

OCF Security Specification

VERSION 2.0.3 | June 2019



OPEN CONNECTIVITY
FOUNDATION™

CONTACT 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	Tables	
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		

472 1 Scope

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

477 2 Normative References

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

482 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)
483 Specification -- Part 1: Core specification
484 <https://www.iso.org/standard/53238.html>
485 Latest version available at:
486 https://openconnectivity.org/specs/OCF_Core_Specification.pdf

487 ISO/IEC 30118-3:2018 Information technology -- Open Connectivity Foundation (OCF)
488 Specification -- Part 3: Bridging specification
489 <https://www.iso.org/standard/74240.html>
490 Latest version available at:
491 https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

492 OCF Wi-Fi Easy Setup, Information technology – Open Connectivity Foundation (OCF)
493 Specification – Part 7: Wi-Fi Easy Setup specification
494 Latest version available at:
495 https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf

496 OCF Cloud Specification, Information technology – Open Connectivity Foundation (OCF)
497 Specification – Part 8: Cloud Specification
498 Latest version available at:
499 https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf

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

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

503 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September
504 2000, <https://tools.ietf.org/html/rfc2898>

505 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November
506 2000, <https://tools.ietf.org/html/rfc2986>

507 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December
508 2005, <https://tools.ietf.org/html/rfc4279>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

543

544 **3 Terms, definitions, and abbreviated terms**

545 **3.1 Terms and definitions**

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

548 ISO and IEC maintain terminological databases for use in standardization at the following
549 addresses:

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

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

552 **3.1.1**

553 **Access Management Service (AMS)**

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

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

557 **3.1.2**

558 **Access Token**

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

561 **3.1.3**

562 **Authorization Provider**

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

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

566 **3.1.4**

567 **Client**

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

569 **3.1.5**

570 **Credential Management Service (CMS)**

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

573 **3.1.6**

574 **Device**

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

576 **3.1.7**

577 **Device Class**

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

581 **3.1.8**

582 **Device ID**

583 a stack instance identifier.

584 **3.1.9**

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

586 a logical entity that establishes device ownership

587 **3.1.10**
588 **Device Registration**
589 a process by which Device is enrolled/registered to the OCF Cloud infrastructure (using Device
590 certificate and unique credential) and becomes ready for further remote operation through the
591 cloud interface (e.g. connection to remote Resources or publishing of its own Resources for
592 access).

593 **3.1.11**
594 **End-Entity**
595 any certificate holder which is not a Root or Intermediate Certificate Authority.

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

597 **3.1.12**
598 **Entity**

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

600 **3.1.13**
601 **OCF Interface**

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

603 **3.1.14**
604 **Intermediary**

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

607 **3.1.15**
608 **OCF Cipher Suite**

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

612 **3.1.16**
613 **OCF Cloud User**

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

616 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
617 organization that onboarded that Device. The OCF Cloud User delegates, to the OCF Cloud authority, authority to route
618 between Devices registered by the OCF Cloud User. The OCF Cloud delegates, to the OCF Cloud User, authority to
619 select the set of Devices which can register and use the services of the OCF Cloud.

620 **3.1.17**
621 **OCF Rooted Certificate Chain**

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

625 **3.1.18**
626 **Onboarding Tool (OBT)**

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

628 **3.1.19**
629 **Out of Band Method**

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

631 **3.1.20**
632 **Owner Credential (OC)**

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

635 **3.1.21**
636 **Platform ID**
637 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

638 **3.1.22**
639 **Property**
640 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

641 **3.1.23**
642 **Resource**
643 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

644 **3.1.24**
645 **Role (Network context)**
646 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

647 **3.1.25**
648 **Role Identifier**
649 a Property of an OCF credentials Resource or element in a role certificate that identifies a
650 privileged role that a Server Device associates with a Client Device for the purposes of making
651 authorization decisions when the Client Device requests access to Device Resources.

652 **3.1.26**
653 **Secure Resource Manager (SRM)**
654 a module in the OCF Core that implements security functionality that includes management of
655 security Resources such as ACLs, credentials and Device owner transfer state.

656 **3.1.27**
657 **Security Virtual Resource (SVR)**
658 a resource supporting security features.
659 Note 1 to entry: For a list of all the SVRs please see clause 13.

660 **3.1.28**
661 **Server**
662 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

663 **3.1.29**
664 **Trust Anchor**
665 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.
666 a Device and an OBT(3.1.18)) can assume trust

667 **3.1.30**
668 **Unique Authenticable Identifier**
669 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is
670 used to create the Device ID.
671 Note 1 to entry: The ownership of a UAID may be authenticated by peer Devices.

672 **3.1.31**
673 **Device Configuration Resource (DCR)**
674 a Resource that is any of the following:

675 a) a Discovery Core Resource, or
676 b) a Security Virtual Resource, or
677 c) a Wi-Fi Easy Setup Resource ("oic.r.easyssetup", "oic.r.wificonf", "oic.r.devconf"), or
678 d) a CoAP Cloud Configuration Resource ("oic.r.coapcloudconf"), or
679 e) a Software Update Resource ("oic.r.softwareupdate"), or

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

681 **3.1.32**

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

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

684 **3.1.33**

685 **Bridged Device**

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

687 **3.1.34**

688 **Bridged Protocol**

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

690 **3.1.35**

691 **Bridge**

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

693 **3.1.36**

694 **Bridging Platform**

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

696 **3.1.37**

697 **Virtual Bridged Device**

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

699 **3.1.38**

700 **Virtual OCF Device**

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

702 **3.1.39**

703 **OCF Security Domain**

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

706 **3.1.40**

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

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

709 **3.1.41**

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

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

712 **3.2 Abbreviated terms**

713 **3.2.1**

714 **AC**

715 Access Control

716 **3.2.2**

717 **ACE**

718 Access Control Entry

719 **3.2.3**

720 **ACL**

721 Access Control List

722 **3.2.4**

723 **AES**

724 Advanced Encryption Standard

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

726 **3.2.5**

727 **AMS**

728 Access Management Service

729 **3.2.6**

730 **CMS**

731 Credential Management Service

732 **3.2.7**

733 **CRUDN**

734 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

735 **3.2.8**

736 **CSR**

737 Certificate Signing Request

738 **3.2.9**

739 **CVC**

740 Code Verification Certificate

741 **3.2.10**

742 **ECC**

743 Elliptic Curve Cryptography

744 **3.2.11**

745 **ECDSA**

746 Elliptic Curve Digital Signature Algorithm

747 **3.2.12**

748 **EKU**

749 Extended Key Usage

750 **3.2.13**

751 **EPC**

752 Embedded Platform Credential

753 **3.2.14**

754 **EPK**

755 Embedded Public Key

756 **3.2.15**

757 **DOTS**

758 Device Ownership Transfer Service

759 **3.2.16**

760 **DPKP**

761 Dynamic Public Key Pair

762 **3.2.17**

763 **ID**

764 Identity/Identifier

765 **3.2.18**

766 **JSON**

767 JavaScript Object Notation.

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

769 **3.2.19**
770 **JWS**
771 JSON Web Signature.

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

773 **3.2.20**
774 **KDF**
775 Key Derivation Function

776 **3.2.21**
777 **MAC**
778 Message Authentication Code

779 **3.2.22**
780 **MITM**
781 Man-in-the-Middle

782 **3.2.23**
783 **NVRAM**
784 Non-Volatile Random-Access Memory

785 **3.2.24**
786 **OC**
787 Owner Credential

788 **3.2.25**
789 **OCSP**
790 Online Certificate Status Protocol

791 **3.2.26**
792 **OBT**
793 Onboarding Tool

794 **3.2.27**
795 **OID**
796 Object Identifier

797 **3.2.28**
798 **OTM**
799 Owner Transfer Method

800 **3.2.29**
801 **OOB**
802 Out of Band

803 **3.2.30**
804 **OWASP**
805 Open Web Application Security Project.

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

807 **3.2.31**
808 **PE**
809 Policy Engine

810 **3.2.32**
811 **PIN**
812 Personal Identification Number

813 **3.2.33**
814 **PPSK**
815 PIN-authenticated pre-shared key

816 **3.2.34**
817 **PRF**
818 Pseudo Random Function

819 **3.2.35**
820 **PSI**
821 Persistent Storage Interface

822 **3.2.36**
823 **PSK**
824 Pre Shared Key

825 **3.2.37**
826 **RBAC**
827 Role Based Access Control

828 **3.2.38**
829 **RM**
830 Resource Manager

831 **3.2.39**
832 **RNG**
833 Random Number Generator

834 **3.2.40**
835 **SACL**
836 Signed Access Control List

837 **3.2.41**
838 **SBAC**
839 Subject Based Access Control

840 **3.2.42**
841 **SEE**
842 Secure Execution Environment

843 **3.2.43**
844 **SRM**
845 Secure Resource Manager

846 **3.2.44**
847 **SVR**
848 Security Virtual Resource

849 **3.2.45**
850 **SW**
851 Software

852 **3.2.46**
853 **UAID**
854 Unique Authenticable Identifier

855 **3.2.47**
856 **URI**
857 Uniform Resource Identifier

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

859 **3.2.48**
860 **VOD**
861 Virtual OCF Device

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

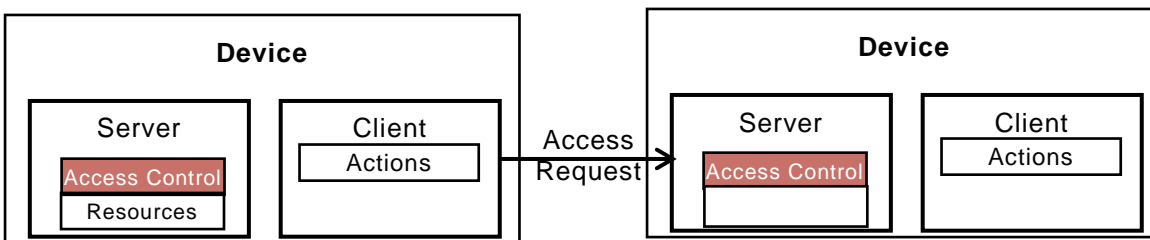
863 **4 Document Conventions and Organization**

864 **4.1 Conventions**

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

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

869 Figure 1 depicts interaction between OCF Devices.



870

871 **Figure 1 – OCF Interaction**

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

876 **4.2 Notation**

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

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

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

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

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

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

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

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

894 **Conditionally allowed** (CA)

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

897 **Conditionally required** (CR)

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

901 **DEPRECATED**

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

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

909 Words that are emphasized are printed in italic.

910 **4.3 Data types**

911 See ISO/IEC 30118-1:2018.

912 **4.4 Document structure**

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

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

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

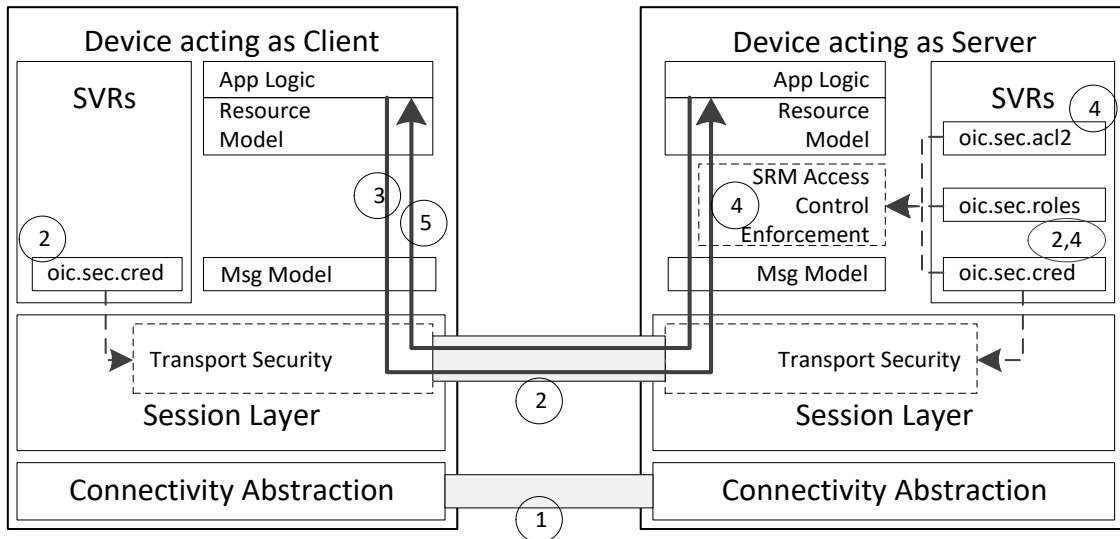
919

920 **5 Security Overview**

921 **5.1 Preamble**

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

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



931
932

933 **Figure 2 – OCF Layers**

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

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

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

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

952 4) The Server receives the request.

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

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

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

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

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

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

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

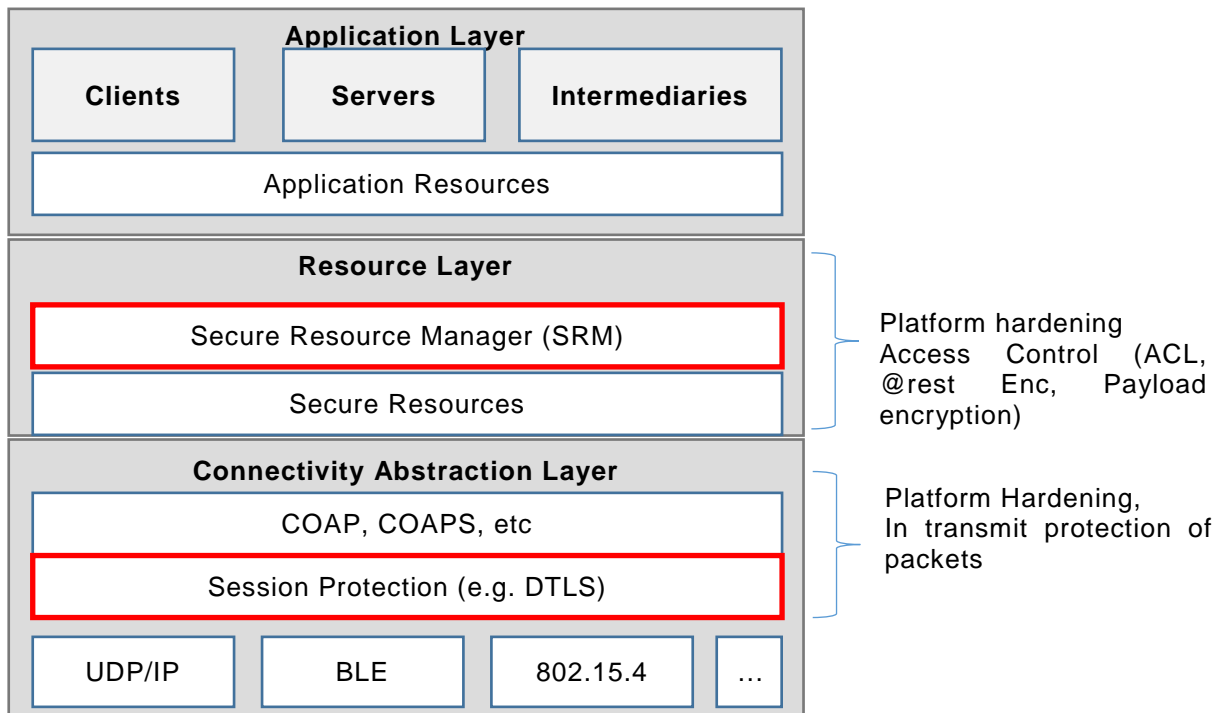
970 5) The Server sends a response back to the Client.

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

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

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

985 Figure 3 depicts OCF Security Enforcement Points.



986
987
988 **Figure 3 – OCF Security Enforcement Points**

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

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

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

999 5.2 Access Control

1000 The OCF framework assumes that Resources are hosted by a Server and are made available to
1001 Clients subject to access control and authorization mechanisms. The Resources at the end point
1002 are protected through implementation of access control, authentication and confidentiality
1003 protection. This clause provides an overview of Access Control (AC) through the use of ACLs.
1004 However, AC in the OCF stack is expected to be transport and connectivity abstraction layer
1005 agnostic.

1006 Implementation of access control relies on a-priori definition of a set of access policies for the
1007 Resource. The policies may be stored by a local ACL or an Access Management Service (AMS)
1008 in form of Access Control Entries (ACE). Two types of access control mechanisms can be applied:

- 1009 – Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of
1010 requestor) of the requesting entity against the subject included in the policy defined for
1011 Resource. Asserting the identity of the requestor requires an authentication process.

1012 – Role-based Access Control (RBAC), where each ACE will match a role identifier included in
1013 the policy for the Resource to a role identifier associated with the requestor.

1014 Some Resources, such as Collections, generate requests to linked Resources when appropriate
1015 Interfaces are used. In such cases, additional access control considerations are necessary.
1016 Additional access control considerations for Collections when using the batch OCF Interface are
1017 found in clause 12.2.7.3.

1018 In the OCF access control model, access to a Resource instance requires an associated ACE.
1019 The lack of such an associated ACE results in the Resource being inaccessible.

1020 The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested
1021 Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier
1022 or (3) wildcard. The way in which the client connects to the server may be relevant context for
1023 making access control decisions. Wildcard matching on authenticated vs. unauthenticated and
1024 encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject
1025 classes.

1026 Example Wildcard Matching Policy:

```
1027 "aclist2": [  
1028   {  
1029     "subject": {"conntype" : "anon-clear" },  
1030     "resources": [  
1031       { "wc": "*" }  
1032     ],  
1033     "permission": 31  
1034   },  
1035   {  
1036     "subject": {"conntype" : "auth-crypt" },  
1037     "resources": [  
1038       { "wc": "*" }  
1039     ],  
1040     "permission": 31  
1041   },  
1042 ]
```

1043 Details of the format for ACL are defined in clause 12. The ACL is composed of one or more
1044 ACEs. The ACL defines the access control policy for the Devices.

1045 ACL Resource requires the same security protection as other sensitive Resources, when it comes
1046 to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW
1047 and SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may
1048 have different scoping levels and thus hardening needs to be specially considered for each
1049 scoping level. For instance, a physical device may host multiple Device implementations and thus
1050 secure storage, usage and isolation of ACLs for different Servers on the same Device needs to
1051 be considered.

1052 **5.2.1 ACL Architecture**

1053 **5.2.1.1 ACL Architecture General**

1054 The Server examines the Resource(s) requested by the client before processing the request. The
1055 access control resource is searched to find one or more ACE entries that match the requestor
1056 and the requested Resources. If a match is found, then permission and period constraints are

1057 applied. If more than one match is found, then the logical UNION of permissions is applied to the
1058 overlapping periods.

1059 The server uses the connection context to determine whether the subject has authenticated or
1060 not and whether data confidentiality has been applied or not. Subject matching wildcard policies
1061 can match on each aspect. If the user has authenticated, then subject matching may happen at
1062 increased granularity based on role or device identity.

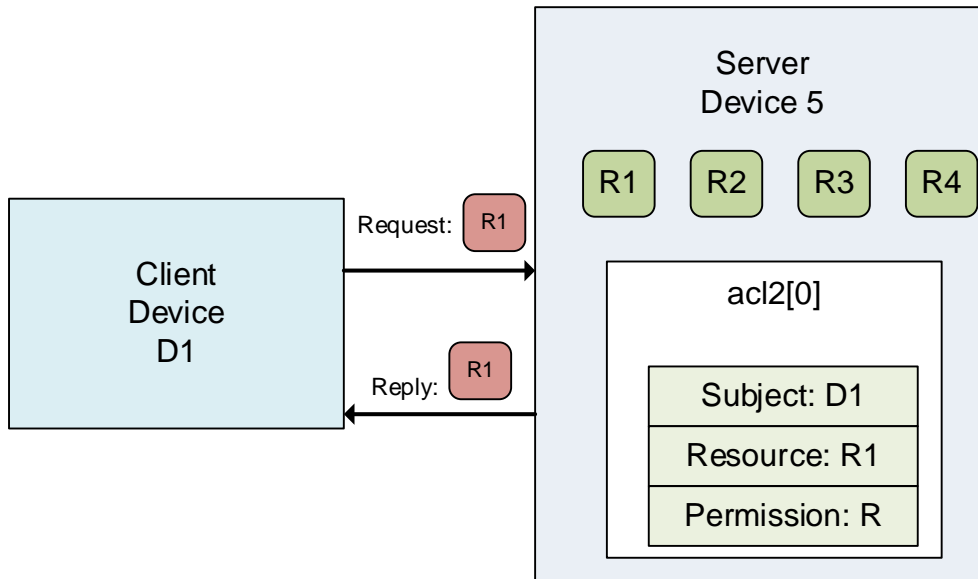
1063 Each ACE contains the permission set that will be applied for a given Resource requestor.
1064 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY
1065 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more
1066 roles. Devices may acquire elevated access permissions when asserting a role. For example, an
1067 ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally
1068 accessible.

1069 5.2.1.2 Use of local ACLs

1070 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control
1071 processing than remote ACL processing by an AMS.
1072 The following use cases describe the operation of access control

1073 Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4).
1074 Client Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0]
1075 corresponds to Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives
1076 access to Resource R1 because the local ACL "/oic/sec/acl2/0" matches the request.

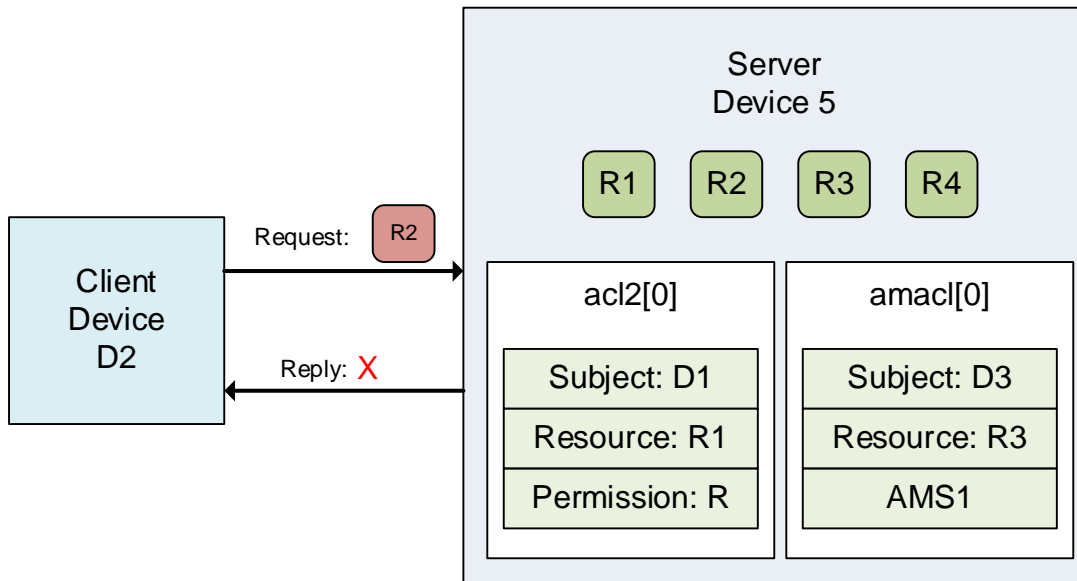
1077



1078

1079 **Figure 4 – Use case-1 showing simple ACL enforcement**

1080 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL
1081 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.



