilities  
4295 resolved.

4296 14) To avoid a malicious device being able to covertly join an OCF Security Domain,  
4297 implementers of any OBT may eliminate completely autonomous sequences where a device  
4298 is brought into the OCF Security Domain without any authorization by the owner. Consider  
4299 either including a confirmation with the OCF Security Domain owner/operator (e.g. "Do you  
4300 want to add 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'? Yes/No/Cancel?") or a  
4301 confirmation with a security policy (e.g. an enterprise policy where the OCF Security Domain  
4302 admin can bulk-onboard devices).

### 4303 **14.2.3 Secure execution engine**

4304 Execution engine is the part of computing Platform that processes security functions, such as  
4305 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine  
4306 requires the following

4307 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This  
4308 includes isolation of CPU caches, and all of execution elements that needed to be considered  
4309 as part of trusted (crypto) boundary.

4310 – Isolation of data paths into and out of execution engine. For instance, both unencrypted but  
4311 sensitive data prior to encryption or after decryption, or cryptographic keys used for  
4312 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

### 4313 **14.2.4 Trusted input/output paths**

4314 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be  
4315 protected. This includes paths into and out secure execution engine and secure memory.

4316 Path protection can be both hardware based (e.g. use of a privileged bus) or software based  
4317 (using encryption over an untrusted bus).

### 4318 **14.2.5 Secure clock**

4319 Many security functions depend on time-sensitive credentials. Examples are time stamped  
4320 Kerberos tickets, OAUTH tokens, X.509 certificates, OSCP response, software upgrades, etc.  
4321 Lack of secure source of clock can mean an attacker can modify the system clock and fool the  
4322 validation mechanism. Thus an SEE needs to provide a secure source of time that is protected  
4323 from tampering. Trustworthiness from security robustness standpoint is not the same as accuracy.  
4324 Protocols such as NTP can provide rather accurate time sources from the network, but are not  
4325 immune to attacks. A secure time source on the other hand can be off by seconds or minutes  
4326 depending on the time-sensitivity of the corresponding security mechanism. Secure time source  
4327 can be external as long as it is signed by a trusted source and the signature validation in the  
4328 local Device is a trusted process (e.g. backed by secure boot).

### 4329 **14.2.6 Approved algorithms**

4330 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and  
4331 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by  
4332 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only

---

<sup>1</sup> A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4333 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic  
4334 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are  
4335 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms  
4336 (even if they deemed stronger by some parties) must be considered non-approved.

4337 The set of algorithms to be considered for approval are algorithms for

- 4338 – Hash functions
- 4339 – Signature algorithms
- 4340 – Encryption algorithms
- 4341 – Key exchange algorithms
- 4342 – Pseudo Random functions (PRF) used for key derivation

4343 This list will be included in this or a separate security robustness rules document and must be  
4344 followed for all security specifications within OCF.

#### 4345 **14.2.7 Hardware tamper protection**

4346 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not  
4347 requirements) regarding tamper protection for cryptographic module

- 4348 – Production-grade (lowest level): this means components that include conformal sealing  
4349 coating applied over the module's circuitry to protect against environmental or other physical  
4350 damage. This does not however require zeroization of secret material during physical  
4351 maintenance. This definition is borrowed from FIPS 140-2 security level 1.
- 4352 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,  
4353 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from  
4354 FIPS 140-2 security level 2.
- 4355 – Tamper resistance (highest level), this means there is a response to physical tempering that  
4356 typically includes zeroization of sensitive material on the module. This definition is borrowed  
4357 from FIPS 140-2 security level 3.

4358 It is difficult of specify quantitative certification test cases for accreditation of these levels.  
4359 Content protection regimes usually talk about different tools (widely available, specialized and  
4360 professional tools) used to circumvent the hardware protections put in place by manufacturing. If  
4361 needed, OCF can follow that model, if and when OCF engage in distributing sensitive key  
4362 material (e.g. PKI) to its members.

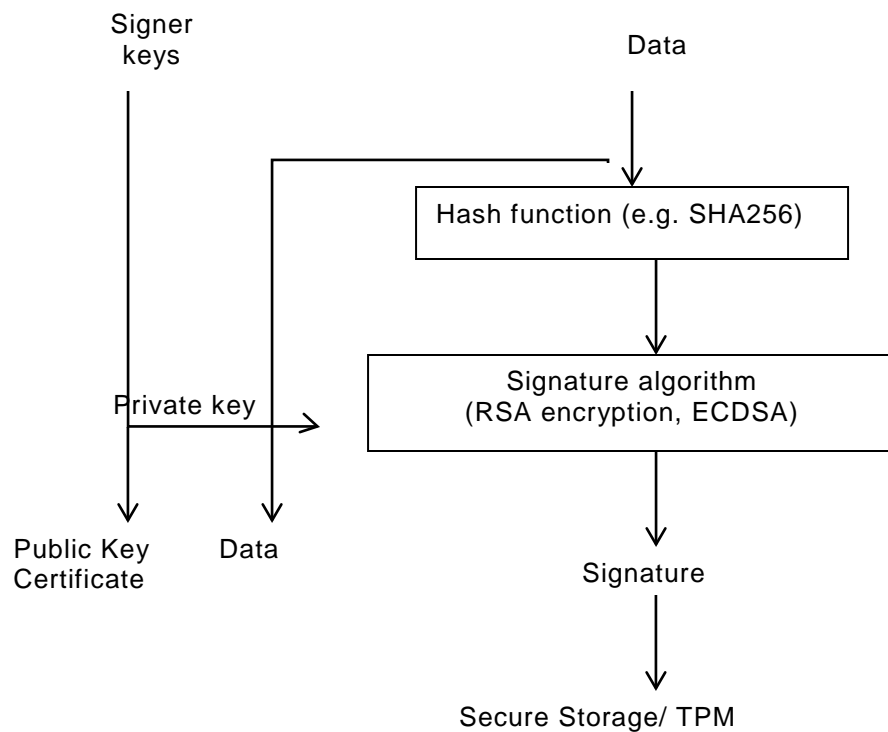
### 4363 **14.3 Secure Boot**

#### 4364 **14.3.1 Concept of software module authentication**

4365 In order to ensure that all components of a Device are operating properly and have not been  
4366 tampered with, it is best to ensure that the Device is booted properly. There may be multiple  
4367 stages of boot. The end result is an application running on top an operating system that takes  
4368 advantage of memory, CPU and peripherals through drivers.

4369 The general concept is that each software module is invoked only after cryptographic integrity  
4370 verification is complete. The integrity verification relies on the software module having been  
4371 hashed (e.g. SHA\_1, SHA\_256) and then signed with a cryptographic signature algorithm with  
4372 (e.g. RSA), with a key that only a signing authority has access to.

4373 Figure 40 depicts software module authentication.

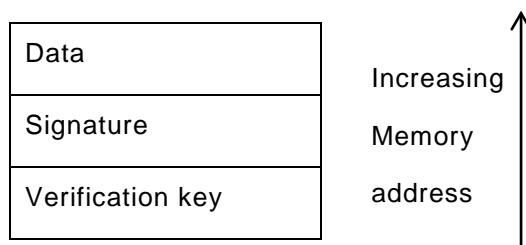


4374

**Figure 40 – Software Module Authentication**

4375 After the data is signed with the signer’s signing key (a private key), the verification key (the  
 4376 public key corresponding to the private signing key) is provided for later verification. For lower  
 4377 level software modules, such as bootloaders, the signatures and verification keys are inserted  
 4378 inside tamper proof memory, such as one-time programmable memory or TPM. For higher level  
 4379 software modules, such as application software, the signing is typically performed according to  
 4380 the PKCS#7 format IETF RFC 2315, where the signedData format includes both indications for  
 4381 signature algorithm, hash algorithm as well as the signature verification key (or certificate).  
 4382 Secure boot does not require use of PKCS#7 format.

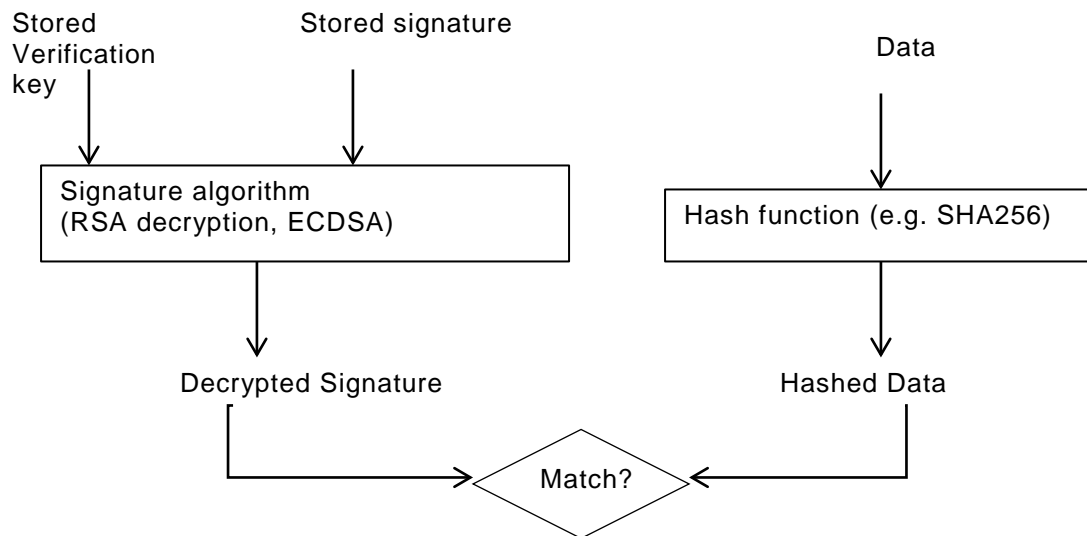
4383 Figure 41 depicts verification software module.



4384

**Figure 41 – Verification Software Module**

4385 As shown in Figure 42. the verification module first decrypts the signature with the verification  
 4386 key (public key of the signer). The verification module also calculates a hash of the data and then  
 4387 compares the decrypted signature (the original) with the hash of data (actual) and if the two  
 4388 values match, the software module is authentic.



4389 **Figure 42 – Software Module Authenticity**

4390 **14.3.2 Secure Boot process**

4391 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/  
 4392 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to  
 4393 find out where the boot code is and then run the boot code (second-stage boot loader). The  
 4394 second stage bootloader is typically the process that loads the operating system (Kernel) and  
 4395 transfers the execution to the where the Kernel code is. Once the Kernel starts, it may load  
 4396 external Kernel modules and drivers.

4397 When performing a secure boot, it is required that the integrity of each boot loader is verified  
 4398 before executing the boot loader stage. As mentioned, while the signature and verification key for  
 4399 the lowest level bootloader is typically stored in tamper-proof memory, the signature and  
 4400 verification key for higher levels should be embedded (but attached in an easily accessible  
 4401 manner) in the data structures software.

4402 **14.3.3 Robustness Requirements**

4403 **14.3.3.1 Robustness General**

4404 To qualify as high robustness secure boot process, the signature and hash algorithms shall be  
 4405 one of the approved algorithms, the signature values and the keys used for verification shall be  
 4406 stored in secure storage and the algorithms shall run inside a secure execution environment and  
 4407 the keys shall be provided the SEE over trusted path.

4408 **14.3.3.2 Next steps**

4409 Develop a list of approved algorithms and data formats

4410 **14.4 Attestation**

4411 **14.5 Software Update**

4412 **14.5.1 Overview:**

4413 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;  
 4414 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and  
 4415 end-of-life stages for the Device remain outstanding. It is possible for the Device to require



4416 update during any of these stages, although the most likely times are during onboarding, regular  
 4417 operation and maintenance. The aspects of the software include, but are not limited to, firmware,  
 4418 operating system, networking stack, application code, drivers, etc.

4419 **14.5.2 Recognition of Current Differences**

4420 Different manufacturers approach software update utilizing a collection of tools and strategies:  
 4421 over-the-air or wired USB connections, full or partial replacement of existing software, signed and  
 4422 verified code, attestation of the delivery package, verification of the source of the code, package  
 4423 structures for the software, etc.

4424 It is recommended that manufacturers review their processes and technologies for compliance  
 4425 with industry best-practices that a thorough security review of these takes place and that periodic  
 4426 review continue after the initial architecture has been established.

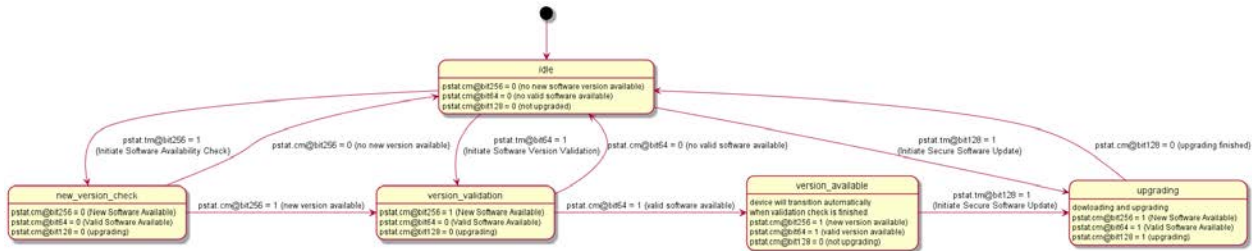
4427 This document applies to software updates as recommended to be implemented by OCF Devices;  
 4428 it does not have any bearing on the above-mentioned alternative proprietary software update  
 4429 mechanisms. The described steps are being triggered by an OCF Client, the actual  
 4430 implementation of the steps and how the software package is downloaded and upgraded is  
 4431 vendor specific.

4432 The triggers that can be invoked from OCF clients can perform:

- 4433 1) Check if new software is available
- 4434 2) Download and verify the integrity of the software package
- 4435 3) Install the verified software package

4436 The triggers are not sequenced, each trigger can be invoked individually.

4437 The state of the transitions of software update is in Figure 43.



4438  
 4439 **Figure 43 – State transitioning diagram for software download**

4440  
 4441 **Table 73 – Description of the software update bits**

Bit	TM property	CM property
Bit 9	Initiate Software Availability Check	New Software Available
Bit 7	Initiate Software Version Validation	Valid Software Available
Bit 8	Initiate Secure Software Update	Upgrading

4442  
 4443 **14.5.2.1 Checking availability of new software**

4444 Setting the Initiate Software Availability Check bit in the "/oic/sec/pstat.tm" Property (see  
 4445 Table 54 of clause 13.8) indicates a request to initiate the process to check if new software is

4446 available, e.g. the process whereby the Device checks if a newer software version is available on  
4447 the external endpoint. Once the Device has determined if a newer software version is available,  
4448 it sets the Initiate Software Availability Check bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE),  
4449 indicating that new software is available or to 0 (FALSE) if no newer software version is available,  
4450 See also Table 73 where the bits in property TM indicates that the action is initiated and the CM  
4451 bits are indicating the result of the action. The Device receiving this trigger is not downloading  
4452 and not validating the software to determine if new software is available. The version check is  
4453 determined by the current software version and the software version on the external endpoint.  
4454 The determination if a software package is newer is vendor defined.

### 4455 **14.5.3 Software Version Validation**

4456 Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see  
4457 Table 54 defines the Properties of "oic.r.pstat".

4458 Table 54 of 13.8) indicates a request to initiate the software version validation process, the  
4459 process whereby the Device validates the software (including firmware, operating system, Device  
4460 drivers, networking stack, etc.) against a trusted source to see if, at the conclusion of the check,  
4461 the software update process will need to be triggered (see clause 14.5.4). When the Initiate  
4462 Software Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged  
4463 Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and  
4464 initiates a software version check. Once the Device has determined if a valid software is available,  
4465 it sets the Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE)  
4466 if an update is available or 0 (FALSE) if no update is available. To signal completion of the  
4467 Software Version Validation process, the Device sets the Initiate Software Version Validation bit  
4468 in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation  
4469 bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation  
4470 process. The Software Version Validation process can download the software from the external  
4471 endpoint to verify the integrity of the software package.

### 4472 **14.5.4 Software Update**

4473 Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 54 of  
4474 clause 13.8) indicates a request to initiate the software update process. When the Initiate Secure  
4475 Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the  
4476 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a  
4477 software update process. Once the Device has completed the software update process, it sets  
4478 the Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if/when  
4479 the software was successfully updated or 0 (FALSE) if no update was performed. To signal  
4480 completion of the Secure Software Update process, the Device sets the Initiate Secure Software  
4481 Update bit in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure Software  
4482 Update bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the update  
4483 process.

#### 4484 **14.5.4.1 State of Device after software update**

4485 The state of all resources implemented in the Device should be the same as after boot, meaning  
4486 that the software update is not resetting user data and retaining a correct state.

4487 User data of a Device is defined as:

- 4488 – Retain the SVR states, e.g. the on boarded state, registered clients.
- 4489 – Retain all created resources
- 4490 – Retain all stored data of a resource
- 4491 – For example the preferences stored for the brewing resource ("oic.r.brewing").

4492 **14.5.5 Recommended Usage**

4493 The Initiate Secure Software Update bit of "/oic/sec/pstat.tm" should only be set by a Client after  
4494 the Initiate Software Version Validation check is complete.

4495 The process of updating Device software may involve state changes that affect the Device  
4496 Operational State ("/oic/sec/pstat.dos"). Devices with an interest in the Device(s) being updated  
4497 should monitor "/oic/sec/pstat.dos" and be prepared for pending software update(s) to affect  
4498 Device state(s) prior to completion of the update.

4499 The Device itself may indicate that it is autonomously initiating a software version check/update  
4500 or that a check/update is complete by setting the "pstat.tm" and "pstat.cm" Initiate Software  
4501 Version Validation and Secure Software Update bits when starting or completing the version  
4502 check or update process. As is the case with a Client-initiated update, Clients can be notified that  
4503 an autonomous version check or software update is pending and/or complete by observing pstat  
4504 resource changes.

4505 The "oic.r.softwareupdate" Resource Type specifies additional features to control the software  
4506 update process see core specification.

4507 **14.6 Non-OCF Endpoint interoperability**

4508 **14.7 Security Levels**

4509 Security Levels are a way to differentiate Devices based on their security criteria. This need for  
4510 differentiation is based on the requirements from different verticals such as industrial and health  
4511 care and may extend into smart home. This differentiation is distinct from Device classification  
4512 (e.g. IETF RFC 7228)

4513 These categories of security differentiation may include, but is not limited to:

- 4514 1) Security Hardening
- 4515 2) Identity Attestation
- 4516 3) Certificate/Trust
- 4517 4) Onboarding Technique
- 4518 5) Regulatory Compliance
  - 4519 a) Data at rest
  - 4520 b) Data in transit
- 4521 6) Cipher Suites – Crypto Algorithms & Curves
- 4522 7) Key Length
- 4523 8) Secure Boot/Update

4524 In the future security levels can be used to define interoperability.

4525 The following applies to the OCF Security Specification 1.1:

4526 The current document does not define any other level beyond Security Level 0. All Devices will  
4527 be designated as Level 0. Future versions may define additional levels.

4528 Additional comments:

- 4529 – The definition of a given security level will remain unchanged between versions of the  
4530 document.
- 4531 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.

4532 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of  
4533 the higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and  
4534 a later document version defines a security level 1, the Device could be evaluated and  
4535 classified as level 1 if it meets level 1 requirements).

4536 – The security levels may need to be visible to the end user.

## 4537 **14.8 Security Profiles**

### 4538 **14.8.1 Security Profiles General**

4539 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This  
4540 need for differentiation is based on the requirements from different verticals such as industrial  
4541 and health care and may extend into smart home. This differentiation is distinct from device  
4542 classification (e.g. IETF RFC 7228)

4543 These categories of security differentiation may include, but is not limited to:

4544 1) Security Hardening and assurances criteria

4545 2) Identity Attestation

4546 3) Certificate/Trust

4547 4) Onboarding Technique

4548 5) Regulatory Compliance

4549 a) Data at rest

4550 b) Data in transit

4551 6) Cipher Suites – Crypto Algorithms & Curves

4552 7) Key Length

4553 8) Secure Boot/Update

4554 Each Security Profile definition must specify the version or versions of the OCF Security  
4555 Specification(s) that form a baseline set of normative requirements. The profile definition may  
4556 include security requirements that supersede baseline requirements (not to relax security  
4557 requirements).

4558 Security Profiles have the following properties:

4559 – A given profile definition is not specific to the version of the document that defines it. For  
4560 example, the profile may remain constant for subsequent OCF Security Specification versions.

4561 – A specific OCF Device and platform combination may be used to satisfy the security profile.

4562 – Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles  
4563 simultaneously.

4564 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found  
4565 to satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the  
4566 document, and a later document version defines a security profile Black, the device could be  
4567 evaluated and classified as profile Black if it meets profile Black requirements).

4568 – A machine-readable representation of compliance results specifically describing profiles  
4569 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or  
4570 manifest may contain security profiles attributes).

### 4571 **14.8.2 Identification of Security Profiles (Normative)**

#### 4572 **14.8.2.1 Security Profiles in Prior Documents**

4573 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles  
4574 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in

4575 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use  
 4576 the OCF Security Specification version to characterize expected security behaviour.

4577 **14.8.2.2 Security Profile Resource Definition**

4578 The "oic.sec.sp" Resource is used by the OCF Device to show which OCF Security Profiles the  
 4579 OCF Device is capable of supporting and which are authorized for use by the OCF Security  
 4580 Domain owner. Properties of the Resource identify which OCF Security Profile is currently  
 4581 operational. The ocfSecurityProfileOID value type shall represent OID values and may reference  
 4582 an entry in the form of strings (UTF-8).

4583 "oic.sec.sp" Resource is defined in Table 74.

4584 **Table 74 – 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

4585 Table 75 defines the Properties of "oic.sec.sp".

4586 **Table 75 – 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")

4587 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or  
 4588 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4589 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)  
 4590 private(4) enterprise(1) OCF(51414) }

4591  
 4592 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4593  
 4594 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4595  
 4596 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }

4597 --The Security Profile is not specified

4598 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }

4599 --This specifies the OCF Baseline Security Profile(s)

4600 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }

4601 --This specifies the OCF Black Security Profile(s)

4602 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }

4603 --This specified the OCF Blue Security Profile(s)

4604 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }

4605 --This specifies the OCF Purple Security Profile(s)

4606  
 4607 --versioned Security Profiles

4608 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)

4609 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4610 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}

4611 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4612 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}

```
4613     --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"
4614     sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
4615     --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"
4616     sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
4617     --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"
4618
4619     ocfSecurityProfileOID ::= UTF8String
4620
```

### 4621 **14.8.3 Security Profiles**

#### 4622 **14.8.3.1 Security Profiles General**

4623 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to  
4624 the Security Profile clauses for additional details).