1083

1084

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

1085

5.2.1.3 Use of AMS

1086

1087

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

1088

1089

1090

1091

1092

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

1093

1094

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

1095

1096

1097

1098

1099

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

1100

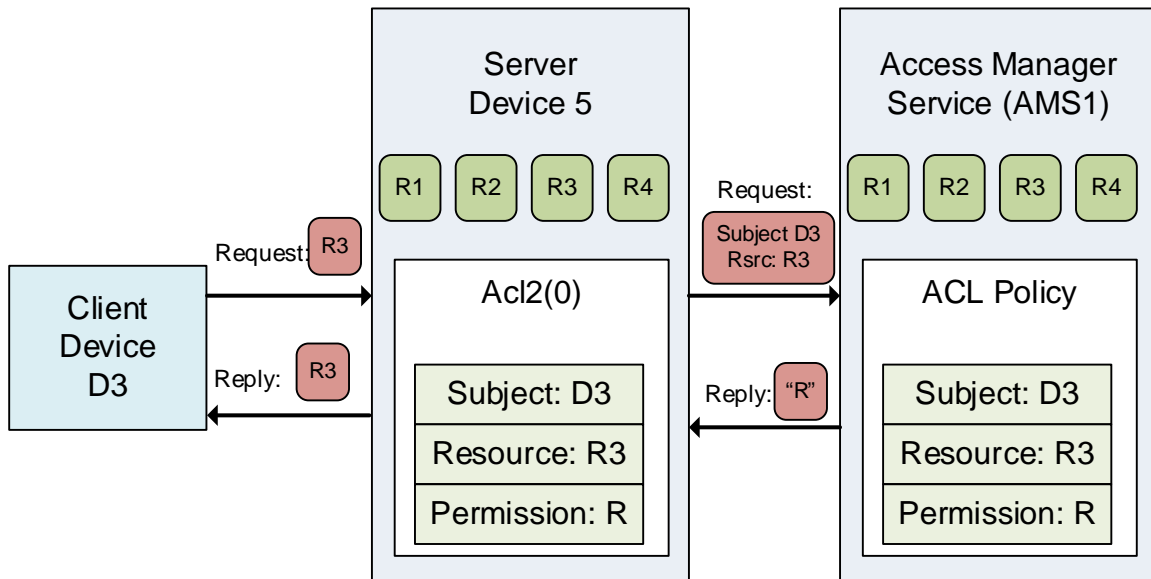
The following use cases describe access control using the AMS:

1101

1102

1103

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



1104

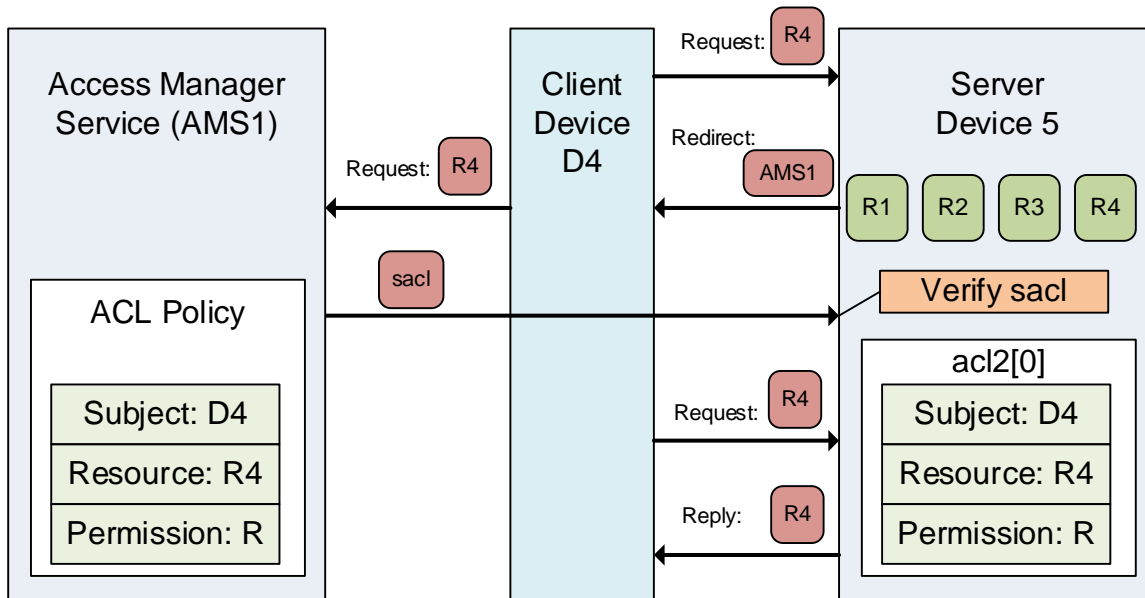
1105

Figure 6 – Use case-3 showing AMS supported ACL

1106 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from
 1107 Server Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1
 1108 by returning an error identifying AMS1 as a "/oic/sec/sacl" Resource issuer. Device D4 obtains
 1109 Sacl1 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the
 1110 "/oic/sec/sacl" Resource and evaluates the ACE policy that grants Perm2 access.

1111 ACE redirection may occur when D4 receives an error result with reason code indicating no
 1112 match exists (i.e. ACCESS_DENIED_NO_ACE). D4 reads the "/oic/sec/acl2" Resource to find the
 1113 "rowneruid" which identifies the AMS and then submits a request to be provisioned, in this
 1114 example the AMS chooses to supply a SACL Resource, however it may choose to re-provision
 1115 the local ACL Resource "/oic/sec/acl2". The request is reissued subsequently. D4 is presumed to
 1116 have been introduced to the AMS as part of Device onboarding or through subsequent credential
 1117 provisioning actions.

1118 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.



1119

1120

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

1121

5.2.2 Access Control Scoping Levels

1122

1123

1124

1125

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

1126

1127

1128

1129

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.

1130

1131

1132

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.

1133

1134

1135

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.

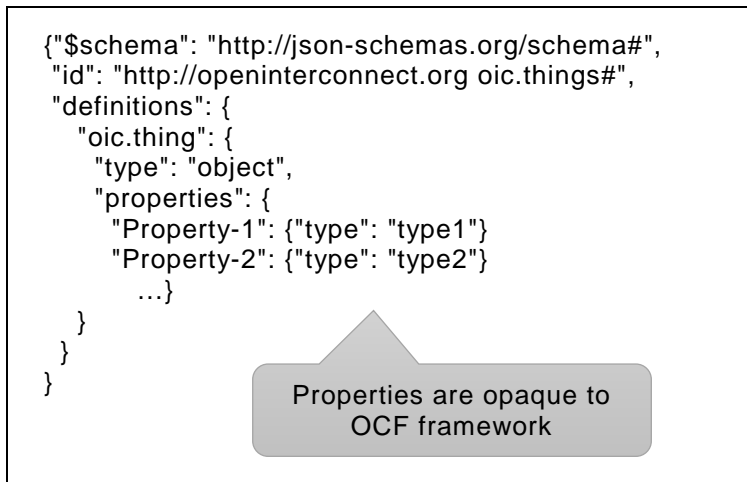
1136

1137

1138

1139

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.

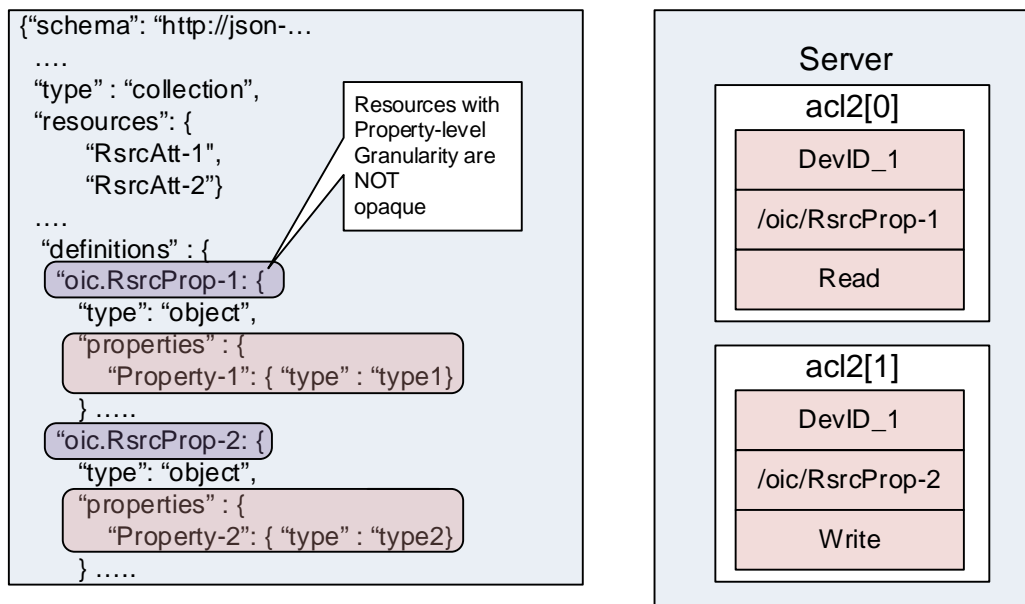


1140

1141

Figure 8 – Example Resource definition with opaque Properties

1142 Currently, OCF framework treats property level information as opaque; therefore, different
 1143 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1
 1144 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into
 1145 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be
 1146 achieved through use of Resource-level ACLs.



1147

1148

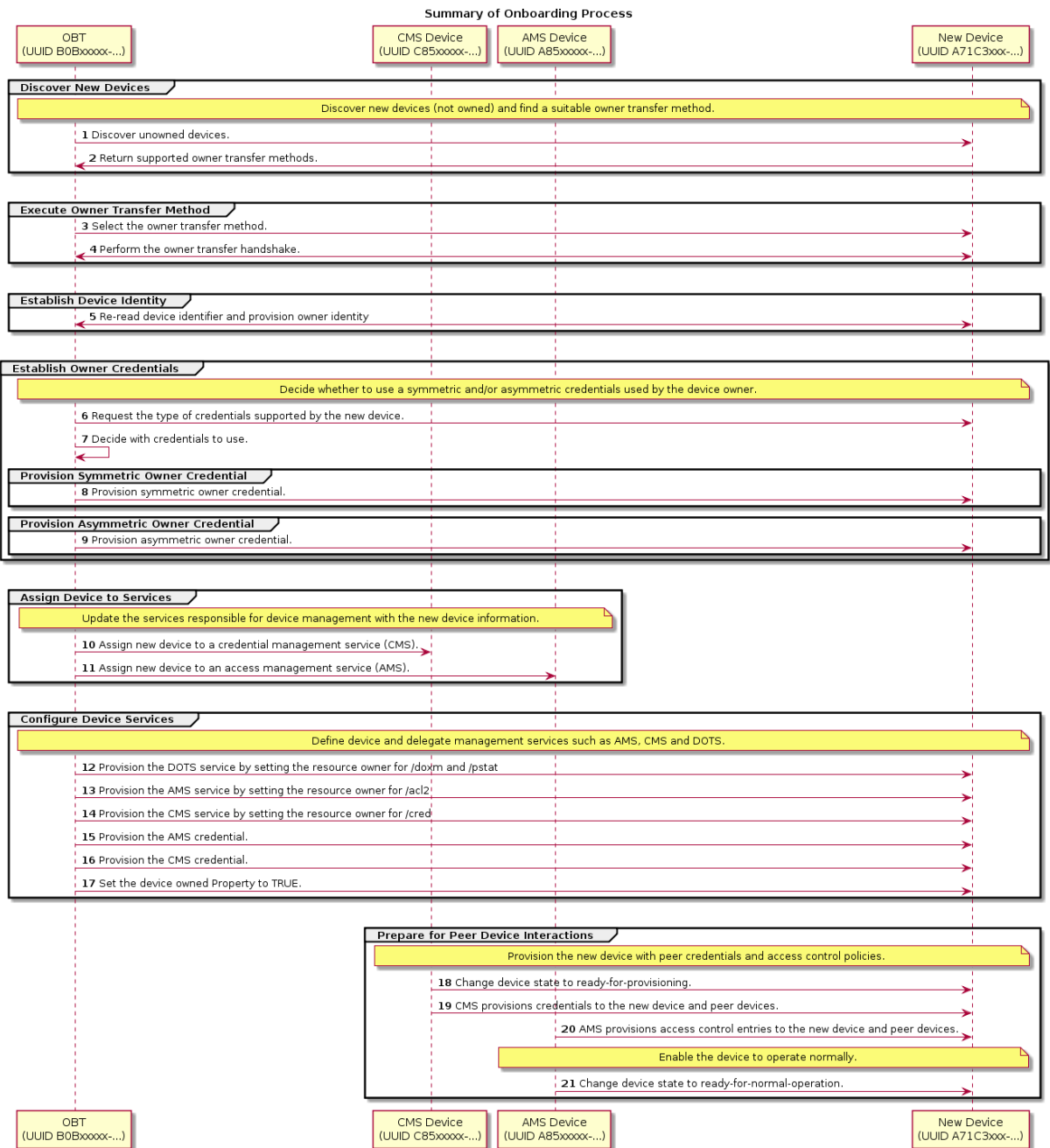
Figure 9 – Property Level Access Control

1149 **5.3 Onboarding Overview**

1150 **5.3.1 Onboarding General**

1151 Before a Device becomes operational in an OCF environment and is able to interact with other
1152 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to
1153 configure the ownership where the legitimate user that owns/purchases the Device uses an
1154 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to
1155 establish ownership. Once ownership is established, the OBT becomes the mechanism through
1156 which the Device can then be provisioned, at the end of which the Device becomes operational
1157 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an
1158 OCF Device.

1159 Figure 10 depicts Onboarding Overview.



1160
1161

Figure 10 – Onboarding Overview

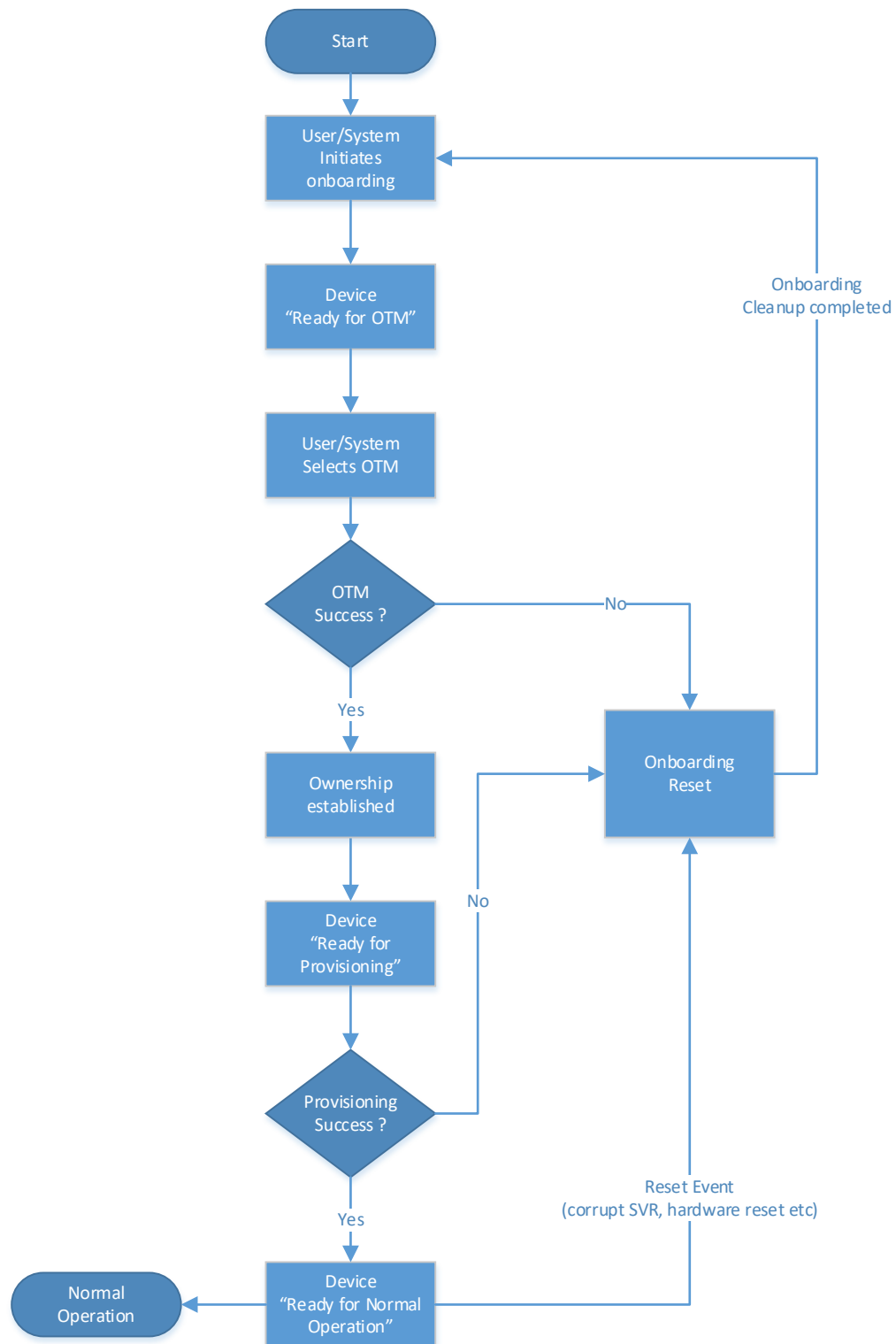
1162 This clause explains the onboarding and security provisioning process but leaves the provisioning
 1163 of non-security aspects to other OCF documents. In the context of security, all Devices are
 1164 required to be provisioned with minimal security configuration that allows the Device to securely
 1165 interact/communicate with other Devices in an OCF environment. This minimal security
 1166 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified
 1167 in 7.5.

1168 Onboarding and provisioning implementations could utilize services defined outside this
 1169 document, it is expected that in using other services, trust between the device being onboarded
 1170 and the various tools is not transitive. This implies that the device being onboarded will
 1171 individually authenticate the credentials of each and every tool used during the onboarding

1172 process; that the tools not share credentials or imply a trust relationship where one has not been
1173 established.

1174 **5.3.2 Onboarding Steps**

1175 The flowchart in Figure 11 shows the typical steps that are involved during onboarding. Although
1176 onboarding may include a variety of non-security related steps, the diagram focus is mainly on
1177 the security related configuration to allow a new Device to function within an OCF environment.
1178 Onboarding typically begins with the Device becoming an Owned Device followed by configuring
1179 the Device for the environment that it will operate in. This would include setting information such
1180 as who can access the Device and what actions can be performed as well as what permissions
1181 the Device has for interacting with other Devices.



1182

1183

Figure 11 – OCF Onboarding Process

1184

5.3.3 Establishing a Device Owner

1185

The objective behind establishing Device ownership is to allow the legitimate user that owns/purchased the Device to assert itself as the owner and manager of the Device. This is done

1186

1187 through the use of an OBT that includes the creation of an ownership context between the new
1188 Device and the OBT tool and asserts operational control and management of the Device. The
1189 OBT can be considered a logical entity hosted by tools/ Servers such as a network management
1190 console, a device management tool, a network-authoring tool, a network provisioning tool, a
1191 home gateway device, or a home automation controller. A physical device hosting the OBT will be
1192 subject to some security hardening requirements, thus preserving integrity and confidentiality of
1193 any credentials being stored. The tool/Server that establishes Device ownership is referred to as
1194 the OBT.

1195 The OBT uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term
1196 owner transfer is used since it is assumed that even for a new Device, the ownership is
1197 transferred from the manufacturer/provider of the Device to the buyer/legitimate user of the new
1198 Device.

1199 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.
1200 Owner transfer establishes the following

- 1201 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"
1202 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during
1203 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property
1204 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the
1205 identity of Device as part of ownership transfer.
- 1206 – The Device owner establishes trust in the Device through the OTM.
- 1207 – Preparing the Device for provisioning by providing credentials that may be needed.

1208 **5.3.4 Provisioning for Normal Operation**

1209 Once the Device has the necessary information to initiate provisioning, the next step is to
1210 provision additional security configuration that allows the Device to become operational. This can
1211 include setting various parameters and may also involve multiple steps. Also provisioning of
1212 ACL's for the various Resources hosted by the Server on the Device is done at this time. The
1213 provisioning step is not limited to this stage only. Device provisioning can happen at multiple
1214 stages in the Device's operational lifecycle. However specific security related provisioning of
1215 Resource and Property state would likely happen at this stage at the end of which, each Device
1216 reaches the Onboarded Device "Ready for Normal Operation" State. The "Ready for Normal
1217 Operation" State is expected to be consistent and well defined regardless of the specific OTM
1218 used or regardless of the variability in what gets provisioned. However individual OTM
1219 mechanisms and provisioning steps may specify additional configuration of Resources and
1220 Property states. The minimal mandatory configuration required for a Device to be in "Ready for
1221 Normal Operation" state is specified in 8.

1222 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview**

1223 As mentioned in the start of clause 5, communication between a Device and OCF Cloud is
1224 subject to different criteria in comparison to Devices which are within a single local network. The
1225 Device is configured in order to connect to the OCF Cloud by a Mediator as specified in the
1226 "CoAPCloudConf" Resource clauses in OCF Cloud Specification. Provisioning includes the
1227 remote connectivity and local details such as URL where the OCF Cloud hosting environment can
1228 be found and the OCF Cloud verifiable Access Token.

1229 **5.3.6 OCF Compliance Management System**

1230 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that
1231 provides Certification status and information for OCF Devices.

1232 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:
1233 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1234 The OBT shall possess the Root Certificate needed to enable https connection to the URI
1235 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1236 The OBT should periodically refresh its copy of the CPL via the URI
1237 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security
1238 Domain owner policy requirements.

1239 **5.4 Provisioning**

1240 **5.4.1 Provisioning General**

1241 In general, provisioning may include processes during manufacturing and distribution of the
1242 Device as well as processes after the Device has been brought into its intended environment
1243 (parts of onboarding process). In this document, security provisioning includes, processes after
1244 ownership transfer (even though some activities during ownership transfer and onboarding may
1245 lead to provisioning of some data in the Device) configuration of credentials for interacting with
1246 provisioning services, configuration of any security related Resources and credentials for dealing
1247 with any services that the Device need to contact later on.

1248 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to
1249 be provisioned with proper security credentials and parameters for regular operation. These
1250 parameters can include:

- 1251 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1252 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the
1253 same OBT, but may be part of AMS in future.

1254 As mentioned, to accommodate a scalable and modular design, these functions are considered
1255 as services that in future could be deployed as separate servers. Currently, the deployment
1256 assumes that these services are all deployed as part of a OBT. Regardless of physical
1257 deployment scenario, the same security-hardening requirement) applies to any physical server
1258 that hosts the tools and security provisioning services discussed here.

1259 Devices are *aware* of their security provisioning status. Self-awareness allows them to be
1260 proactive about provisioning or re-provisioning security Resources as needed to achieve the
1261 devices operational goals.

1262 **5.4.2 Provisioning other services**

1263 To be able to support the use of potentially different device management service hosts, each
1264 Device Secure Virtual Resource (SVR) has an associated Resource owner identified in the
1265 Resource's rowneruuid Property.

1266 The DOTS shall update the rowneruuid Property of the `"/oic/sec/doxm"` and `"/oic/sec/pstat"`
1267 resources with the DOTS resource owner identifier.

1268 The DOTS shall update the rowneruuid Property of the `"/oic/sec/cred"` resource with the CMS
1269 resource owner identifier.

1270 The DOTS shall update the rowneruuid Property of the `"/oic/sec/acl2"` resource with the AMS
1271 resource owner identifier

1272 When these OCF Services are configured, the Device may proactively request provisioning and
1273 verify provisioning requests are authorized. The DOTS shall provision credentials that enable
1274 secure connections between OCF Services and the new Device. The DOTS may initiate client-
1275 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed
1276 provisioning by setting tm Property of the `"/oic/sec/pstat"` Resource.

1277 **5.4.3 Provisioning Credentials for Normal Operation**

1278 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1279 – Pairwise symmetric keys
- 1280 – Group symmetric keys
- 1281 – Certificates
- 1282 – Raw asymmetric keys

1283 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS
1284 credential provisioned by the DOTS.

1285 The following example describes how a Device updates a symmetric key credential involving a
1286 peer Device. The Device discovers the credential to be updated; for example, a secure
1287 connection attempt fails. The Device requests its CMS to supply the updated credential. The
1288 CMS returns an updated symmetric key credential. The CMS updates the corresponding
1289 symmetric key credential on the peer Device.

1290 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1291 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)
1292 asserted by the Client requesting the Resource and compare that role identifier(s) with the
1293 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned
1294 with one or more role credentials.

1295 Each Device holds the role information as a Property within the credential Resource.

1296 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a
1297 certificate role credential.

1298 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned
1299 for a client, access to a Resource is granted if it would be granted under any of the roles.

1300 **5.4.5 ACL provisioning**

1301 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.
1302 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL
1303 policy by updating the Device's ACL Resources.

1304 The AMS shall digitally sign an ACL as part of issuing a "/oic/sec/sacl" Resource if the Device
1305 supports the "/oic/sec/sacl" Resource. The public key used by the Device to verify the signature
1306 shall be provisioned by the CMS as needed. A "/oic/sec/cred" Resource with an asymmetric key
1307 type or signed asymmetric key type is used. The "PublicData" Property contains the AMS's public
1308 key.

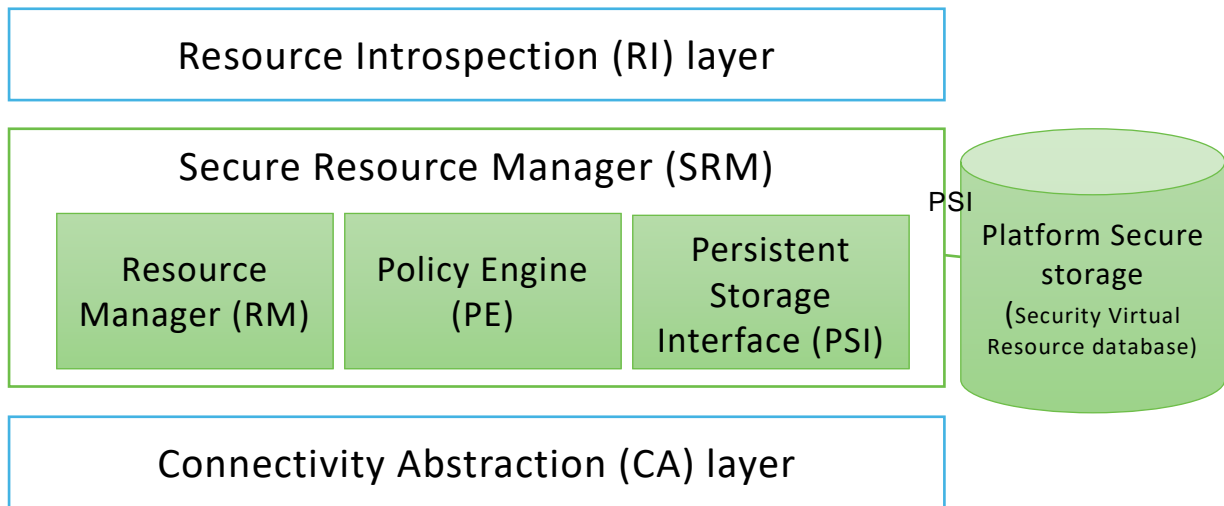
1309 **5.5 Secure Resource Manager (SRM)**

1310 SRM plays a key role in the overall security operation. In short, SRM performs both management
1311 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3
1312 main functional elements:

- 1313 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using
1314 PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3)
1315 Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a
1316 format that is consistent with device-specific data store format. However, the RM will use
1317 JSON format to marshal SVR data structures before being passed to PSI for storage, or travel
1318 off-device.

- 1319 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control
1320 policies responds to the requests with either "ACCESS_GRANTED" or "ACCESS_DENIED".
1321 To make the access decisions, the PE consults the appropriate ACL and looks for best
1322 Access Control Entry (ACE) that can serve the request given the subject (Device or role) that
1323 was authenticated by DTLS.
- 1324 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in
1325 its own memory and storage. The SRM design is modular such that it may be implemented in
1326 the Platform's secure execution environment; if available.