4625 The OCF Conformance criteria may require vendor attestation that establishes the expected  
4626 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific  
4627 requirements).

#### 4628 **14.8.3.2 Security Profile Unspecified (sp-undefined-v0)**

4629 The Security Profile "sp-undefined-v0" is reserved for future use.

#### 4630 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4631 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where  
4632 the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the  
4633 "supportedprofiles" Property of the "/oic/sec/sp" Resource.

4634 It indicates the OCF Device satisfies the normative security requirements for this document.

4635 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-  
4636 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other  
4637 profiles.

4638 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to  
4639 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

#### 4640 **14.8.3.4 Security Profile Black (sp-black-v0)**

##### 4641 **14.8.3.4.1 Black Profile General**

4642 The need for Security Profile Black v0 is to support devices and manufacturers who wish to  
4643 certify their devices meeting this specific set of security criteria. A Device may satisfy the Black  
4644 requirements as well as requirements of other profiles, the Black Security Profile is not  
4645 necessarily mutually exclusive with other Security Profiles unless those requirements conflict with  
4646 the explicit requirements of the Black Security Profile.

##### 4647 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4648 Security Profile Black devices could include any device a manufacturer wishes to certify at this  
4649 profile, but healthcare devices and industrial devices with additional security requirements are  
4650 the initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or  
4651 devices with exceptional profiles of trust bestowed upon them, may wish to certify at this profile;  
4652 these types of devices may include, but are not limited to the following:

- 4653 – Bridges (Mapping devices between ecosystems handling virtual devices from different  
4654 ecosystems)
- 4655 – Resource Directories (Devices trusted to manage OCF Security Domain resources)
- 4656 – Remote Access (Devices which have external access but can also act within the OCF  
4657 Security Domain)

- 4658 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)
- 4659 – Industrial Devices (Devices with advanced management, security and attestation
- 4660 requirements)

#### 4661 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4662 Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security  
4663 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4664 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4665 – Support for AES 128 encryption for data at rest and in transit.
- 4666 – Hardening minimums: manufacturer assertion of secure credential storage
- 4667 – In 13) in enumerated item #10 "The "/oic/sec/cred" Resource should contain credential(s) if  
4668 required by the selected OTM" is changed to require the credential be stored: "The  
4669 "/oic/sec/cred" Resource shall contain credential(s)."
- 4670 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4671 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by  
4672 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4673 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-  
4674 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4675 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to  
4676 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4677 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework  
4678 described in the supporting documents:

- 4679 – Certificate Profile (See 9.4.2)
- 4680 – Certificate Policy (see Certificate Policy document:  
4681 <https://openconnectivity.org/specs/OCF%20Certificate%20Policy.pdf>)

#### 4682 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

##### 4683 **14.8.3.5.1 Blue Profile General**

4684 The Security Profile Blue is used when manufacturers issue platform certificates for platforms  
4685 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is  
4686 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF  
4687 Security Domain owners evaluate manufacturer supplied certificates and attributed data to  
4688 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.  
4689 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may  
4690 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4691 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting  
4692 Criteria defined by OCF.

##### 4693 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4694 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from  
4695 OCF Device vendor and where platform vendors may implement trusted platforms that may  
4696 conform to industry standards defining trusted platforms. The OCF Security Profile Blue specifies  
4697 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance  
4698 criteria produced by OCF conformance operations. The OCF Security Domain owner evaluates  
4699 these data when an OCF Device is onboarded into the OCF Security Domain. Based on this  
4700 evaluation the OCF Security Domain owner determines which Security Profile may be applied  
4701 during OCF Device operation. All OCF Device types may be considered for evaluation using the  
4702 OCF Security Profile Blue.

4703 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4704 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for  
4705 this document version are satisfied and the following additional criteria are satisfied.

4706 OCF Blue profile defines the following OCF Device quality assurances:

- 4707 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF  
4708 Device was hosted on one or more platforms that satisfies OCF Blue platform security  
4709 assurances and platform security and privacy functionality requirements.
- 4710 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF  
4711 and published by OCF in a machine readable format.
- 4712 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned  
4713 signing key.
- 4714 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4715 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by  
4716 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4717 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in  
4718 its certificate.
- 4719 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF  
4720 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's  
4721 "securityprofiles" field.

4722 OCF Blue profile defines the following OCF Device security functionality:

- 4723 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage  
4724 functions are hardened by the platform.
- 4725 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials  
4726 using the "/oic/sec/cred" Resource where the "credusage" Property contains the value  
4727 "oic.sec.cred.mfgcert".
- 4728 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an  
4729 IEEE802.1AR IDevID and should verify the "TCG Endorsement Key Credential". All TCG-  
4730 defined manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the  
4731 "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to  
4732 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4733 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See  
4734 NIST SP 800-57).
- 4735 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST  
4736 SP 800-57).
- 4737 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST  
4738 SP 800-57).
- 4739 – OCF Device(s) should protect the "/oic/sec/cred" resource using the platform provided secure  
4740 storage.
- 4741 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned  
4742 certificates) using platform provided secure storage.
- 4743 – OCF Device(s) should check certificate revocation status for locally issued certificates.
- 4744 – OCF OBTs (aka DOTS) shall check certificate revocation status for all certificates in  
4745 manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation  
4746 fails and the connection is refused. The DOTS may disregard revocation status results if  
4747 unavailable.

4748 OCF Blue profile defines the following platform security assurances:



4749 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage  
4750 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL  
4751 Level 2.

4752 – Platforms implementing trusted platform functionality should be evaluated with a minimum  
4753 Common Criteria EAL Level 1.

4754 OCF Blue profile defines the following platform security and privacy functionality:

4755 – The Platform shall implement cryptographic service provider (CSP) functionality.

4756 – Platform CSP functionality shall include cryptographic algorithms, random number generation,  
4757 secure time.

4758 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST  
4759 SP 800-57).

4760 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See  
4761 NIST SP 800-57).

4762 – Platforms hosting OCF Device(s) should implement a platform identifier following  
4763 IEEE802.1AR or Trusted Computing Group(TCG) specifications.

4764 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF  
4765 Device(s) should supply TCG-defined manufacture certificates; also known as "TCG  
4766 Endorsement Key Credential" (which complies with IETF RFC 5280) and "TCG Platform  
4767 Credential" (which complies with IETF RFC 5755).

4768 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,  
4769 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

4770 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to  
4771 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

4772 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"  
4773 Property to one of the other values found in the "supportedprofiles" Property.

#### 4774 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**

4775 Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause  
4776 14.8.2 must support following minimum requirements

4777 – Hardening minimums: secure credential storage, software integrity validation, secure update.

4778 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension  
4779 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-  
4780 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"

4781 – The OCF Device shall include a X.509v3 OCF CPLAttributes Extension (clause 9.4.2.2.7) in its  
4782 End-Entity Certificate when manufacturer certificate is used.

4783 Security Profile Purple has following optional security hardening requirements that the device can  
4784 additionally support.

4785 – Hardening additions: secure boot, hardware backed secure storage

4786 – The OCF Device shall include a X.509v3 OCF SecurityClaims Extension (clause 9.4.2.2.6) in  
4787 its End-Entity Certificate and it shall include corresponding OIDs to the hardening additions  
4788 implemented and attested by the vendor. If there is no additional support for hardening  
4789 requirements, X.509v3 OCF SecurityClaims Extension shall be omitted.

4790 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism  
4791 for security critical executables such as cryptographic modules or secure service applications,  
4792 and they should be validated before the execution. The key used for validating the integrity must  
4793 be pinned at the least to the validating software module.

4794 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4795 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM)  
4796 to be executed by the processor on power-on, and secure boot parameters to be provisioned by  
4797 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the  
4798 security critical executables and stop the boot process if any integrity of them is compromised.

4799 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile  
4800 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic  
4801 attacks.

4802 More details on security hardening guidelines for software integrity validation, secure boot,  
4803 secure update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4804 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA  
4805 Vetting Criteria defined by OCF.

4806 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-  
4807 purple-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other  
4808 profiles.

4809 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to  
4810 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

4811 **15 Device Type Specific Requirements**

4812 **15.1 Bridging Security**

4813 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4814 The Bridge shall go through OCF ownership transfer as any other onboarder would.

4815 The software of an Bridge shall be field updatable. (This requirement need not be tested but can  
4816 be certified via a vendor declaration.)

4817 Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be  
4818 provisioned as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged  
4819 Devices are treated the same way as physical Devices. They are entities that have to be  
4820 provisioned in their network.

4821 Each VOD shall implement the behaviour required by ISO/IEC 30118-1:2018 and this document.  
4822 Each VOD shall perform authentication, access control, and encryption according to the security  
4823 settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security  
4824 requirements of the Bridged Protocol.

4825 In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall  
4826 use appropriate ecosystem-specific security options for communication between the Virtual  
4827 Bridged Devices instantiated by the Bridge and Bridged Devices. This security shall include  
4828 mutual authentication, and encryption and integrity protection of messages in the bridged  
4829 ecosystem.

4830 A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see  
4831 clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which  
4832 instantiated that VOD.

4833 A VOD may authenticate itself to the OCF Cloud (see clause 10.5.2) using the Manufacturer  
4834 Certificate and corresponding private key of the Bridge which instantiated that VOD.

4835 A Bridge and the VODs created by that Bridge shall operate as independent Devices, with the  
4836 following exceptions:

4837 – If a Bridge creates a VOD while the Bridge is in an Unowned State, then the VOD shall be  
4838 created in an Unowned State.

4839 – An Unowned VOD shall not accept DTLS connection attempts nor TLS connection attempts  
4840 nor any other requests, including discovery requests, while the Bridge (that created that VOD)  
4841 is Unowned.

4842 – At any time when a Bridge is transitioning from Owned to Unowned State, all Unowned VODs  
4843 (created by that Bridge prior to the transition) shall drop any existing TLS and/or DTLS  
4844 connections.

4845 – At any time when a Bridge is transitioning from Unowned to Owned State, the Bridge shall  
4846 trigger all Unowned VODs (created by that Bridge prior to the transition) to become  
4847 accessible in RFOTM state, with internal state as if the VOD has just transitioned from  
4848 RESET to RFOTM.

4849 – If a Bridge creates a VOD while the Bridge is in an Owned State, then the VOD shall become  
4850 accessible in RFOTM state, with internal state as if the VOD has just transitioned from  
4851 RESET to RFOTM.

4852 Table 76 intends to clarify this behaviour.

4853  
4854

**Table 76 – Dependencies of VOD Behaviour on Bridge state, as clarification of accompanying text**

Bridge state	Additional dependencies on VOD behaviour	
	VOD is Unowned (either just created, or created previously)	VOD is Owned
From unboxing Bridge until just prior to the end of transition of Bridge from Unowned to Owned	No accepting DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests	Not applicable
At end of transition from Unowned to Owned	VOD becomes accessible in RFOTM following Bridge's transition. Internal state as if just transitioned from RESET.	As per normal Device
Owned	As per normal Device	As per normal Device
At Start of transition from Owned to Unowned	Drop any established TLS/DTLS connections, even if already partway through Device ownership	As per normal Device
Start of transition from Owned to Unowned, until just prior to the end of transition from Unowned to Owned.	No accepting DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests	As per normal Device

4855 The "vods" Property of the "oic.r.vodlist" Resource on a Bridge reflects the details of all currently  
4856 Owned VODs which have been created by that Bridge since the most recent hardware reset (if  
4857 any) of the Bridge Platform (which removes all the created VODs), regardless of whether the  
4858 VODs have the same owner as the Bridge or not. The entries in the "vods" Property are added  
4859 and removed according to the following criteria:

- 4860 – Whenever a VOD created by a Bridge transitions from being Unowned to being Owned, then  
4861 an entry for that VOD shall be added to the "vods" Property of the "oic.r.vodlist" Resource of  
4862 that Bridge.
- 4863 – Whenever a VOD created by a Bridge transitions from being Owned to being Unowned, then  
4864 entry for that VOD shall be removed from the "vods" Property of the "oic.r.vodlist" Resource of  
4865 that Bridge. If that Bridge is currently in Unowned state, then the "oic.r.vodlist" Resource is  
4866 not accessible, and the entry for that VOD shall be removed from the "vods" Property before  
4867 or during the transition of that Bridge to the Owned state.
- 4868 – All other modifications of the list are not allowed.

4869 A Bridge shall only expose a secure OCF Endpoint for the "oic.r.vodlist" Resource.

4870 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4871 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4872 For AllJoyn translator, an OCF OBT shall be able to block the communication of all OCF Devices  
4873 with all Bridged Devices that don't communicate securely with the Bridge, by using the Bridge  
4874 Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

4875 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4876 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4877 communicate securely with the Bridge.

4878 **15.1.2.3 Additional Security Requirements specific to the oneM2M Protocols**

4879 The Bridge shall implement oneM2M application access control as defined in the oneM2M  
4880 Release 3 Specifications.

4881 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4882 communicate securely with the Bridge.

4883 **15.1.2.4 Additional Security Requirements specific to the U+ Protocol**

4884 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4885 communicate securely with the Bridge.

4886 **15.1.2.5 Additional Security Requirements specific to the Z-Wave Protocol**

4887 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4888 communicate securely with the Bridge.

4889 **15.1.2.6 Additional Security Requirements specific to the Zigbee Protocol**

4890 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4891 communicate securely with the Bridge.

4892

4893

4894

4895

4896

4897

4898

4899

4900

4901

4902

4903

4904

4905

4906

4907

4908

4909

4910

4911

4912

4913 .

4914  
4915  
4916

## Annex A (informative) Access Control Examples

4917

### Example OCF ACL Resource

4918 Figure A-1 shows how a "/oic/sec/acl2" Resource could be configured to enforce an example  
4919 access policy on the Server.

```
4920 {
4921   "aclist2": [
4922     {
4923       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve,
4924       Update, Delete and Notify)
4925       "subject": {"uuid": "XXX-...-XX01"},
4926       "resources": [
4927         {"href": "/oic/sh/light/1"},
4928         {"href": "/oic/sh/temp/0"}
4929       ],
4930       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC
4931       "validity": [
4932         // The period starting at 18:00:00 UTC, on January 1, 2015 and
4933         // ending at 07:00:00 UTC on January 2, 2015
4934         "period": ["20150101T180000Z/20150102T070000Z"],
4935         // Repeats the {period} every week until the last day of Jan. 2015.
4936         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
4937       ],
4938       "aceid": 1
4939     }
4940   ],
4941   // An ACL provisioning and management service should be identified as
4942   // the resource owner
4943   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
4944 }
4945
```

4945 **Figure A-1 – Example "/oic/sec/acl2" Resource**

4946

### Example AMS

4947 Figure A-2 demonstrates how the "/oic/sec/amacl" Resource should be configured to achieve this  
4948 objective.

```
4949 {
4950   "resources": [
4951     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
4952     // supplied then use the sacl validation credential to enforce access.
4953     {"href": "/oic/sh/light/1"},
4954     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
4955     // supplied then use the sacl validation credential to enforce access.
4956     {"href": "/oma/3"},

```

```
4957 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4958 // the sacl validation credential to enforce access.
4959 {"wc": ""}]
4960 }
4961
```

**Figure A-2 Example "/oic/sec/amacl" Resource**

4962  
4963  
4964

## Annex B (Informative) Execution Environment Security Profiles

4965 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security  
4966 robustness requirements meeting all IOT applications and services will not serve the needs of  
4967 OCF, and security profiles of varying degree of robustness (trustworthiness), cost and complexity  
4968 have to be defined. To address a large ecosystem of vendors, the profiles can only be defined as  
4969 requirements and the exact solutions meeting those requirements are specific to the vendors'  
4970 open or proprietary implementations, and thus in most part outside scope of this document.

4971 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228  
4972 (Terminology for constrained node networks) methodology, we limit the number of security  
4973 profiles to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities  
4974 criteria for each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4975 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are  
4976 either capable of no security functionality or easily breakable security that depend on  
4977 environmental (e.g. availability of human) factors to perform security functions. This means the  
4978 class 0 will not be equipped with an SEE.

4979

**Table B.1 – OCF Security Profile**

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4980 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of  
4981 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,  
4982 memory, storage).



4983  
4984  
4985

## Annex C (normative) Resource Type definitions

### 4986 C.1 List of Resource Type definitions

4987 Table C.1 contains the list of defined security resources in this document.

4988 **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

4989

### 4990 C.2 Account Token

#### 4991 C.2.1 Introduction

4992 Sign-up using generic account provider.

#### 4993 C.2.2 Well-known URI

4994 /oic/sec/account

#### 4995 C.2.3 Resource type

4996 The Resource Type is defined as: "oic.r.account".

#### 4997 C.2.4 OpenAPI 2.0 definition

```
4998 {  
4999   "swagger": "2.0",  
5000   "info": {  
5001     "title": "Account Token",  
5002     "version": "20190111",  
5003     "license": {  
5004       "name": "OCF Data Model License",  
5005       "url":  
5006       "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
5007       CENSE.md",  
5008       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
5009       reserved."  
5010     }  
5011     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
```

```

5012 },
5013 "schemes": ["http"],
5014 "consumes": ["application/json"],
5015 "produces": ["application/json"],
5016 "paths": {
5017   "/oic/sec/account" : {
5018     "post": {
5019       "description": "Sign-up using generic account provider.\n",
5020       "parameters": [
5021         { "$ref": "#/parameters/interface" },
5022         {
5023           "name": "body",
5024           "in": "body",
5025           "required": true,
5026           "schema": { "$ref": "#/definitions/Account-request" },
5027           "x-example":
5028             {
5029               "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
5030               "authprovider" : "github",
5031               "accesstoken" : "8802f2eaf8b5e147a936"
5032             }
5033         }
5034       ],
5035       "responses": {
5036         "204": {
5037           "description": "2.04 Changed respond with required and optional information\n",
5038           "x-example":
5039             {
5040               "rt": ["oic.r.account"],
5041               "accesstoken" : "0f3d9f7fe5491d54077d",
5042               "refreshToken" : "00fe4644a6fbe5324eec",
5043               "expiresin" : 3600,
5044               "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
5045               "redirecturi" : "coaps+tcp://example.com:443"
5046             },
5047           "schema": { "$ref": "#/definitions/Account-response" }
5048         }
5049       }
5050     },
5051     "delete": {
5052       "description": "Delete a device. This also removes all resources in the device on cloud
5053 side.\nexample: /oic/account?di=9cfbeb8e-5ale-4dlc-9d01-
5054 00c04fd430c8&accesstoken=0f3d9f7fe5491d54077d\n",
5055       "parameters": [
5056         { "$ref": "#/parameters/interface" }
5057       ],
5058       "responses": {
5059         "202": {
5060           "description": "2.02 Deleted response informing the device is successfully
5061 deleted.\n"
5062         }
5063       }
5064     }
5065   },
5066 },
5067 "parameters": {
5068   "interface" : {
5069     "in" : "query",
5070     "name" : "if",
5071     "type" : "string",
5072     "enum" : ["oic.if.baseline"]
5073   }
5074 },
5075 "definitions": {
5076   "Account-request" : {
5077     "properties": {
5078       "authprovider": {
5079         "description": "The name of Authorization Provider through which Access Token was
5080 obtained",
5081         "type": "string"
5082       }
5083     }
5084   }
5085 }

```

```

5083     "accesstoken" : {
5084         "description": "Access-Token used for communication with OCF Cloud after account creation",
5085         "pattern": "(?!$|\\s+).*",
5086         "type": "string"
5087     },
5088     "di": {
5089         "description": "Format pattern according to IETF RFC 4122.",
5090         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5091         "type": "string"
5092     }
5093 },
5094 "type" : "object",
5095 "required": ["di", "accesstoken"]
5096 },
5097 "Account-response": {
5098     "properties": {
5099         "expiresin" : {
5100             "description": "Access-Token remaining life time in seconds (-1 if permanent)",
5101             "readOnly": true,
5102             "type": "integer"
5103         },
5104         "rt": {
5105             "description": "Resource Type of the Resource",
5106             "items": {
5107                 "maxLength": 64,
5108                 "type": "string",
5109                 "enum" : ["oic.r.account"]
5110             },
5111             "minItems": 1,
5112             "maxItems": 1,
5113             "readOnly": true,
5114             "type": "array"
5115         },
5116         "refreshtoken" : {
5117             "description": "Refresh token can be used to refresh the Access Token before getting
5118 expired",
5119             "pattern": "(?!$|\\s+).*",
5120             "readOnly": true,
5121             "type": "string"
5122         },
5123         "uid" : {
5124             "description": "Format pattern according to IETF RFC 4122.",
5125             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5126             "type": "string"
5127         },
5128         "accesstoken" : {
5129             "description": "Access-Token used for communication with cloud after account creation",
5130             "pattern": "(?!$|\\s+).*",
5131             "type": "string"
5132         },
5133         "n": {
5134             "$ref":
5135 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5136 schema.json#/definitions/n"
5137         },
5138         "id": {
5139             "$ref":
5140 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5141 schema.json#/definitions/id"
5142         },
5143         "redirecturi" : {
5144             "description": "Using this URI, the Client needs to reconnect to a redirected OCF Cloud.
5145 If provided, this value shall be used by the Device instead of Mediator-provided URI during the
5146 Device Registration.",
5147             "readOnly": true,
5148             "type": "string"
5149         },
5150         "if": {
5151             "description": "The interface set supported by this resource",
5152             "items": {
5153                 "enum": [

```

```

5154         "oic.if.baseline"
5155     ],
5156     "type": "string"
5157 },
5158 "minItems": 1,
5159 "maxItems": 1,
5160 "uniqueItems": true,
5161 "readOnly": true,
5162 "type": "array"
5163 }
5164 },
5165 "type" : "object",
5166 "required": ["accesstoken", "refreshtoken", "expiresin", "uid"]
5167 }
5168 }
5169 }
5170