1327 Figure 12 depicts OCF's SRM Architecture.



1328

1329

Figure 12 – OCF's SRM Architecture

1330 **5.6 Credential Overview**

1331 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional
1332 communication. Credentials can be symmetric or asymmetric. Each device stores secret and
1333 public parts of its own credentials where applicable, as well as credentials for other devices that
1334 have been provided by the DOTS or a CMS. These credentials are then used in the
1335 establishment of secure communication sessions (e.g. using DTLS) to validate the identities of
1336 the participating parties. Role credentials are used once an authenticated session is established,
1337 to assert one or more roles for a device.

1338 Access Tokens are provided to an OCF Cloud once an authenticated session with an OCF Cloud
1339 is established, to verify the User ID with which the Device is to be associated.

1340

1341 **6 Security for the Discovery Process**

1342 **6.1 Preamble**

1343 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,
1344 called links) for the Resources hosted by the Server, complemented by attributes about those
1345 Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-
1346 1:2018)

1347 **6.2 Security Considerations for Discovery**

1348 When defining discovery process, care must be taken that only a minimum set of Resources are
1349 exposed to the discovering entity without violating security of sensitive information or privacy
1350 requirements of the application at hand. This includes both data included in the Resources, as
1351 well as the corresponding metadata.

1352 To achieve extensibility and scalability, this document does not provide a mandate on
1353 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on
1354 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any
1355 of the Resources.

1356 The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted
1357 Resources. (See 13.5)

1358 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery
1359 process itself needs to be secured. This document sets the following requirements for the
1360 discovery process:

- 1361 1) Providing integrity protection for discovered Resources.
- 1362 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1363 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)
1364 on the known "/oic/res" Resource.

1365 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a
1366 Server cannot determine the identity of the requester. In such cases, a Server that wants to
1367 authenticate the Client before responding can list the secure discovery URI (e.g.
1368 coaps://IP:PORT/oic/res) in the unsecured "/oic/res" Resource response. This means the secure
1369 discovery URI is by default discoverable by any Client. The Client will then be required to send a
1370 separate unicast request using DTLS to the secure discovery URI.

1371 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to
1372 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN
1373 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1374 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource
1375 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1376 {  
1377   "aclist2": [  
1378     {  
1379       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1380       "resources": [{"href": "/door"}],  
1381       "permission": 2, // RETRIEVE  
1382       "aceid": 1  
1383     }  
1384   ],
```

```

1385     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1386   }
1387   {
1388     "aclist2": [
1389       {
1390         "subject": {"authority": "owner", "role": "owner"}
1391         "resources": [{"href": "/door"}],
1392         "permission": 2, // RETRIEVE
1393         "aceid": 2
1394       }
1395     ],
1396     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1397   }
1398   {
1399     "aclist2": [
1400       {
1401         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1402         "resources": [{"href": "/door/lock"}],
1403         "permission": 4, // UPDATE
1404         "aceid": 3
1405       }
1406     ],
1407     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1408   }
1409   {
1410     "aclist2": [
1411       {
1412         "subject": {"conntype": "anon-clear"},
1413         "resources": [{"href": "/light"}],
1414         "permission": 2, // RETRIEVE
1415         "aceid": 4
1416       }
1417     ],
1418     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1419   }

```

1420 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when
1421 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will
1422 include the URI of the "/door" Resource metadata. Client "d2" will have access to both the
1423 Resources. ACE2 will prevent "d4" from update.

1424 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1425   [
1426     {
1427       "href": "/door",
1428       "rt": ["oic.r.door"],
1429       "if": ["oic.if.b", "oic.if.ll"],

```

```
1430     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1431   }
1432 ]
```

1433 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1434 [
1435   {
1436     "href": "/door",
1437     "rt": ["oic.r.door"],
1438     "if": ["oic.if.b", "oic.if.ll"],
1439     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1440   },
1441   {
1442     "href": "/door/lock",
1443     "rt": ["oic.r.lock"],
1444     "if": ["oic.if.b"],
1445     "type": ["application/json", "application/exi+xml"]
1446   }
1447 ]
```

1448 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

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

1458 Discovery results delivered to any device regarding d3's "/oic/res" Resource from the unsecure interface:

```
1460 [
1461   {
1462     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1463     "href": "/light",
1464     "rt": ["oic.r.light"],
1465     "if": ["oic.if.s"]
1466   }
1467 ]
1468
```

1469 **7 Security Provisioning**

1470 **7.1 Device Identity**

1471 **7.1.1 General Device Identity**

1472 Each Device, which is a logical device, is identified with a Device ID.

1473 Devices shall be identified by a Device ID value that is established as part of device onboarding.
1474 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. "urn:uuid"). Device IDs shall
1475 be unique within the scope of operation of the corresponding OCF Security Domain, and should
1476 be universally unique. The DOTS shall ensure Device ID of the new Device is unique within the
1477 scope of the owner's OCF Security Domain. The DOTS shall verify the chosen new device
1478 identifier does not conflict with Device IDs previously introduced into the OCF Security Domain.

1479 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"
1480 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying
1481 authentication credentials of a peer device.

1482 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of
1483 credentials, both its own and other Device credentials, in the "/oic/sec/cred" Resource. The
1484 device ID can be used to distinguish between a device's own credential, and credentials for other
1485 devices. Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the
1486 device.

1487 Device ID shall be:

- 1488 – Unique
- 1489 – Immutable
- 1490 – Verifiable

1491 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the
1492 device as described later in this clause.

1493 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The
1494 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in
1495 the device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1496 An OCF Platform may have a secure execution environment, which shall be used to secure
1497 unique identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to
1498 provide each Device with the appropriate and separate security.

1499 **7.1.2 Device Identity for Devices with UAID [Deprecated]**

1500 This clause is intentionally left blank.

1501 **7.2 Device Ownership**

1502 This is an informative clause. Devices are logical entities that are security endpoints that have an
1503 identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first
1504 initialized. Establishing device ownership is a process by which the device asserts its identity to
1505 the DOTS and the DOTS provisions an owner identity. This exchange results in the device
1506 changing its ownership state, thereby preventing a different DOTS from asserting administrative
1507 control over the device.

1508 The ownership transfer process starts with the OBT discovering a new device that is in Unowned
1509 state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new
1510 device. At the end of ownership transfer, the following is accomplished:

- 1511 1) The DOTS shall establish a secure session with new device.
1512 2) Optionally asserts any of the following:
1513 a) Proximity (using PIN) of the OBT to the Platform.
1514 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific
1515 attributes.
1516 3) Determines the device identifier.
1517 4) Determines the device owner.
1518 5) Specifies the device owner (e.g. Device ID of the OBT).
1519 6) Provisions the device with owner's credentials.
1520 7) Sets the "Owned" state of the new device to TRUE.

1521 NOTE A Device which connects to the OCF Cloud still retains the ownership established at onboarding with the DOTS.

1522 **7.3 Device Ownership Transfer Methods**

1523 **7.3.1 OTM implementation requirements**

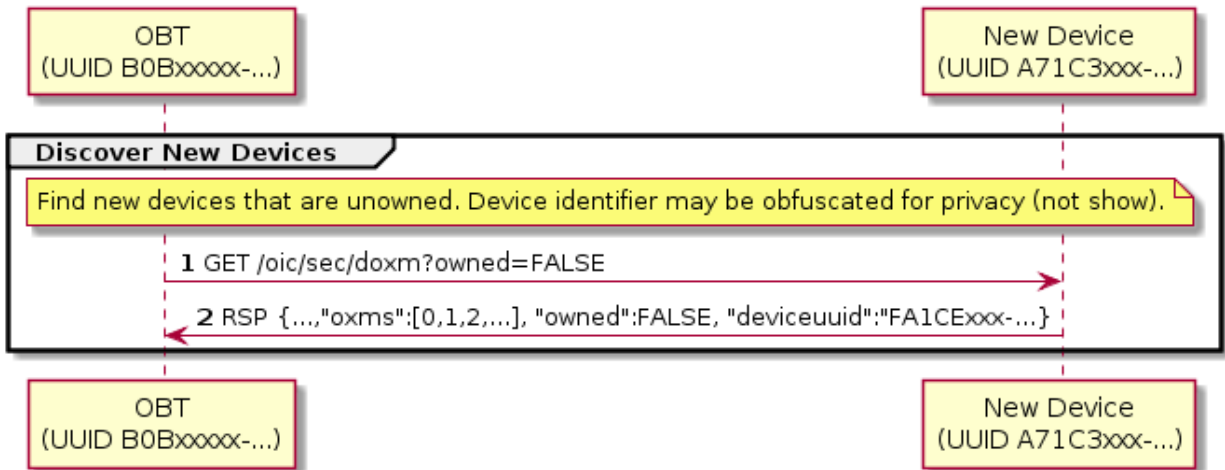
1524 This document provides specifications for several methods for ownership transfer.
1525 Implementation of each individual ownership transfer method is considered optional. However,
1526 each device shall implement at least one of the ownership transfer methods not including vendor
1527 specific methods.

1528 All OTMs included in this document are considered optional. Each vendor is required to choose
1529 and implement at least one of the OTMs specified in this document. The OCF, does however,
1530 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability
1531 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with
1532 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the
1533 preferred approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in
1534 designing vendor-specific OTMs.

1535 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer
1536 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.
1537 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the
1538 "/oic/sec/doxm" Resource. The DOTS shall query the Device's supported credential types using
1539 the "credtype" Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision
1540 credentials according to the credential types supported.

1541 Figure 13 depicts new Device discovery sequence.

Discover New Devices Sequence



1542

1543

Figure 13 – Discover New Device Sequence

1544

1545

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.

1546 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs
 1547 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for
 1548 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the
 1549 new Device.

1550 The new device may have to perform some initialization steps at the beginning of an OTM. For
 1551 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN
 1552 value. The OBT shall POST to the oxmsel property of "/oic/sec/doxm" the value corresponding to
 1553 the OTM being used, before performing other OTM steps. This POST notifies the new device that
 1554 ownership transfer is starting.

1555 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT
 1556 and the OBT to authenticate to the new device.

1557 The DOTS may perform additional provisioning steps subsequent to owner transfer success
 1558 leveraging the established OTM session.

1559 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall
 1560 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array
 1561 includes a SVR.

1562 7.3.2 SharedKey Credential Calculation

1563 The SharedKey credential is derived using a PRF that accepts the key_block value resulting from
1564 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following
1565 calculation to ensure interoperability across vendor products:

1566 SharedKey = PRF(Secret, Message);

1567 Where:

- 1568 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1569 - Secret is the key_block resulting from the DTLS handshake
 - 1570 ▪ See IETF RFC 5246 clause 6.3
 - 1571 ▪ The length of key_block depends on cipher suite.
 - 1572 • (e.g. 96 bytes for TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - 1573 40 bytes for TLS_PSK_WITH_AES_128_CCM_8)
- 1574 - Message is a concatenation of the following:
 - 1575 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
 - 1576 • See clause 13.2.4 for specific DoxmTypes
 - 1577 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
 - 1578 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
 - 1579 ▪ Device ID is new device's UUID Device ID
 - 1580 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1581 - SharedKey Length will be 32 octets.
 - 1582 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used.
 - 1583 DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

1584 7.3.3 Certificate Credential Generation

1585 The Certificate Credential will be used by Devices for secure bidirectional communication. The
1586 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used
1587 to mutually establish the authenticity of the Device. The onboarding details for certificate
1588 generation will be specified in a later version of this document.

1589 7.3.4 Just-Works OTM

1590 7.3.4.1 Just-Works OTM General

1591 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a
1592 secure connection through which a device should be provisioned for use within the owner's OCF
1593 Security Domain. Provisioning additional credentials and Resources is a typical step following
1594 ownership establishment. The pre-shared key is called SharedKey.

1595 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite
1596 defined for the Just-works OTM.

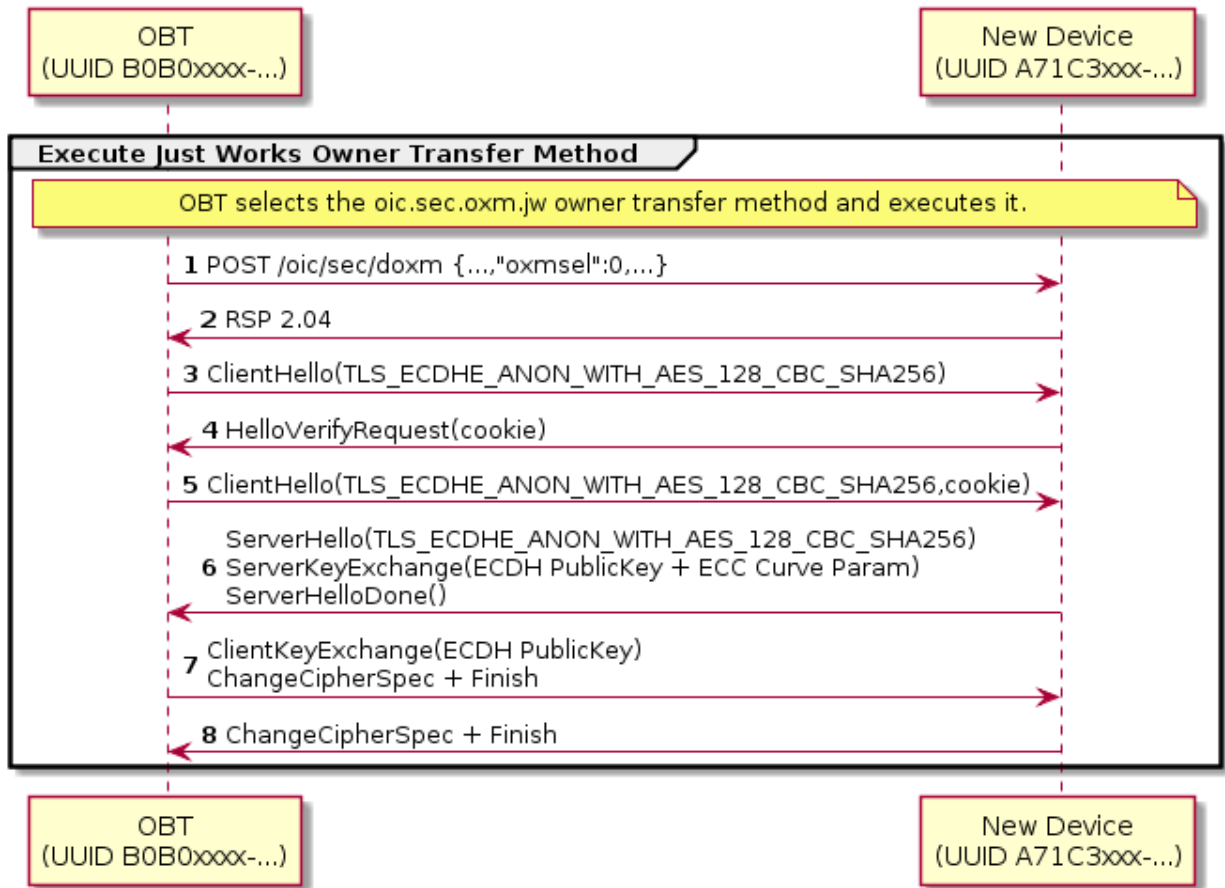
1597 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1598 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
1599 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

1600 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for
1601 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1602 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.

Perform Just-Works Owner Transfer Method



1603

1604

Figure 14 – A Just Works OTM

1605

1606

Table 2 – A Just Works OTM Details

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

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

1607 7.3.4.2 Security Considerations

1608 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this
 1609 method presumes that both the OBT and the new device perform the "just-works" method
 1610 assumes onboarding happens in a relatively safe environment absent of an attack device.

1611 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to
 1612 the device.

1613 The new device should use a temporal device ID prior to transitioning to an owned device while it
 1614 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-

1615 temporal device ID that could differ from the temporal value during the secure session in which
1616 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not
1617 conflict with a Device ID already in use. If it is already in use the existing credentials are used to
1618 establish a secure session.

1619 An un-owned Device that also has established device credentials might be an indication of a
1620 corrupted or compromised device.

1621 **7.3.5 Random PIN Based OTM**

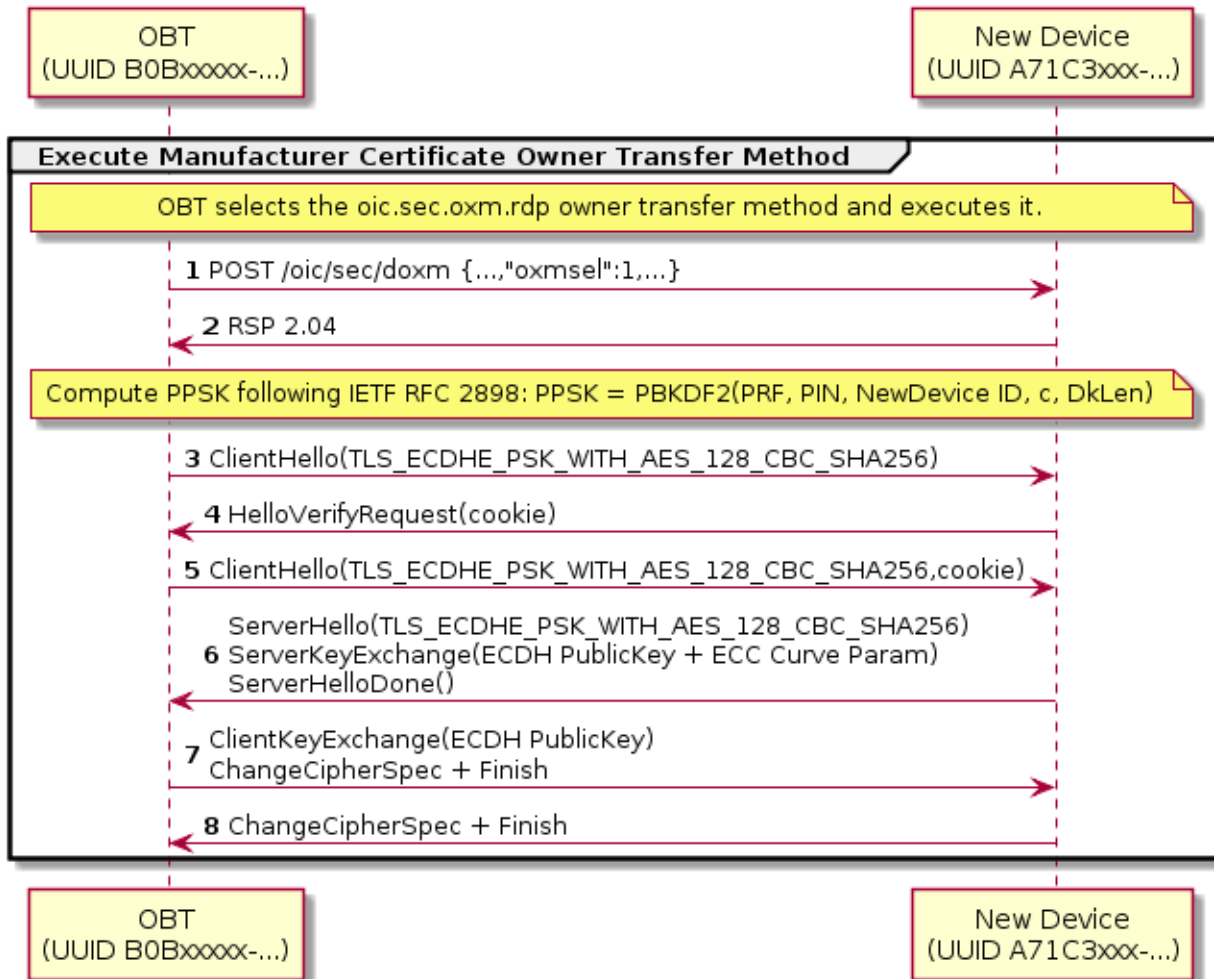
1622 **7.3.5.1 Random PIN OTM General**

1623 The Random PIN method establishes physical proximity between the new device and the OBT
1624 can prevent man-in-the-middle attacks. The Device generates a random number that is
1625 communicated to the OBT over an out-of-band channel. The definition of out-of-band
1626 communications channel is outside the scope of the definition of device OTMs. The OBT and new
1627 Device use the PIN in a key exchange as evidence that someone authorized the transfer of
1628 ownership by having physical access to the new Device via the out-of-band-channel.

1629 **7.3.5.2 Random PIN Owner Transfer Sequence**

1630 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.

Perform Random PIN Device Owner Transfer Method



1631

1632

Figure 15 – Random PIN-based OTM

1633

1634

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.

1635 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF
 1636 RFC 2898 and a PIN exchanged via an out-of-band method to generate a pre-shared key. The
 1637 PIN-authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a PSK.

1638 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)
1639 The PBKDF2 function has the following parameters:
1640 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.
1641 - PIN – obtain via out-of-band channel.
1642 - Device ID – UUID of the new device.
1643 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
1644 - c – Iteration count initialized to 1000
1645 - dkLen – Desired length of the derived PSK in octets.

1646 **7.3.5.3 Security Considerations**

1647 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with
1648 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials
1649 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,
1650 allows an attacker to masquerade as the onboarded device.

1651 It is recommended that the entropy of the PIN be enough to withstand an online brute-force
1652 attack, 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-
1653 9a-z), or a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack
1654 (MITM) is when the attacker is active on the network and can intercept and modify messages
1655 between the OBT and device. In the MITM attack, the attacker must recover the PIN from the key
1656 exchange messages in "real time", i.e., before the peer's time out and abort the connection
1657 attempt. Having recovered the PIN, he can complete the authentication step of key exchange.
1658 The guidance given here calls for a minimum of 40 bits of entropy, however, the assurance this
1659 provides depends on the resources available to the attacker. Given the parallelizable nature of a
1660 brute force guessing attack, the attack enjoys a linear speedup as more cores/threads are added.
1661 A more conservative amount of entropy would be 64 bits. Since the Random PIN OTM requires
1662 using a DTLS ciphersuite that includes an ECDHE key exchange, the security of the Random PIN
1663 OTM is always at least equivalent to the security of the JustWorks OTM.

1664 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN.
1665 The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess
1666 in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an
1667 effective way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify
1668 the reduction, since an X-fold increase in time spent by the honest peers does not directly
1669 translate to an X-fold increase in time by the attacker. This asymmetry is because the attacker
1670 may use specialized implementations and hardware not available to honest peers. For this
1671 reason, when deciding how much entropy to use for a PIN, it is recommended that implementers
1672 assume PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

1673 The Random PIN device OTM security depends on an assumption that a secure out-of-band
1674 method for communicating a randomly generated PIN from the new device to the OBT exists. If
1675 the OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN,
1676 and the attacks described above apply. The out-of-band mechanism should be chosen such that
1677 it requires proximity between the OBT and the new device. The attacker is assumed to not have
1678 compromised the out-of-band-channel. As an example OOB channel, the device may display a
1679 PIN to be entered into the OBT software. Another example is for the device to encode the PIN as
1680 a 2D barcode and display it for a camera on the OBT device to capture and decode.

1681 **7.3.6 Manufacturer Certificate Based OTM**

1682 **7.3.6.1 Manufacturer Certificate Based OTM General**

1683 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the
1684 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device
1685 and the OBT.

1686 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust
1687 anchor.

1688 For some environments, policies or administrators, additional information about device
1689 characteristics may be sought. This list of additional attestations that OCF may or may not have
1690 tested (understanding that some attestations are incapable of testing or for which testing may be
1691 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3
1692 extension described in 9.4.2.2.6.

1693 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with
1694 certificate data to authenticate their identities with the OBT in the process of bringing a new
1695 device into operation on an OCF Security Domain. The onboarding process involves several
1696 discrete steps:

1697 1) Pre-on-board conditions

1698 a) The credential element of the Device's credential Resource ("/oic/sec/cred") containing the
1699 manufacturer certificate shall be identified by the "credusage" Property containing the
1700 string "oic.sec.cred.mfgcert" to indicate that the credential contains a manufacturer
1701 certificate.

1702 b) The manufacturer certificate chain shall be contained in the identified credential element's
1703 "publicdata" Property.

1704 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1705 d) If the device requires authentication of the OBT as part of ownership transfer, it is
1706 presumed that the OBT has been registered and has obtained a certificate for its unique
1707 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.

1708 e) User has configured the OBT app with network access info and account info (if any).

1709 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the
1710 Device may authenticate the OBT to verify the OBT signature.

1711 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready
1712 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted
1713 link in accordance with the negotiated cipher suite.

1714 **7.3.6.2 Certificate Profiles**

1715 See 9.4.2 for details.

1716 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

1717 In order for full, mutual authentication to occur between the device and the OBT, both the device
1718 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This
1719 implies that OCF may need to obtain services from a Certificate Authority (e.g. Symantec,
1720 Verisign, etc.) to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors
1721 are derived.

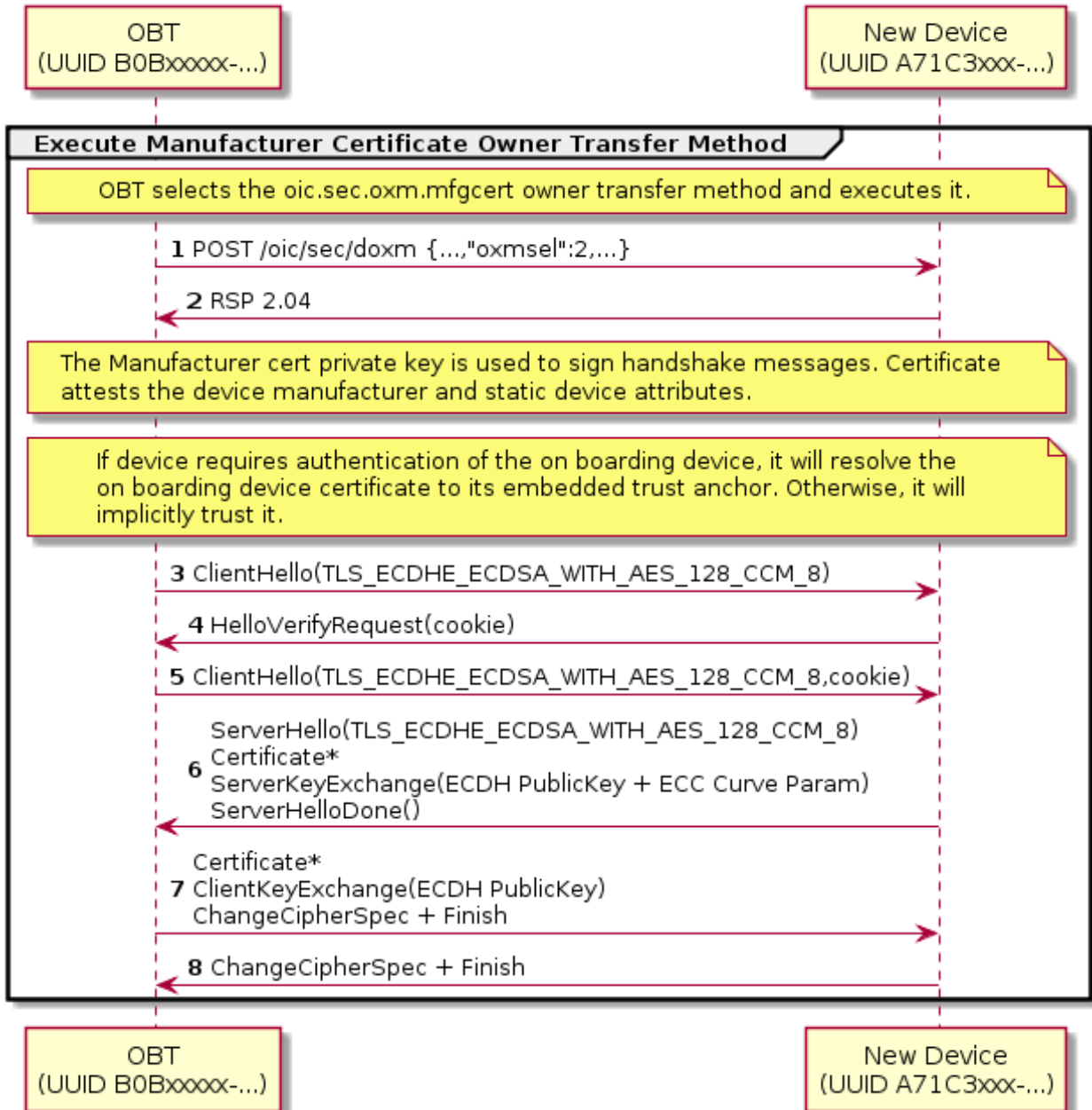
1722 The OBT shall authenticate the device during onboarding. However, the device is not required to
1723 authenticate the OBT due to potential resource constraints on the device.

1724 In the case where the Device does NOT authenticate the OBT software, there is the possibility of
1725 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,
1726 which can compromise OCF Security Domain access credentials and/or personal information.

1727 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1728 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.

Perform Manufacturer Certificate Owner Transfer Method



1729

1730

1731

1732

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.

1733 **7.3.6.5 Security Considerations**

1734 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of
1735 assurance that the private key cannot be compromised.

1736 The Platform manufacturer issues the manufacturer certificate and attests the private key
1737 protection mechanism.

1738 **7.3.7 Vendor Specific OTMs**

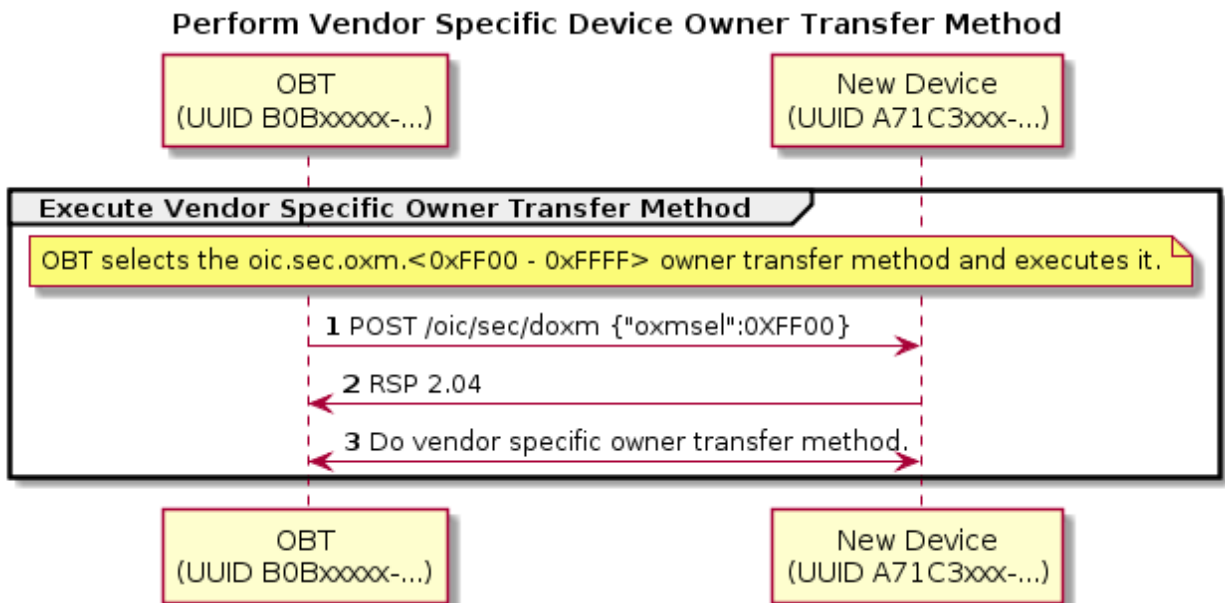
1739 **7.3.7.1 Vendor Specific OTM General**

1740 The OCF anticipates situations where a vendor will need to implement an OTM that
1741 accommodates manufacturing or Device constraints. The Device OTM resource is extensible for
1742 this purpose. Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1743 – The OBT must determine which credential types are supported by the Device. This is
1744 accomplished by querying the Device's "/oic/sec/doxm" Resource to identify supported
1745 credential types.
- 1746 – The OBT provisions the Device with OC(s).
- 1747 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1748 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security
1749 Domain subsequent to ownership establishment.
- 1750 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks
1751 to be performed by a security service.

1752 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1753 Vendor-specific OTM sequence example is shown in Figure 17 and steps described in Table 5.



1754
1755 **Figure 17 – Vendor-specific Owner Transfer Sequence**
1756

1757

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

1758 **7.3.7.3 Security Considerations**

1759 The vendor is responsible for considering security threats and mitigation strategies.

1760 **7.3.8 Establishing Owner Credentials**

1761 Once the OBT and the new Device have authenticated and established an encrypted connection
1762 using one of the defined OTM methods.

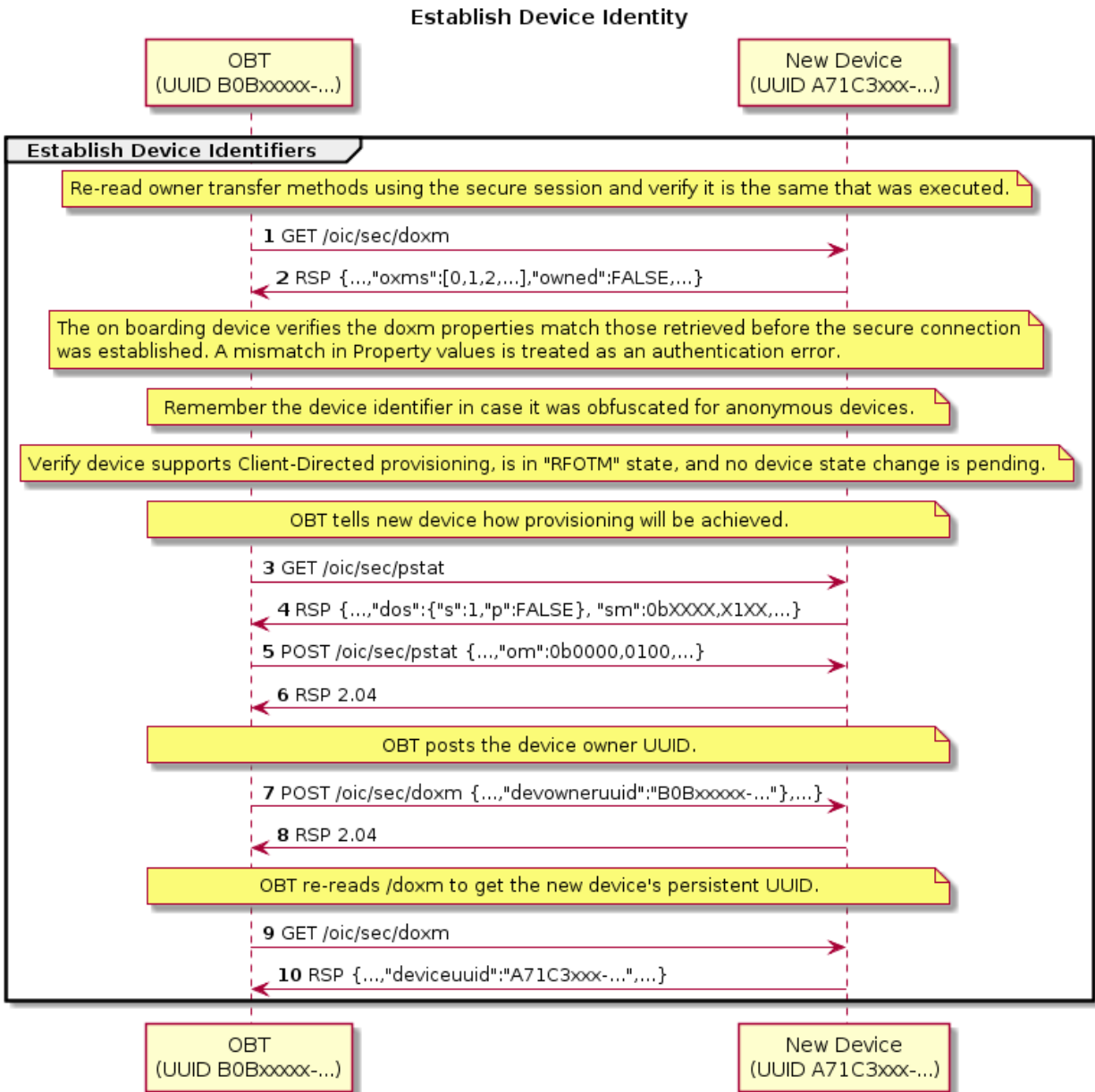
1763 Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security
1764 Domain access information, provisioning functions, shared keys, or Kerberos tickets.

1765 The OBT might then provision the new Device with additional credentials for Device management
1766 and Device-to-Device communications. These credentials may consist of certificates with
1767 signatures, UAID based on the Device public key, PSK, etc.

1768 The steps for establishing Device's owner credentials (OC) are:

- 1769 1) The OBT shall establish the Device ID and Device owner uuid - See Figure 18 and Table 6.
- 1770 2) The OBT then establishes Device's OC - See Figure 19 and Table 7. This can be either:
 - 1771 a) Symmetric credential - See Figure 20 and Table 8.
 - 1772 b) Asymmetric credential - See Figure 21 and Table 9.
- 1773 3) Configure Device services - See Figure 22 and Table 10.
- 1774 4) Configure Device for peer to peer interaction - See Figure 23 and Table 11.

1775



1776
1777
1778
1779

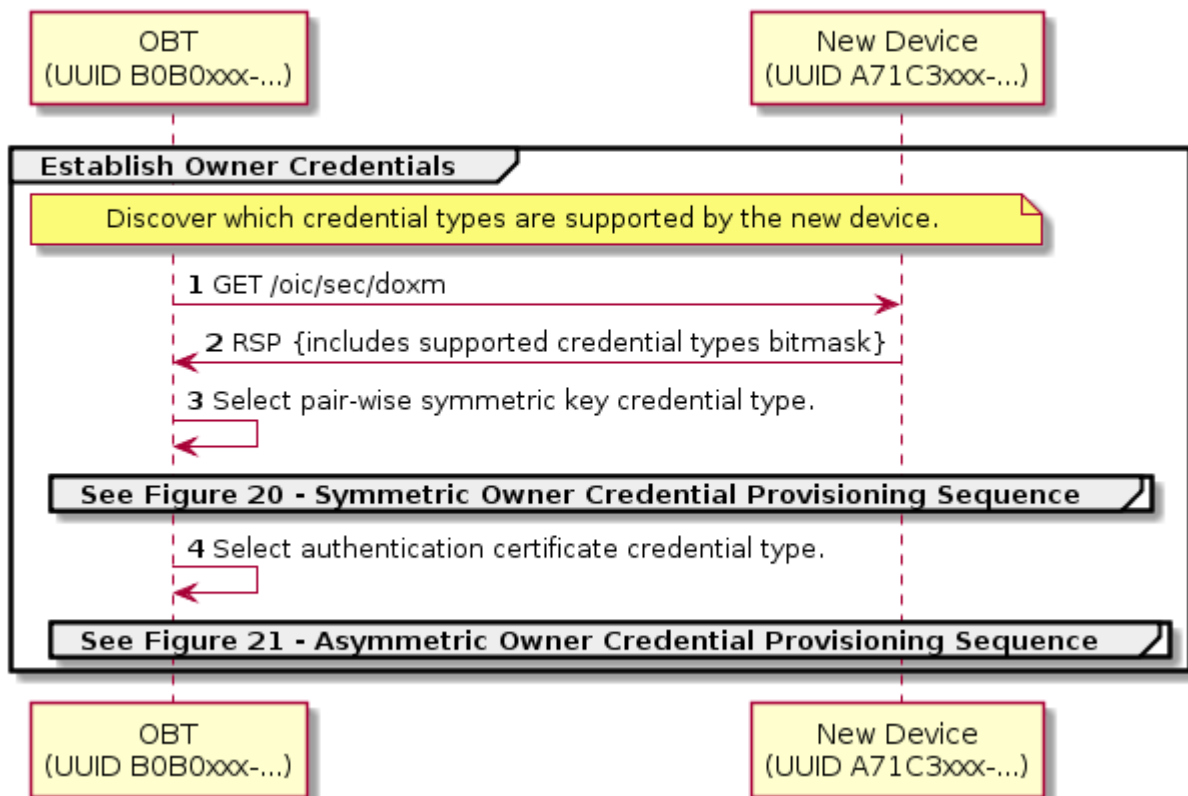
Figure 18 – Establish Device Identity Flow

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



1780

1781

Figure 19 – Owner Credential Selection Provisioning Sequence

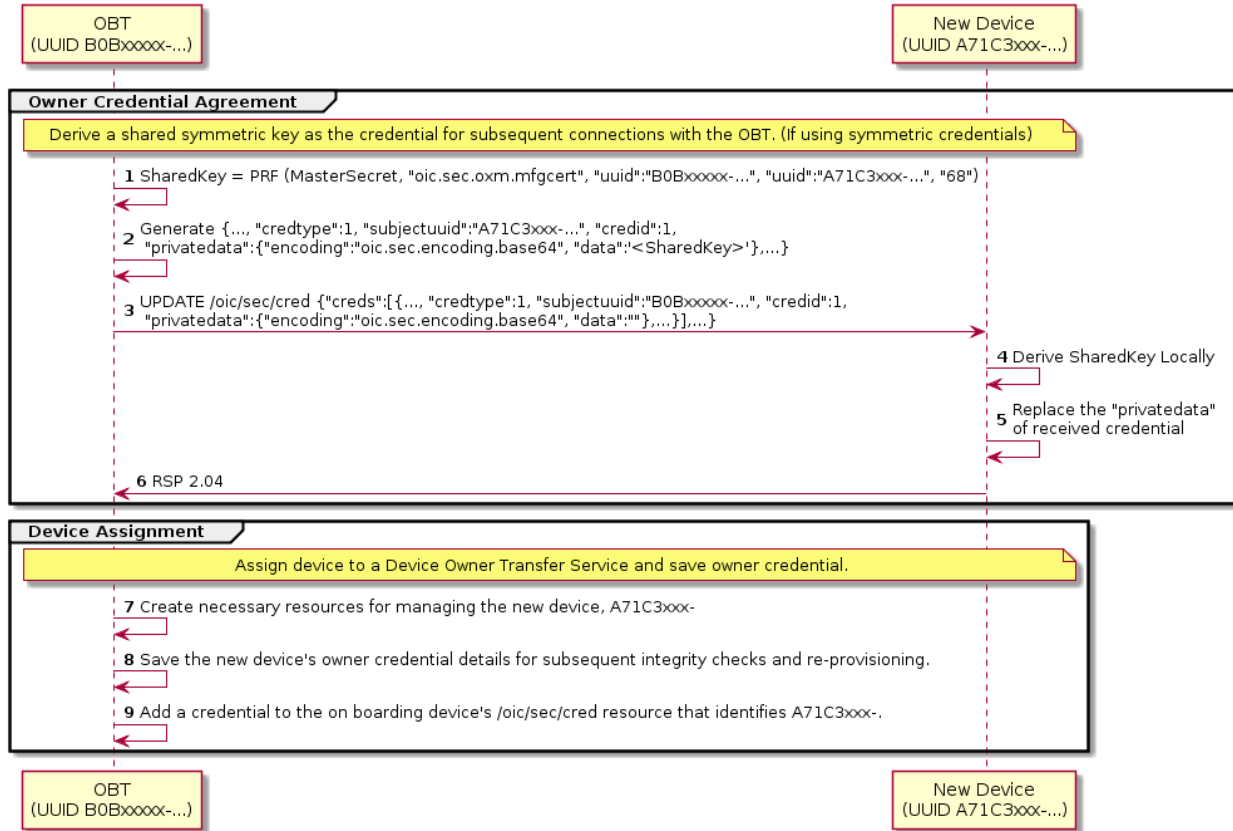
1782

1783

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



1784

1785

1786

1787

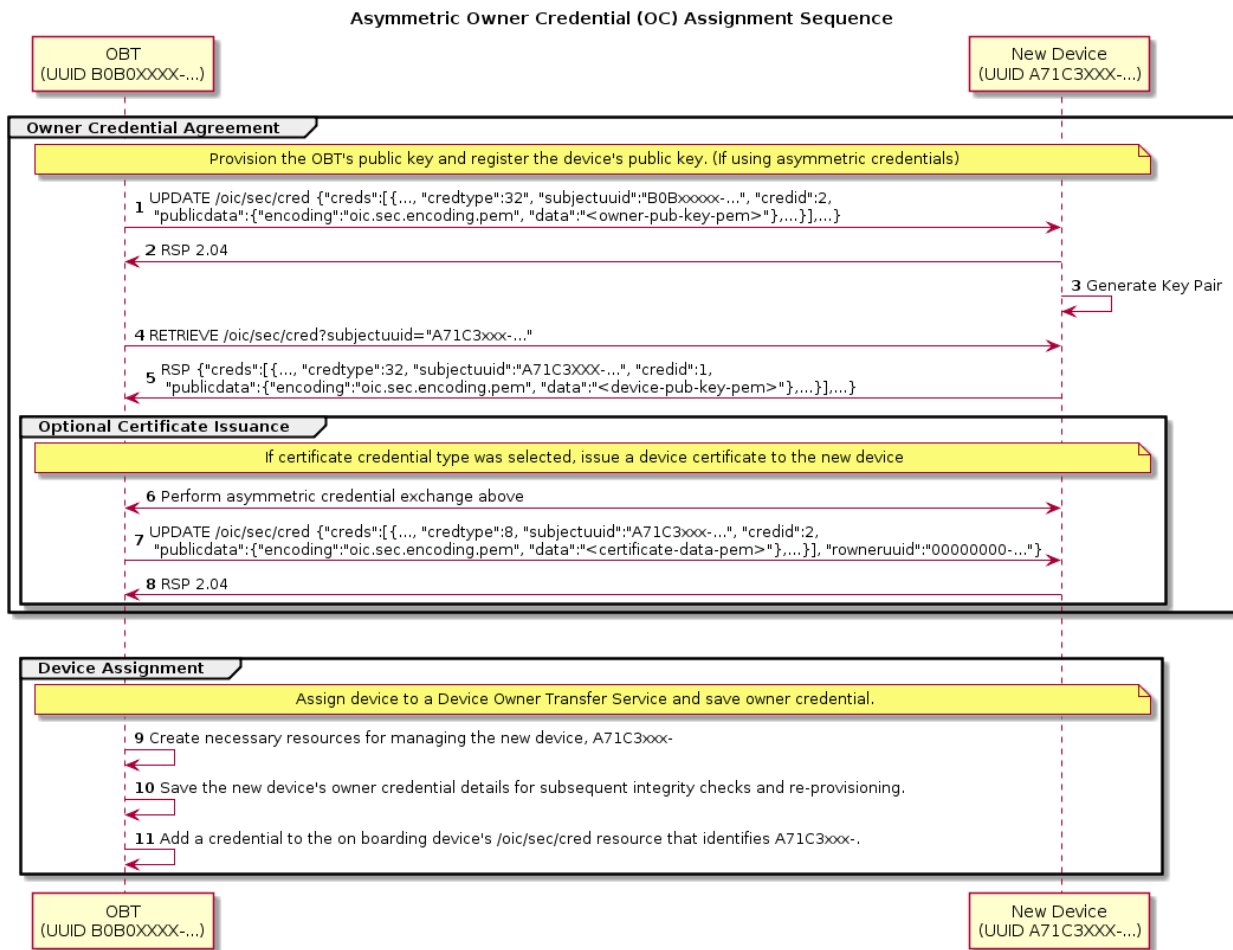
Figure 20 – Symmetric Owner Credential Provisioning Sequence

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.

- 1788 In particular, if the OBT selects symmetric owner credentials:
- 1789 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
 - 1790 described in 7.3.2.
 - 1791 – The OBT shall send an empty key to the new Device's "/oic/sec/cred" Resource, identified as
 - 1792 a symmetric pair-wise key.
 - 1793 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
 - 1794 generate the Shared Key using the SharedKey Credential Calculation method described in
 - 1795 7.3.2 and store it with the owner credential.
 - 1796 – The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred"
 - 1797 Resource to authenticate the owner during subsequent connections.



1798

Figure 21 – Asymmetric Owner Credential Provisioning Sequence

1799

1800

1801

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

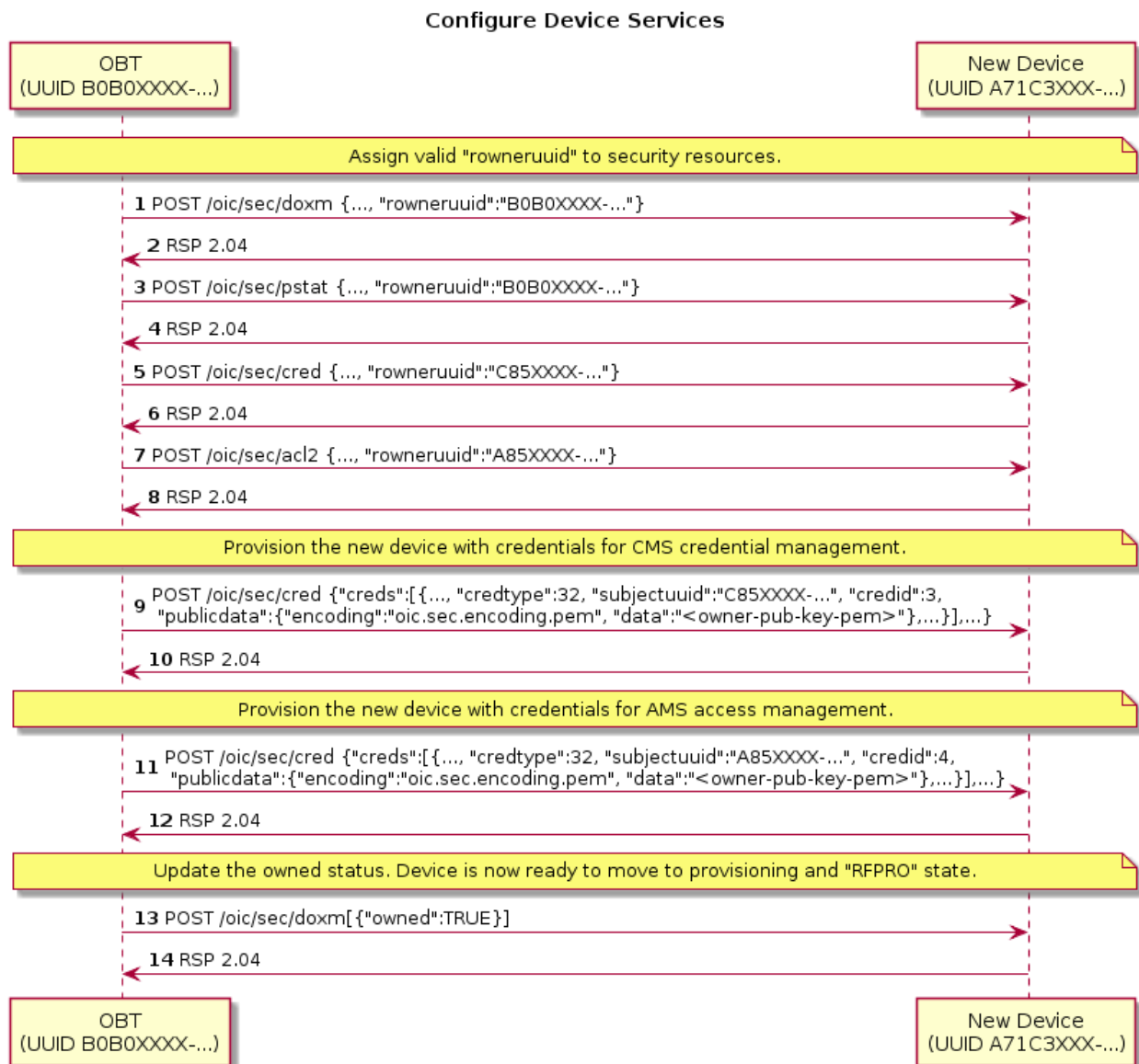
	Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric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

1802 If the OBT selects asymmetric owner credentials:

- 1803 – The OBT shall add its public key to the new Device's "/oic/sec/cred" Resource, identified as
1804 an Asymmetric Encryption Key.
- 1805 – The OBT shall query the "/oic/sec/cred" Resource from the new Device, supplying the new
1806 Device's UUID via the SubjectID query parameter. In response, the new Device shall return
1807 the public Asymmetric Encryption Key, which the OBT shall retain for future owner
1808 authentication of the new Device.

1809 If the OBT selects certificate owner credentials:

- 1810 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the
1811 public key returned by the new Device, signed by a mutually-trusted CA, and complying with
1812 the Certificate Credential Generation requirements defined in 7.3.3.
- 1813 – The OBT shall add the newly-created certificate chain to the "/oic/sec/cred" Resource,
1814 identified as an Asymmetric Signing Key with Certificate.



1815

1816

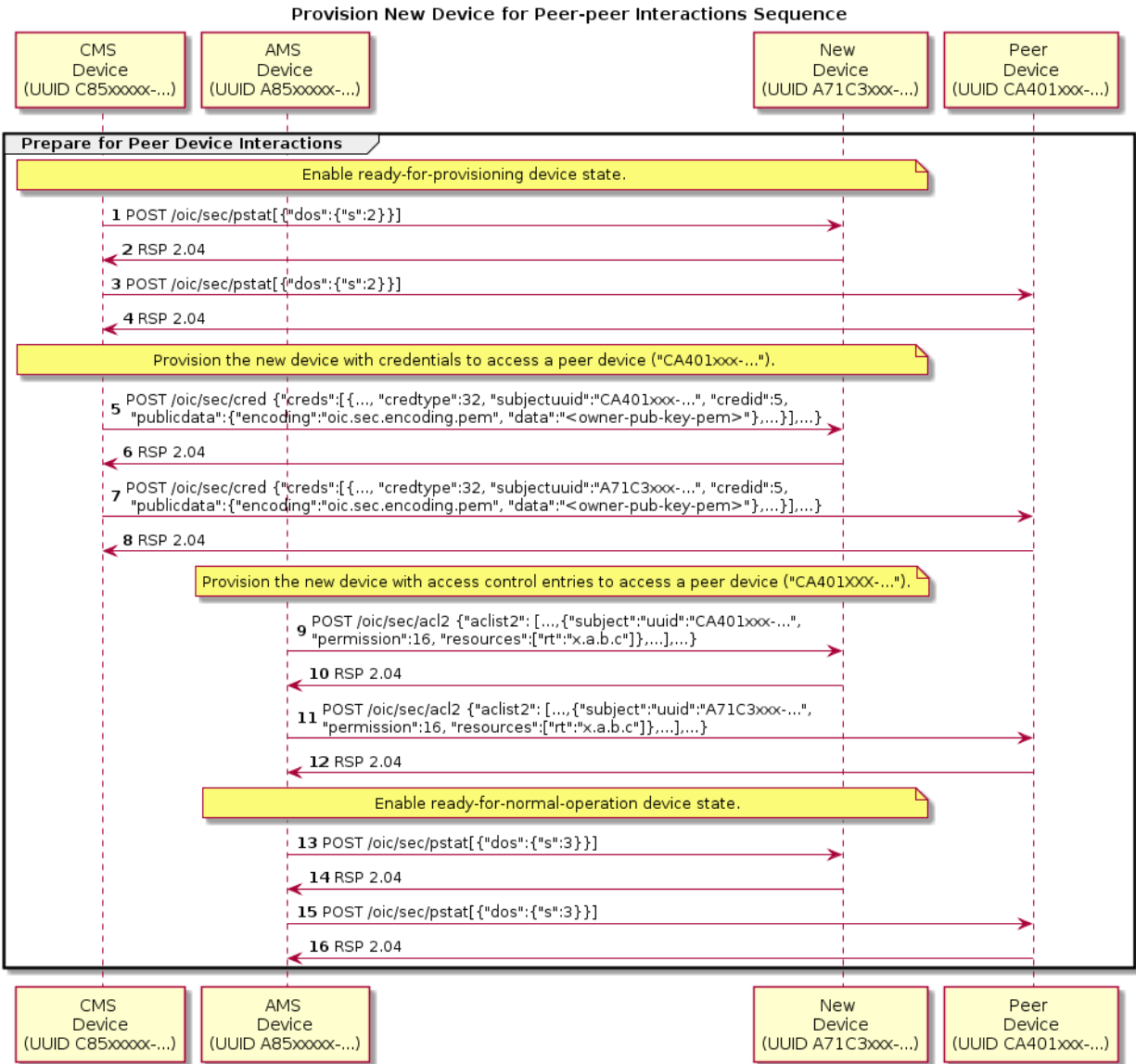
1817

1818

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.



1819

1820

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

1821

1822

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.

1823 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1824 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are
1825 acceptable when transferring ownership of a new Device. Some of the factors to be considered
1826 when determining those requirements are:

- 1827 – The security considerations described for each of the OTMs
- 1828 – The probability that a man-in-the-middle attacker might be present in the environment used to
1829 perform the ownership transfer

1830 For example, the operator of an OBT might require that all of the Devices being onboarded
1831 support either the Random PIN or the Manufacturer Certificate OTM.

1832 When such a local OTM policy exists, the OBT should try to use just the OTMs that are
1833 acceptable according to that policy, regardless of the doxm contents obtained during step 1 from
1834 the sequence diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an
1835 unauthenticated and/or unencrypted connection between the OBT and the Device, the contents of
1836 the response to the GET request might have been tampered by a man-in-the-middle attacker. For
1837 example, the list of OTMs supported by the new Device might have been altered by the attacker.

1838 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new
1839 Device to fail. In such cases, the OBT has no way of determining if the session failed because
1840 the new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle
1841 injected such a failure into the communication between the OBT and the new Device.

1842 The current version of this document leaves the design and user experience related to the OTM
1843 policy as OBT implementation details.

1844 **7.3.10 Security Profile Assignment**

1845 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results
1846 could be accessed from a manufacturer's certificate, OCF web server or other public repository.
1847 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device
1848 is authorized to possess and configures the Device with the subset of evaluated security profiles
1849 best suited for the OCF Security Domain owner's intended segmentation strategy.

1850 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles"
1851 Property of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification
1852 process. The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the
1853 values contained in the "supportedprofiles". The manufacturer default value shall be re-asserted
1854 when the Device transitions to RESET Device State.

1855 The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in
1856 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as
1857 directed by a Security Profile.

1858 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a
1859 subset of the OCF Security Profiles values the Device achieved as part of OCF Conformance
1860 testing. The DOTS may locate conformance results by inspecting manufacturer certificates
1861 supplied with the OCF Device by selecting the "credusage" Property of the "/oic/sec/cred"
1862 Resource having the value of "oic.sec.cred.mfgcert". The DOTS may further locate conformance
1863 results by visiting a well-known OCF web site URI corresponding to the ocCPLAttributes
1864 extension fields (clause 9.4.2.2.7). The DOTS may select a subset of Security Profiles (from
1865 those evaluated by OCF conformance testing) based on a local policy.

1866 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries
1867 to allow DOTS access subsequent to onboarding.

1868 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the
1869 value that most correctly depicts the OCF Security Domain owner's intended Device deployment
1870 strategy.

1871 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID")
1872 to indicate the OCF Security Domain owner's intention to segment the OCF Security Domain
1873 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the
1874 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security
1875 Profiles when issuing role credentials.

1876 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control
1877 entries that include the role designation(s).

1878 **7.4 Provisioning**

1879 **7.4.1 Provisioning Flows**

1880 **7.4.1.1 Provisioning Flows General**

1881 As part of onboarding a new Device a secure channel is formed between the new Device and the
1882 OBT. Subsequent to the Device ownership status being changed to "owned", there is an
1883 opportunity to begin provisioning. The OBT decides how the new Device will be managed going
1884 forward and provisions the support services that should be subsequently used to complete
1885 Device provisioning and on-going Device management.

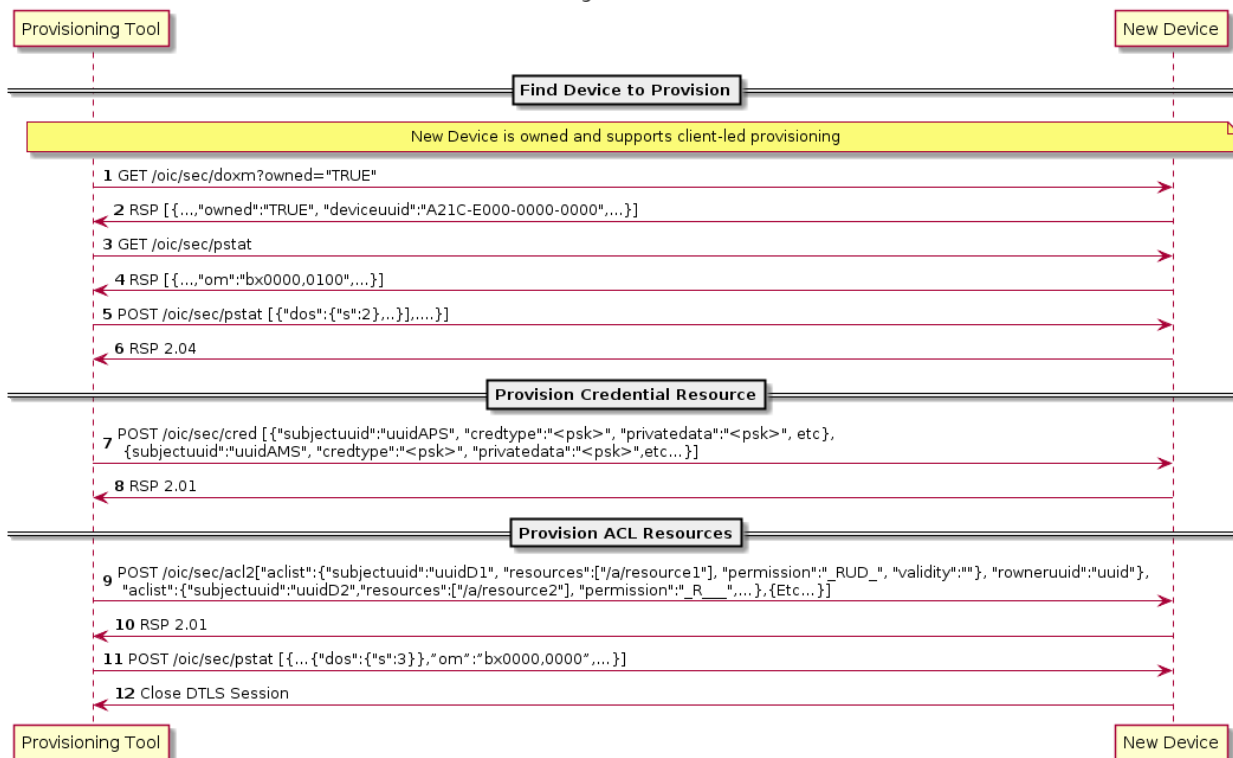
1886 The Device employs a Server-directed or Client-directed provisioning strategy. The
1887 "/oic/sec/pstat" Resource identifies the provisioning strategy and current provisioning status. The
1888 provisioning service should determine which provisioning strategy is most appropriate for the
1889 OCF Security Domain. See 13.8 for additional detail.

1890 **7.4.1.2 Client-directed Provisioning**

1891 Client-directed provisioning relies on a provisioning service that identifies Servers in need of
1892 provisioning then performs all necessary provisioning duties.

1893 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



1894

1895 **Figure 24 – Example of Client-directed provisioning**

1896

1897 **Table 12 – Steps describing Client -directed provisioning**

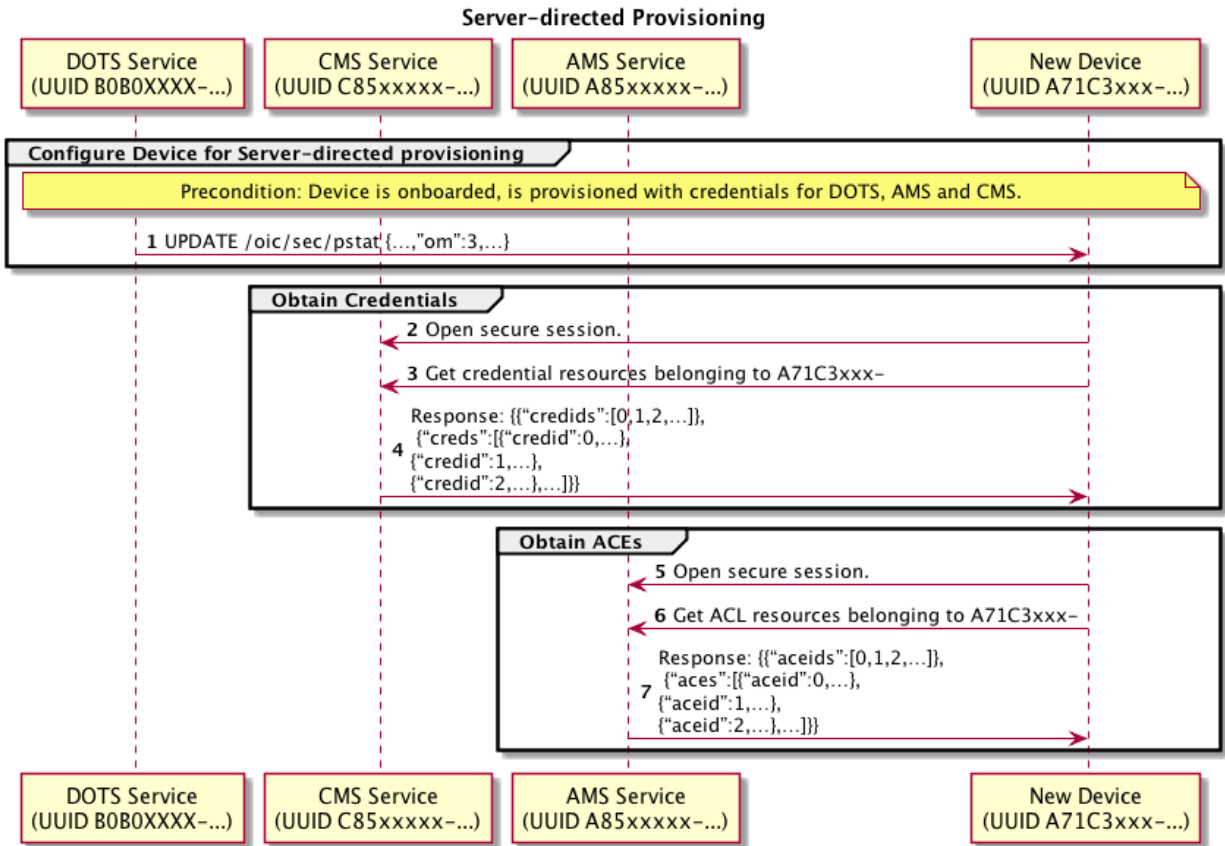
Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The "/oic/sec/doxm" Resource identifies the Device and it's owned status.
3	Provisioning Tool (PT) obtains the new Device's provisioning status found in "/oic/sec/pstat" Resource
4	The "pstat" Resource describes 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.
----	-------------------------------

1898 **7.4.1.3 Server-directed Provisioning**

1899 Server-directed provisioning relies on the Server (i.e. new Device) for directing much of the
 1900 provisioning work. As part of the onboarding process the support services used by the Server to
 1901 seek additional provisioning are provisioned. The new Device uses a self-directed, state-driven