```

5171 **C.2.5 Property definition**

5172 Table C.2 defines the Properties that are part of the "oic.r.account" Resource Type.

5173 **Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account".**

Property name	Value type	Mandatory	Access mode	Description
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
authprovider	string	No	Read Write	The name of Authorization Provider through which Access Token was obtained
accesstoken	string	Yes	Read Write	Access-Token used for communication with OCF Cloud after account creation
id	multiple types: see schema	No	Read Write	
refreshtoken	string	Yes	Read Only	Refresh token can be used to refresh the Access Token before getting expired
rt	array: see schema	No	Read Only	Resource Type of the Resource
accesstoken	string	Yes	Read Write	Access-Token used for communication with cloud after account creation
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds

				(-1 if permanent)
if	array: see schema	No	Read Only	The interface set supported by this 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.
n	multiple types: see schema	No	Read Write	

5174 **C.2.6 CRUDN behaviour**

5175 Table C.3 defines the CRUDN operations that are supported on the "oic.r.account" Resource  
5176 Type.

5177 **Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account".**

Create	Read	Update	Delete	Notify
		post	delete	

5178 **C.3 Access Control List [DEPRECATED]**

5179 This clause intentionally left empty.

5180 **C.4 Access Control List-2**

5181 **C.4.1 Introduction**

5182 This Resource specifies the local access control list.  
5183 When used without query parameters, all the ACE entries are returned.  
5184 When used with a query parameter, only the ACEs matching the specified  
5185 parameter are returned.  
5186

5187 **C.4.2 Well-known URI**

5188 /oic/sec/acl2

5189 **C.4.3 Resource type**

5190 The Resource Type is defined as: "oic.r.acl2".

5191 **C.4.4 OpenAPI 2.0 definition**

5192 {  
5193 "swagger": "2.0",  
5194 "info": {  
5195 "title": "Access Control List-2",  
5196 "version": "20190111",  
5197 "license": {  
5198 "name": "OCF Data Model License",  
5199 "url":  
5200 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI

```

5201 CENSE.md",
5202     "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5203 reserved."
5204 },
5205     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5206 },
5207     "schemes": ["http"],
5208     "consumes": ["application/json"],
5209     "produces": ["application/json"],
5210     "paths": {
5211         "/oic/sec/acl2" : {
5212             "get": {
5213                 "description": "This Resource specifies the local access control list.\nWhen used without
5214 query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
5215 matching the specified\nparameter are returned.\n",
5216                 "parameters": [
5217                     {"$ref": "#/parameters/interface"},
5218                     {"$ref": "#/parameters/ace-filtered"}
5219                 ],
5220                 "responses": {
5221                     "200": {
5222                         "description": "",
5223                         "x-example":
5224                             {
5225                                 "rt" : ["oic.r.acl2"],
5226                                 "aclist2": [
5227                                     {
5228                                         "aceid": 1,
5229                                         "subject": {
5230                                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5231                                             "role": "SOME_STRING"
5232                                         },
5233                                         "resources": [
5234                                             {
5235                                                 "href": "/light",
5236                                                 "rt": ["oic.r.light"],
5237                                                 "if": ["oic.if.baseline", "oic.if.a"]
5238                                             },
5239                                             {
5240                                                 "href": "/door",
5241                                                 "rt": ["oic.r.door"],
5242                                                 "if": ["oic.if.baseline", "oic.if.a"]
5243                                             }
5244                                         ],
5245                                         "permission": 24
5246                                     },
5247                                     {
5248                                         "aceid": 2,
5249                                         "subject": {
5250                                             "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5251                                         },
5252                                         "resources": [
5253                                             {
5254                                                 "href": "/light",
5255                                                 "rt": ["oic.r.light"],
5256                                                 "if": ["oic.if.baseline", "oic.if.a"]
5257                                             },
5258                                             {
5259                                                 "href": "/door",
5260                                                 "rt": ["oic.r.door"],
5261                                                 "if": ["oic.if.baseline", "oic.if.a"]
5262                                             }
5263                                         ],
5264                                         "permission": 24
5265                                     },
5266                                     {
5267                                         "aceid": 3,
5268                                         "subject": {"conntype": "anon-clear"},
5269                                         "resources": [
5270                                             {
5271                                                 "href": "/light",

```

```

5272         "rt": ["oic.r.light"],
5273         "if": ["oic.if.baseline", "oic.if.a"]
5274     },
5275     {
5276         "href": "/door",
5277         "rt": ["oic.r.door"],
5278         "if": ["oic.if.baseline", "oic.if.a"]
5279     }
5280 ],
5281 "permission": 16,
5282 "validity": [
5283     {
5284         "period": "20160101T180000Z/20170102T070000Z",
5285         "recurrence": [ "DSTART:XXXXX",
5286 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5287     },
5288     {
5289         "period": "20160101T180000Z/PT5H30M",
5290         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5291     }
5292 ]
5293 },
5294 ],
5295 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5296 },
5297 "schema": { "$ref": "#/definitions/Acl2" }
5298 },
5299 "400": {
5300     "description": "The request is invalid."
5301 }
5302 },
5303 },
5304 "post": {
5305     "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
5306 update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
5307 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
5308 the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
5309     "parameters": [
5310         { "$ref": "#/parameters/interface" },
5311         { "$ref": "#/parameters/ace-filtered" },
5312         {
5313             "name": "body",
5314             "in": "body",
5315             "required": true,
5316             "schema": { "$ref": "#/definitions/Acl2-Update" },
5317             "x-example":
5318                 {
5319                     "aclist2": [
5320                         {
5321                             "aceid": 1,
5322                             "subject": {
5323                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5324                                 "role": "SOME_STRING"
5325                             },
5326                             "resources": [
5327                                 {
5328                                     "href": "/light",
5329                                     "rt": ["oic.r.light"],
5330                                     "if": ["oic.if.baseline", "oic.if.a"]
5331                                 },
5332                                 {
5333                                     "href": "/door",
5334                                     "rt": ["oic.r.door"],
5335                                     "if": ["oic.if.baseline", "oic.if.a"]
5336                                 }
5337                             ],
5338                             "permission": 24
5339                         },
5340                         {
5341                             "aceid": 3,
5342                             "subject": {

```

```

5343         "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5344     },
5345     "resources": [
5346         {
5347             "href": "/light",
5348             "rt": ["oic.r.light"],
5349             "if": ["oic.if.baseline", "oic.if.a"]
5350         },
5351         {
5352             "href": "/door",
5353             "rt": ["oic.r.door"],
5354             "if": ["oic.if.baseline", "oic.if.a"]
5355         }
5356     ],
5357     "permission": 24
5358 }
5359 ],
5360 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5361 }
5362 },
5363 ],
5364 "responses": {
5365     "400": {
5366         "description": "The request is invalid."
5367     },
5368     "201": {
5369         "description": "The ACL entry is created."
5370     },
5371     "204": {
5372         "description": "The ACL entry is updated."
5373     }
5374 }
5375 },
5376 "delete": {
5377     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5378 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
5379 the\nspecified parameter are deleted.\n",
5380     "parameters": [
5381         {"$ref": "#/parameters/interface"},
5382         {"$ref": "#/parameters/ace-filtered"}
5383     ],
5384     "responses": {
5385         "200": {
5386             "description": "The matching ACEs or the entire ACL Resource has been successfully
5387 deleted."
5388         },
5389         "400": {
5390             "description": "The request is invalid."
5391         }
5392     }
5393 }
5394 },
5395 },
5396 "parameters": {
5397     "interface": {
5398         "in": "query",
5399         "name": "if",
5400         "type": "string",
5401         "enum": ["oic.if.baseline"]
5402     },
5403     "ace-filtered": {
5404         "in": "query",
5405         "name": "aceid",
5406         "required": false,
5407         "type": "integer",
5408         "description": "Only applies to the ACE with the specified aceid.",
5409         "x-example": 2112
5410     }
5411 },
5412 "definitions": {
5413     "Acl2": {

```



```

5414     "properties": {
5415         "owneruuiid" : {
5416             "description": "The value identifies the unique Resource owner\nFormat pattern according
5417 to IETF RFC 4122.",
5418             "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}$",
5419             "type": "string"
5420         },
5421         "rt" : {
5422             "description": "Resource Type of the Resource.",
5423             "items": {
5424                 "maxLength": 64,
5425                 "type": "string",
5426                 "enum": ["oic.r.acl2"]
5427             },
5428             "minItems": 1,
5429             "maxItems": 1,
5430             "readOnly": true,
5431             "type": "array"
5432         },
5433         "aclist2" : {
5434             "description": "Access Control Entries in the ACL Resource.",
5435             "items": {
5436                 "properties": {
5437                     "aceid": {
5438                         "description": "An identifier for the ACE that is unique within the ACL. In cases
5439 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5440                         "minimum": 1,
5441                         "type": "integer"
5442                     },
5443                     "permission": {
5444                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5445 permissions.",
5446                         "x-detail-desc": [
5447                             "0 - No permissions",
5448                             "1 - Create permission is granted",
5449                             "2 - Read, observe, discover permission is granted",
5450                             "4 - Write, update permission is granted",
5451                             "8 - Delete permission is granted",
5452                             "16 - Notify permission is granted"
5453                         ],
5454                         "maximum": 31,
5455                         "minimum": 0,
5456                         "type": "integer"
5457                     },
5458                     "resources": {
5459                         "description": "References the application's Resources to which a security policy
5460 applies.",
5461                         "items": {
5462                             "description": "Each Resource must have at least one of these properties set.",
5463                             "properties": {
5464                                 "href": {
5465                                     "description": "When present, the ACE only applies when the href matches\nThis
5466 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5467                                     "format": "uri",
5468                                     "maxLength": 256,
5469                                     "type": "string"
5470                                 },
5471                                 "if": {
5472                                     "description": "When present, the ACE only applies when the if (interface)
5473 matches\nThe interface set supported by this Resource.",
5474                                     "items": {
5475                                         "enum": [
5476                                             "oic.if.baseline",
5477                                             "oic.if.ll",
5478                                             "oic.if.b",
5479                                             "oic.if.rw",
5480                                             "oic.if.r",
5481                                             "oic.if.a",
5482                                             "oic.if.s"
5483                                         ],
5484                                         "type": "string"

```

```

5485         },
5486         "minItems": 1,
5487         "type": "array"
5488     },
5489     "rt": {
5490         "description": "When present, the ACE only applies when the rt (Resource type)
matches\nResource Type of the Resource.",
5491         "items": {
5492             "maxLength": 64,
5493             "type": "string"
5494         },
5495         "minItems": 1,
5496         "type": "array"
5497     },
5498     },
5499     "wc": {
5500         "description": "A wildcard matching policy.",
5501         "pattern": "^[+*]$",
5502         "type": "string"
5503     }
5504 },
5505 "type": "object"
5506 },
5507 "type": "array"
5508 },
5509 "subject": {
5510     "anyOf": [
5511         {
5512             "description": "This is the Device identifier.",
5513             "properties": {
5514                 "uuid": {
5515                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4122.",
5516                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5517 fA-F0-9]{12}$",
5518                     "type": "string"
5519                 }
5520             },
5521         },
5522         "required": [
5523             "uuid"
5524         ],
5525         "type": "object"
5526     },
5527     {
5528         "description": "Security role specified as an <Authority> & <Rolename>. A NULL
<Authority> refers to the local entity or Device.",
5529         "properties": {
5530             "authority": {
5531                 "description": "The Authority component of the entity being identified. A
5532 NULL <Authority> refers to the local entity or Device.",
5533                 "type": "string"
5534             },
5535             "role": {
5536                 "description": "The ID of the role being identified.",
5537                 "type": "string"
5538             }
5539         },
5540     },
5541     "required": [
5542         "role"
5543     ],
5544     "type": "object"
5545 },
5546 {
5547     "properties": {
5548         "conntype": {
5549             "description": "This property allows an ACE to be matched based on the
5550 connection or message type.",
5551             "x-detail-desc": [
5552                 "auth-crypt - ACE applies if the Client is authenticated and the data
5553 channel or message is encrypted and integrity protected",
5554                 "anon-clear - ACE applies if the Client is not authenticated and the data
5555 channel or message is not encrypted but may be integrity protected"

```

```

5556         ],
5557         "enum": [
5558             "auth-crypt",
5559             "anon-clear"
5560         ],
5561         "type": "string"
5562     }
5563 },
5564 "required": [
5565     "conntype"
5566 ],
5567 "type": "object"
5568 }
5569 ]
5570 },
5571 "validity": {
5572     "description": "validity is an array of time-pattern objects.",
5573     "items": {
5574         "description": "The time-pattern contains a period and recurrence expressed in
5575 RFC5545 syntax.",
5576         "properties": {
5577             "period": {
5578                 "description": "String represents a period using the RFC5545 Period.",
5579                 "type": "string"
5580             },
5581             "recurrence": {
5582                 "description": "String array represents a recurrence rule using the RFC5545
5583 Recurrence.",
5584                 "items": {
5585                     "type": "string"
5586                 },
5587                 "type": "array"
5588             }
5589         },
5590         "required": [
5591             "period"
5592         ],
5593         "type": "object"
5594     },
5595     "type": "array"
5596 }
5597 },
5598 "required": [
5599     "aceid",
5600     "resources",
5601     "permission",
5602     "subject"
5603 ],
5604 "type": "object"
5605 },
5606 "type": "array"
5607 },
5608 "n": {
5609     "$ref":
5610     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5611     schema.json#/definitions/n"
5612 },
5613 "id": {
5614     "$ref":
5615     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5616     schema.json#/definitions/id"
5617 },
5618 "if" : {
5619     "description": "The interface set supported by this Resource.",
5620     "items": {
5621         "enum": [
5622             "oic.if.baseline"
5623         ],
5624         "type": "string"
5625     },
5626     "minItems": 1,

```

```

5627         "maxItems": 1,
5628         "readOnly": true,
5629         "type": "array"
5630     },
5631 },
5632 "type" : "object",
5633 "required": ["acllist2", "rowneruuid"]
5634 },
5635 "Acl2-Update" : {
5636     "properties": {
5637         "rowneruuid" : {
5638             "description": "The value identifies the unique Resource owner\n Format pattern according
5639 to IETF RFC 4122.",
5640             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5641 9]{12}$",
5642             "type": "string"
5643         },
5644         "acllist2" : {
5645             "description": "Access Control Entries in the ACL Resource.",
5646             "items": {
5647                 "properties": {
5648                     "aceid": {
5649                         "description": "An identifier for the ACE that is unique within the ACL. In cases
5650 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5651                         "minimum": 1,
5652                         "type": "integer"
5653                     },
5654                     "permission": {
5655                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5656 permissions.",
5657                         "x-detail-desc": [
5658                             "0 - No permissions",
5659                             "1 - Create permission is granted",
5660                             "2 - Read, observe, discover permission is granted",
5661                             "4 - Write, update permission is granted",
5662                             "8 - Delete permission is granted",
5663                             "16 - Notify permission is granted"
5664                         ],
5665                         "maximum": 31,
5666                         "minimum": 0,
5667                         "type": "integer"
5668                     },
5669                     "resources": {
5670                         "description": "References the application's Resources to which a security policy
5671 applies.",
5672                         "items": {
5673                             "description": "Each Resource must have at least one of these properties set.",
5674                             "properties": {
5675                                 "href": {
5676                                     "description": "When present, the ACE only applies when the href matches\nThis
5677 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5678                                     "format": "uri",
5679                                     "maxLength": 256,
5680                                     "type": "string"
5681                                 },
5682                                 "if": {
5683                                     "description": "When present, the ACE only applies when the if (interface)
5684 matches\nThe interface set supported by this Resource.",
5685                                     "items": {
5686                                         "enum": [
5687                                             "oic.if.baseline",
5688                                             "oic.if.ll",
5689                                             "oic.if.b",
5690                                             "oic.if.rw",
5691                                             "oic.if.r",
5692                                             "oic.if.a",
5693                                             "oic.if.s"
5694                                         ],
5695                                         "type": "string"
5696                                     },
5697                                     "minItems": 1,

```

```

5698         "type": "array"
5699     },
5700     "rt": {
5701         "description": "When present, the ACE only applies when the rt (Resource type)
5702 matches\nResource Type of the Resource.",
5703         "items": {
5704             "maxLength": 64,
5705             "type": "string"
5706         },
5707         "minItems": 1,
5708         "type": "array"
5709     },
5710     "wc": {
5711         "description": "A wildcard matching policy.",
5712         "x-detail-desc": [
5713             "+ - Matches all discoverable Resources",
5714             "- - Matches all non-discoverable Resources",
5715             "* - Matches all Resources"
5716         ],
5717         "enum": [
5718             "+",
5719             "-",
5720             "*"
5721         ],
5722         "type": "string"
5723     }
5724 },
5725 "type": "object"
5726 },
5727 "type": "array"
5728 },
5729 "subject": {
5730     "anyOf": [
5731         {
5732             "description": "This is the Device identifier.",
5733             "properties": {
5734                 "uuid": {
5735                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5736 4122.",
5737                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5738 fA-F0-9]{12}$",
5739                     "type": "string"
5740                 }
5741             },
5742             "required": [
5743                 "uuid"
5744             ],
5745             "type": "object"
5746         },
5747         {
5748             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5749 <Authority> refers to the local entity or Device.",
5750             "properties": {
5751                 "authority": {
5752                     "description": "The Authority component of the entity being identified. A
5753 NULL <Authority> refers to the local entity or Device.",
5754                     "type": "string"
5755                 },
5756                 "role": {
5757                     "description": "The ID of the role being identified.",
5758                     "type": "string"
5759                 }
5760             },
5761             "required": [
5762                 "role"
5763             ],
5764             "type": "object"
5765         },
5766         {
5767             "properties": {
5768                 "conntype": {

```

```

5769         "description": "This property allows an ACE to be matched based on the
5770 connection or message type.",
5771         "x-detail-desc": [
5772             "auth-crypt - ACE applies if the Client is authenticated and the data
5773 channel or message is encrypted and integrity protected",
5774             "anon-clear - ACE applies if the Client is not authenticated and the data
5775 channel or message is not encrypted but may be integrity protected"
5776         ],
5777         "enum": [
5778             "auth-crypt",
5779             "anon-clear"
5780         ],
5781         "type": "string"
5782     }
5783 },
5784     "required": [
5785         "conntype"
5786     ],
5787     "type": "object"
5788 }
5789 ],
5790 },
5791     "validity": {
5792         "description": "validity is an array of time-pattern objects.",
5793         "items": {
5794             "description": "The time-pattern contains a period and recurrence expressed in
5795 RFC5545 syntax.",
5796             "properties": {
5797                 "period": {
5798                     "description": "String represents a period using the RFC5545 Period.",
5799                     "type": "string"
5800                 },
5801                 "recurrence": {
5802                     "description": "String array represents a recurrence rule using the RFC5545
5803 Recurrence.",
5804                     "items": {
5805                         "type": "string"
5806                     },
5807                     "type": "array"
5808                 }
5809             },
5810             "required": [
5811                 "period"
5812             ],
5813             "type": "object"
5814         },
5815         "type": "array"
5816     }
5817 },
5818     "required": [
5819         "resources",
5820         "permission",
5821         "subject"
5822     ],
5823     "type": "object"
5824 },
5825     "type": "array"
5826 }
5827 },
5828     "type": "object"
5829 }
5830 }
5831 }
5832

```

### 5833 C.4.5 Property definition

5834 Table C.4 defines the Properties that are part of the "oic.r.acl2" Resource Type.

**Table C.4 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".**

Property name	Value type	Mandatory	Access mode	Description
aclist2	array: see schema	No	Read Write	Access Control Entries in the ACL Resource.
rowneruuid	string	No	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL Resource.
rowneruuid	string	Yes	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

**5836 C.4.6 CRUDN behaviour**

5837 Table C.5 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

**5838 Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

**5839 C.5 Managed Access Control****5840 C.5.1 Introduction**

5841 This Resource specifies the host Resources with access permission that is managed by an AMS.

**5842 C.5.2 Well-known URI**

5843 /oic/sec/amacl

**5844 C.5.3 Resource type**

5845 The Resource Type is defined as: "oic.r.amacl".

**5846 C.5.4 OpenAPI 2.0 definition**

```
5847 {
5848   "swagger": "2.0",
5849   "info": {
5850     "title": "Managed Access Control",
```

```

5851     "version": "20190111",
5852     "license": {
5853         "name": "OCF Data Model License",
5854         "url":
5855         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5856         CENSE.md",
5857         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5858         reserved."
5859     },
5860     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5861 },
5862 "schemas": ["http"],
5863 "consumes": ["application/json"],
5864 "produces": ["application/json"],
5865 "paths": {
5866     "/oic/sec/amacl" : {
5867         "get": {
5868             "description": "This Resource specifies the host Resources with access permission that is
5869 managed by an AMS.\n",
5870             "parameters": [
5871                 {"$ref": "#/parameters/interface"}
5872             ],
5873             "responses": {
5874                 "200": {
5875                     "description": "",
5876                     "x-example":
5877                     {
5878                         "rt" : ["oic.r.amacl"],
5879                         "resources": [
5880                             {
5881                                 "href": "/temp",
5882                                 "rt": ["oic.r.temperature"],
5883                                 "if": ["oic.if.baseline", "oic.if.a"]
5884                             },
5885                             {
5886                                 "href": "/temp",
5887                                 "rt": ["oic.r.temperature"],
5888                                 "if": ["oic.if.baseline", "oic.if.s"]
5889                             }
5890                         ]
5891                     },
5892                     "schema": { "$ref": "#/definitions/Amacl" }
5893                 }
5894             }
5895         },
5896         "post": {
5897             "description": "Sets the new amacl data.\n",
5898             "parameters": [
5899                 {"$ref": "#/parameters/interface"},
5900                 {
5901                     "name": "body",
5902                     "in": "body",
5903                     "required": true,
5904                     "schema": { "$ref": "#/definitions/Amacl" },
5905                     "x-example":
5906                     {
5907                         "resources": [
5908                             {
5909                                 "href": "/temp",
5910                                 "rt": ["oic.r.temperature"],
5911                                 "if": ["oic.if.baseline", "oic.if.a"]
5912                             },
5913                             {
5914                                 "href": "/temp",
5915                                 "rt": ["oic.r.temperature"],
5916                                 "if": ["oic.if.baseline", "oic.if.s"]
5917                             }
5918                         ]
5919                     }
5920                 }
5921             ],

```



```

5922     "responses": {
5923         "400": {
5924             "description": "The request is invalid."
5925         },
5926         "201": {
5927             "description": "The AMACL entry is created."
5928         },
5929         "204": {
5930             "description": "The AMACL entry is updated."
5931         }
5932     },
5933 },
5934 "put": {
5935     "description": "Creates the new acl data.\n",
5936     "parameters": [
5937         {"$ref": "#/parameters/interface"},
5938         {
5939             "name": "body",
5940             "in": "body",
5941             "required": true,
5942             "schema": { "$ref": "#/definitions/Amacl" },
5943             "x-example":
5944                 {
5945                     "resources": [
5946                         {
5947                             "href": "/temp",
5948                             "rt": ["oic.r.temperature"],
5949                             "if": ["oic.if.baseline", "oic.if.a"]
5950                         },
5951                         {
5952                             "href": "/temp",
5953                             "rt": ["oic.r.temperature"],
5954                             "if": ["oic.if.baseline", "oic.if.s"]
5955                         }
5956                     ]
5957                 }
5958         }
5959     ],
5960     "responses": {
5961         "400": {
5962             "description": "The request is invalid."
5963         },
5964         "201": {
5965             "description": "The AMACL entry is created."
5966         }
5967     }
5968 },
5969 "delete": {
5970     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
5971     entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
5972     matched entry is deleted.\n",
5973     "parameters": [
5974         {"$ref": "#/parameters/interface"},
5975         {
5976             "in": "query",
5977             "description": "Delete the ACE identified by the string matching the subject value.\n",
5978             "type": "string",
5979             "name": "subject"
5980         }
5981     ],
5982     "responses": {
5983         "200": {
5984             "description": "The ACE instance or the the entire AMACL Resource has been
5985     successfully deleted."
5986         },
5987         "400": {
5988             "description": "The request is invalid."
5989         }
5990     }
5991 }
5992 }

```

```

5993     },
5994     "parameters": {
5995         "interface" : {
5996             "in" : "query",
5997             "name" : "if",
5998             "type" : "string",
5999             "enum" : ["oic.if.baseline"]
6000         }
6001     },
6002     "definitions": {
6003         "Amacl" : {
6004             "properties": {
6005                 "rt" : {
6006                     "description": "Resource Type of the Resource.",
6007                     "items": {
6008                         "maxLength": 64,
6009                         "type": "string",
6010                         "enum": ["oic.r.amacl"]
6011                     },
6012                     "minItems": 1,
6013                     "maxItems": 1,
6014                     "readOnly": true,
6015                     "type": "array"
6016                 },
6017                 "n": {
6018                     "$ref":
6019 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6020 schema.json#/definitions/n"
6021                 },
6022                 "id": {
6023                     "$ref":
6024 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6025 schema.json#/definitions/id"
6026                 },
6027                 "resources" : {
6028                     "description": "Multiple links to this host's Resources.",
6029                     "items": {
6030                         "description": "Each Resource must have at least one of these properties set.",
6031                         "properties": {
6032                             "href": {
6033                                 "description": "When present, the ACE only applies when the href matches\nThis is
6034 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6035                                 "format": "uri",
6036                                 "maxLength": 256,
6037                                 "type": "string"
6038                             },
6039                             "if": {
6040                                 "description": "When present, the ACE only applies when the if (interface)
6041 matches\nThe interface set supported by this Resource.",
6042                                 "items": {
6043                                     "enum": [
6044                                         "oic.if.baseline",
6045                                         "oic.if.ll",
6046                                         "oic.if.b",
6047                                         "oic.if.rw",
6048                                         "oic.if.r",
6049                                         "oic.if.a",
6050                                         "oic.if.s"
6051                                     ],
6052                                     "type": "string"
6053                                 },
6054                                 "minItems": 1,
6055                                 "type": "array"
6056                             },
6057                             "rt": {
6058                                 "description": "When present, the ACE only applies when the rt (Resource type)
6059 matches\nResource Type of the Resource.",
6060                                 "items": {
6061                                     "maxLength": 64,
6062                                     "type": "string"
6063                                 },

```

```

6064         "minItems": 1,
6065         "type": "array"
6066     },
6067     "wc": {
6068         "description": "A wildcard matching policy.",
6069         "pattern": "^[--+]*$",
6070         "type": "string"
6071     }
6072 },
6073     "type": "object"
6074 },
6075     "type": "array"
6076 },
6077     "if" : {
6078         "description": "The interface set supported by this Resource.",
6079         "items": {
6080             "enum": [
6081                 "oic.if.baseline"
6082             ],
6083             "type": "string"
6084         },
6085         "minItems": 1,
6086         "maxItems": 1,
6087         "readOnly": true,
6088         "type": "array"
6089     }
6090 },
6091     "type" : "object",
6092     "required": ["resources"]
6093 }
6094 }
6095 }
6096

```

6097 **C.5.5 Property definition**

6098 Table C.6 defines the Properties that are part of the "oic.r.amacl" Resource Type.

6099 **Table C.6 – The Property definitions of the Resource with type "rt" = "oic.r.amacl".**

Property name	Value type	Mandatory	Access mode	Description
resources	array: see schema	Yes	Read Write	Multiple links to this host's Resources.
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
id	multiple types: see schema	No	Read Write	

6100 **C.5.6 CRUDN behaviour**

6101 Table C.7 defines the CRUDN operations that are supported on the "oic.r.amacl" Resource Type.

6102 **Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl".**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

## 6103 C.6 Credential

### 6104 C.6.1 Introduction

6105 This Resource specifies credentials a Device may use to establish secure communication.

6106 Retrieves the credential data.

6107 When used without query parameters, all the credential entries are returned.

6108 When used with a query parameter, only the credentials matching the specified  
6109 parameter are returned.

6110

6111 Note that write-only credential data will not be returned.

6112

### 6113 C.6.2 Well-known URI

6114 /oic/sec/cred

### 6115 C.6.3 Resource type

6116 The Resource Type is defined as: "oic.r.cred".

### 6117 C.6.4 OpenAPI 2.0 definition

```
6118 {
6119   "swagger": "2.0",
6120   "info": {
6121     "title": "Credential",
6122     "version": "v1.0-20181031",
6123     "license": {
6124       "name": "OCF Data Model License",
6125       "url":
6126         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6127         CENSE.md",
6128       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6129         reserved."
6130     },
6131     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6132   },
6133   "schemes": ["http"],
6134   "consumes": ["application/json"],
6135   "produces": ["application/json"],
6136   "paths": {
6137     "/oic/sec/cred" : {
6138       "get": {
6139         "description": "This Resource specifies credentials a Device may use to establish secure
6140         communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
6141         credential entries are returned.\nWhen used with a query parameter, only the credentials matching
6142         the specified\nparameter are returned.\n\nNote that write-only credential data will not be
6143         returned.\n",
6144         "parameters": [
6145           {"$ref": "#/parameters/interface"}
6146           , {"$ref": "#/parameters/cred-filtered-credid"}
6147           , {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6148         ],
6149         "responses": {
6150           "200": {
6151             "description": "",
6152             "x-example":
6153               {
6154                 "xt": ["oic.r.cred"],
6155                 "creds": [
6156                   {
6157                     "credid": 55,
6158                     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6159                     "roleid": {
6160                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6161                       "role": "SOME_STRING"
6162                     },
6163                     "credtype": 32,
6164                     "publicdata": {
```

```

6165         "encoding": "oic.sec.encoding.base64",
6166         "data": "BASE-64-ENCODED-VALUE"
6167     },
6168     "privatedata": {
6169         "encoding": "oic.sec.encoding.base64",
6170         "data": "BASE-64-ENCODED-VALUE",
6171         "handle": 4
6172     },
6173     "optionaldata": {
6174         "revstat": false,
6175         "encoding": "oic.sec.encoding.base64",
6176         "data": "BASE-64-ENCODED-VALUE"
6177     },
6178     "period": "20160101T180000Z/20170102T070000Z",
6179     "crms": [ "oic.sec.crm.pk10" ]
6180 },
6181 {
6182     "credid": 56,
6183     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6184     "roleid": {
6185         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6186         "role": "SOME_STRING"
6187     },
6188     "credtype": 1,
6189     "publicdata": {
6190         "encoding": "oic.sec.encoding.base64",
6191         "data": "BASE-64-ENCODED-VALUE"
6192     },
6193     "privatedata": {
6194         "encoding": "oic.sec.encoding.base64",
6195         "data": "BASE-64-ENCODED-VALUE",
6196         "handle": 4
6197     },
6198     "optionaldata": {
6199         "revstat": false,
6200         "encoding": "oic.sec.encoding.base64",
6201         "data": "BASE-64-ENCODED-VALUE"
6202     },
6203     "period": "20160101T180000Z/20170102T070000Z",
6204     "crms": [ "oic.sec.crm.pk10" ]
6205 }
6206 ],
6207     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6208 }
6209 ,
6210     "schema": { "$ref": "#/definitions/Cred" }
6211 },
6212     "400": {
6213         "description": "The request is invalid."
6214     }
6215 },
6216 },
6217     "post": {
6218         "description": "Updates the credential Resource with the provided
6219 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
6220 Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
6221 Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
6222 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
6223 Resource.\n",
6224         "parameters": [
6225             { "$ref": "#/parameters/interface" },
6226             {
6227                 "name": "body",
6228                 "in": "body",
6229                 "required": true,
6230                 "schema": { "$ref": "#/definitions/Cred-Update" },
6231                 "x-example":
6232                 {
6233                     "creds": [
6234                         {
6235                             "credid": 55,

```

```

6236         "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6237         "roleid": {
6238             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6239             "role": "SOME_STRING"
6240         },
6241         "credtype": 32,
6242         "publicdata": {
6243             "encoding": "oic.sec.encoding.base64",
6244             "data": "BASE-64-ENCODED-VALUE"
6245         },
6246         "privatedata": {
6247             "encoding": "oic.sec.encoding.base64",
6248             "data": "BASE-64-ENCODED-VALUE",
6249             "handle": 4
6250         },
6251         "optionaldata": {
6252             "revstat": false,
6253             "encoding": "oic.sec.encoding.base64",
6254             "data": "BASE-64-ENCODED-VALUE"
6255         },
6256         "period": "20160101T180000Z/20170102T070000Z",
6257         "crms": [ "oic.sec.crm.pk10" ]
6258     },
6259     {
6260         "credid": 56,
6261         "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6262         "roleid": {
6263             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6264             "role": "SOME_STRING"
6265         },
6266         "credtype": 1,
6267         "publicdata": {
6268             "encoding": "oic.sec.encoding.base64",
6269             "data": "BASE-64-ENCODED-VALUE"
6270         },
6271         "privatedata": {
6272             "encoding": "oic.sec.encoding.base64",
6273             "data": "BASE-64-ENCODED-VALUE",
6274             "handle": 4
6275         },
6276         "optionaldata": {
6277             "revstat": false,
6278             "encoding": "oic.sec.encoding.base64",
6279             "data": "BASE-64-ENCODED-VALUE"
6280         },
6281         "period": "20160101T180000Z/20170102T070000Z",
6282         "crms": [ "oic.sec.crm.pk10" ]
6283     }
6284 ],
6285     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6286 }
6287 }
6288 ],
6289 "responses": {
6290     "400": {
6291         "description": "The request is invalid."
6292     },
6293     "201": {
6294         "description": "The credential entry is created."
6295     },
6296     "204": {
6297         "description": "The credential entry is updated."
6298     }
6299 }
6300 },
6301 "delete": {
6302     "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
6303 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
6304 matching\nthe query parameter are deleted.\n",
6305     "parameters": [
6306         {"$ref": "#/parameters/interface"},

```

```

6307         {"$ref": "#/parameters/cred-filtered-credid"},
6308         {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6309     ],
6310     "responses": {
6311         "400": {
6312             "description": "The request is invalid."
6313         },
6314         "204": {
6315             "description": "The specific credential(s) or the the entire credential Resource has
6316 been successfully deleted."
6317         }
6318     }
6319 }
6320 }
6321 },
6322 "parameters": {
6323     "interface" : {
6324         "in" : "query",
6325         "name" : "if",
6326         "type" : "string",
6327         "enum" : ["oic.if.baseline"]
6328     },
6329     "cred-filtered-credid" : {
6330         "in" : "query",
6331         "name" : "credid",
6332         "required" : false,
6333         "type" : "integer",
6334         "description" : "Only applies to the credential with the specified credid.",
6335         "x-example" : 2112
6336     },
6337     "cred-filtered-subjectuuid" : {
6338         "in" : "query",
6339         "name" : "subjectuuid",
6340         "required" : false,
6341         "type" : "string",
6342         "description" : "Only applies to credentials with the specified subject UUID.",
6343         "x-example" : "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6344     }
6345 },
6346 "definitions": {
6347     "Cred" : {
6348         "properties": {
6349             "rowneruuid" : {
6350                 "description": "Format pattern according to IETF RFC 4122.",
6351                 "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}$",
6352                 "type": "string"
6353             },
6354             "rt" : {
6355                 "description": "Resource Type of the Resource.",
6356                 "items": {
6357                     "maxLength": 64,
6358                     "type": "string",
6359                     "enum": ["oic.r.cred"]
6360                 },
6361                 "minItems": 1,
6362                 "readOnly": true,
6363                 "type": "array",
6364                 "uniqueItems": true
6365             },
6366             "n": {
6367                 "$ref":
6368 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6369 schema.json#/definitions/n"
6370             },
6371             "id": {
6372                 "$ref":
6373 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6374 schema.json#/definitions/id"
6375             },
6376             "creds" : {
6377                 "description": "List of credentials available at this Resource.",

```

```

6378     "items": {
6379         "properties": {
6380             "credid": {
6381                 "description": "Local reference to a credential Resource.",
6382                 "type": "integer"
6383             },
6384             "credtype": {
6385                 "description": "Representation of this credential's type\nCredential Types - Cred
6386 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6387 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6388 password32 - Asymmetric encryption key.",
6389                 "maximum": 63,
6390                 "minimum": 0,
6391                 "type": "integer"
6392             },
6393             "credusage": {
6394                 "description": "A string that provides hints about how/where the cred is used\nThe
6395 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6396 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6397 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6398                 "enum": [
6399                     "oic.sec.cred.trustca",
6400                     "oic.sec.cred.cert",
6401                     "oic.sec.cred.rolecert",
6402                     "oic.sec.cred.mfgtrustca",
6403                     "oic.sec.cred.mfgcert"
6404                 ],
6405                 "type": "string"
6406             },
6407             "crms": {
6408                 "description": "The refresh methods that may be used to update this credential.",
6409                 "items": {
6410                     "description": "Each enum represents a method by which the credentials are
6411 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6412 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6413 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6414 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6415                     "enum": [
6416                         "oic.sec.crm.pro",
6417                         "oic.sec.crm.psk",
6418                         "oic.sec.crm.rdp",
6419                         "oic.sec.crm.skdc",
6420                         "oic.sec.crm.pk10"
6421                     ],
6422                     "type": "string"
6423                 },
6424                 "type": "array",
6425                 "uniqueItems": true
6426             },
6427             "optionaldata": {
6428                 "description": "Credential revocation status information\nOptional credential
6429 contents describes revocation status for this credential.",
6430                 "properties": {
6431                     "data": {
6432                         "description": "The encoded structure.",
6433                         "type": "string"
6434                     },
6435                     "encoding": {
6436                         "description": "A string specifying the encoding format of the data contained in
6437 the optdata.",
6438                         "x-detail-desc": [
6439                             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6440                             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6441                             "oic.sec.encoding.base64 - Base64 encoded object.",
6442                             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6443                             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6444                             "oic.sec.encoding.raw - Raw hex encoded data."
6445                         ],
6446                         "enum": [
6447                             "oic.sec.encoding.jwt",
6448                             "oic.sec.encoding.cwt",

```



```

6449         "oic.sec.encoding.base64",
6450         "oic.sec.encoding.pem",
6451         "oic.sec.encoding.der",
6452         "oic.sec.encoding.raw"
6453     ],
6454     "type": "string"
6455 },
6456 "revstat": {
6457     "description": "Revocation status flag - true = revoked.",
6458     "type": "boolean"
6459 },
6460 },
6461 "required": [
6462     "revstat"
6463 ],
6464 "type": "object"
6465 },
6466 "period": {
6467     "description": "String with RFC5545 Period.",
6468     "type": "string"
6469 },
6470 "privatedata": {
6471     "description": "Private credential information\nCredential Resource non-public
6472 contents.",
6473     "properties": {
6474         "data": {
6475             "description": "The encoded value.",
6476             "maxLength": 3072,
6477             "type": "string"
6478         },
6479         "encoding": {
6480             "description": "A string specifying the encoding format of the data contained in
6481 the privdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6482 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6483 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6484 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data.",
6485             "enum": [
6486                 "oic.sec.encoding.jwt",
6487                 "oic.sec.encoding.cwt",
6488                 "oic.sec.encoding.base64",
6489                 "oic.sec.encoding.uri",
6490                 "oic.sec.encoding.handle",
6491                 "oic.sec.encoding.raw"
6492             ],
6493             "type": "string"
6494         },
6495         "handle": {
6496             "description": "Handle to a key storage Resource.",
6497             "type": "integer"
6498         }
6499     },
6500     "required": [
6501         "encoding"
6502     ],
6503     "type": "object"
6504 },
6505 "publicdata": {
6506     "description": "Public credential information.",
6507     "properties": {
6508         "data": {
6509             "description": "The encoded value.",
6510             "maxLength": 3072,
6511             "type": "string"
6512         },
6513         "encoding": {
6514             "description": "A string specifying the encoding format of the data contained in
6515 the pubdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6516 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6517 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.pem - Encoding for PEM encoded
6518 certificate or chain\noic.sec.encoding.der - Encoding for DER encoded
6519 certificate\noic.sec.encoding.raw - Raw hex encoded data.",

```

```

6520         "enum": [
6521             "oic.sec.encoding.jwt",
6522             "oic.sec.encoding.cwt",
6523             "oic.sec.encoding.base64",
6524             "oic.sec.encoding.uri",
6525             "oic.sec.encoding.pem",
6526             "oic.sec.encoding.der",
6527             "oic.sec.encoding.raw"
6528         ],
6529         "type": "string"
6530     },
6531 },
6532 "type": "object"
6533 },
6534 "roleid": {
6535     "description": "The role this credential possesses\nSecurity role specified as an
6536 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6537     "properties": {
6538         "authority": {
6539             "description": "The Authority component of the entity being identified. A NULL
6540 <Authority> refers to the local entity or Device.",
6541             "type": "string"
6542         },
6543         "role": {
6544             "description": "The ID of the role being identified.",
6545             "type": "string"
6546         }
6547     },
6548     "required": [
6549         "role"
6550     ],
6551     "type": "object"
6552 },
6553 "subjectuuid": {
6554     "anyOf": [
6555         {
6556             "description": "The id of the Device, which the cred entry applies to or \"*\n
6557 for wildcard identity.",
6558             "pattern": "^[\\*]*$",
6559             "type": "string"
6560         },
6561         {
6562             "description": "Format pattern according to IETF RFC 4122.",
6563             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6564 F0-9]{12}$",
6565             "type": "string"
6566         }
6567     ]
6568 },
6569 },
6570 "type": "object"
6571 },
6572 "type": "array"
6573 },
6574 "if" : {
6575     "description": "The interface set supported by this Resource.",
6576     "items": {
6577         "enum": [
6578             "oic.if.baseline"
6579         ],
6580         "type": "string"
6581     },
6582     "minItems": 1,
6583     "readOnly": true,
6584     "type": "array"
6585 }
6586 },
6587 "type" : "object",
6588 "required": ["creds", "rowneruuid"]
6589 },
6590 "Cred-Update" : {

```

```

6591     "properties": {
6592         "rowneruuid" : {
6593             "description": "Format pattern according to IETF RFC 4122.",
6594             "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}$",
6595             "type": "string"
6596         },
6597         "creds" : {
6598             "description": "List of credentials available at this Resource.",
6599             "items": {
6600                 "properties": {
6601                     "credid": {
6602                         "description": "Local reference to a credential Resource.",
6603                         "type": "integer"
6604                     },
6605                     "credtype": {
6606                         "description": "Representation of this credential's type\nCredential Types - Cred
6607 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6608 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6609 password32 - Asymmetric encryption key.",
6610                         "maximum": 63,
6611                         "minimum": 0,
6612                         "type": "integer"
6613                     },
6614                     "credusage": {
6615                         "description": "A string that provides hints about how/where the cred is used\nThe
6616 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6617 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6618 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6619                         "enum": [
6620                             "oic.sec.cred.trustca",
6621                             "oic.sec.cred.cert",
6622                             "oic.sec.cred.rolecert",
6623                             "oic.sec.cred.mfgtrustca",
6624                             "oic.sec.cred.mfgcert"
6625                         ],
6626                         "type": "string"
6627                     },
6628                     "crms": {
6629                         "description": "The refresh methods that may be used to update this credential.",
6630                         "items": {
6631                             "description": "Each enum represents a method by which the credentials are
6632 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6633 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6634 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6635 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6636                             "enum": [
6637                                 "oic.sec.crm.pro",
6638                                 "oic.sec.crm.psk",
6639                                 "oic.sec.crm.rdp",
6640                                 "oic.sec.crm.skdc",
6641                                 "oic.sec.crm.pk10"
6642                             ],
6643                             "type": "string"
6644                         },
6645                         "type": "array"
6646                     },
6647                     "optionaldata": {
6648                         "description": "Credential revocation status information\nOptional credential
6649 contents describes revocation status for this credential.",
6650                         "properties": {
6651                             "data": {
6652                                 "description": "The encoded structure.",
6653                                 "type": "string"
6654                             },
6655                             "encoding": {
6656                                 "description": "A string specifying the encoding format of the data contained in
6657 the optdata.",
6658                                 "x-detail-desc": [
6659                                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6660                                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6661                                     "oic.sec.encoding.base64 - Base64 encoded object.",