 1902 approach to analyse current provisioning state, and tries to drive toward target state. This
 1903 example assumes a single support service is used to provision the new Device.

1904 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1905 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**
 1906

1907 **Table 13 – Steps for Server-directed provisioning using a single provisioning service**

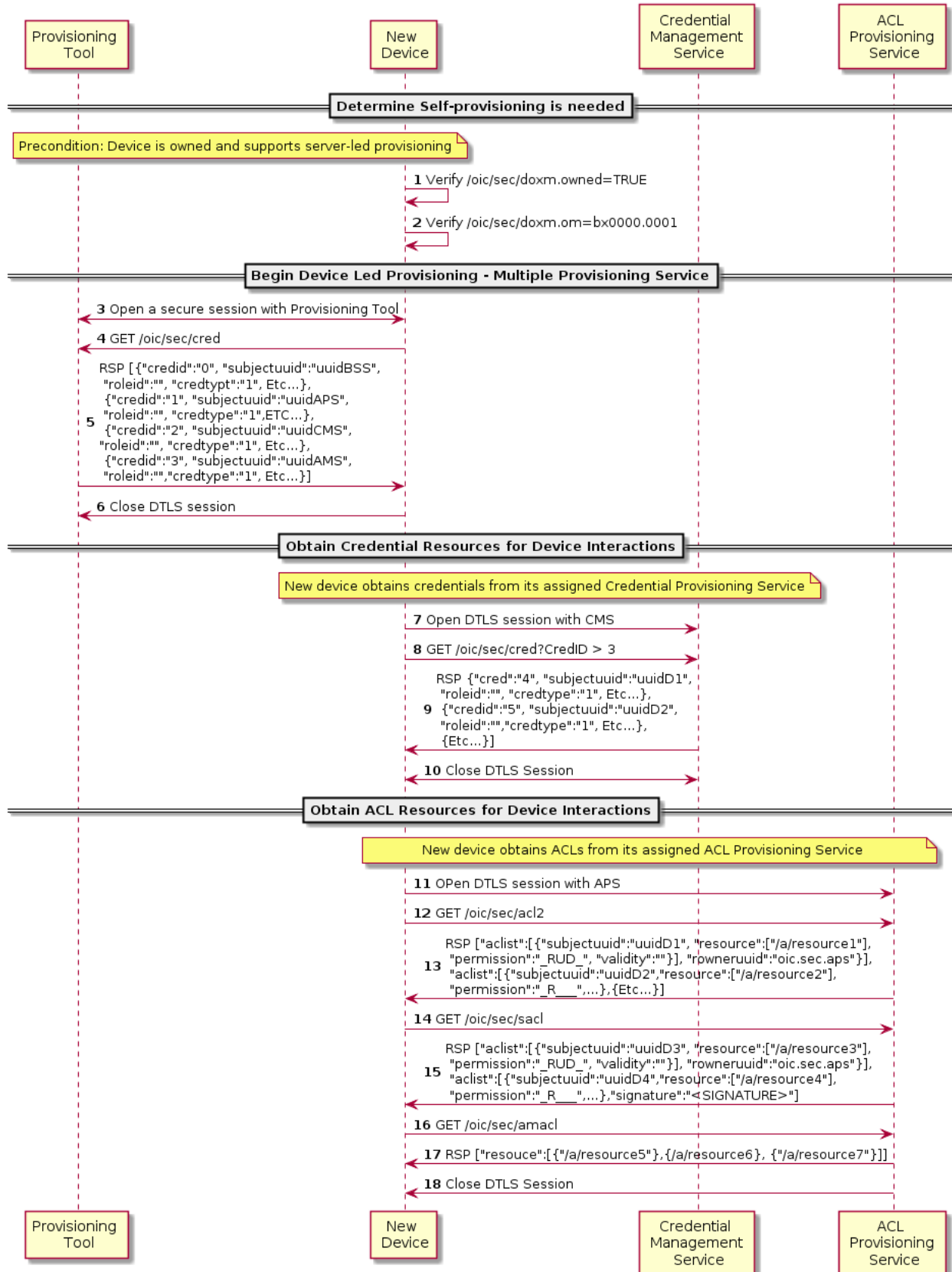
Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a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

1908 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1909 A Server-directed provisioning flow, involving multiple support services distributes the
 1910 provisioning work across multiple support services. Employing multiple support services is an
 1911 effective way to distribute provisioning workload or to deploy specialized support. The example in
 1912 Figure 26 demonstrates using a provisioning tool to configure two support services, a CMS and
 1913 an AMS. Steps for the example are described in Table 14.

OCF Server Led Provisioning with Multiple Service Providers



1915 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1916 **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.

1917 **7.5 Device Provisioning for OCF Cloud**

1918 **7.5.1 Cloud Provisioning General**

1919 The Device that connects to the OCF Cloud shall support the "oic.r.coapcloudconf" Resource on
 1920 Device and following SVRs on the OCF Cloud: "/oic/sec/account", "/oic/sec/session",
 1921 "/oic/sec/tokenrefresh".

1922 The OCF Cloud is expected to use a secure mechanism for associating a Mediator with an OCF
 1923 Cloud User. The choice of mechanism is up to the OCF Cloud. Example, mechanisms include
 1924 HTTP authentication (with username and password) or OAuth 2.0 (using an Authorization Server
 1925 which could be operated by the OCF Cloud provider or a third party). OCF Cloud is expected to
 1926 ensure that the suitable authentication mechanism is used to authenticate the OCF Cloud User.

1927 **7.5.2 Device Provisioning by Mediator**

1928 The Mediator and the Device shall use the secure session to provision the Device to connect with
 1929 the OCF Cloud.

1930 The Mediator obtains an Access Token from the OCF Cloud as described in OCF Cloud
 1931 Specification. This Access Token is then used by the Device for registering with the OCF Cloud

1932 as described in 10.5. The OCF Cloud maintains a map where Access Token and Mediator
 1933 provided Device ID are stored. At the time of Device Registration OCF Cloud validates the
 1934 Access Token and associates the TLS session with corresponding Device ID.

1935 The Mediator provisions the Device, as described in OCF Cloud Specification. The Mediator
 1936 provisions OCF Cloud URI to the "cis" Property of "oic.r.coapcloudconf" Resource, OCF Cloud
 1937 UUID to the "sid" Property of "oic.r.coapcloudconf" Resource and per-device Access Token to the
 1938 "at" Property of "oic.r.coapcloudconf" Resource on Device. Provisioned "at" is to be treated by
 1939 Device as an Access Token with "Bearer" token type as defined in IETF RFC 6750.

1940 For the purposes of access control, the Device shall identify the OCF Cloud using the OCF Cloud
 1941 UUID in the Common Name field of the End-Entity certificate used to authenticate the OCF Cloud.

1942 AMS should configure the ACE2 entries on a Device so that the Mediator(s) is the only Device(s)
 1943 with UPDATE permission for the "oic.r.coapcloudconf" Resource.

1944 The AMS should configure the ACE2 entries on the Device to allow request from the OCF Cloud.
 1945 By request from the Mediator, the AMS removes old ACL2 entries with previous OCF Cloud UUID.
 1946 This request happens before "oic.r.coapcloudconf" is configured by the Mediator for the new OCF
 1947 Cloud. The Mediator also requests AMS to set the OCF Cloud UUID as the "subject" Property for
 1948 the new ACL2 entries. AMS may use "sid" Property of "oic.r.coapcloudconf" Resource as the
 1949 current OCF Cloud UUID. AMS could either provision a wildcard entry for the OCF Cloud or
 1950 provision an entry listing each Resource published on the Device.

1951 If OCF Cloud provides "redirecturi" Value as response during Device Registration, the redirected-
 1952 to OCF Cloud is assumed to have the same OCF Cloud UUID and to use the same trust anchor.
 1953 Otherwise, presented OCF Cloud UUID wouldn't match the provisioned ACL2 entries.

1954 The Mediator should provision the "oic.r.coapcloudconf" Resource with the Properties in Table 15.
 1955 These details once provisioned are used by the Device to perform Device Registration to the
 1956 OCF Cloud. After the initial registration, the Device should use updated values received from the
 1957 OCF Cloud instead. If OCF Cloud User wants the Device to re-register with the OCF Cloud, they
 1958 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.

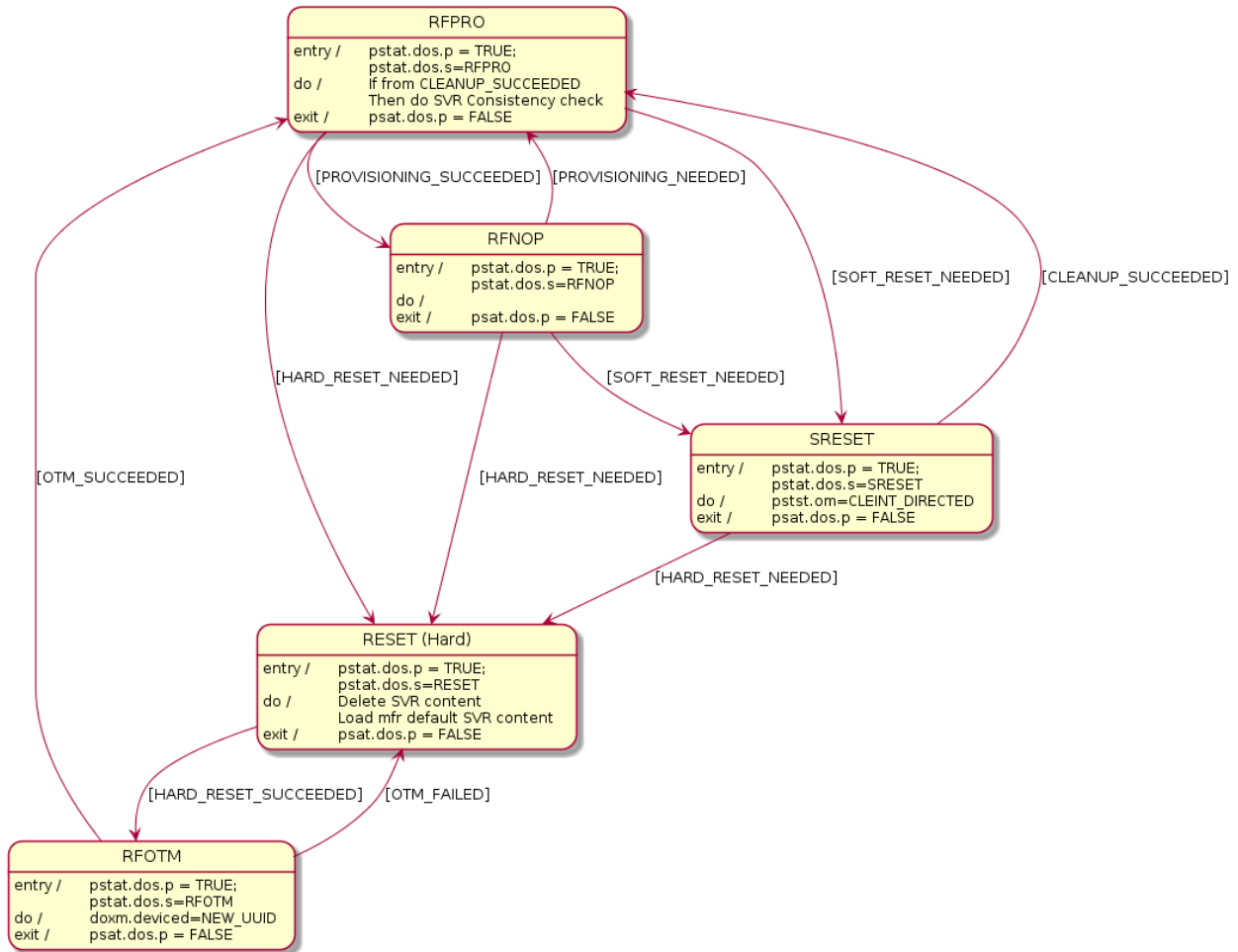
1961 **8 Device Onboarding State Definitions**

1962 **8.1 Device Onboarding General**

1963 As explained in 5.3, the process of onboarding completes after the ownership of the Device has
 1964 been transferred and the Device has been provisioned with relevant configuration/services as

1965 explained in 5.4. The Figure 27 shows the various states a Device can be in during the Device
 1966 lifecycle.

1967 The "/pstat.dos.s" Property is RW by the "/oic/sec/pstat" resource owner (e.g. "doxs" service) so
 1968 that the resource owner can remotely update the Device state. When the Device is in RFNOP or
 1969 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the
 1970 Device state is SRESET the Device OC may be the only indication of authorization to access the
 1971 Device. The Device owner may perform low-level consistency checks and re-provisioning to get
 1972 the Device suitable for a transition to RFPRO.



1973 **Figure 27 – Device state model**

1974
 1975 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the
 1976 "Ready for Normal Operation" state where it has all it needs in order to start interoperating with
 1977 other Devices. Clause 8.5 specifies the minimum mandatory configuration that a Device shall
 1978 hold in order to be considered as "Ready for Normal Operation".

1979 In the event of power loss or Device failure, the Device should remain in the same state that it
 1980 was in prior to the power loss / failure

1981 If a Device or resource owner OBSERVEs "/pstat.dos.s", then transitions to SRESET will give
 1982 early warning notification of Devices that may require SVR consistency checking.

1983 In order for onboarding to function, the Device shall have the following Resources installed:

1984 1) "/oic/sec/doxm" Resource

1985 2) "/oic/sec/pstat" Resource

1986 3) "/oic/sec/cred" Resource

1987 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5
1988 and 8.6.

1989 **8.2 Device Onboarding-Reset State Definition**

1990 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset
1991 also defines a state where the Device asset is ready to be transferred to another party.

1992 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces
1993 Platform reset. All Devices hosted on the same Platform transition their Device states to RESET
1994 when the Platform reset is asserted.

1995 The following Resources and their specific properties shall have the value as specified:

1996 1) The "owned" Property of the "/oic/sec/doxm" Resource shall transition to FALSE.

1997 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.

1998 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
1999 implemented.

2000 4) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
2001 default value.

2002 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
2003 default value, if this Property is implemented.

2004 6) The "sct" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
2005 default value.

2006 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
2007 default value.

2008 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.

2009 9) The "dos" Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal
2010 "RESET" state and dos.p shall equal "FALSE".

2011 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the
2012 manufacturer default value.

2013 11) The "sm" (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set
2014 to the manufacturer default value.

2015 12) The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and
2016 "/oic/sec/cred" Resources shall be nil UUID.

2017 13) The "supportedprofiles" Property of the "/oic/sec/sp" Resource shall be set to the
2018 manufacturer default value.

2019 14) The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer
2020 default value.

2021 **8.3 Device Ready-for-OTM State Definition**

2022 The following Resources and their specific properties shall have the value as specified when the
2023 Device enters ready for ownership transfer:

2024 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to
2025 TRUE.

- 2026 2) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
2027 implemented.
- 2028 3) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2029 4) The "deviceid" Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is
2030 implemented. The value of the di Property in "/oic/d" is undefined.
- 2031 5) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
2032 default value.
- 2033 6) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2034 7) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFOTM"
2035 state and dos.p shall equal "FALSE".
- 2036 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

2037 **8.4 Device Ready-for-Provisioning State Definition**

2038 The following Resources and their specific properties shall have the value as specified when the
2039 Device enters ready for provisioning:

- 2040 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2041 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2042 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be
2043 set to the value that was determined during RFOTM processing. Also the value of the "di"
2044 Property in "/oic/d" Resource shall be the same as the "deviceid" Property in the
2045 "/oic/sec/doxm" Resource.
- 2046 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual
2047 OTM used during ownership transfer.
- 2048 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2049 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFPRO"
2050 state and "dos.p" shall equal "FALSE".
- 2051 7) The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner
2052 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
2053 "rowneruuid" may result in an orphan Resource.
- 2054 8) The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by
2055 "rowneruuid" and "devowneruuid" Properties.

2056 **8.5 Device Ready-for-Normal-Operation State Definition**

2057 The following Resources and their specific properties shall have the value as specified when the
2058 Device enters ready for normal operation:

- 2059 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2060 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2061 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be
2062 set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d"
2063 shall be the same as the deviceuuid.
- 2064 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual
2065 OTM used during ownership transfer.
- 2066 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once
2067 transition to RFNOP is otherwise complete.
- 2068 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFNOP"
2069 state and dos.p shall equal "FALSE".

2070 7) The "rowneruuid" Property of every installed Resource shall be set to a valid resource owner
2071 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
2072 "rowneruuid" results in an orphan Resource.

2073 8) The "/oic/sec/cred" Resource shall contain credentials for each service referenced by
2074 "rowneruuid" and "devowneruuid" Properties.

2075 **8.6 Device Soft Reset State Definition**

2076 The soft reset state is defined (e.g. "/pstat.dos.s" = SRESET) where entrance into this state
2077 means the Device is not operational but remains owned by the current owner. The Device may
2078 exit SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during
2079 original onboarding (but should not require use of an OTM /doxm.oxms).

2080 The DOTS should perform a consistency check of the SVR and if necessary, re-provision them
2081 sufficiently to allow the Device to transition to RFPRO.

2082 Figure 28 depicts OBT Sanity Check Sequence in SRESET.

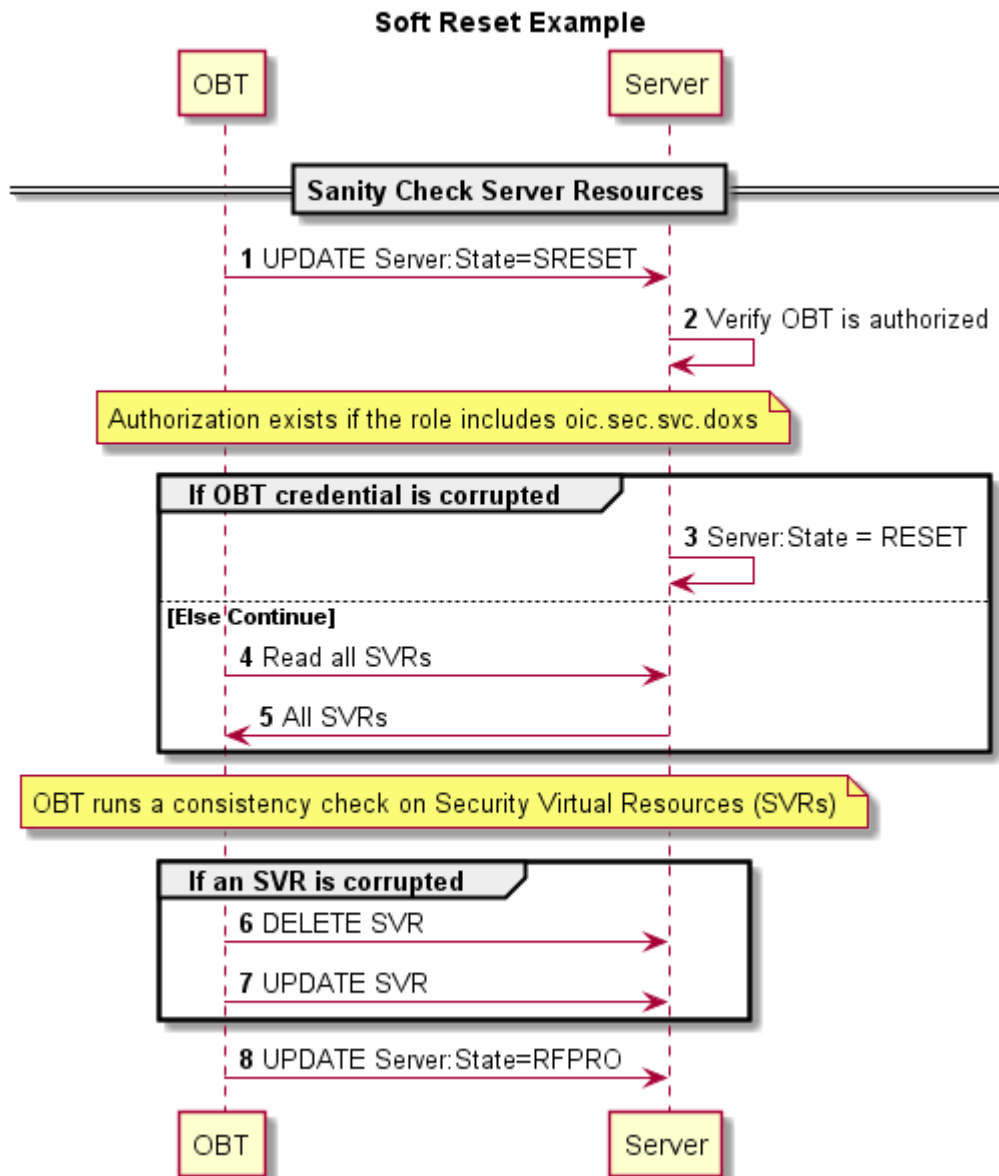


Figure 28 – OBT Sanity Check Sequence in SRESET

2083
2084

2085 The DOTS should perform a sanity check of SVRs before final transition to RFPRO Device state.
2086 If the DOTS credential cannot be found or is determined to be corrupted, the Device state
2087 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to
2088 validate the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS
2089 Devices.

2090 When in SRESET, the following Resources and their specific Properties shall have the values as
2091 specified.

- 2092 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
2093 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
2094 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be non-null, if this Property is
2095 implemented.

- 2096 4) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2097 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2098 6) The "sct" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 2099 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall retains its value.
- 2100 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2101 9) The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- 2102 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-
- 2103 directed mode".
- 2104 11) The "sm" (supported operational modes) Property of "/oic/sec/pstat" Resource may be
- 2105 updated by the Device owner (aka DOTS).
- 2106 12) The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2",
- 2107 "/oic/sec/amacl", "/oic/sec/sacl", and "/oic/sec/cred" Resources may be reset by the Device
- 2108 owner (aka DOTS) and re-provisioned.
- 2109

2110 **9 Security Credential Management**

2111 **9.1 Preamble**

2112 This clause provides an overview of the credential types in OCF, along with details of credential
2113 use, provisioning and ongoing management.

2114 **9.2 Credential Lifecycle**

2115 **9.2.1 Credential Lifecycle General**

2116 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)
2117 issuance and (5) revocation.

2118 **9.2.2 Creation**

2119 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is
2120 authorized by matching the rowneruuid Property of the "/oic/sec/cred" resource to the DeviceID of
2121 the credential the CMS used to establish the secure connection.

2122 Credential Resources created using a CMS may involve specialized credential issuance protocols
2123 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate
2124 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of
2125 a provisioning action by a DOTS, CMS or AMS.

2126 **9.2.3 Deletion**

2127 The CMS should delete known compromised credential Resources. The Device (e.g. the Device
2128 where the credential Resource is hosted) should delete credential Resources that have expired.

2129 An expired credential Resource may be deleted to manage memory and storage space.

2130 Deletion in OCF key management is equivalent to credential suspension.

2131 **9.2.4 Refresh**

2132 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2133 The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may
2134 be applied when a credential is about to expire or is about to exceed a maximum threshold for
2135 bytes encrypted.

2136 A credential refresh method specifies the options available when performing key refresh. The
2137 Period Property informs when the credential should expire. The Device may proactively obtain a
2138 new credential using a credential refresh method using current unexpired credentials to refresh
2139 the existing credential. If the Device does not have an internal time source, the current time
2140 should be obtained from a CMS at regular intervals.

2141 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the
2142 CMS credentials to the Device. If the onboarding established credentials are allowed to expire
2143 the DOTS shall re-onboard the Device to re-apply device owner transfer steps.

2144 All Devices shall support at least one credential refresh method.

2145 **9.2.5 Revocation**

2146 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where
2147 the revocation method involves provisioning of a revocation object that identifies a credential that
2148 is to be revoked prior to its normal expiration period, a credential Resource is created containing
2149 the revocation information that supersedes the originally issued credential. The revocation object

2150 expiration should match that of the revoked credential so that the revocation object is cleaned up
2151 upon expiry.

2152 It is conceptually reasonable to consider revocation applying to a credential or to a Device.
2153 Device revocation asserts all credentials associated with the revoked Device should be
2154 considered for revocation. Device revocation is necessary when a Device is lost, stolen or
2155 compromised. Deletion of credentials on a revoked Device might not be possible or reliable.

2156 **9.3 Credential Types**

2157 **9.3.1 Preamble**

2158 The "/oic/sec/cred" Resource maintains a credential type Property that supports several
2159 cryptographic keys and other information used for authentication and data protection. The
2160 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric
2161 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.
2162 PIN/password).

2163 **9.3.2 Pair-wise Symmetric Key Credentials**

2164 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The
2165 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2166 Pair-wise keys could be established through ad-hoc key agreement protocols.

2167 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2168 The PublicData Property may contain a token encrypted to the peer Device containing the pair-
2169 wise key.

2170 The OptionalData Property may contain revocation status.

2171 The Device implementer should apply hardened key storage techniques that ensure the
2172 PrivateData remains private.

2173 The Device implementer should apply appropriate integrity, confidentiality and access protection
2174 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2175 unauthorized modifications.

2176 **9.3.3 Group Symmetric Key Credentials**

2177 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are
2178 used for efficient sharing of data among group participants.

2179 Group keys do not provide authentication of Devices but only establish membership in a group.

2180 The CMS shall provision group symmetric key credentials to the group members. The CMS
2181 maintains the group memberships.

2182 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2183 The PublicData Property may contain the group name.

2184 The OptionalData Property may contain revocation status.

2185 The Device implementer should apply hardened key storage techniques that ensure the
2186 PrivateData remains private.

2187 The Device implementer should apply appropriate integrity, confidentiality and access protection
2188 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2189 unauthorized modifications.

2190 **9.3.4 Asymmetric Authentication Key Credentials**

2191 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2192 Asymmetric authentication key credentials contain either a public and private key pair or only a
2193 public key. The private key is used to sign Device authentication challenges. The public key is
2194 used to verify a device authentication challenge-response.

2195 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2196 The PublicData Property contains the public key.

2197 The OptionalData Property may contain revocation status.

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

2200 Devices should generate asymmetric authentication key pairs internally to ensure the private key
2201 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material
2202 between Devices.

2203 The Device implementer should apply appropriate integrity, confidentiality and access protection
2204 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2205 unauthorized modifications.

2206 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2207 Devices should employ industry-standard high-assurance techniques when allowing off-device
2208 key pair creation and provisioning. Use of such key pairs should be minimized, particularly if the
2209 key pair is immutable and cannot be changed or replaced after provisioning.

2210 When used as part of onboarding, these key pairs can be used to prove the Device possesses
2211 the manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept
2212 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the
2213 Device, and then provisions new OCF Security Domain credentials for use.

2214 **9.3.5 Asymmetric Key Encryption Key Credentials**

2215 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when
2216 distributing or storing the key.

2217 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2218 The PublicData Property contains the public key.

2219 The OptionalData Property may contain revocation status.

2220 The Device implementer should apply hardened key storage techniques that ensure the
2221 PrivateData remains private.

2222 The Device implementer should apply appropriate integrity, confidentiality and access protection
2223 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2224 unauthorized modifications.

2225 **9.3.6 Certificate Credentials**

2226 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a
2227 CMS or an external certificate authority (CA).

2228 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2229 The issued certificate is stored with the asymmetric key credential Resource.

2230 Other objects useful in managing certificate lifecycle such as certificate revocation status are
2231 associated with the credential Resource.

2232 Either an asymmetric key credential Resource or a self-signed certificate credential is used to
2233 terminate a path validation.

2234 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2235 The PublicData Property contains the issued certificate.

2236 The OptionalData Property may contain revocation status.

2237 The Device implementer should apply hardened key storage techniques that ensure the
2238 PrivateData remains private.

2239 The Device implementer should apply appropriate integrity, confidentiality and access protection
2240 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2241 unauthorized modifications.

2242 **9.3.7 Password Credentials**

2243 Shared secret credentials are used to maintain a PIN or password that authorizes Device access
2244 to a foreign system or Device that doesn't support any other OCF credential types.

2245 The PrivateData Property in the "/oic/sec/cred" Resource contains the PIN, password and other
2246 values useful for changing and verifying the password.

2247 The PublicData Property may contain the user or account name if applicable.

2248 The OptionalData Property may contain revocation status.

2249 The Device implementer should apply hardened key storage techniques that ensure the
2250 PrivateData remains private.

2251 The Device implementer should apply appropriate integrity, confidentiality and access protection
2252 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
2253 unauthorized modifications.

2254 **9.4 Certificate Based Key Management**

2255 **9.4.1 Overview**

2256 To achieve authentication and transport security during communications in OCF Security Domain,
2257 certificates containing public keys of communicating parties and private keys can be used.

2258 The certificate and private key may be issued by a local or remote certificate authority (CA). For
2259 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of
2260 identity. In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to
2261 validate proof of identity and validity.

2262 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,
2263 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in
2264 X.509 are not supported so that the format intends to meet the constrained Device's requirement.

2265 As for the certificate and CRL management in the Server, the process of storing, retrieving and
2266 parsing Resources of the certificates and CRL will be performed at the security resource
2267 manager layer; the relevant interfaces may be exposed to the upper layer.

2268 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of
2269 certificates and CRL will be stored and managed in SVR database.

2270 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a
2271 CMS to a Device when it is newly onboarded. The issuing CMS should process certificate
2272 revocations for certificates it issues. If a certificate private key is compromised, the CMS should
2273 revoke the certificate. If CRLs are used by a Device, the CMS should regularly (for example;
2274 every 3 months) update the "/oic/sec/crl" resource for the Devices it manages.

2275 **9.4.2 X.509 Digital Certificate Profiles**

2276 **9.4.2.1 Digital Certificate Profile General**

2277 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in
2278 IETF RFC 5280.

2279 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications
2280 for those communities wishing to make use of X.509 technology. The X.509 v3 certificate format
2281 is described in detail, with additional information regarding the format and semantics of OCF
2282 specific extension(s). The supported standard certificate extensions are also listed.

2283 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this
2284 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are
2285 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,
2286 compliant entities shall ignore their contents.

2287 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as
2288 defined in ISO/IEC 8825-1 to encode certificates.

2289 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public
2290 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs
2291 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and
2292 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2293 The following clauses specify the supported standard and custom extensions for the OCF
2294 certificates profile.

2295 **9.4.2.2 Certificate Profile and Fields**

2296 **9.4.2.2.1 Root CA Certificate Profile**

2297 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2298 **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)

2299 Table 17 describes X.509 v3 extensions required for Root CA Certificates.

2300 **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)

2301 **9.4.2.2.2 Intermediate CA Certificate Profile**

2302 Table 18 describes X.509 v1 fields required for Intermediate CA Certificates.

2303 **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)

2304 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

2306 **9.4.2.2.3 End-Entity Black Certificate Profile**

2307 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security
2308 profile.

2309 **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)

2310 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"> • serverAuthentication - 1.3.6.1.5.5.7.3.1 • clientAuthentication - 1.3.6.1.5.5.7.3.2 Exactly ONE of the following OIDs SHALL be present: <ul style="list-style-type: none"> • Identity certificate - 1.3.6.1.4.1.44924.1.6 • Role certificate - 1.3.6.1.4.1.44924.1.7 End-Entity certificates SHALL NOT contain the anyExtendedKeyUsage OID (2.5.29.37.0)
subjectAlternativeName	REQUIRED UNDER CERTAIN CONDITIONS	Non-critical	The subjectAltName extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. When the extendedKeyUsage (EKU) extension contains the Identity Certificate OID (1.3.6.1.4.1.44924.1.6), the subjectAltName extension SHOULD NOT be present. If the EKU extension contains the Role Certificate

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

2312 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2313 The OCF Compliance Extension defines required parameters to correctly identify the type of
2314 Device, its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2315 The extension carries an "ocfVersion" field which provides the specific base version of the OCF
2316 documents the device implements. The "ocfVersion" field shall contain a sequence of three
2317 integers ("major", "minor", and "build"). For example, if an entity is certified to be compliant with

2318 OCF specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be
2319 set to "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote
2320 compliance to a specified base version of the OCF documents.

2321 The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more
2322 supported Security Profiles associated with the certificate in string form (UTF-8). All Security
2323 Profiles associated with the certificate should be identified by this field.

2324 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".
2325 The fields carry human-readable descriptions of the Device's name and manufacturer,
2326 respectively.

2327 The ASN.1 definition of the OCFCCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined
2328 as follows:

```
2329 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2330                               private(4) enterprise(1) OCF(51414) }
2331
2332   id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2333
2334   id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2335
2336 ocfVersion ::= SEQUENCE {
2337     major  INTEGER,
2338           --Major version number
2339     minor  INTEGER,
2340           --Minor version number
2341     build  INTEGER,
2342           --Build/Micro version number
2343 }
2344
2345 ocfCompliance ::= SEQUENCE {
2346     version          ocfVersion,
2347                   --Device/OCF version
2348     securityProfile  SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2349                   --Sequence of OCF Security Profile OID strings
2350                   --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2351     deviceName       UTF8String,
2352                   --Name of the device
2353     deviceManufacturer UTF8String,
2354                   --Human-Readable Manufacturer
2355                   --of the device
2356 }
```

2357 **9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension**

2358 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for
2359 devices to signal to the network the access and network functionality they require to properly
2360 function. Access controls can be more easily achieved and deployed at scale when the MUD
2361 extension is used. The current draft of the MUD v3 extension at this time of writing is:

2362 <https://tools.ietf.org/html/rfc8520#section-11>

2363 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2364 MUDURLExtnModule-2016 {   iso(1) identified-organization(3) dod(6)
2365                           internet(1) security(5) mechanisms(5) pkix(7)
2366                           id-mod(0) id-mod-mudURLExtn2016(88) }
2367
2368   DEFINITIONS IMPLICIT TAGS ::= BEGIN
2369   -- EXPORTS ALL --
2370   IMPORTS
```



```

2371     EXTENSION
2372     FROM PKIX-CommonTypes-2009
2373         { iso(1) identified-organization(3) dod(6) internet(1)
2374           security(5) mechanisms(5) pkix(7) id-mod(0)
2375           id-mod-pkixCommon-02(57) }
2376     id-pe
2377     FROM PKIX1Explicit-2009
2378         { iso(1) identified-organization(3) dod(6) internet(1)
2379           security(5) mechanisms(5) pkix(7) id-mod(0)
2380           id-mod-pkix1-explicit-02(51) } ;
2381     MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2382     ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2383                               IDENTIFIED BY id-pe-mud-url }
2384
2385     id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2386
2387     MUDURLSyntax ::= IA5String
2388
2389     END