```

```

6662         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6663         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6664         "oic.sec.encoding.raw - Raw hex encoded data."
6665     ],
6666     "enum": [
6667         "oic.sec.encoding.jwt",
6668         "oic.sec.encoding.cwt",
6669         "oic.sec.encoding.base64",
6670         "oic.sec.encoding.pem",
6671         "oic.sec.encoding.der",
6672         "oic.sec.encoding.raw"
6673     ],
6674     "type": "string"
6675 },
6676 "revstat": {
6677     "description": "Revocation status flag - true = revoked.",
6678     "type": "boolean"
6679 }
6680 },
6681 "required": [
6682     "revstat"
6683 ],
6684 "type": "object"
6685 },
6686 "period": {
6687     "description": "String with RFC5545 Period.",
6688     "type": "string"
6689 },
6690 "privatedata": {
6691     "description": "Private credential information\nCredential Resource non-public
6692 contents.",
6693     "properties": {
6694         "data": {
6695             "description": "The encoded value.",
6696             "maxLength": 3072,
6697             "type": "string"
6698         },
6699         "encoding": {
6700             "description": "A string specifying the encoding format of the data contained in
6701 the privdata.",
6702             "x-detail-desc": [
6703                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6704                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6705                 "oic.sec.encoding.base64 - Base64 encoded object.",
6706                 "oic.sec.encoding.uri - URI reference.",
6707                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
6708 referenced using a handle.",
6709                 "oic.sec.encoding.raw - Raw hex encoded data."
6710             ],
6711             "enum": [
6712                 "oic.sec.encoding.jwt",
6713                 "oic.sec.encoding.cwt",
6714                 "oic.sec.encoding.base64",
6715                 "oic.sec.encoding.uri",
6716                 "oic.sec.encoding.handle",
6717                 "oic.sec.encoding.raw"
6718             ],
6719             "type": "string"
6720         },
6721         "handle": {
6722             "description": "Handle to a key storage Resource.",
6723             "type": "integer"
6724         }
6725     },
6726     "required": [
6727         "encoding"
6728     ],
6729     "type": "object"
6730 },
6731 "publicdata": {
6732     "properties": {

```

```

6733         "data": {
6734             "description": "The encoded value.",
6735             "maxLength": 3072,
6736             "type": "string"
6737         },
6738         "encoding": {
6739             "description": "Public credential information\nA string specifying the encoding
6740 format of the data contained in the pubdata.",
6741             "x-detail-desc": [
6742                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6743                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6744                 "oic.sec.encoding.base64 - Base64 encoded object.",
6745                 "oic.sec.encoding.uri - URI reference.",
6746                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6747                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6748                 "oic.sec.encoding.raw - Raw hex encoded data."
6749             ],
6750             "enum": [
6751                 "oic.sec.encoding.jwt",
6752                 "oic.sec.encoding.cwt",
6753                 "oic.sec.encoding.base64",
6754                 "oic.sec.encoding.uri",
6755                 "oic.sec.encoding.pem",
6756                 "oic.sec.encoding.der",
6757                 "oic.sec.encoding.raw"
6758             ],
6759             "type": "string"
6760         }
6761     },
6762     "type": "object"
6763 },
6764 "roleid": {
6765     "description": "The role this credential possesses\nSecurity role specified as an
6766 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6767     "properties": {
6768         "authority": {
6769             "description": "The Authority component of the entity being identified. A NULL
6770 <Authority> refers to the local entity or Device.",
6771             "type": "string"
6772         },
6773         "role": {
6774             "description": "The ID of the role being identified.",
6775             "type": "string"
6776         }
6777     },
6778     "required": [
6779         "role"
6780     ],
6781     "type": "object"
6782 },
6783 "subjectuuid": {
6784     "anyOf": [
6785         {
6786             "description": "The id of the Device, which the cred entry applies to or \"*\n
6787 for wildcard identity.",
6788             "pattern": "^\\*$",
6789             "type": "string"
6790         },
6791         {
6792             "description": "Format pattern according to IETF RFC 4122.",
6793             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6794 F0-9]{12}$",
6795             "type": "string"
6796         }
6797     ]
6798 },
6799 },
6800 "type": "object"
6801 },
6802 "type": "array"
6803 },

```

```

6804     "if" :
6805         {
6806             "description": "The interface set supported by this Resource.",
6807             "items": {
6808                 "enum": [
6809                     "oic.if.baseline"
6810                 ],
6811                 "type": "string"
6812             },
6813             "minItems": 1,
6814             "readOnly": true,
6815             "type": "array"
6816         },
6817     },
6818     "type" : "object"
6819 }
6820 }
6821 }
6822

```

6823 **C.6.5 Property definition**

6824 Table C.8 defines the Properties that are part of the "oic.r.cred" Resource Type.

6825 **Table C.8 – The Property definitions of the Resource with type "rt" = "oic.r.cred".**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	No	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
creds	array: see schema	No	Read Write	List of credentials available at this Resource.
id	multiple types: see schema	No	Read Write	
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
creds	array: see schema	Yes	Read Write	List of credentials available at this Resource.

6826 **C.6.6 CRUDN behaviour**

6827 Table C.9 defines the CRUDN operations that are supported on the "oic.r.cred" Resource Type.

6828 **Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 6829 C.7 Certificate Revocation

### 6830 C.7.1 Introduction

6831 This Resource specifies certificate revocation lists as X.509 objects.

### 6832 C.7.2 Well-known URI

6833 /oic/sec/crl

### 6834 C.7.3 Resource type

6835 The Resource Type is defined as: "oic.r.crl".

### 6836 C.7.4 OpenAPI 2.0 definition

```
6837 {
6838   "swagger": "2.0",
6839   "info": {
6840     "title": "Certificate Revocation",
6841     "version": "v1.0-20150819",
6842     "license": {
6843       "name": "OCF Data Model License",
6844       "url":
6845         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6846         CENSE.md",
6847       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6848       reserved."
6849     },
6850     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6851   },
6852   "schemes": ["http"],
6853   "consumes": ["application/json"],
6854   "produces": ["application/json"],
6855   "paths": {
6856     "/oic/sec/crl" : {
6857       "get": {
6858         "description": "This Resource specifies certificate revocation lists as X.509 objects.\n",
6859         "parameters": [
6860           { "$ref": "#/parameters/interface" }
6861         ],
6862         "responses": {
6863           "200": {
6864             "description": "",
6865             "x-example":
6866               {
6867                 "rt": ["oic.r.crl"],
6868                 "crlid": 1,
6869                 "thisupdate": "2016-04-12T23:20:50.52Z",
6870                 "crldata": "Base64ENCODEDCRL"
6871               },
6872             "schema": { "$ref": "#/definitions/Crl" }
6873           }
6874         }
6875       },
6876       "post": {
6877         "description": "Updates the CRL data.\n",
6878         "parameters": [
6879           { "$ref": "#/parameters/interface" },
6880           {
6881             "name": "body",
6882             "in": "body",
6883             "required": true,
6884             "schema": { "$ref": "#/definitions/Crl-Update" },
6885             "x-example":
6886               {
6887                 "crlid": 1,
6888                 "thisupdate": "2016-04-12T23:20:50.52Z",
6889                 "crldata": "Base64ENCODEDCRL"
6890               }
6891           }
6892         ]
6893       }
6894     }
6895   }
6896 }
```

```

6893     "responses": {
6894         "400": {
6895             "description": "The request is invalid."
6896         },
6897         "204": {
6898             "description": "The CRL entry is updated."
6899         }
6900     }
6901 }
6902 },
6903 },
6904 "parameters": {
6905     "interface": {
6906         "in": "query",
6907         "name": "if",
6908         "type": "string",
6909         "enum": ["oic.if.baseline"]
6910     }
6911 },
6912 "definitions": {
6913     "Crl": {
6914         "properties": {
6915             "rt": {
6916                 "description": "Resource Type of the Resource.",
6917                 "items": {
6918                     "maxLength": 64,
6919                     "type": "string",
6920                     "enum": ["oic.r.crl"]
6921                 },
6922                 "minItems": 1,
6923                 "readOnly": true,
6924                 "type": "array"
6925             },
6926             "n": {
6927                 "$ref":
6928 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6929 schema.json#/definitions/n"
6930             },
6931             "id": {
6932                 "$ref":
6933 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6934 schema.json#/definitions/id"
6935             },
6936             "crlldata": {
6937                 "description": "Base64 BER encoded CRL data.",
6938                 "type": "string"
6939             },
6940             "crlid": {
6941                 "description": "Local reference to a CRL Resource.",
6942                 "type": "integer"
6943             },
6944             "thisupdate": {
6945                 "description": "UTC time of last CRL update.",
6946                 "type": "string"
6947             },
6948             "if": {
6949                 "description": "The interface set supported by this Resource.",
6950                 "items": {
6951                     "enum": [
6952                         "oic.if.baseline"
6953                     ],
6954                     "type": "string"
6955                 },
6956                 "minItems": 1,
6957                 "readOnly": true,
6958                 "type": "array"
6959             }
6960         },
6961         "type": "object",
6962         "required": ["crlid", "thisupdate", "crlldata"]
6963     }

```



```

6964     ,
6965     "Crl-Update": {
6966         "properties": {
6967             "crldata": {
6968                 "description": "Base64 BER encoded CRL data.",
6969                 "type": "string"
6970             },
6971             "crlid": {
6972                 "description": "Local reference to a CRL Resource.",
6973                 "type": "integer"
6974             },
6975             "thisupdate": {
6976                 "description": "UTC time of last CRL update.",
6977                 "type": "string"
6978             }
6979         },
6980         "type" : "object"
6981     }
6982 }
6983 }
6984

```

### 6985 C.7.5 Property definition

6986 Table C.10 defines the Properties that are part of the "oic.r.crl" Resource Type.

6987 **Table C.10 – The Property definitions of the Resource with type "rt" = "oic.r.crl".**

Property name	Value type	Mandatory	Access mode	Description
crldata	string	Yes	Read Write	Base64 BER encoded CRL data.
thisupdate	string	Yes	Read Write	UTC time of last CRL update.
n	multiple types: see schema	No	Read Write	
crlid	integer	Yes	Read Write	Local reference to a CRL Resource.
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
crldata	string		Read Write	Base64 BER encoded CRL data.
thisupdate	string		Read Write	UTC time of last CRL update.
crlid	integer		Read Write	Local reference to a CRL Resource.

### 6988 C.7.6 CRUDN behaviour

6989 Table C.11 defines the CRUDN operations that are supported on the "oic.r.crl" Resource Type.

6990 **Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".**

Create	Read	Update	Delete	Notify
	get	post		observe

## 6991 C.8 Certificate Signing Request

### 6992 C.8.1 Introduction

6993 This Resource specifies a Certificate Signing Request.

### 6994 C.8.2 Well-known URI

6995 /oic/sec/csr

### 6996 C.8.3 Resource type

6997 The Resource Type is defined as: "oic.r.csr".

### 6998 C.8.4 OpenAPI 2.0 definition

```
6999 {
7000   "swagger": "2.0",
7001   "info": {
7002     "title": "Certificate Signing Request",
7003     "version": "v1.0-20150819",
7004     "license": {
7005       "name": "OCF Data Model License",
7006       "url":
7007         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7008         CENSE.md",
7009       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7010         reserved."
7011     },
7012     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7013   },
7014   "schemes": ["http"],
7015   "consumes": ["application/json"],
7016   "produces": ["application/json"],
7017   "paths": {
7018     "/oic/sec/csr" : {
7019       "get": {
7020         "description": "This Resource specifies a Certificate Signing Request.\n",
7021         "parameters": [
7022           {"$ref": "#/parameters/interface"}
7023         ],
7024         "responses": {
7025           "200": {
7026             "description": "",
7027             "x-example":
7028               {
7029                 "rt": ["oic.r.csr"],
7030                 "encoding" : "oic.sec.encoding.pem",
7031                 "csr": "PEMENCODEDCSR"
7032               },
7033             "schema": { "$ref": "#/definitions/Csr" }
7034           },
7035           "404": {
7036             "description" : "The Device does not support certificates and generating CSRs."
7037           },
7038           "503": {
7039             "description" : "The Device is not yet ready to return a response. Try again later."
7040           }
7041         }
7042       }
7043     }
7044   },
7045   "parameters": {
7046     "interface" : {
7047       "in" : "query",
7048       "name" : "if",
7049       "type" : "string",
7050       "enum" : ["oic.if.baseline"]
7051     }
7052   },
7053   "definitions": {
7054     "Csr" : {
```

```

7055     "properties": {
7056       "rt" : {
7057         "description": "Resource Type of the Resource.",
7058         "items": {
7059           "maxLength": 64,
7060           "type": "string",
7061           "enum": ["oic.r.csr"]
7062         },
7063         "minItems": 1,
7064         "readOnly": true,
7065         "type": "array"
7066       },
7067       "encoding": {
7068         "description": "A string specifying the encoding format of the data contained in CSR.",
7069         "x-detail-desc": [
7070           "oic.sec.encoding.pem - Encoding for PEM encoded CSR.",
7071           "oic.sec.encoding.der - Encoding for DER encoded CSR."
7072         ],
7073         "enum": [
7074           "oic.sec.encoding.pem",
7075           "oic.sec.encoding.der"
7076         ],
7077         "readOnly": true,
7078         "type": "string"
7079       },
7080       "n": {
7081         "$ref":
7082         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7083         schema.json#/definitions/n"
7084       },
7085       "id": {
7086         "$ref":
7087         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7088         schema.json#/definitions/id"
7089       },
7090       "csr": {
7091         "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
7092         "maxLength": 3072,
7093         "readOnly": true,
7094         "type": "string"
7095       },
7096       "if": {
7097         "description": "The interface set supported by this Resource.",
7098         "items": {
7099           "enum": [
7100             "oic.if.baseline"
7101           ],
7102           "type": "string"
7103         },
7104         "minItems": 1,
7105         "readOnly": true,
7106         "type": "array"
7107       }
7108     },
7109     "type" : "object",
7110     "required": ["csr", "encoding"]
7111   }
7112 }
7113 }
7114

```

### 7115 C.8.5 Property definition

7116 Table C.12 defines the Properties that are part of the "oic.r.csr" Resource Type.

7117 **Table C.12 – The Property definitions of the Resource with type "rt" = "oic.r.csr".**

Property name	Value type	Mandatory	Access mode	Description
n	multiple types: see schema	No	Read Write	

id	multiple types: see schema	No	Read Write	
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in CSR.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property.

7118 **C.8.6 CRUDN behaviour**

7119 Table C.13 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

7120 **Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".**

Create	Read	Update	Delete	Notify
	get			observe

7121 **C.9 Device Owner Transfer Method**

7122 **C.9.1 Introduction**

7123 This Resource specifies properties needed to establish a Device owner.

7124

7125 **C.9.2 Well-known URI**

7126 /oic/sec/doxm

7127 **C.9.3 Resource type**

7128 The Resource Type is defined as: "oic.r.doxm".

7129 **C.9.4 OpenAPI 2.0 definition**

```

7130 {
7131   "swagger": "2.0",
7132   "info": {
7133     "title": "Device Owner Transfer Method",
7134     "version": "v1.0-20181001",
7135     "license": {
7136       "name": "OCF Data Model License",
7137       "url":
7138 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7139 CENSE.md",
7140       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7141 reserved."
7142     },
7143     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7144   },
7145   "schemes": ["http"],
7146   "consumes": ["application/json"],
7147   "produces": ["application/json"],
7148   "paths": {
7149     "/oic/sec/doxm" : {

```

```

7150     "get": {
7151         "description": "This Resource specifies properties needed to establish a Device owner.\n",
7152         "parameters": [
7153             {"$ref": "#/parameters/interface"}
7154         ],
7155         "responses": {
7156             "200": {
7157                 "description": "",
7158                 "x-example":
7159                 {
7160                     "rt": ["oic.r.doxm"],
7161                     "oxms": [ 0, 2, 3 ],
7162                     "oxmsel": 0,
7163                     "sct": 16,
7164                     "owned": true,
7165                     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7166                     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7167                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7168                 }
7169             },
7170             "schema": { "$ref": "#/definitions/Doxm" }
7171         },
7172         "400": {
7173             "description": "The request is invalid."
7174         }
7175     },
7176 },
7177 "post": {
7178     "description": "Updates the DOXM Resource data.\n",
7179     "parameters": [
7180         {"$ref": "#/parameters/interface"},
7181         {
7182             "name": "body",
7183             "in": "body",
7184             "required": true,
7185             "schema": { "$ref": "#/definitions/Doxm-Update" },
7186             "x-example":
7187             {
7188                 "oxmsel": 0,
7189                 "owned": true,
7190                 "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7191                 "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7192                 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7193             }
7194         }
7195     ],
7196     "responses": {
7197         "400": {
7198             "description": "The request is invalid."
7199         },
7200         "204": {
7201             "description": "The DOXM entry is updated."
7202         }
7203     }
7204 },
7205 },
7206 },
7207 "parameters": {
7208     "interface": {
7209         "in": "query",
7210         "name": "if",
7211         "type": "string",
7212         "enum": ["oic.if.baseline"]
7213     }
7214 },
7215 "definitions": {
7216     "Doxm": {
7217         "properties": {
7218             "rowneruuid": {
7219                 "description": "Format pattern according to IETF RFC 4122.",
7220                 "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}$",

```

```

7221         "type": "string"
7222     },
7223     "oxms": {
7224         "description": "List of supported owner transfer methods.",
7225         "items": {
7226             "description": "The Device owner transfer methods that may be selected at Device on-
7227 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
7228 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
7229 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
7230 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
7231 (deprecated).",
7232             "type": "integer"
7233         },
7234         "readOnly": true,
7235         "type": "array"
7236     },
7237     "devowneruuid": {
7238         "description": "Format pattern according to IETF RFC 4122.",
7239         "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}$",
7240         "type": "string"
7241     },
7242     "deviceuuid": {
7243         "description": "The uuid formatted identity of the Device\nFormat pattern according to
7244 IETF RFC 4122.",
7245         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7246         "type": "string"
7247     },
7248     "owned": {
7249         "description": "Ownership status flag.",
7250         "type": "boolean"
7251     },
7252     "n": {
7253         "$ref":
7254 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7255 schema.json#/definitions/n"
7256     },
7257     "id": {
7258         "$ref":
7259 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7260 schema.json#/definitions/id"
7261     },
7262     "oxmsel": {
7263         "description": "The selected owner transfer method used during on-boarding\nThe Device
7264 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7265 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7266 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7267 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7268 method (oic.sec.doxm.dcap) (deprecated).",
7269         "type": "integer"
7270     },
7271     "sct": {
7272         "description": "Bitmask encoding of supported credential types\nCredential Types -
7273 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7274 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7275 password32 - Asymmetric encryption key.",
7276         "maximum": 63,
7277         "minimum": 0,
7278         "type": "integer",
7279         "readOnly": true
7280     },
7281     "rt" : {
7282         "description": "Resource Type of the Resource.",
7283         "items": {
7284             "maxLength": 64,
7285             "type": "string",
7286             "enum": ["oic.r.doxm"]
7287         },
7288         "minItems": 1,
7289         "readOnly": true,
7290         "type": "array"
7291     },

```

```

7292     "if": {
7293         "description": "The interface set supported by this Resource.",
7294         "items": {
7295             "enum": [
7296                 "oic.if.baseline"
7297             ],
7298             "type": "string"
7299         },
7300         "minItems": 1,
7301         "readOnly": true,
7302         "type": "array"
7303     }
7304 },
7305 "type" : "object",
7306 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
7307 },
7308 "Doxm-Update" : {
7309     "properties": {
7310         "rowneruuid": {
7311             "description": "Format pattern according to IETF RFC 4122.",
7312             "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}$",
7313             "type": "string"
7314         },
7315         "devowneruuid": {
7316             "description": "Format pattern according to IETF RFC 4122.",
7317             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7318             "type": "string"
7319         },
7320         "deviceuuid": {
7321             "description": "The uuid formatted identity of the Device\nFormat pattern according to
7322 IETF RFC 4122.",
7323             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7324 9]{12}$",
7325             "type": "string"
7326         },
7327         "owned": {
7328             "description": "Ownership status flag.",
7329             "type": "boolean"
7330         },
7331         "oxmsel": {
7332             "description": "The selected owner transfer method used during on-boarding\nThe Device
7333 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7334 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7335 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7336 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7337 method (oic.sec.doxm.dcap) (deprecated).",
7338             "type": "integer"
7339         }
7340     },
7341     "type" : "object"
7342 }
7343 }
7344 }
7345

```

### 7346 C.9.5 Property definition

7347 Table C.14 defines the Properties that are part of the "oic.r.doxm" Resource Type.

7348 **Table C.14 – The Property definitions of the Resource with type "rt" = "oic.r.doxm".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
owned	boolean	Yes	Read Write	Ownership status flag.
oxmsel	integer	Yes	Read Write	The selected owner transfer method used during on-boarding

				The Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the Device Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
n	multiple types: see schema	No	Read Write	
oxms	array: see schema	Yes	Read Only	List of supported owner transfer methods.
devowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
sct	integer	Yes	Read Only	Bitmask encoding of supported credential types Credential Types - Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 - Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with



				certificate16 - PIN or password32 - Asymmetric encryption key.
rowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
owned	boolean		Read Write	Ownership status flag.
oxmsel	integer		Read Write	The selected owner transfer method used during on-boarding. The Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).
devowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string		Read Write	The uuid formatted identity of the Device. Format pattern according to IETF RFC 4122.

7349 **C.9.6 CRUDN behaviour**

7350 Table C.15 defines the CRUDN operations that are supported on the "oic.r.doxm" Resource Type.

7351 **Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".**

Create	Read	Update	Delete	Notify
	get	post		observe

7352 **C.10 Device Provisioning Status**

7353 **C.10.1 Introduction**

7354 This Resource specifies Device provisioning status.

7355

7356 **C.10.2 Well-known URI**

7357 /oic/sec/pstat

7358 **C.10.3 Resource type**

7359 The Resource Type is defined as: "oic.r.pstat".

7360 **C.10.4 OpenAPI 2.0 definition**

```
7361 {
7362   "swagger": "2.0",
7363   "info": {
7364     "title": "Device Provisioning Status",
7365     "version": "v1.0-20191001",
7366     "license": {
7367       "name": "OCF Data Model License",
7368       "url":
7369         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7370 CENSE.md",
7371       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7372 reserved."
7373     },
7374     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7375   },
7376   "schemes": ["http"],
7377   "consumes": ["application/json"],
7378   "produces": ["application/json"],
7379   "paths": {
7380     "/oic/sec/pstat" : {
7381       "get": {
7382         "description": "This Resource specifies Device provisioning status.\n",
7383         "parameters": [
7384           { "$ref": "#/parameters/interface" }
7385         ],
7386         "responses": {
7387           "200": {
7388             "description": "",
7389             "x-example":
7390               {
7391                 "rt": ["oic.r.pstat"],
7392                 "dos": {"s": 3, "p": true},
7393                 "isop": true,
7394                 "cm": 8,
7395                 "tm": 60,
7396                 "om": 2,
7397                 "sm": 7,
7398                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7399               },
7400             "schema": { "$ref": "#/definitions/Pstat" }
7401           },
7402           "400": {
7403             "description": "The request is invalid."
7404           }
7405         }
7406       },
7407       "post": {
7408         "description": "Sets or updates Device provisioning status data.\n",
7409         "parameters": [
7410           { "$ref": "#/parameters/interface" },
7411           {
7412             "name": "body",
7413             "in": "body",
7414             "required": true,
7415             "schema": { "$ref": "#/definitions/Pstat-Update" },
7416             "x-example":
7417               {
7418                 "dos": {"s": 3},
7419                 "tm": 60,
7420                 "om": 2,
7421                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7422               }
7423           }
7424         ]
7425       }
7426     }
7427   }
7428 }
```

```

7423     }
7424   ],
7425   "responses": {
7426     "400": {
7427       "description": "The request is invalid."
7428     },
7429     "204": {
7430       "description": "The PSTAT entry is updated."
7431     }
7432   }
7433 }
7434 },
7435 },
7436 "parameters": {
7437   "interface": {
7438     "in": "query",
7439     "name": "if",
7440     "type": "string",
7441     "enum": ["oic.if.baseline"]
7442   }
7443 },
7444 "definitions": {
7445   "Pstat": {
7446     "properties": {
7447       "rowneruuid": {
7448         "description": "The UUID formatted identity of the Resource owner\nFormat pattern
according to IETF RFC 4122.",
7449         "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}$",
7450         "type": "string"
7451       },
7452     },
7453     "rt": {
7454       "description": "Resource Type of the Resource.",
7455       "items": {
7456         "maxLength": 64,
7457         "type": "string",
7458         "enum": ["oic.r.pstat"]
7459       },
7460       "minItems": 1,
7461       "readOnly": true,
7462       "type": "array"
7463     },
7464     "om": {
7465       "description": "Current operational mode\nDevice provisioning operation may be server
directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
- Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7466       "maximum": 7,
7467       "minimum": 1,
7468       "type": "integer"
7469     },
7470     "cm": {
7471       "description": "Current Device provisioning mode\nDevice provisioning mode maintains a
bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
- Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
Software Version Validation128 - Initiate Secure Software Update.",
7472       "maximum": 255,
7473       "minimum": 0,
7474       "type": "integer",
7475       "readOnly": true
7476     },
7477     "n": {
7478       "$ref":
7479       "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
7480     },
7481     "id": {
7482       "$ref":
7483       "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7484       schema.json#/definitions/id"
7485     }
7486   }
7487 }
7488 }
7489 }
7490 }
7491 }
7492 }
7493 }

```

```

7494 schema.json#/definitions/id"
7495     },
7496     "isop": {
7497         "description": "true indicates Device is operational.",
7498         "readOnly": true,
7499         "type": "boolean"
7500     },
7501     "tm": {
7502         "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7503 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7504 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7505 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7506 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7507 Software Version Validation128 - Initiate Secure Software Update.",
7508         "maximum": 255,
7509         "minimum": 0,
7510         "type": "integer"
7511     },
7512     "sm": {
7513         "description": "Supported operational modes\nDevice provisioning operation may be server
7514 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7515 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7516 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7517 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7518         "maximum": 7,
7519         "minimum": 1,
7520         "type": "integer",
7521         "readOnly": true
7522     },
7523     "dos": {
7524         "description": "Device on-boarding state\nDevice operation state machine.",
7525         "properties": {
7526             "p": {
7527                 "default": true,
7528                 "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7529 to Device Resources are complete.",
7530                 "readOnly": true,
7531                 "type": "boolean"
7532             },
7533             "s": {
7534                 "description": "The current or pending operational state.",
7535                 "x-detail-desc": [
7536                     "0 - RESET - Device reset state.",
7537                     "1 - RFOTM - Ready for Device owner transfer method state.",
7538                     "2 - RFPPO - Ready for Device provisioning state.",
7539                     "3 - RFNOP - Ready for Device normal operation state.",
7540                     "4 - SRESET - The Device is in a soft reset state."
7541                 ],
7542                 "maximum": 4,
7543                 "minimum": 0,
7544                 "type": "integer"
7545             }
7546         },
7547         "required": [
7548             "s"
7549         ],
7550         "type": "object"
7551     },
7552     "if" : {
7553         "description": "The interface set supported by this Resource.",
7554         "items": {
7555             "enum": [
7556                 "oic.if.baseline"
7557             ],
7558             "type": "string"
7559         },
7560         "minItems": 1,
7561         "readOnly": true,
7562         "type": "array"
7563     }
7564 },

```

```

7565     "type" : "object",
7566     "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
7567 },
7568 "Pstat-Update" : {
7569     "properties": {
7570         "rowneruuid": {
7571             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7572 according to IETF RFC 4122.",
7573             "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}$",
7574             "type": "string"
7575         },
7576         "om": {
7577             "description": "Current operational mode\nDevice provisioning operation may be server
7578 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7579 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7580 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7581 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7582             "maximum": 7,
7583             "minimum": 1,
7584             "type": "integer"
7585         },
7586         "tm": {
7587             "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7588 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7589 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7590 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7591 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7592 Software Version Validation128 - Initiate Secure Software Update.",
7593             "maximum": 255,
7594             "minimum": 0,
7595             "type": "integer"
7596         },
7597         "dos": {
7598             "description": "Device on-boarding state\nDevice operation state machine.",
7599             "properties": {
7600                 "p": {
7601                     "default": true,
7602                     "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7603 to Device Resources are complete.",
7604                     "readOnly": true,
7605                     "type": "boolean"
7606                 },
7607                 "s": {
7608                     "description": "The current or pending operational state.",
7609                     "x-detail-desc": [
7610                         "0 - RESET - Device reset state.",
7611                         "1 - RFOTM - Ready for Device owner transfer method state.",
7612                         "2 - RFPRO - Ready for Device provisioning state.",
7613                         "3 - RFNOP - Ready for Device normal operation state.",
7614                         "4 - SRESET - The Device is in a soft reset state."
7615                     ],
7616                     "maximum": 4,
7617                     "minimum": 0,
7618                     "type": "integer"
7619                 }
7620             },
7621             "required": [
7622                 "s"
7623             ],
7624             "type": "object"
7625         }
7626     },
7627     "type" : "object"
7628 }
7629 }
7630 }
7631

```

## 7632 C.10.5 Property definition

7633 Table C.16 defines the Properties that are part of the "oic.r.pstat" Resource Type.

**Table C.16 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".**

Property name	Value type	Mandatory	Access mode	Description
dos	object: see schema	No	Read Write	Device on-boarding state machine. Device operation state machine.
rowneruuid	string	No	Read Write	The UUID formatted identity of the Resource owner. Format pattern according to IETF RFC 4122.
tm	integer	No	Read Write	Target Device provisioning mode. Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If it's only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
om	integer	No	Read Write	Current

				operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
isop	boolean	Yes	Read Only	true indicates Device is operational.
cm	integer	Yes	Read Only	Current Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If it's only 8 characters it represents the lower byte value1 - Manufacturer reset state2 -

				Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
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.
om	integer	Yes	Read Write	Current operational mode



				<p>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.</p>
tm	integer	Yes	Read Write	<p>Target Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If it's only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of</p>

				credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.
dos	object: see schema	Yes	Read Write	Device on-boarding state Device operation state machine.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.

7635 **C.10.6 CRUDN behaviour**

7636 Table C.17 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

7637 **Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".**

Create	Read	Update	Delete	Notify
	get	post		observe

7638 **C.11 Asserted Roles**

7639 **C.11.1 Introduction**

7640 This Resource specifies roles that have been asserted.

7641

7642 **C.11.2 Well-known URI**

7643 /oic/sec/roles

7644 **C.11.3 Resource type**

7645 The Resource Type is defined as: "oic.r.roles".

#### 7646 C.11.4 OpenAPI 2.0 definition

```
7647 {
7648   "swagger": "2.0",
7649   "info": {
7650     "title": "Asserted Roles",
7651     "version": "v1.0-20170323",
7652     "license": {
7653       "name": "OCF Data Model License",
7654       "url":
7655         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7656         CENSE.md",
7657       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7658       reserved."
7659     },
7660     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7661   },
7662   "schemes": ["http"],
7663   "consumes": ["application/json"],
7664   "produces": ["application/json"],
7665   "paths": {
7666     "/oic/sec/roles" : {
7667       "get": {
7668         "description": "This Resource specifies roles that have been asserted.\n",
7669         "parameters": [
7670           {"$ref": "#/parameters/interface"}
7671         ],
7672         "responses": {
7673           "200": {
7674             "description": "",
7675             "x-example":
7676               {
7677                 "roles" :[
7678                   {
7679                     "credid":1,
7680                     "credtype":8,
7681                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7682                     "publicdata":
7683                       {
7684                         "encoding":"oic.sec.encoding.pem",
7685                         "data":"PEMENCODEDROLECERT"
7686                       },
7687                     "optionaldata":
7688                       {
7689                         "revstat": false,
7690                         "encoding":"oic.sec.encoding.pem",
7691                         "data":"PEMENCODEDISSUERCERT"
7692                       }
7693                   },
7694                   {
7695                     "credid":2,
7696                     "credtype":8,
7697                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7698                     "publicdata":
7699                       {
7700                         "encoding":"oic.sec.encoding.pem",
7701                         "data":"PEMENCODEDROLECERT"
7702                       },
7703                     "optionaldata":
7704                       {
7705                         "revstat": false,
7706                         "encoding":"oic.sec.encoding.pem",
7707                         "data":"PEMENCODEDISSUERCERT"
7708                       }
7709                   }
7710                 ],
7711                 "rt":["oic.r.roles"],
7712                 "if":["oic.if.baseline"]
7713             }
7714           },
7715           "schema": { "$ref": "#/definitions/Roles" }
7716         }
7717       }
7718     }
7719   }
7720 }
```

```

7716         },
7717         "400": {
7718             "description" : "The request is invalid."
7719         }
7720     },
7721 },
7722 "post": {
7723     "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
7724 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
7725 same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
7726 ignored, the Resource assigns its own.\n",
7727     "parameters": [
7728         { "$ref": "#/parameters/interface" },
7729         {
7730             "name": "body",
7731             "in": "body",
7732             "required": true,
7733             "schema": { "$ref": "#/definitions/Roles-update" },
7734             "x-example":
7735                 {
7736                     "roles" :[
7737                         {
7738                             "credid":1,
7739                             "credtype":8,
7740                             "subjectuuid":"00000000-0000-0000-0000-000000000000",
7741                             "publicdata":
7742                                 {
7743                                     "encoding":"oic.sec.encoding.pem",
7744                                     "data":"PEMENCODEDROLECERT"
7745                                 },
7746                             "optionaldata":
7747                                 {
7748                                     "revstat": false,
7749                                     "encoding":"oic.sec.encoding.pem",
7750                                     "data":"PEMENCODEDISSUERCERT"
7751                                 }
7752                         },
7753                         {
7754                             "credid":2,
7755                             "credtype":8,
7756                             "subjectuuid":"00000000-0000-0000-0000-000000000000",
7757                             "publicdata":
7758                                 {
7759                                     "encoding":"oic.sec.encoding.pem",
7760                                     "data":"PEMENCODEDROLECERT"
7761                                 },
7762                             "optionaldata":
7763                                 {
7764                                     "revstat": false,
7765                                     "encoding":"oic.sec.encoding.pem",
7766                                     "data":"PEMENCODEDISSUERCERT"
7767                                 }
7768                         }
7769                     ]
7770                 }
7771         }
7772     ],
7773     "responses": {
7774         "400": {
7775             "description" : "The request is invalid."
7776         },
7777         "204": {
7778             "description" : "The roles entry is updated."
7779         }
7780     }
7781 },
7782 "delete": {
7783     "description": "Deletes roles Resource entries.\n\nWhen DELETE is used without query
7784 parameters, all the roles entries are deleted.\n\nWhen DELETE is used with a query parameter, only the
7785 entries matching\nthe query parameter are deleted.\n",
7786     "parameters": [

```

```

7787         {"$ref": "#/parameters/interface"},
7788         {"$ref": "#/parameters/roles-filtered"}
7789     ],
7790     "responses": {
7791         "200": {
7792             "description": "The specified or all roles Resource entries have been successfully
7793 deleted."
7794         },
7795         "400": {
7796             "description": "The request is invalid."
7797         }
7798     }
7799 }
7800 }
7801 },
7802 "parameters": {
7803     "interface": {
7804         "in": "query",
7805         "name": "if",
7806         "type": "string",
7807         "enum": ["oic.if.baseline"]
7808     },
7809     "roles-filtered": {
7810         "in": "query",
7811         "name": "credid",
7812         "required": false,
7813         "type": "integer",
7814         "description": "Only applies to the credential with the specified credid.",
7815         "x-example": 2112
7816     }
7817 },
7818 "definitions": {
7819     "Roles": {
7820         "properties": {
7821             "rt": {
7822                 "description": "Resource Type of the Resource.",
7823                 "items": {
7824                     "maxLength": 64,
7825                     "type": "string",
7826                     "enum": ["oic.r.roles"]
7827                 },
7828                 "minItems": 1,
7829                 "readOnly": true,
7830                 "type": "array"
7831             },
7832             "n": {
7833                 "$ref":
7834 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7835 schema.json#/definitions/n"
7836             },
7837             "id": {
7838                 "$ref":
7839 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7840 schema.json#/definitions/id"
7841             },
7842             "roles": {
7843                 "description": "List of role certificates.",
7844                 "items": {
7845                     "properties": {
7846                         "credid": {
7847                             "description": "Local reference to a credential Resource.",
7848                             "type": "integer"
7849                         },
7850                         "credtype": {
7851                             "description": "Representation of this credential's type\nCredential Types - Cred
7852 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7853 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7854 password32 - Asymmetric encryption key.",
7855                             "maximum": 63,
7856                             "minimum": 0,
7857                             "type": "integer"

```

```

7858     },
7859     "credusage": {
7860         "description": "A string that provides hints about how/where the cred is used\nThe
7861 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7862 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7863 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7864         "enum": [
7865             "oic.sec.cred.trustca",
7866             "oic.sec.cred.cert",
7867             "oic.sec.cred.rolecert",
7868             "oic.sec.cred.mfgtrustca",
7869             "oic.sec.cred.mfgcert"
7870         ],
7871         "type": "string"
7872     },
7873     "crms": {
7874         "description": "The refresh methods that may be used to update this credential.",
7875         "items": {
7876             "description": "Each enum represents a method by which the credentials are
7877 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7878 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7879 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7880 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7881             "enum": [
7882                 "oic.sec.crm.pro",
7883                 "oic.sec.crm.psk",
7884                 "oic.sec.crm.rdp",
7885                 "oic.sec.crm.skdc",
7886                 "oic.sec.crm.pk10"
7887             ],
7888             "type": "string"
7889         },
7890         "type": "array"
7891     },
7892     "optionaldata": {
7893         "description": "Credential revocation status information\nOptional credential
7894 contents describes revocation status for this credential.",
7895         "properties": {
7896             "data": {
7897                 "description": "This is the encoded structure.",
7898                 "type": "string"
7899             },
7900             "encoding": {
7901                 "description": "A string specifying the encoding format of the data contained in
7902 the optdata.",
7903                 "x-detail-desc": [
7904                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7905                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7906                     "oic.sec.encoding.base64 - Base64 encoded object.",
7907                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7908                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7909                     "oic.sec.encoding.raw - Raw hex encoded data."
7910                 ],
7911                 "enum": [
7912                     "oic.sec.encoding.jwt",
7913                     "oic.sec.encoding.cwt",
7914                     "oic.sec.encoding.base64",
7915                     "oic.sec.encoding.pem",
7916                     "oic.sec.encoding.der",
7917                     "oic.sec.encoding.raw"
7918                 ],
7919                 "type": "string"
7920             },
7921             "revstat": {
7922                 "description": "Revocation status flag - true = revoked.",
7923                 "type": "boolean"
7924             }
7925         },
7926         "required": [
7927             "revstat"
7928         ],

```

```

7929         "type": "object"
7930     },
7931     "period": {
7932         "description": "String with RFC5545 Period.",
7933         "type": "string"
7934     },
7935     "privatedata": {
7936         "description": "Private credential information\nCredential Resource non-public
7937 contents.",
7938         "properties": {
7939             "data": {
7940                 "description": "The encoded value.",
7941                 "maxLength": 3072,
7942                 "type": "string"
7943             },
7944             "encoding": {
7945                 "description": "A string specifying the encoding format of the data contained in
7946 the privdata.",
7947                 "x-detail-desc": [
7948                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7949                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7950                     "oic.sec.encoding.base64 - Base64 encoded object.",
7951                     "oic.sec.encoding.uri - URI reference.",
7952                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
7953 referenced using a handle.",
7954                     "oic.sec.encoding.raw - Raw hex encoded data."
7955                 ],
7956                 "enum": [
7957                     "oic.sec.encoding.jwt",
7958                     "oic.sec.encoding.cwt",
7959                     "oic.sec.encoding.base64",
7960                     "oic.sec.encoding.uri",
7961                     "oic.sec.encoding.handle",
7962                     "oic.sec.encoding.raw"
7963                 ],
7964                 "type": "string"
7965             },
7966             "handle": {
7967                 "description": "Handle to a key storage Resource.",
7968                 "type": "integer"
7969             }
7970         },
7971         "required": [
7972             "encoding"
7973         ],
7974         "type": "object"
7975     },
7976     "publicdata": {
7977         "description": "Public credential information.",
7978         "properties": {
7979             "data": {
7980                 "description": "This is the encoded value.",
7981                 "maxLength": 3072,
7982                 "type": "string"
7983             },
7984             "encoding": {
7985                 "description": "A string specifying the encoding format of the data contained in
7986 the pubdata.",
7987                 "x-detail-desc": [
7988                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7989                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7990                     "oic.sec.encoding.base64 - Base64 encoded object.",
7991                     "oic.sec.encoding.uri - URI reference.",
7992                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7993                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7994                     "oic.sec.encoding.raw - Raw hex encoded data."
7995                 ],
7996                 "enum": [
7997                     "oic.sec.encoding.jwt",
7998                     "oic.sec.encoding.cwt",
7999                     "oic.sec.encoding.base64",

```

```

8000         "oic.sec.encoding.uri",
8001         "oic.sec.encoding.pem",
8002         "oic.sec.encoding.der",
8003         "oic.sec.encoding.raw"
8004     ],
8005     "type": "string"
8006 }
8007 },
8008 "type": "object"
8009 },
8010 "roleid": {
8011     "description": "The role this credential possesses\nSecurity role specified as an
8012 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8013     "properties": {
8014         "authority": {
8015             "description": "The Authority component of the entity being identified. A NULL
8016 <Authority> refers to the local entity or Device.",
8017             "type": "string"
8018         },
8019         "role": {
8020             "description": "The ID of the role being identified.",
8021             "type": "string"
8022         }
8023     },
8024     "required": [
8025         "role"
8026     ],
8027     "type": "object"
8028 },
8029 "subjectuuid": {
8030     "anyOf": [
8031         {
8032             "description": "The id of the Device, which the cred entry applies to or \"*\n"
8033 for wildcard identity.",
8034             "pattern": "^[\\*]*$",
8035             "type": "string"
8036         },
8037         {
8038             "description": "Format pattern according to IETF RFC 4122.",
8039             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8040 F0-9]{12}$",
8041             "type": "string"
8042         }
8043     ]
8044 }
8045 },
8046 "type": "object"
8047 },
8048 "type": "array"
8049 },
8050 "if": {
8051     "description": "The interface set supported by this Resource.",
8052     "items": {
8053         "enum": [
8054             "oic.if.baseline"
8055         ],
8056         "type": "string"
8057     },
8058     "minItems": 1,
8059     "readOnly": true,
8060     "type": "array"
8061 }
8062 },
8063 "type": "object",
8064 "required": ["roles"]
8065 },
8066 "Roles-update" : {
8067     "properties": {
8068         "roles": {
8069             "description": "List of role certificates.",
8070             "items": {

```



```

8071     "properties": {
8072         "credid": {
8073             "description": "Local reference to a credential Resource.",
8074             "type": "integer"
8075         },
8076         "credtype": {
8077             "description": "Representation of this credential's type\nCredential Types - Cred
8078 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8079 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
8080 password32 - Asymmetric encryption key.",
8081             "maximum": 63,
8082             "minimum": 0,
8083             "type": "integer"
8084         },
8085         "credusage": {
8086             "description": "A string that provides hints about how/where the cred is used\nThe
8087 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8088 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8089 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
8090             "enum": [
8091                 "oic.sec.cred.trustca",
8092                 "oic.sec.cred.cert",
8093                 "oic.sec.cred.rolecert",
8094                 "oic.sec.cred.mfgtrustca",
8095                 "oic.sec.cred.mfgcert"
8096             ],
8097             "type": "string"
8098         },
8099         "crms": {
8100             "description": "The refresh methods that may be used to update this credential.",
8101             "items": {
8102                 "description": "Each enum represents a method by which the credentials are
8103 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8104 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8105 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8106 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
8107                 "enum": [
8108                     "oic.sec.crm.pro",
8109                     "oic.sec.crm.psk",
8110                     "oic.sec.crm.rdp",
8111                     "oic.sec.crm.skdc",
8112                     "oic.sec.crm.pk10"
8113                 ],
8114                 "type": "string"
8115             },
8116             "type": "array"
8117         },
8118         "optionaldata": {
8119             "description": "Credential revocation status information\nOptional credential
8120 contents describes revocation status for this credential.",
8121             "properties": {
8122                 "data": {
8123                     "description": "This is the encoded structure.",
8124                     "type": "string"
8125                 },
8126                 "encoding": {
8127                     "description": "A string specifying the encoding format of the data contained in
8128 the optdata.",
8129                     "x-detail-desc": [
8130                         "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8131                         "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8132                         "oic.sec.encoding.base64 - Base64 encoded object.",
8133                         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8134                         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8135                         "oic.sec.encoding.raw - Raw hex encoded data."
8136                     ],
8137                     "enum": [
8138                         "oic.sec.encoding.jwt",
8139                         "oic.sec.encoding.cwt",
8140                         "oic.sec.encoding.base64",
8141                         "oic.sec.encoding.pem",