```

2390 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2391 The OCF Security Claims Extension defines a list of OIDs representing security claims that the
2392 manufacturer/integrator is making as to the security posture of the device above those required
2393 by the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2394 The purpose of this extension is to allow for programmatic evaluation of assertions made about
2395 security to enable some platforms/policies/administrators to better understand what is being
2396 onboarded or challenged.

2397 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is
2398 defined as follows:

```

2399 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2400                               private(4) enterprise(1) OCF(51414) }
2401
2402     id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2403
2404     id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2405
2406     claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2407     --Device claims that the boot process follows a procedure trusted
2408     --by the firmware and the BIOS
2409
2410     claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2411     --Device claims that credentials are stored in a specialized hardware
2412     --protection environment such as a Trusted Platform Module (TPM) or
2413     --similar mechanism.
2414
2415     ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2416
2417     ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

2418 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2419 The OCF Certified Product List Extension defines required parameters to utilize the OCF
2420 Compliance Management System Certified Product List (OCMS-CPL). This clause is only
2421 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to
2422 verify the compliance level of a device.

2423 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model
2424 and Version.

2425 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique
2426 PEN established in the IANA PEN list located at: [https://www.iana.org/enterprise-](https://www.iana.org/assignments/enterprise-)
2427 numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as
2428 reported during OCF Certification.

2429 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field
2430 found in end-products shall be the same information as reported during OCF Certification.

2431 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found
2432 in end-products shall be the same information as reported during OCF Certification.

2433 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is
2434 defined as follows:

```
2435 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2436                               private(4) enterprise(1) OCF(51414) }
2437
2438 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2439
2440     id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2441
2442     cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2443     cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2444     cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2445
2446
2447     ocfCPLAttributes ::= SEQUENCE {
2448         cpl-at-IANAPen      UTF8String,
2449         --Manufacturer's registered IANA Private Enterprise Number
2450         cpl-at-model       UTF8String,
2451         --Device OCF Security Profile
2452         cpl-at-version     UTF8String
2453         --Name of the device
2454     }
```

2455 9.4.2.3 Supported Certificate Extensions

2456 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the
2457 clause number from that RFC to include it by reference. Each extension is summarized here, and
2458 any modifications to the RFC definition are listed. Devices MUST implement and understand the
2459 extensions listed here; other extensions from the RFC are not included in this document and
2460 therefore are not required. 10.4 describes what Devices must implement when validating
2461 certificate chains, including processing of extensions, and actions to take when certain
2462 extensions are absent.

2463 – Authority Key Identifier (4.2.1.1)

2464 The Authority Key Identifier (AKI) extension provides a means of identifying the public key
2465 corresponding to the private key used to sign a certificate. This document makes the following
2466 modifications to the referenced definition of this extension:

2467 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier
2468 sequence are not permitted; only keyIdentifier is allowed. This results in the following
2469 grammar definition:

```
2470 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2471
2472 AuthorityKeyIdentifier ::= SEQUENCE {
2473     keyIdentifier          [0] KeyIdentifier
2474 }
2475 KeyIdentifier ::= OCTET STRING
```

2476 – Subject Key Identifier (4.2.1.2)

2477 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that
2478 contain a particular public key.

2479 This document makes the following modification to the referenced definition of this extension:

2480 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's
2481 SubjectPublicKeyInfo field or a method that generates unique values. This document
2482 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey
2483 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains
2484 must not assume any particular method of computing key identifiers, however, and must only
2485 base matching AKI's and SKI's in certification path constructions on key identifiers seen in
2486 certificates.

2487 – Subject Alternative Name

2488 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role
2489 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and
2490 interpreted as described below. When no EKU is present, or has another value, the
2491 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to
2492 encode one or more Role ID values in role certificates, binding the roles to the subject public
2493 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2494 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2495
2496 SubjectAltName ::= GeneralNames
2497
2498 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2499
2500 GeneralName ::= CHOICE {
2501     otherName                [0]     OtherName,
2502     rfc5322Name              [1]     IA5String,
2503     dNSName                  [2]     IA5String,
2504     x400Address              [3]     ORAddress,
2505     directoryName            [4]     Name,
2506     ediPartyName             [5]     EDIPartyName,
2507     uniformResourceIdentifier [6]     IA5String,
2508     iPAddress                [7]     OCTET STRING,
2509     registeredID             [8]     OBJECT IDENTIFIER }
2510
2511     EDIPartyName ::= SEQUENCE {
2512         nameAssigner          [0]     DirectoryString OPTIONAL,
2513         partyName             [1]     DirectoryString }
2514
```

2515 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a
2516 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name
2517 shall contain exactly one CN (Common Name) component, and zero or one OU
2518 (Organizational Unit) components. The OU component, if present, shall specify the authority
2519 that defined the semantics of the role. If the OU component is absent, the certificate issuer
2520 has defined the role. The CN component shall encode the role ID. Other GeneralName types
2521 in the SEQUENCE may be present, but shall not be interpreted as roles. Therefore, if the
2522 certificate issuer includes non-role names in the subjectAltName extension, the extension
2523 should not be marked critical.

2524 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted
2525 character set [0-9a-z-A-z '()+,.-/:=?].

2526 – Key Usage (4.2.1.3)

2527 The key usage extension defines the purpose (e.g., encipherment, signature, certificate
2528 signing) of the key contained in the certificate. The usage restriction might be employed when
2529 a key that could be used for more than one operation is to be restricted.

2530 This document does not modify the referenced definition of this extension.

2531 – Basic Constraints (4.2.1.9)

2532 The basic constraints extension identifies whether the subject of the certificate is a CA and
2533 the maximum depth of valid certification paths that include this certificate. Without this
2534 extension, a certificate cannot be an issuer of other certificates.

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

2536 – Extended Key Usage (4.2.1.12)

2537

2538 Extended Key Usage describes allowed purposes for which the certified public key may can
2539 be used. When a Device receives a certificate, it determines the purpose based on the
2540 context of the interaction in which the certificate is presented, and verifies the certificate can
2541 be used for that purpose.

2542 This document makes the following modifications to the referenced definition of this extension:
2543 CAs SHOULD mark this extension as critical.

2544 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).
2545

2546 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2547 – Identity certificate 1.3.6.1.4.1.44924.1.6
2548 – Role certificate 1.3.6.1.4.1.44924.1.7

2549 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2550 See 9.4.3.5 for details.

2551 **9.4.2.5 Encoding of Certificate**

2552 See 9.4.2 for details.

2553 **9.4.3 Certificate Revocation List (CRL) Profile**

2554 **9.4.3.1 CRL General**

2555 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use
2556 within OCF applications for those communities wishing to support revocation features in their
2557 PKIs.

2558 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in
2559 IETF RFC 5280 – Clause 5.1

2560 **9.4.3.2 CRL Profile and Fields**

2561 This clause intentionally left empty.

2562 **9.4.3.3 Encoding of CRL**

2563 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]
2564 should be used to encode CRL.

2565 **9.4.3.4 CRLs Supported Standard Extensions**

2566 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55]
2567 provide methods for associating additional attributes with CRLs. The following list of X.509
2568 extensions should be supported in this certificate profile:

2569 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides
2570 a means of identifying the public key corresponding to the private key used to sign a CRL.
2571 Conforming CRL issuers should use the key identifier method, and shall include this extension
2572 in all CRLs issued

2573 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that
2574 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer
2575 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for
2576 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]
2577 [X9.55]. Although this document does not provide any recommendation about the use of specific
2578 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not
2579 marked critical.

2580 **9.4.3.5 Encryption Ciphers and TLS support**

2581 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support
2582 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite as defined in IETF RFC 7251 and
2583 may support additional ciphers as defined in the TLS v1.2 specifications.

2584 **9.4.4 Resource Model**

2585 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated
2586 with a separate crl Resource that is defined for maintaining the revocation list.

2587 The cred Resource contains the certificate information pertaining to the Device. The PublicData
2588 Property holds the device certificate and CA certificate chain. PrivateData Property holds the
2589 Device private key paired to the certificate. (See 13.3 for additional detail regarding the
2590 "/oic/sec/cred" Resource).

2591 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained
2592 through the CMS. The Device must consider revoked certificates as part of certificate path
2593 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain
2594 a full list, the Device must query the CMS for current revocation status. (See 13.4 for additional
2595 detail regarding the "/oic/sec/crl" Resource).

2596 **9.4.5 Certificate Provisioning**

2597 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have
2598 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)
2599 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate
2600 shall have the format described in 9.4.2.

2601 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,
2602 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer
2603 them to the Device including its CA certificate chain. Optionally, the CMS may also transfer one
2604 or more role certificates, which shall have the format described in clause 9.4.2. . The
2605 subjectPublicKey of each role certificate shall match the subjectPublicKey in the Device
2606 certificate.

2607 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in
2608 IETF RFC 2986, and is included here by reference.

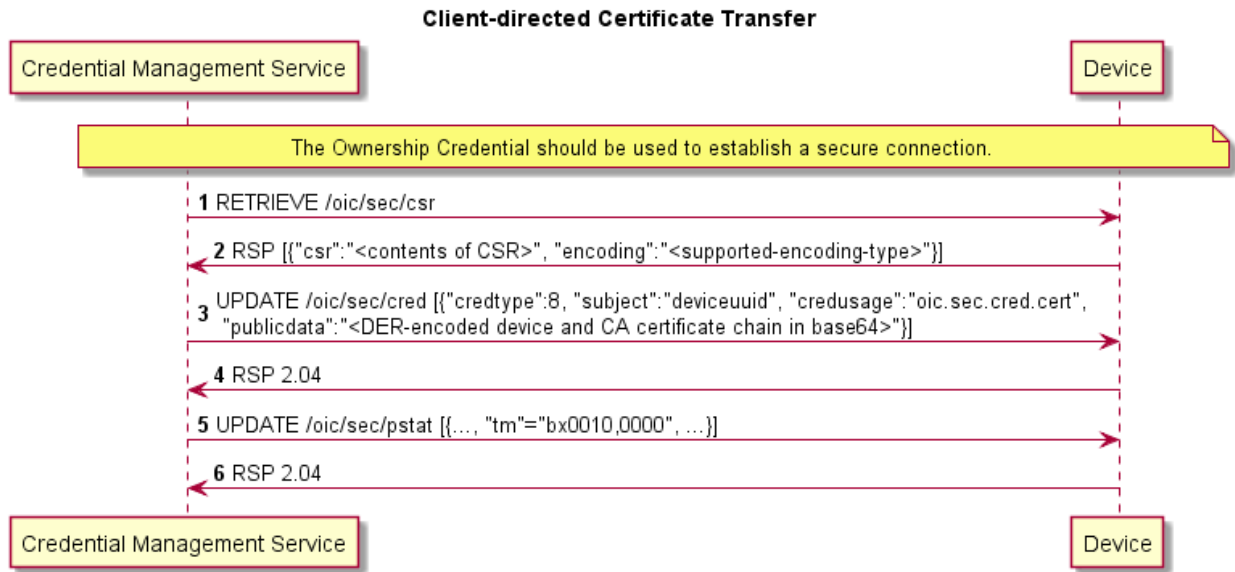
2609 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2610 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device
2611 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.
2612 The Device determines the public key to present; this may be an already-provisioned key it
2613 has selected for use with authentication, or if none is present, it may generate a new key pair
2614 internally and provide the public part. The key pair shall be compatible with the allowed
2615 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF
2616 authentication.

2617 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on
2618 its own, then it is not capable of using certificates. The Device shall advertise this fact both by

2619 setting the 0x8 bit position in the sct Property of "/oic/sec/doxm" to 0, and return an error that
 2620 the "/oic/sec/csr" resource does not exist.

2621 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using
 2622 the same credid, to maintain the association with the private key. The credential type
 2623 ("oic.sec.cred") used to transfer certificates in Figure 29 is also used to transfer role
 2624 certificates, by including multiple credentials in the POST from CMS to Device. Identity
 2625 certificates shall be stored with the credusage Property set to "oic.sec.cred.cert" and role
 2626 certificates shall be stored with the credusage Property set to "oic.sec.cred.rolecert".



2627

2628 **Figure 29 – Client-directed Certificate Transfer**

2629 **9.4.6 CRL Provisioning**

2630 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the
 2631 function to register revocation certificates, to sign CRL and to transfer it to Devices.

2632 The CMS sends the CRL to the Device.

2633 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2634 – change of issuer name
- 2635 – change of association between Devices and CA
- 2636 – certificate compromise
- 2637 – suspected compromise of the corresponding private key

2638 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In
 2639 some special cases, Devices may request CRL to a given CMS.

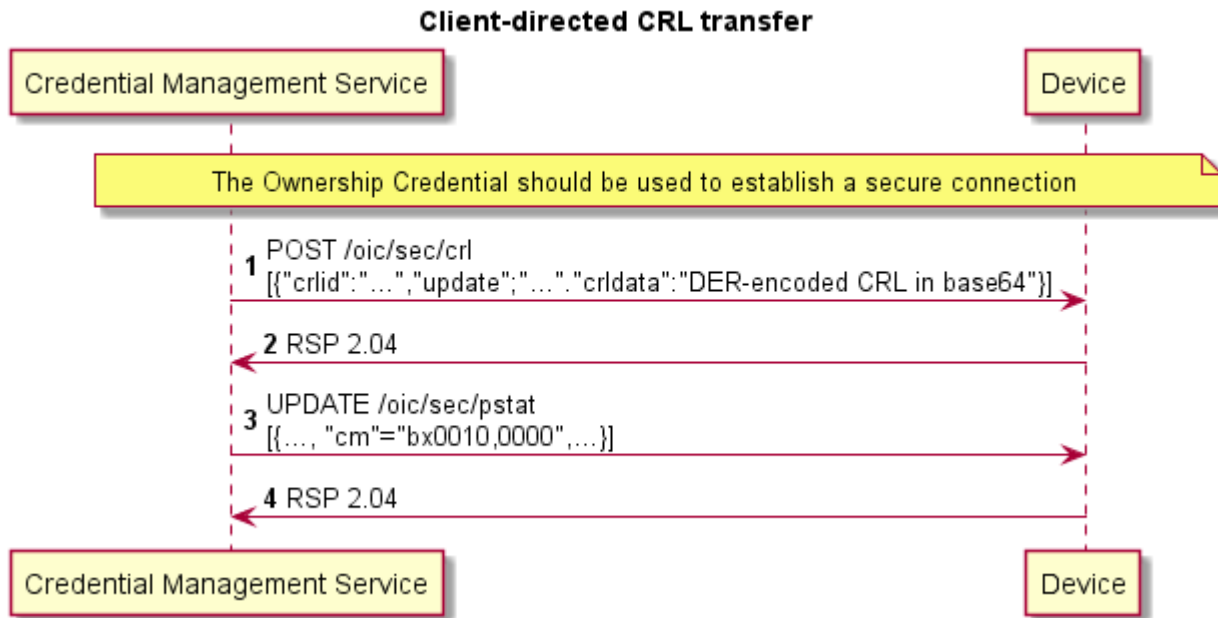
2640 There are two options to update and deliver CRL;

- 2641 – CMS pushes CRL to each Device
- 2642 – each Device periodically requests to update CRL

2643 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

2644 1) The CMS may retrieve the CRL Resource Property.

2645 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.
 2646
 2647



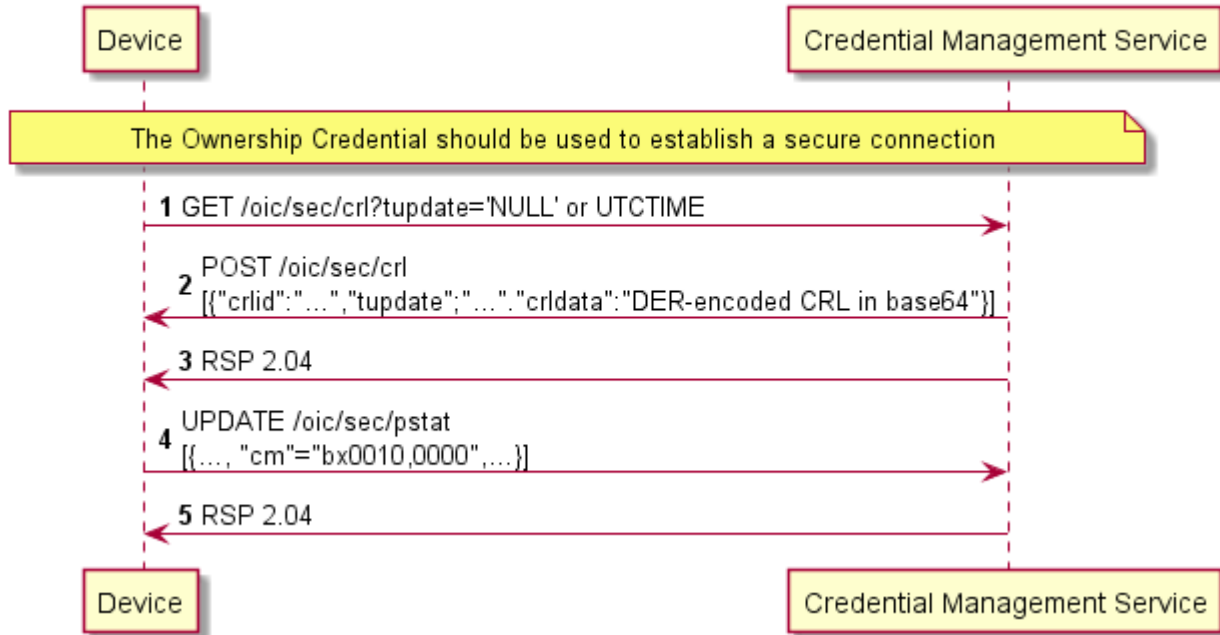
2648

Figure 30 – Client-directed CRL Transfer

2649 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.

- 2650 1) The Device retrieves the CRL Resource Property "tupdate" to the CMS.
 2651 2) If the CMS recognizes the updated CRL information after the designated "tupdate" time, it
 2652 may transfer its CRL to the Device.

Server-directed CRL transfer



2653
2654
2655

Figure 31 – Server-directed CRL Transfer

2656 **10 Device Authentication**

2657 **10.1 Device Authentication General**

2658 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the
2659 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or
2660 more roles that the server can use in access control decisions. Roles may be asserted when the
2661 Device authentication is done with certificates.

2662 **10.2 Device Authentication with Symmetric Key Credentials**

2663 When using symmetric keys to authenticate, the Server Device shall include the
2664 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client
2665 shall validate that it has a credential with the Subject ID set to the Server's Device ID, and a
2666 credential type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error
2667 or other suitable error.

2668 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that
2669 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a
2670 credential with the matching Subject ID and type. If it does not, the Server shall respond with an
2671 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS
2672 protocol, and both Client and Server shall compute the resulting premaster secret.

2673 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2674 When using raw asymmetric keys to authenticate, the Client and the Server shall include a
2675 suitable public key from a credential that is bound to their Device. Each Device shall verify that
2676 the provided public key matches the `PublicData` field of a credential they have, and use the
2677 corresponding Subject ID of the credential to identify the peer Device.

2678 **10.4 Device Authentication with Certificates**

2679 **10.4.1 Device Authentication with Certificates General**

2680 When using certificates to authenticate, the Client and Server shall each include their certificate
2681 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.
2682 Each Device shall validate the certificate chain presented by the peer Device. Each certificate
2683 signature shall be verified until a public key is found within the `"/oic/sec/cred"` Resource with the
2684 `"oic.sec.cred.trustca"` credusage. Credential Resource found in `"/oic/sec/cred"` is used to
2685 terminate certificate path validation. Also, the validity period and revocation status should be
2686 checked for all above certificates, but at this time a failure to obtain a certificate's revocation
2687 status (CRL or OCSP response) MAY continue to allow the use of the certificate if all other
2688 verification checks succeed.

2689 If available, revocation information should be used to verify the revocation status of the certificate.
2690 The URL referencing the revocation information should be retrieved from the certificate (via the
2691 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used
2692 to gather relevant revocation information like CRLs or OCSP responses.

2693 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2694 Devices must follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In
2695 particular:

- 2696 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is
2697 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate
2698 chain **MUST** be rejected. If the `pathLenConstraint` field is present, Devices will confirm the
2699 number of certificates between this certificate and the End-Entity certificate is less than or
2700 equal to `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity

- 2701 certificate can be issued by this certificate. If the pathLenConstraint field is absent, there is no
2702 limit to the chain length.
- 2703 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is
2704 present, and that the keyCertSign bit is asserted.
- 2705 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.
2706 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt
2707 validation with the public keys of possible issuer certificates whose subject name equals the
2708 issuer name of this certificate.
- 2709 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage
2710 (EKU) suitable to the purpose for which it is being presented. An End-Entity certificate which
2711 contains no EKU extension is not valid for any purpose and must be rejected. Any certificate
2712 which contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other
2713 valid EKUs are also present.
- 2714 – Devices MUST verify "transitive EKU" for certificate chains. Issuer certificates (any certificate
2715 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the
2716 certificate chain is being presented. An issuer certificate is valid for a purpose if it contains an
2717 EKU extension and the EKU OID for that purpose is listed in the extension, OR it does not
2718 have an EKU extension. An issuer certificate SHOULD contain an EKU extension and a
2719 complete list of EKUs for the purposes for which it is authorized to issue certificates. An
2720 issuer certificate without an EKU extension is valid for all purposes; this differs from End-
2721 Entity certificates without an EKU extension.
- 2722 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2723 If the Device does not recognize an extension, it must examine the `critical` field. If the field is
2724 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the
2725 certificate as if the extension were absent and proceed accordingly. This applies to all certificates
2726 in a chain.

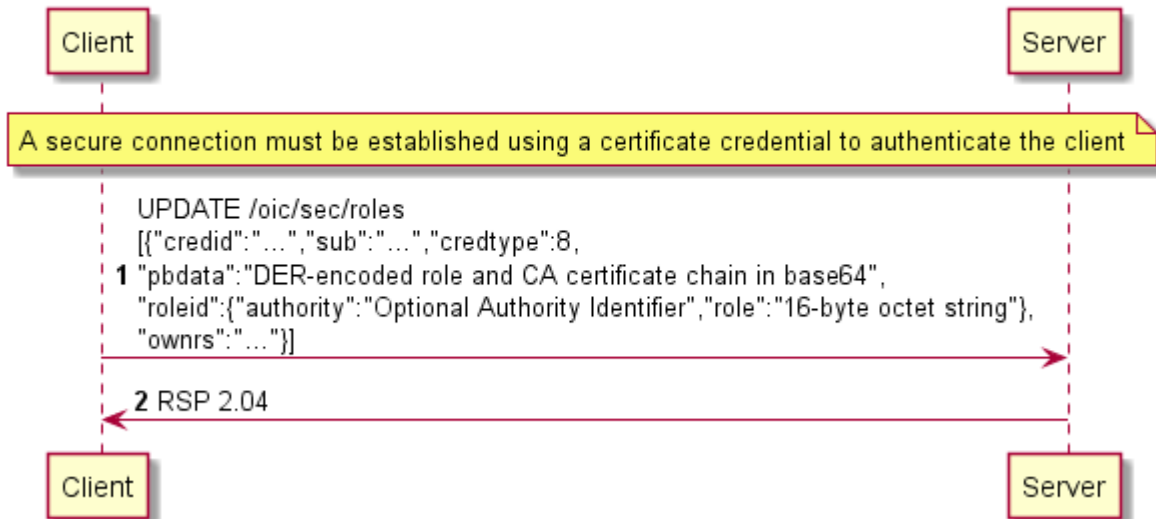
2727 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

2728 **10.4.2 Role Assertion with Certificates**

2729 This clause describes role assertion by a client to a server using a certificate role credential. If a
2730 server does not support the certificate credential type, clients should not attempt to assert roles
2731 with certificates.

2732 Following authentication with a certificate, a client may assert one or more roles by updating the
2733 server's roles resource with the role certificates it wants to use. The role credentials must be
2734 certificate credentials and shall include a certificate chain. The server shall validate each
2735 certificate chain as specified in clause 10.3. Additionally, the public key in the End-Entity
2736 certificate used for Device authentication must be identical to the public key in all role (End-Entity)
2737 certificates. Also, the subject distinguished name in the End-Entity authentication and role
2738 certificates must match. The roles asserted are encoded in the subjectAltName extension in the
2739 certificate. The subjectAltName field can have multiple values, allowing a single certificate to
2740 encode multiple roles that apply to the client. The server shall also check that the EKU extension
2741 of the role certificate(s) contains the value 1.3.6.1.4.1.44924.1.7 (see clause 9.4.2.2) indicating
2742 the certificate may be used to assert roles. Figure 32 describes how a client Device asserts roles
2743 to a server.

Asserting Certificate Role Credentials



2744

2745

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

2746 Additional comments for Figure 32

2747 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If
2748 the server does not support certificate credentials, it should return "501 Not Implemented"

2749 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration
2750 shall not exceed the validity period of the role certificate. When fresh CRL information is
2751 obtained, the certificates in "/oic/sec/roles" should be checked, and the role removed if the
2752 certificate is revoked or expired.

2753 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role
2754 by a client. It is recommended that servers use the validity period of the certificate as a
2755 duration, effectively allowing the CMS to decide the duration.

2756 4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see
2757 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not
2758 be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the
2759 value in the certificate shall be used for validation. For example, if the Period field is set in
2760 the credential, the server shall treat the validity period in the certificate as authoritative.
2761 Similar for the roleid data (authority, role).

2762 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)

2763 6) Clients may GET the "/oic/sec/roles" resource to determine the roles that have been
2764 previously asserted. An array of credential objects shall be returned. If there are no valid
2765 certificates corresponding to the currently connected and authenticated Client's identity, then
2766 an empty array (i.e. []) shall be returned.

2767 10.4.3 OCF PKI Roots

2768 This clause intentionally left empty.

2769 10.4.4 PKI Trust Store

2770 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store
2771 the OCF Root CA certificates in the "oic/sec/cred" resource and SHOULD physically store this
2772 resource in a hardened memory location where the certificates cannot be tampered with.