```

```

8142         "oic.sec.encoding.der",
8143         "oic.sec.encoding.raw"
8144     ],
8145     "type": "string"
8146 },
8147     "revstat": {
8148         "description": "Revocation status flag - true = revoked.",
8149         "type": "boolean"
8150     }
8151 },
8152     "required": [
8153         "revstat"
8154     ],
8155     "type": "object"
8156 },
8157     "period": {
8158         "description": "String with RFC5545 Period.",
8159         "type": "string"
8160     },
8161     "privatedata": {
8162         "description": "Private credential information\nCredential Resource non-public
8163 contents.",
8164         "properties": {
8165             "data": {
8166                 "description": "The encoded value.",
8167                 "maxLength": 3072,
8168                 "type": "string"
8169             },
8170             "encoding": {
8171                 "description": "A string specifying the encoding format of the data contained in
8172 the privdata.",
8173                 "x-detail-desc": [
8174                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8175                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8176                     "oic.sec.encoding.base64 - Base64 encoded object.",
8177                     "oic.sec.encoding.uri - URI reference.",
8178                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
8179 referenced using a handle.",
8180                     "oic.sec.encoding.raw - Raw hex encoded data."
8181                 ],
8182                 "enum": [
8183                     "oic.sec.encoding.jwt",
8184                     "oic.sec.encoding.cwt",
8185                     "oic.sec.encoding.base64",
8186                     "oic.sec.encoding.uri",
8187                     "oic.sec.encoding.handle",
8188                     "oic.sec.encoding.raw"
8189                 ],
8190                 "type": "string"
8191             },
8192             "handle": {
8193                 "description": "Handle to a key storage Resource.",
8194                 "type": "integer"
8195             }
8196         },
8197         "required": [
8198             "encoding"
8199         ],
8200         "type": "object"
8201     },
8202     "publicdata": {
8203         "description": "Public credential information.",
8204         "properties": {
8205             "data": {
8206                 "description": "The encoded value.",
8207                 "maxLength": 3072,
8208                 "type": "string"
8209             },
8210             "encoding": {
8211                 "description": "A string specifying the encoding format of the data contained in
8212 the pubdata.",

```

```

8213         "x-detail-desc": [
8214             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8215             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8216             "oic.sec.encoding.base64 - Base64 encoded object.",
8217             "oic.sec.encoding.uri - URI reference.",
8218             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8219             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8220             "oic.sec.encoding.raw - Raw hex encoded data."
8221         ],
8222         "enum": [
8223             "oic.sec.encoding.jwt",
8224             "oic.sec.encoding.cwt",
8225             "oic.sec.encoding.base64",
8226             "oic.sec.encoding.uri",
8227             "oic.sec.encoding.pem",
8228             "oic.sec.encoding.der",
8229             "oic.sec.encoding.raw"
8230         ],
8231         "type": "string"
8232     }
8233 },
8234 "type": "object"
8235 },
8236 "roleid": {
8237     "description": "The role this credential possesses\nSecurity role specified as an
8238 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8239     "properties": {
8240         "authority": {
8241             "description": "The Authority component of the entity being identified. A NULL
8242 <Authority> refers to the local entity or Device.",
8243             "type": "string"
8244         },
8245         "role": {
8246             "description": "The ID of the role being identified.",
8247             "type": "string"
8248         }
8249     },
8250     "required": [
8251         "role"
8252     ],
8253     "type": "object"
8254 },
8255 "subjectuuid": {
8256     "anyOf": [
8257         {
8258             "description": "The id of the Device, which the cred entry applies to or \"*\n
8259 for wildcard identity.",
8260             "pattern": "^[\\*$]",
8261             "type": "string"
8262         },
8263         {
8264             "description": "Format pattern according to IETF RFC 4122.",
8265             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8266 F0-9]{12}$",
8267             "type": "string"
8268         }
8269     ]
8270 }
8271 },
8272 "type": "object"
8273 },
8274 "type": "array"
8275 }
8276 },
8277 "type": "object",
8278 "required": ["roles"]
8279 }
8280 }
8281 }
8282

```

8283 **C.11.5 Property definition**

8284 Table C.18 defines the Properties that are part of the "oic.r.roles" Resource Type.

8285 **Table C.18 – The Property definitions of the Resource with type "rt" = "oic.r.roles".**

Property name	Value type	Mandatory	Access mode	Description
roles	array: see schema	Yes	Read Write	List of role certificates.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
roles	array: see schema	Yes	Read Write	List of role certificates.

8286 **C.11.6 CRUDN behaviour**

8287 Table C.19 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

8288 **Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

8289 **C.12 Signed Access Control List**

8290 **C.12.1 Introduction**

8291 This Resource specifies a signed ACL object.

8292

8293 **C.12.2 Well-known URI**

8294 /oic/sec/sacl

8295 **C.12.3 Resource type**

8296 The Resource Type is defined as: "oic.r.sacl".

8297 **C.12.4 OpenAPI 2.0 definition**

```

8298 {
8299   "swagger": "2.0",
8300   "info": {
8301     "title": "Signed Access Control List",
8302     "version": "v1.0-20150819",
8303     "license": {
8304       "name": "OCF Data Model License",
8305       "url":
8306         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8307         CENSE.md",
8308       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8309         reserved."
8310     },
8311     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8312   },
8313   "schemes": ["http"],
8314   "consumes": ["application/json"],
8315   "produces": ["application/json"],
8316   "paths": {
8317     "/oic/sec/sacl" : {

```

```

8318 "get": {
8319   "description": "This Resource specifies a signed ACL object.\n",
8320   "parameters": [
8321     {"$ref": "#/parameters/interface"}
8322   ],
8323   "responses": {
8324     "200": {
8325       "description": "",
8326       "x-example":
8327         {
8328           "rt": ["oic.r.sacl"],
8329           "aclist2": [
8330             {
8331               "aceid": 1,
8332               "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8333               "resources": [
8334                 {
8335                   "href": "/temp",
8336                   "rt": ["oic.r.temperature"],
8337                   "if": ["oic.if.baseline", "oic.if.a"]
8338                 },
8339                 {
8340                   "href": "/temp",
8341                   "rt": ["oic.r.temperature"],
8342                   "if": ["oic.if.baseline", "oic.if.s"]
8343                 }
8344               ],
8345               "permission": 31,
8346               "validity": [
8347                 {
8348                   "period": "20160101T180000Z/20170102T070000Z",
8349                   "recurrence": [ "DSTART:XXXXX",
8350 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8351                 },
8352                 {
8353                   "period": "20160101T180000Z/PT5H30M",
8354                   "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8355                 }
8356               ]
8357             },
8358             {
8359               "aceid": 2,
8360               "subject": {
8361                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8362                 "role": "SOME_STRING"
8363               },
8364               "resources": [
8365                 {
8366                   "href": "/light",
8367                   "rt": ["oic.r.light"],
8368                   "if": ["oic.if.baseline", "oic.if.a"]
8369                 },
8370                 {
8371                   "href": "/door",
8372                   "rt": ["oic.r.door"],
8373                   "if": ["oic.if.baseline", "oic.if.a"]
8374                 }
8375               ],
8376               "permission": 15
8377             }
8378           ],
8379           "signature": {
8380             "sigtype": "oic.sec.sigtype.pk7",
8381             "sigvalue": "ENCODED-SIGNATURE-VALUE"
8382           }
8383         },
8384         "schema": { "$ref": "#/definitions/Sacl" }
8385       }
8386     },
8387     "post": {
8388

```

```

8389     "description": "Sets the sacl Resource data.\n",
8390     "parameters": [
8391         {"$ref": "#/parameters/interface"},
8392         {
8393             "name": "body",
8394             "in": "body",
8395             "required": true,
8396             "schema": { "$ref": "#/definitions/Sacl" },
8397             "x-example":
8398                 {
8399                     "aclist2": [
8400                         {
8401                             "aceid": 1,
8402                             "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8403                             "resources": [
8404                                 {
8405                                     "href": "/temp",
8406                                     "rt": ["oic.r.temperature"],
8407                                     "if": ["oic.if.baseline", "oic.if.a"]
8408                                 },
8409                                 {
8410                                     "href": "/temp",
8411                                     "rt": ["oic.r.temperature"],
8412                                     "if": ["oic.if.baseline", "oic.if.s"]
8413                                 }
8414                             ],
8415                             "permission": 31,
8416                             "validity": [
8417                                 {
8418                                     "period": "20160101T180000Z/20170102T070000Z",
8419                                     "recurrence": [ "DSTART:XXXXX",
8420 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8421                                 },
8422                                 {
8423                                     "period": "20160101T180000Z/PT5H30M",
8424                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8425                                 }
8426                             ]
8427                             },
8428                             {
8429                                 "aceid": 2,
8430                                 "subject": {
8431                                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8432                                     "role": "SOME_STRING"
8433                                 },
8434                                 "resources": [
8435                                     {
8436                                         "href": "/light",
8437                                         "rt": ["oic.r.light"],
8438                                         "if": ["oic.if.baseline", "oic.if.a"]
8439                                     },
8440                                     {
8441                                         "href": "/door",
8442                                         "rt": ["oic.r.door"],
8443                                         "if": ["oic.if.baseline", "oic.if.a"]
8444                                     }
8445                                 ],
8446                                 "permission": 15
8447                             }
8448                             ],
8449                             "signature": {
8450                                 "sigtype": "oic.sec.sigtype.pk7",
8451                                 "sigvalue": "ENCODED-SIGNATURE-VALUE"
8452                             }
8453                         }
8454                     ]
8455                 },
8456                 "responses": {
8457                     "400": {
8458                         "description": "The request is invalid."
8459                     }
8460                 }
8461             }
8462         }
8463     ]
8464 }

```

```

8460         "201": {
8461             "description": "The ACL entry is created."
8462         },
8463         "204": {
8464             "description": "The ACL entry is updated."
8465         }
8466     },
8467 },
8468 "put": {
8469     "description": "Sets the sacl Resource data\n",
8470     "parameters": [
8471         { "$ref": "#/parameters/interface" },
8472         {
8473             "name": "body",
8474             "in": "body",
8475             "required": true,
8476             "schema": { "$ref": "#/definitions/Sacl" },
8477             "x-example":
8478                 {
8479                     "aclist2": [
8480                         {
8481                             "aceid": 1,
8482                             "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8483                             "resources": [
8484                                 {
8485                                     "href": "/temp",
8486                                     "rt": [ "oic.r.temperature" ],
8487                                     "if": [ "oic.if.baseline", "oic.if.a" ]
8488                                 },
8489                                 {
8490                                     "href": "/temp",
8491                                     "rt": [ "oic.r.temperature" ],
8492                                     "if": [ "oic.if.baseline", "oic.if.s" ]
8493                                 }
8494                             ],
8495                             "permission": 31,
8496                             "validity": [
8497                                 {
8498                                     "period": "20160101T180000Z/20170102T070000Z",
8499                                     "recurrence": [ "DSTART:XXXXX",
8500 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8501                                 },
8502                                 {
8503                                     "period": "20160101T180000Z/PT5H30M",
8504                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8505                                 }
8506                             ]
8507                         },
8508                     {
8509                         "aceid": 2,
8510                         "subject": {
8511                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8512                             "role": "SOME_STRING"
8513                         },
8514                         "resources": [
8515                             {
8516                                 "href": "/light",
8517                                 "rt": [ "oic.r.light" ],
8518                                 "if": [ "oic.if.baseline", "oic.if.a" ]
8519                             },
8520                             {
8521                                 "href": "/door",
8522                                 "rt": [ "oic.r.door" ],
8523                                 "if": [ "oic.if.baseline", "oic.if.a" ]
8524                             }
8525                         ],
8526                         "permission": 15
8527                     }
8528                 ],
8529                 "signature": {
8530                     "sigtype": "oic.sec.sigtype.pk7",

```

```

8531         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8532     }
8533 }
8534 }
8535 ],
8536 "responses": {
8537     "400": {
8538         "description": "The request is invalid."
8539     },
8540     "201": {
8541         "description": "The signed ACL entry is created."
8542     }
8543 }
8544 },
8545 "delete": {
8546     "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
8547 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
8548 specified, only the matched entry is deleted.\n",
8549     "parameters": [
8550         { "$ref": "#/parameters/interface" },
8551         {
8552             "in": "query",
8553             "description": "Delete the signed ACL identified by the string containing subject
8554 UUID.\n",
8555             "type": "string",
8556             "name": "subject"
8557         }
8558     ],
8559     "responses": {
8560         "200": {
8561             "description": "The signed ACL instance or the the entire signed ACL Resource has
8562 been successfully deleted."
8563         },
8564         "400": {
8565             "description": "The request is invalid."
8566         }
8567     }
8568 }
8569 },
8570 "parameters": {
8571     "interface": {
8572         "in": "query",
8573         "name": "if",
8574         "type": "string",
8575         "enum": ["oic.if.baseline"]
8576     }
8577 },
8578 "definitions": {
8579     "Sacl": {
8580         "properties": {
8581             "rt": {
8582                 "description": "Resource Type of the Resource.",
8583                 "items": {
8584                     "maxLength": 64,
8585                     "type": "string",
8586                     "enum": ["oic.r.sacl"]
8587                 },
8588                 "minItems": 1,
8589                 "readOnly": true,
8590                 "type": "array"
8591             },
8592             "aclist2": {
8593                 "description": "Access Control Entries in the ACL Resource.",
8594                 "items": {
8595                     "properties": {
8596                         "aceid": {
8597                             "description": "An identifier for the ACE that is unique within the ACL. In cases
8598 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
8599                             "minimum": 1,
8600                             "type": "integer"

```



```

8602     },
8603     "permission": {
8604         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
8605 permissions.",
8606         "x-detail-desc": [
8607             "0 - No permissions.",
8608             "1 - Create permission is granted.",
8609             "2 - Read, observe, discover permission is granted.",
8610             "4 - Write, update permission is granted.",
8611             "8 - Delete permission is granted.",
8612             "16 - Notify permission is granted."
8613         ],
8614         "maximum": 31,
8615         "minimum": 0,
8616         "type": "integer"
8617     },
8618     "resources": {
8619         "description": "References the application's Resources to which a security policy
8620 applies.",
8621         "items": {
8622             "description": "Each Resource must have at least one of these properties set.",
8623             "properties": {
8624                 "href": {
8625                     "allOf": [
8626                         {
8627                             "description": "When present, the ACE only applies when the href matches."
8628                         },
8629                         {
8630                             "description": "This is the target URI, it can be specified as a Relative
8631 Reference or fully-qualified URI.",
8632                             "format": "uri",
8633                             "maxLength": 256,
8634                             "type": "string"
8635                         }
8636                     ]
8637                 },
8638                 "if": {
8639                     "description": "When present, the ACE only applies when the if (interface)
8640 matches\nThe interface set supported by this Resource.",
8641                     "items": {
8642                         "enum": [
8643                             "oic.if.baseline",
8644                             "oic.if.ll",
8645                             "oic.if.b",
8646                             "oic.if.rw",
8647                             "oic.if.r",
8648                             "oic.if.a",
8649                             "oic.if.s"
8650                         ],
8651                         "type": "string"
8652                     },
8653                     "minItems": 1,
8654                     "type": "array"
8655                 },
8656                 "rt": {
8657                     "description": "When present, the ACE only applies when the rt (resource type)
8658 matches\nResource Type of the Resource.",
8659                     "items": {
8660                         "maxLength": 64,
8661                         "type": "string"
8662                     },
8663                     "minItems": 1,
8664                     "type": "array"
8665                 },
8666                 "wc": {
8667                     "description": "A wildcard matching policy.",
8668                     "pattern": "^[+*]$",
8669                     "type": "string"
8670                 }
8671             },
8672             "type": "object"

```

```

8673     },
8674     "type": "array"
8675 },
8676 "subject": {
8677   "anyOf": [
8678     {
8679       "description": "Device identifier.",
8680       "properties": {
8681         "uuid": {
8682           "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4122.",
8684           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
8685 fA-F0-9]{12}$",
8686           "type": "string"
8687         }
8688       },
8689       "required": [
8690         "uuid"
8691       ],
8692       "type": "object"
8693     },
8694     {
8695       "description": "Security role specified as an <Authority> & <Rolename>. A NULL
8696 <Authority> refers to the local entity or Device.",
8697       "properties": {
8698         "authority": {
8699           "description": "The Authority component of the entity being identified. A
8700 NULL <Authority> refers to the local entity or Device.",
8701           "type": "string"
8702         },
8703         "role": {
8704           "description": "The ID of the role being identified.",
8705           "type": "string"
8706         }
8707       },
8708       "required": [
8709         "role"
8710       ],
8711       "type": "object"
8712     },
8713     {
8714       "properties": {
8715         "conntype": {
8716           "description": "This property allows an ACE to be matched based on the
8717 connection or message type.",
8718           "x-detail-desc": [
8719             "auth-crypt - ACE applies if the Client is authenticated and the data
8720 channel or message is encrypted and integrity protected.",
8721             "anon-clear - ACE applies if the Client is not authenticated and the data
8722 channel or message is not encrypted but may be integrity protected."
8723           ],
8724           "enum": [
8725             "auth-crypt",
8726             "anon-clear"
8727           ],
8728           "type": "string"
8729         }
8730       },
8731       "required": [
8732         "conntype"
8733       ],
8734       "type": "object"
8735     }
8736   ]
8737 },
8738 "validity": {
8739   "description": "validity is an array of time-pattern objects.",
8740   "items": {
8741     "description": "The time-pattern contains a period and recurrence expressed in
8742 RFC5545 syntax.",
8743     "properties": {

```

```

8744         "period": {
8745             "description": "String represents a period using the RFC5545 Period.",
8746             "type": "string"
8747         },
8748         "recurrence": {
8749             "description": "String array represents a recurrence rule using the RFC5545
Recurrence.",
8750             "items": {
8751                 "type": "string"
8752             },
8753             "type": "array"
8754         }
8755     },
8756     "required": [
8757         "period"
8758     ],
8759     "type": "object"
8760 },
8761 "type": "array"
8762 }
8763 },
8764 "required": [
8765     "aceid",
8766     "resources",
8767     "permission",
8768     "subject"
8769 ],
8770 "type": "object"
8771 },
8772 "type": "array"
8773 },
8774 "n": {
8775     "$ref":
8776     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8777     schema.json#/definitions/n"
8778 },
8779 "id": {
8780     "$ref":
8781     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8782     schema.json#/definitions/id"
8783 },
8784 "signature": {
8785     "description": "The signature over the ACL Resource\nEncoded signature data.",
8786     "properties": {
8787         "sigtype": {
8788             "description": "The string specifies the predefined signature format.",
8789             "x-detail-desc": [
8790                 "RFC7515 JSON web signature (JWS) object.",
8791                 "RFC2315 base64 encoded object.",
8792                 "CBOR encoded JWS object."
8793             ],
8794             "enum": [
8795                 "oic.sec.sigtype.jws",
8796                 "oic.sec.sigtype.pk7",
8797                 "oic.sec.sigtype.cws"
8798             ],
8799             "type": "string"
8800         },
8801         "sigvalue": {
8802             "description": "The encoded signature.",
8803             "type": "string"
8804         }
8805     }
8806 },
8807 "required": [
8808     "sigtype",
8809     "sigvalue"
8810 ],
8811 "type": "object"
8812 },
8813 "if": {
8814     "description": "The interface set supported by this Resource.",

```

```

8815         "items": {
8816             "enum": [
8817                 "oic.if.baseline"
8818             ],
8819             "type": "string"
8820         },
8821         "minItems": 1,
8822         "readOnly": true,
8823         "type": "array"
8824     }
8825 },
8826 "type" : "object",
8827 "required": ["aclist2", "signature"]
8828 }
8829 }
8830 }
8831

```

8832 **C.12.5 Property definition**

8833 Table C.20 defines the Properties that are part of the "oic.r.sacl" Resource Type.

8834 **Table C.20 – The Property definitions of the Resource with type "rt" = "oic.r.sacl".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
id	multiple types: see schema	No	Read Write	
signature	object: see schema	Yes	Read Write	The signature over the ACL Resource Encoded signature data.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL Resource.
n	multiple types: see schema	No	Read Write	

8835 **C.12.6 CRUDN behaviour**

8836 Table C.21 defines the CRUDN operations that are supported on the "oic.r.sacl" Resource Type.

8837 **Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl".**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

8838 **C.13 Session**

8839 **C.13.1 Introduction**

8840 Resource that manages the persistent session between a Device and OCF Cloud.

8841 **C.13.2 Well-known URI**

8842 /oic/sec/session

8843 **C.13.3 Resource type**

8844 The Resource Type is defined as: "oic.r.session".

### 8845 C.13.4 OpenAPI 2.0 definition

```
8846 {
8847   "swagger": "2.0",
8848   "info": {
8849     "title": "Session",
8850     "version": "v1.0-20181001",
8851     "license": {
8852       "name": "OCF Data Model License",
8853       "url":
8854         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8855         CENSE.md",
8856       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8857         reserved."
8858     },
8859     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8860   },
8861   "schemes": ["http"],
8862   "consumes": ["application/json"],
8863   "produces": ["application/json"],
8864   "paths": {
8865     "/oic/sec/session" : {
8866       "post": {
8867         "description": "Resource that manages the persistent session between a Device and OCF
8868         Cloud.",
8869         "parameters": [
8870           { "$ref": "#/parameters/interface" },
8871           {
8872             "name": "body",
8873             "in": "body",
8874             "required": true,
8875             "schema": { "$ref": "#/definitions/Account-Session-Request" },
8876             "x-example":
8877               {
8878                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
8879                 "di" : "9cfbeb8e-5ale-4d1c-9d01-00c04fd430c8",
8880                 "accesstoken" : "0f3d9f7fe5491d54077d",
8881                 "login" : true
8882               }
8883           }
8884         ],
8885         "responses": {
8886           "204": {
8887             "description" : "",
8888             "x-example":
8889               {
8890                 "rt": ["oic.r.session"],
8891                 "expiresin" : 3600
8892               },
8893             "schema": { "$ref": "#/definitions/Account-Session-Response" }
8894           }
8895         }
8896       }
8897     }
8898   },
8899   "parameters": {
8900     "interface" : {
8901       "in" : "query",
8902       "name" : "if",
8903       "type" : "string",
8904       "enum" : ["oic.if.baseline"]
8905     }
8906   },
8907   "definitions": {
8908     "Account-Session-Request" : {
8909       "properties": {
8910         "uid": {
8911           "description": "Format pattern according to IETF RFC 4122.",
8912           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
8913           "type": "string"
8914         },

```

```

8915     "di": {
8916         "description": "The Device ID\nFormat pattern according to IETF RFC 4122.",
8917         "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}$",
8918         "type": "string"
8919     },
8920     "accesstoken": {
8921         "description": "Access-Token used to grant access right for the Device to sign-in.",
8922         "pattern": "(?!$|\\s+).*",
8923         "type": "string"
8924     },
8925     "login": {
8926         "description": "Action for the request: true = login, false = logout.",
8927         "type": "boolean"
8928     }
8929 },
8930 "type" : "object",
8931 "required": ["uid", "di", "accesstoken", "login"]
8932 },
8933 "Account-Session-Response" : {
8934     "properties": {
8935         "expiresin": {
8936             "description": "Access-Token remaining life time in seconds (-1 if permanent).",
8937             "readOnly": true,
8938             "type": "integer"
8939         },
8940         "rt": {
8941             "description": "Resource Type of the Resource.",
8942             "items": {
8943                 "maxLength": 64,
8944                 "type": "string",
8945                 "enum": ["oic.r.session"]
8946             },
8947             "minItems": 1,
8948             "readOnly": true,
8949             "type": "array"
8950         },
8951         "n": {
8952             "$ref":
8953 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8954 schema.json#/definitions/n"
8955         },
8956         "id": {
8957             "$ref":
8958 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8959 schema.json#/definitions/id"
8960         },
8961         "if": {
8962             "description": "The interface set supported by this Resource.",
8963             "items": {
8964                 "enum": [
8965                     "oic.if.baseline"
8966                 ],
8967                 "type": "string"
8968             },
8969             "minItems": 1,
8970             "readOnly": true,
8971             "type": "array"
8972         }
8973     },
8974     "type" : "object",
8975     "required" : ["expiresin"]
8976 }
8977 }
8978 }
8979

```

### 8980 C.13.5 Property definition

8981 Table C.22 defines the Properties that are part of the "oic.r.session" Resource Type.

**Table C.22 – The Property definitions of the Resource with type "rt" = "oic.r.session".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds (-1 if permanent).
rt	array: see schema	No	Read Only	Resource Type of the Resource.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
di	string	Yes	Read Write	The Device ID Format pattern according to IETF RFC 4122.
accesstoken	string	Yes	Read Write	Access-Token used to grant access right for the Device to sign-in.
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
login	boolean	Yes	Read Write	Action for the request: true = login, false = logout.

8983 **C.13.6 CRUDN behaviour**8984 Table C.23 defines the CRUDN operations that are supported on the "oic.r.session" Resource  
8985 Type.8986 **Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.session".**

Create	Read	Update	Delete	Notify
		post		

8987 **C.14 Security Profile**8988 **C.14.1 Introduction**

8989 Resource specifying supported and active security profile(s).

8990

8991 **C.14.2 Well-known URI**

8992 /oic/sec/sp

8993 **C.14.3 Resource type**

8994 The Resource Type is defined as: "oic.r.sp".

#### 8995 C.14.4 OpenAPI 2.0 definition

```
8996 {
8997   "swagger": "2.0",
8998   "info": {
8999     "title": "Security Profile",
9000     "version": "v1.0-20190208",
9001     "license": {
9002       "name": "OCF Data Model License",
9003       "url":
9004         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9005 CENSE.md",
9006       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9007 reserved."
9008     },
9009     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9010   },
9011   "schemes": ["http"],
9012   "consumes": ["application/json"],
9013   "produces": ["application/json"],
9014   "paths": {
9015     "/oic/sec/sp" : {
9016       "get": {
9017         "description": "Resource specifying supported and active security profile(s).\n",
9018         "parameters": [
9019           { "$ref": "#/parameters/interface" }
9020         ],
9021         "responses": {
9022           "200": {
9023             "description": "",
9024             "x-example":
9025               {
9026                 "rt": ["oic.r.sp"],
9027                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9028                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9029               },
9030             "schema": { "$ref": "#/definitions/SP" }
9031           },
9032           "400": {
9033             "description": "The request is invalid."
9034           }
9035         }
9036       },
9037       "post": {
9038         "description": "Sets or updates Device provisioning status data.\n",
9039         "parameters": [
9040           { "$ref": "#/parameters/interface" },
9041           {
9042             "name": "body",
9043             "in": "body",
9044             "required": true,
9045             "schema": { "$ref": "#/definitions/SP-Update" },
9046             "x-example":
9047               {
9048                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9049                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9050               }
9051           }
9052         ],
9053         "responses": {
9054           "200": {
9055             "description": "",
9056             "x-example":
9057               {
9058                 "rt": ["oic.r.sp"],
9059                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9060                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9061               },
9062             "schema": { "$ref": "#/definitions/SP" }
9063           },
9064           "400": {
```



```

9065         "description" : "The request is invalid."
9066     }
9067 }
9068 }
9069 },
9070 },
9071 "parameters": {
9072     "interface" : {
9073         "in" : "query",
9074         "name" : "if",
9075         "type" : "string",
9076         "enum" : ["oic.if.baseline"]
9077     }
9078 },
9079 "definitions": {
9080     "SP" : {
9081         "properties": {
9082             "rt": {
9083                 "description": "Resource Type of the Resource.",
9084                 "items": {
9085                     "maxLength": 64,
9086                     "type": "string",
9087                     "enum": ["oic.r.sp"]
9088                 },
9089                 "minItems": 1,
9090                 "readOnly": true,
9091                 "type": "array"
9092             },
9093             "n": {
9094                 "$ref":
9095 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9096 schema.json#/definitions/n"
9097             },
9098             "id": {
9099                 "$ref":
9100 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9101 schema.json#/definitions/id"
9102             },
9103             "currentprofile": {
9104                 "description": "Security Profile currently active.",
9105                 "type": "string"
9106             },
9107             "supportedprofiles": {
9108                 "description": "Array of supported Security Profiles.",
9109                 "items": {
9110                     "type": "string"
9111                 },
9112                 "type": "array"
9113             },
9114             "if": {
9115                 "description": "The interface set supported by this Resource.",
9116                 "items": {
9117                     "enum": [
9118                         "oic.if.baseline"
9119                     ],
9120                     "type": "string"
9121                 },
9122                 "minItems": 1,
9123                 "readOnly": true,
9124                 "type": "array"
9125             }
9126         },
9127         "type" : "object",
9128         "required": ["supportedprofiles", "currentprofile"]
9129     },
9130     "SP-Update" : {
9131         "properties": {
9132             "currentprofile": {
9133                 "description": "Security Profile currently active.",
9134                 "type": "string"
9135             },

```

```

9136     "supportedprofiles": {
9137         "description": "Array of supported Security Profiles.",
9138         "items": {
9139             "type": "string"
9140         },
9141         "type": "array"
9142     },
9143 },
9144 "type" : "object"
9145 }
9146 }
9147 }
9148

```

9149 **C.14.5 Property definition**

9150 Table C.24 defines the Properties that are part of the "oic.r.sp" Resource Type.

9151 **Table C.24 – The Property definitions of the Resource with type "rt" = "oic.r.sp".**

Property name	Value type	Mandatory	Access mode	Description
supportedprofiles	array: see schema		Read Write	Array of supported Security Profiles.
currentprofile	string		Read Write	Security Profile currently active.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
currentprofile	string	Yes	Read Write	Security Profile currently active.
supportedprofiles	array: see schema	Yes	Read Write	Array of supported Security Profiles.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

9152 **C.14.6 CRUDN behaviour**

9153 Table C.25 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

9154 **Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".**

Create	Read	Update	Delete	Notify
	get	post		observe

9155 **C.15 Token Refresh**

9156 **C.15.1 Introduction**

9157 Obtain fresh access-token using the refresh token, client should refresh access-token before it expires.

9159 **C.15.2 Well-known URI**

9160 /oic/sec/tokenrefresh

9161 **C.15.3 Resource type**

9162 The Resource Type is defined as: "oic.r.tokenrefresh".

#### 9163 C.15.4 OpenAPI 2.0 definition

```
9164 {
9165   "swagger": "2.0",
9166   "info": {
9167     "title": "Token Refresh",
9168     "version": "v1.0-20181001",
9169     "license": {
9170       "name": "OCF Data Model License",
9171       "url":
9172         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9173 CENSE.md",
9174       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9175 reserved."
9176     },
9177     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9178   },
9179   "schemes": ["http"],
9180   "consumes": ["application/json"],
9181   "produces": ["application/json"],
9182   "paths": {
9183     "/oic/sec/tokenrefresh" : {
9184       "post": {
9185         "description": "Obtain fresh access-token using the refresh token, client should refresh
9186 access-token before it expires.\n",
9187         "parameters": [
9188           { "$ref": "#/parameters/interface" },
9189           {
9190             "name": "body",
9191             "in": "body",
9192             "required": true,
9193             "schema": { "$ref": "#/definitions/TokenRefresh-Request" },
9194             "x-example":
9195               {
9196                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9197                 "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
9198                 "refreshtoken" : "00fe4644a6fbe5324eec"
9199               }
9200           }
9201         ],
9202         "responses": {
9203           "204": {
9204             "description" : "2.04 Changed respond with new access-token.\n",
9205             "x-example":
9206               {
9207                 "rt": ["oic.r.tokenrefresh"],
9208                 "accesstoken" : "8ce598980761869837be",
9209                 "refreshtoken" : "d4922312b6df0518e146",
9210                 "expiresin" : 3600
9211               }
9212             ,
9213             "schema": { "$ref": "#/definitions/TokenRefresh-Response" }
9214           }
9215         }
9216       }
9217     }
9218   },
9219   "parameters": {
9220     "interface" : {
9221       "in" : "query",
9222       "name" : "if",
9223       "type" : "string",
9224       "enum" : ["oic.if.baseline"]
9225     }
9226   },
9227   "definitions": {
9228     "TokenRefresh-Request" : {
9229       "properties": {
9230         "refreshtoken": {
9231           "description": "Refresh token received by account management or during token refresh
9232 procedure.",
```

```

9233         "pattern": "(?!$|\\s+).*",
9234         "type": "string"
9235     },
9236     "uid": {
9237         "description": "Format pattern according to IETF RFC 4122.",
9238         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9239         "type": "string"
9240     },
9241     "di": {
9242         "description": "Format pattern according to IETF RFC 4122.",
9243         "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}$",
9244         "type": "string"
9245     }
9246 },
9247 "type": "object",
9248 "required": ["uid", "di", "refreshToken"]
9249 },
9250 "TokenRefresh-Response" : {
9251     "properties": {
9252         "expiresin": {
9253             "description": "Access-Token life time in seconds (-1 if permanent).",
9254             "readOnly": true,
9255             "type": "integer"
9256         },
9257         "rt": {
9258             "description": "Resource Type of the Resource.",
9259             "items": {
9260                 "maxLength": 64,
9261                 "type": "string",
9262                 "enum": ["oic.r.tokenrefresh"]
9263             },
9264             "minItems": 1,
9265             "readOnly": true,
9266             "type": "array"
9267         },
9268         "refreshToken": {
9269             "description": "Refresh token received by account management or during token refresh
9270 procedure.",
9271             "pattern": "(?!$|\\s+).*",
9272             "type": "string"
9273         },
9274         "accessToken": {
9275             "description": "Granted Access-Token.",
9276             "pattern": "(?!$|\\s+).*",
9277             "readOnly": true,
9278             "type": "string"
9279         },
9280         "n": {
9281             "$ref":
9282 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9283 schema.json#/definitions/n"
9284         },
9285         "id": {
9286             "$ref":
9287 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9288 schema.json#/definitions/id"
9289         },
9290         "if" :
9291         {
9292             "description": "The interface set supported by this Resource.",
9293             "items": {
9294                 "enum": [
9295                     "oic.if.baseline"
9296                 ],
9297                 "type": "string"
9298             },
9299             "minItems": 1,
9300             "readOnly": true,
9301             "type": "array"
9302         }
9303     },

```

```

9304     "type" : "object",
9305     "required": ["accesstoken", "refreshtoken", "expiresin"]
9306   }
9307 }
9308 }
9309

```

9310 **C.15.5 Property definition**

9311 Table C.26 defines the Properties that are part of the "oic.r.tokenrefresh" Resource Type.

9312 **Table C.26 – The Property definitions of the Resource with type "rt" = "oic.r.tokenrefresh".**

Property name	Value type	Mandatory	Access mode	Description
refreshtoken	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure.
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
expiresin	integer	Yes	Read Only	Access-Token life time in seconds (-1 if permanent).
accesstoken	string	Yes	Read Only	Granted Access-Token.
refreshtoken	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure.
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
id	multiple types: see schema	No	Read Write	

9313 **C.15.6 CRUDN behaviour**

9314 Table C.27 defines the CRUDN operations that are supported on the "oic.r.tokenrefresh"  
 9315 Resource Type.

9316 **Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh".**

Create	Read	Update	Delete	Notify
		post		

9317  
9318  
9319  
9320

## Annex D (informative)

### OID definitions

9321 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to  
9322 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of  
9323 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
9324 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
9325     private(4) enterprise(1) OCF(51414) }
9326
9327 -- OCF Security specific OIDs
9328
9329 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
9330 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9331
9332 -- OCF Security Categories
9333
9334 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
9335 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
9336
9337 -- OCF Security Profiles
9338
9339 sp-undefined ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
9340 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
9341 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
9342 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
9343 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
9344
9345 sp-undefined-v0 ::= ocfSecurityProfileOID (id-sp-undefined 0)
9346 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
9347 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
9348 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
9349 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
9350
9351 ocfSecurityProfileOID ::= UTF8String
9352
9353 -- OCF Security Certificate Policies
9354
9355 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
9356
9357 -- OCF X.509v3 Extensions
9358
9359 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9360 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
9361 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
9362 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
9363
9364 ocfVersion ::= SEQUENCE {
9365     major    INTEGER,
9366     minor    INTEGER,
9367     build    INTEGER}
9368
9369 ocfCompliance ::= SEQUENCE {
9370     version      ocfVersion,
9371     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
9372     deviceName    UTF8String,
9373     deviceManufacturer UTF8String}
9374
9375 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
9376 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
9377
9378 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
9379
9380 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
9381
9382 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
9383 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
9384 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
9385
9386 ocfCPLAttributes ::= SEQUENCE {
9387     cpl-at-IANAPen UTF8String,
9388     cpl-at-model UTF8String,
9389     cpl-at-version UTF8String}
```

**Annex E  
(informative)**

**Security considerations specific to Bridged Protocols**

9390  
9391  
9392  
9393

9394 The text in this Annex is provided for information only. This Annex has no normative impact. This  
9395 information is applicable at the time of initial publication and may become out of date.

**E.1 Security Considerations specific to the AllJoyn Protocol**

9396  
9397

This clause intentionally left empty.

**E.2 Security Considerations specific to the Bluetooth LE Protocol**

9398

9399 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security  
9400 mode has several security levels (see Table E.1)

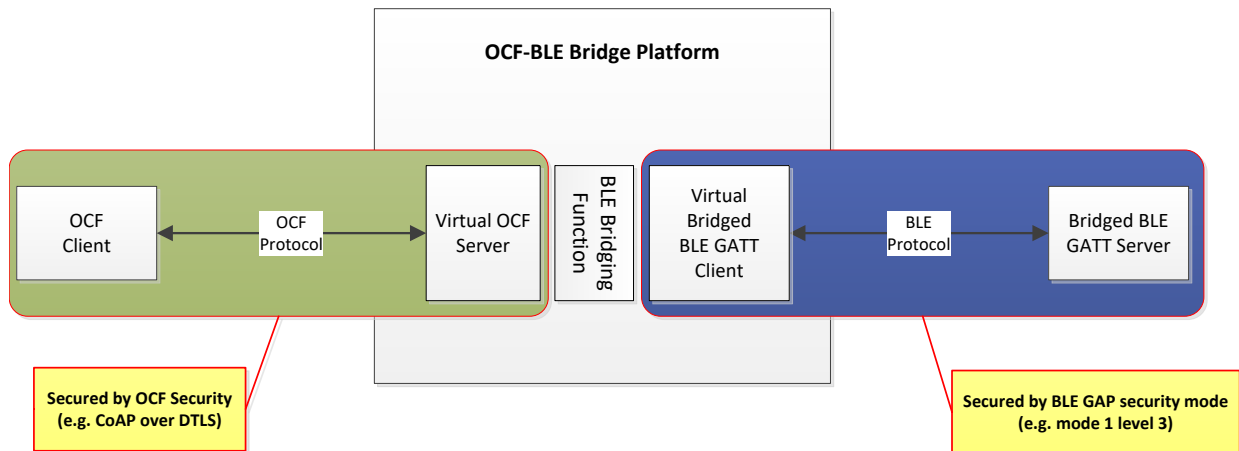
9401 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF  
9402 perspective. The appropriate selection of security mode and level is left to the vendor.

9403

**Table E.1 GAP security mode**

GAP security mode	security level
Security mode 1	1 (no security)
	2 (Unauthenticated pairing with encryption)
	3 (Authenticated pairing with encryption)
	4 (Authenticated LE Secure Connections pairing with encryption)
Security mode 2	1 (Unauthenticated pairing with data signing)
	2 (Authenticated pairing with data signing)

9404 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are  
9405 secured by their own security.



9406  
9407

**Figure E-1 Security Considerations for BLE Bridge**

**E.3 Security Considerations specific to the oneM2M Protocol**

9409

This clause intentionally left empty.

9410



9411 **E.4 Security Considerations specific to the U+ Protocol**

9412 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

9413 **Table E.2 TLS 1.2 Cipher Suites used by U+**

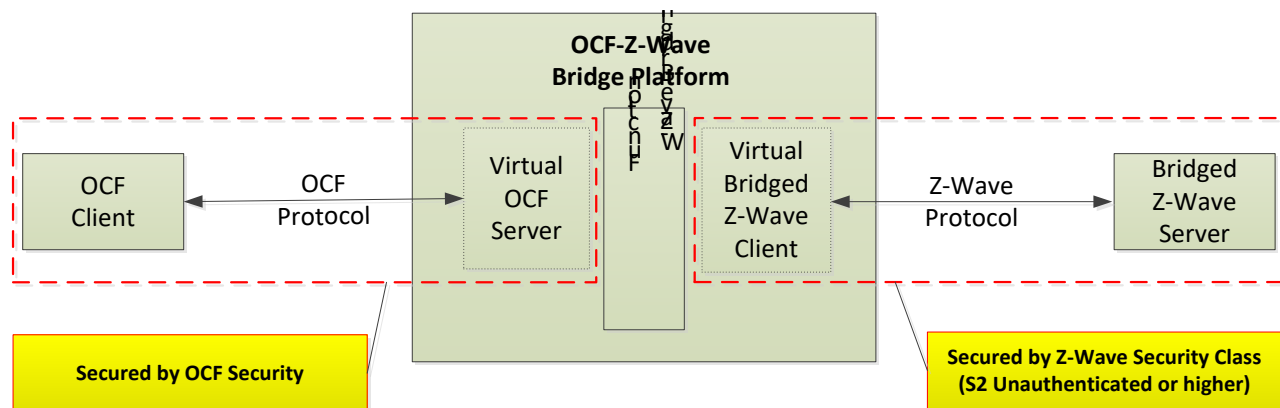
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

9414 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

9415 **E.5 Security Considerations specific to the Z-Wave Protocol**

9416 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2  
 9417 Security Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for  
 9418 communication with a Virtual Bridged Client would typically be considered secure from an OCF  
 9419 perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

9420 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their  
 9421 own security.



9422  
9423

9424

**Figure E-2 Security Considerations for Z-Wave Bridge**

9425 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2  
9426 Unauthenticated provides the following advantages from the security perspective;

- 9427 – The unique device specific key for every secure device enables validation of device identity  
9428 and prevents man-in-the-middle compromises to security
- 9429 – The Secure cryptographic key exchange methods during inclusion achieves high level of  
9430 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 9431 – Out of band key exchange for product authentication which is combined with device specific  
9432 key prevents eavesdropping and man-in-the-middle attack vectors.

9433 See Table E.3 for a summary of Z-Wave Security Classes.

9434

**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

9435 On the other hand, S0 Security Class has the vulnerability of security during inclusion by  
9436 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the  
9437 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices  
9438 might be no longer secure in that case.

9439 **E.6 Security Considerations specific to the Zigbee Protocol**

9440 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the  
9441 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0  
9442 stack, "nwkSecurityLevel", represents the security level of a device.

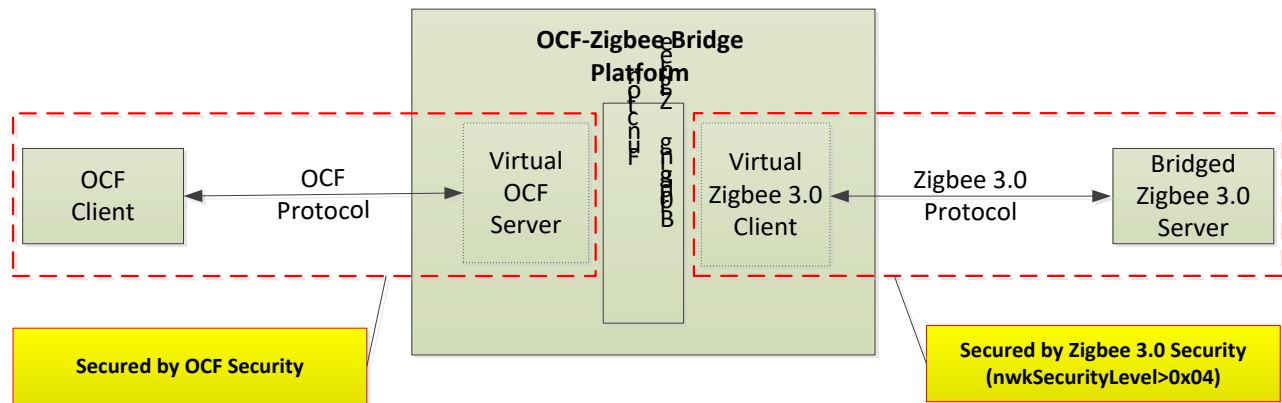
9443 The security level `nwkSecurityLevel > 0x04` provides message integrity code (MIC) and/or  
 9444 AES128-CCM encryption (ENC). Zigbee Servers using `nwkSecurityLevel > 0x04` would typically  
 9445 be considered secure from an OCF perspective. The appropriate selection for `nwkSecurityLevel`  
 9446 is left to the vendor.

9447 See Table E.4 for a summary of the Zigbee Security Levels.

9448 **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)

9449 Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are  
 9450 secured by their own security.



9451  
 9452

9453 **Figure E-3 Security Considerations for Zigbee Bridge**