2773 **10.4.5 Path Validation and extension processing**

2774 Devices SHALL follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In
2775 addition, the following are best practices and SHALL be adhered to by any OCF-compliant
2776 application handling digital certificates

2777 – Validity Period checking

2778 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and
2779 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,
2780 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2781 – Revocation checking

2782 Relying applications SHOULD check the revocation status for all certificates, but at this time,
2783 an application MAY continue to allow the use of the certificate upon a failure to obtain a
2784 certificate's revocation status (CRL or OCSP response), if all other verification checks
2785 succeed.

2786 – basicConstraints

2787 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify
2788 that the basicConstraints extension is present, flagged critical, and that the cA boolean value
2789 in the extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2790 If the pathLenConstraint field is present, Devices will confirm the number of certificates
2791 between this certificate and the End-Entity certificate is less than or equal to
2792 pathLenConstraint. In particular, if pathLenConstraint is zero, only an End-Entity certificate
2793 can be issued by this certificate. If the pathLenConstraint field is absent, there is no limit to
2794 the chain length.

2795 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged
2796 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a
2797 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a
2798 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a
2799 null value.

2800 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-
2801 compliant Root CA certificates SHALL NOT contain a pathLenConstraint. This allows
2802 additional tiers of Intermediate CAs to be implemented in the future without changing the Root
2803 CA trust anchors, should such a requirement emerge.

2804 – keyUsage

2805 For all certificates, Devices shall verify that the key usage extension is present and flagged
2806 critical.

2807 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits
2808 SHALL be asserted.

2809 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be
2810 asserted.

2811 – extendedKeyUsage:

2812 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be
2813 rejected.

2814 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)
2815 are required for compatibility with various TLS implementations.

2816 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)
2817 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be
2818 present and End-Entity certificates with EKU extensions containing both OIDs SHALL be
2819 rejected.

2820 – certificatePolicies
2821 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one
2822 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)
2823 corresponding to the version of the OCF Certificate Policy under which it was issued.
2824 Additional manufacturer-specific CP OIDs may also be populated.

2825 **10.5 Device Authentication with OCF Cloud**

2826 **10.5.1 Device Authentication with OCF Cloud General**

2827 The mechanisms for Device Authentication in clauses 10.2, 10.3 and 10.4 imply that a Device is
2828 authorized to communicate with any other Device meeting the criteria provisioned in
2829 "/oic/sec/cred"; the "/oic/sec/acl2" Resource is additionally used to restrict access to specific
2830 Resources. The present clause describes Device authentication for OCF Cloud, which uses
2831 slightly different criteria as described in clause 5. A Device accessing an OCF Cloud shall
2832 establish a TLS session. The mutual authenticated TLS session is established using Server
2833 certificate and Client certificate.

2834 Each Device is identified based on the Access Token it is assigned during Device Registration.
2835 The OCF Cloud holds an OCF Cloud association table that maps Access Token, User ID and
2836 Device ID. The Device Registration shall happen while the Device is in RFNOP state. After
2837 Device Registration, the updated Access Token, Device ID and User ID are used by the Device
2838 for the subsequent connection with the OCF Cloud.

2839 **10.5.2 Device Connection with the OCF Cloud**

2840 The Device should establish the TLS connection using the certificate based credential. The
2841 connection should be established after Device is provisioned by Mediator.

2842 The TLS session is established between Device and the OCF Cloud as specified in IETF RFC
2843 8323. The OCF Cloud is expected to provide certificate signed by trust anchor that is present in
2844 cred entries of the Device. These cred entries are expected to be configured by the Mediator.

2845 The Device shall validate the OCF Cloud's identity based on the credentials that are contained in
2846 "/oic/sec/cred" Resource entries of the Device.

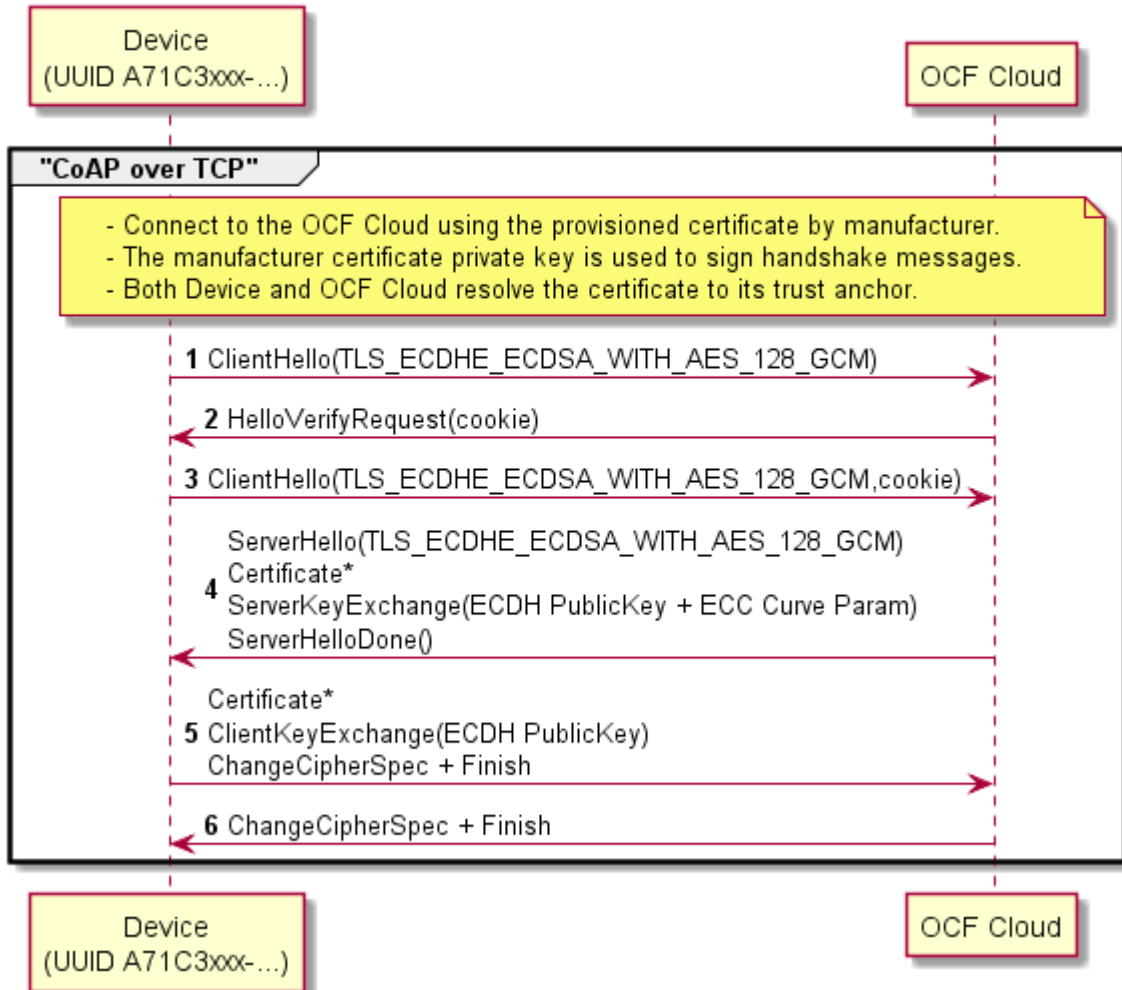
2847 The OCF Cloud is expected to validate the manufacturer certificate provided by the Device.

2848 The assumption is that the OCF Cloud User trusts the OCF Cloud that the Device connects. The
2849 OCF Cloud connection should not happen without the consent of the OCF Cloud User. The
2850 assumption is that the OCF Cloud User has either service agreement with the OCF Cloud
2851 provider or uses manufacturer provided OCF Cloud.

2852 If authentication fails, the "clec" Property of "oic.r.coapcloudconf" Resource on the Device shall
2853 be updated about the failed state, if it is supported by the Device. If authentication succeeds, the
2854 Device and OCF Cloud should establish an encrypted link in accordance with the negotiated
2855 cipher suite.

2856 Figure 33 depicts sequence for Device connection with OCF Cloud and steps described in
2857 Table 22.

Device Connection with OCF Cloud



2858

2859

Figure 33 – Device connection with OCF Cloud

2860

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

2861

10.5.3 Security Considerations

2862

2863

2864

2865

2866

2867

2868

2869

2870

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.

2871 The security considerations about the manufacturer certificate as described in 7.3.6.5 are also
2872 applicable in the Device authentication with the OCF Cloud.

2873 The Device should validate the OCF Cloud's TLS certificate as defined by IETF RFC 6125 and in
2874 accordance with its requirements for Server identity authentication.

2875 The "uid" and "di" Property Value of "/oic/d" Resource may be considered personally identifiable
2876 information in some regulatory regions, and the OCF Cloud is expected to provide protections
2877 appropriate to its governing regulatory bodies.

2878

2879 **11 Message Integrity and Confidentiality**

2880 **11.1 Preamble**

2881 Secured communications between Clients and Servers are protected against eavesdropping,
2882 tampering, or message replay, using security mechanisms that provide message confidentiality
2883 and integrity.

2884 **11.2 Session Protection with DTLS**

2885 **11.2.1 DTLS Protection General**

2886 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices
2887 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See
2888 11.3 for a list of required and optional cipher suites for message communication.

2889 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1
2890 or lower.

2891 Multicast session semantics are not yet defined in this version of the security document.

2892 **11.2.2 Unicast Session Semantics**

2893 For unicast messages between a Client and a Server, both Devices shall authenticate each other.
2894 See clause 10 for details on Device Authentication.

2895 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.
2896 The sending Device shall encrypt and authenticate messages as defined by the selected cipher
2897 suite and the receiving Device shall verify and decrypt the messages before processing them.

2898 **11.2.3 Cloud Session Semantics**

2899 The messages between the OCF Cloud and Device shall be exchanged only if the Device and
2900 OCF Cloud authenticate each other as described in 10.4.3. The asymmetric cipher suites as
2901 described in 11.3.5 shall be employed for establishing a secured session and for
2902 encrypting/decrypting between the OCF Cloud and the Device. The OCF Endpoint sending the
2903 message shall encrypt and authenticate the message using the cipher suite as described in
2904 11.3.5 and the OCF Endpoint shall verify and decrypt the message before processing it.

2905 **11.3 Cipher Suites**

2906 **11.3.1 Cipher Suites General**

2907 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher
2908 suites allowed during ownership transfer and normal operation. The following RFCs provide
2909 additional information about the cipher suites used in OCF.

2910 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2911 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2912 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and
2913 PSKs

2914 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2915 **11.3.2 Cipher Suites for Device Ownership Transfer**

2916 **11.3.2.1 Just Works Method Cipher Suites**

2917 The Just Works OTM may use the following (D)TLS cipher suites.

2918 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,

2919 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

2920 All Devices supporting Just Works OTM shall implement:

2921 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (with the value 0xFF00)

2922 All Devices supporting Just Works OTM should implement:

2923 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256 (with the value 0xFF01)

2924 **11.3.2.2 Random PIN Method Cipher Suites**

2925 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2926 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2927 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2928 All Devices supporting Random Pin Based OTM shall implement:

2929 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

2930 **11.3.2.3 Certificate Method Cipher Suites**

2931 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2932 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

2933 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2934 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2935 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2936 Using the following curve:

2937 secp256r1 (See IETF RFC 4492)

2938 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2939 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2940 Devices supporting Manufacturer Certificate Based OTM should implement:

2941 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2942 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2943 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2944 **11.3.3 Cipher Suites for Symmetric Keys**

2945 The following cipher suites are defined for (D)TLS communication using PSKs:

2946 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2947 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2948 TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET Authentication tag *)

2949 TLS_PSK_WITH_AES_256_CCM_8,

2950 TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET Authentication tag *)

2951 TLS_PSK_WITH_AES_256_CCM,

2952 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2953 All Devices shall implement the following:

2954 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2955

2956 Devices should implement the following:

2957 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2958 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2959 TLS_PSK_WITH_AES_128_CCM_8,

2960 TLS_PSK_WITH_AES_256_CCM_8,

2961 TLS_PSK_WITH_AES_128_CCM,

2962 TLS_PSK_WITH_AES_256_CCM

2963 **11.3.4 Cipher Suites for Asymmetric Credentials**

2964 The following cipher suites are defined for (D)TLS communication with asymmetric keys or
2965 certificates:

2966 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

2967 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2968 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2969 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2970 Using the following curve:

2971 secp256r1 (See IETF RFC 4492)

2972 All Devices supporting Asymmetric Credentials shall implement:

2973 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2974 All Devices supporting Asymmetric Credentials should implement:

2975 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2976 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2977 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2978 **11.3.5 Cipher suites for OCF Cloud Credentials**

2979 The following cipher suites are defined for TLS communication with certificates:

2980 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,

2981 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,

2982 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,

2983 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,

2984 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,

2985 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

2986 All Devices supporting OCF Cloud Certificate Credentials shall implement:

2987 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

2988 All Devices supporting OCF Cloud Certificate Credentials should implement:

2989 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,

2990 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,

2991 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
2992 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
2993 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
2994

2995 **12 Access Control**

2996 **12.1 ACL Generation and Management**

2997 This clause will be expanded in a future version of the document.

2998 **12.2 ACL Evaluation and Enforcement**

2999 **12.2.1 ACL Evaluation and Enforcement General**

3000 The Server enforces access control over application Resources before exposing them to the
3001 requestor. The Security Layer in the Server authenticates the requestor when access is received
3002 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL
3003 entries that specify the requestor's identity, role or may match authenticated requestors using a
3004 subject wildcard.

3005 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a
3006 subject wildcard match of anonymous requestors.

3007 Access is denied if a requested Resource is not matched by an ACL entry.

3008 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources
3009 may be granted prior to provisioning of ACL Resources.

3010 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries
3011 (ACE2) that employ a Resource matching algorithm that uses an array of Resource references to
3012 match Resources to which the ACE2 access policy applies. Matching consists of comparing the
3013 values of the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources
3014 are matched in two ways:

- 3015 1) host reference ("href")
- 3016 2) resource wildcard ("wc").

3017 **12.2.2 Host Reference Matching**

3018 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for
3019 Resource matching.

3020 – The href Property shall be used to find an exact match of the Resource name if present.

3021 **12.2.3 Resource Wildcard Matching**

3022 When present, a wildcard (wc) expression shall be used to match multiple Resources using a
3023 wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

3024 A wildcard expression may be used to match multiple Resources using a wildcard Property
3025 contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined
3026 in Table 23.

3027 **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.

3028 NOTE Discoverable resources appear in the "/oic/res" Resource, while non-discoverable resources may appear in
3029 other collection resources but do not appear in the /res collection.

3030 **12.2.4 Multiple Criteria Matching**

3031 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for
3032 each array element. For example, if a first array element of the "resources" Property contains
3033 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",
3034 then Resources that match either of the two "href" criteria shall be included in the set of matched
3035 Resources.

3036 Example 1 JSON for Resource matching

```
3037 {  
3038 //Matches Resources named "/x/door1" or "/x/door2"  
3039 "resources": [  
3040   {  
3041     "href": "/x/door1"  
3042   },  
3043   {  
3044     "href": "/x/door2"  
3045   },  
3046 ]  
3047 }
```

3048 Example 2 JSON for Resource matching

```
3049 {  
3050 // Matches all Resources  
3051 "resources": [  
3052   {  
3053     "wc": "*"   
3054   }  
3055 ]  
3056 }
```

3057 **12.2.5 Subject Matching using Wildcards**

3058 When the ACE subject is specified as the wildcard string "*" any requestor is matched. The OCF
3059 server may authenticate the OCF client, but is not required to.

3060 Examples: JSON for subject wildcard matching

```
3061 //matches all subjects that have authenticated and confidentiality protections in place.  
3062 "subject" : {  
3063   "conntype" : "auth-crypt"  
3064 }  
3065 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
3066 "subject" : {  
3067   "conntype" : "anon-clear"  
3068 }
```

3069 **12.2.6 Subject Matching using Roles**

3070 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 3071 1) The requestor authenticated with a symmetric key credential, and the role is present in the
3072 roleid Property of the credential's entry in the credential resource, or

3073 2) The requestor authenticated with a certificate, and a valid role certificate is present in the
3074 roles resource with the requestor's certificate's public key at the time of evaluation. Validating
3075 role certificates is defined in 10.3.1.

3076 **12.2.7 ACL Evaluation**

3077 **12.2.7.1 ACE2 matching algorithm**

3078 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 3079 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these
3080 ACE2 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the
3081 signature, at least once, following update of the "/oic/sec/sacl" Resource.
- 3082 2) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.
- 3083 3) Access shall be granted when all these criteria are met:
 - 3084 a) The requestor is matched by the ACE2 "subject" Property.
 - 3085 b) The requested Resource is matched by the ACE2 resources Property and the requested
3086 Resource shall exist on the local Server.
 - 3087 c) The "period" Property constraint shall be satisfied.
 - 3088 d) The "permission" Property constraint shall be applied.

3089 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching
3090 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then
3091 UNION (Perm1, Perm2)=CRUDN.

3092 The Server shall enforce access based on the effective permissions granted.

3093 Batch requests to Resource containing Links require additional considerations when accessing
3094 the linked Resources. ACL considerations for batch request to the Atomic Measurement
3095 Resource Type are provided in clause 12.2.7.2. ACL considerations for batch request to the
3096 Collection Resource Type are provided in 12.2.7.3.

3097 **12.2.7.2 (Currently blank)**

3098 This clause intentionally left empty.

3099 **12.2.7.3 ACL considerations for a batch OCF Interface request to a Collection**

3100 This clause addresses the additional authorization processes which take place when a Server
3101 receives a batch OCF Interface request from a Client to a Collection hosted on that Server,
3102 assuming there is an ACE matching the Collection which permits the original Client request. For
3103 the purposes of this clause, the Server hosting this Collection is called the "Collection host". The
3104 additional authorization process is dependent on whether the linked Resource is hosted on the
3105 Collection host or the linked Resource is hosted on another Server:

- 3106 – For each generated request to a linked Resource hosted on the Collection host, the Collection
3107 host shall apply the ACE2 matching algorithm in clause 12.2.7.1 to determine whether the
3108 linked Resource is permitted to process the generated request, with the following
3109 clarifications:
 - 3110 – The requestor in clause 12.2.7.1 shall be the Client which sent the original Client request.
 - 3111 – The requested Resource in clause 12.2.7.1 shall be the linked Resource, which shall be
3112 matched using at least one of:
 - 3113 – a Resource Wildcard matching the linked Resource, or
 - 3114 – an exact match of the local path of the linked Resource with a "href" Property in the
3115 "resources" array in the ACE2.

3116 – an exact match of the full URI of the linked Resource with a "href" Property in the
3117 "resources" array in the ACE2.

3118 NOTE The full URI of a linked Resource is obtained by concatenating the "anchor" Property of the Link, if present,
3119 and the "href" Property of the Link. The local path can then be determined from the full URI.

3120 If the linked Resource is not permitted to process the generated request, then the Collection host
3121 shall treat such cases as a linked Resource which cannot process the request when composing
3122 the aggregated response to the original Client Request, as specified for the batch OCF Interface
3123 in the ISO/IEC 30118-1:2018.

3124 **13 Security Resources**

3125 **13.1 Security Resources General**

3126 OCF Security Resources are shown in Figure 34.

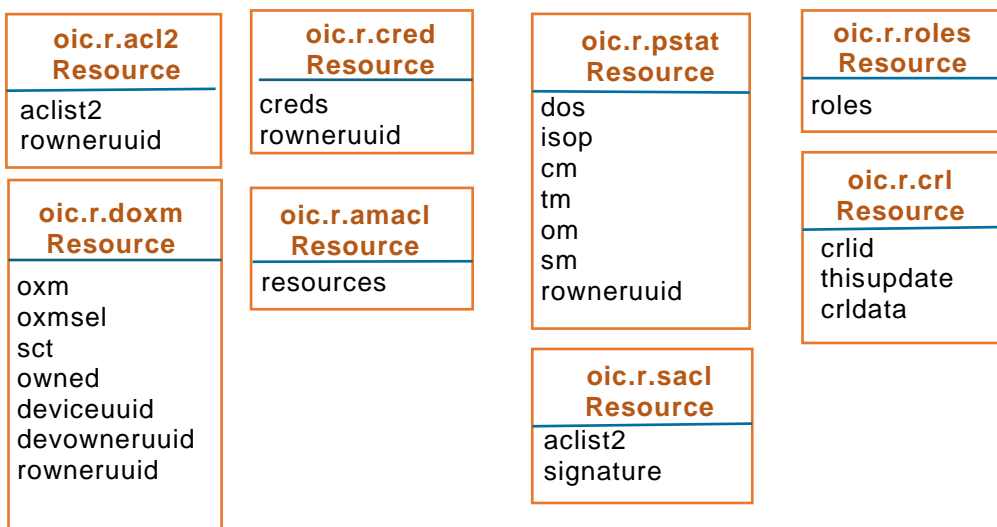
3127 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

3128 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

3129 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

3130 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

3131



3132

Figure 34 – OCF Security Resources

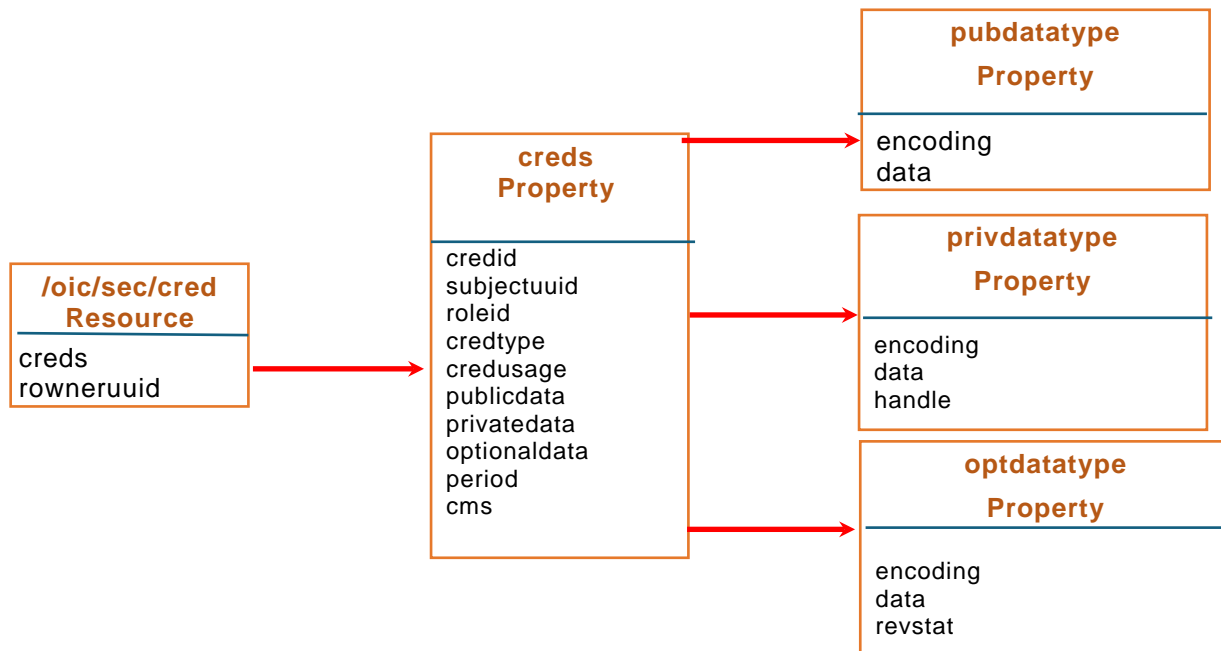


Figure 35 – "/oic/sec/cred" Resource and Properties

3133

3134

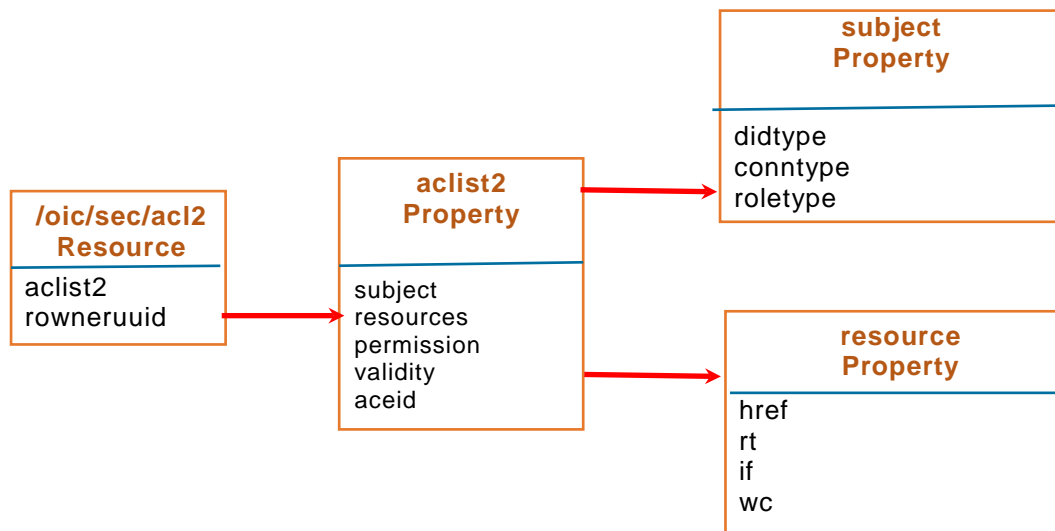
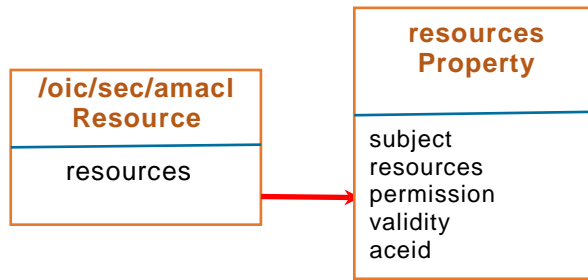


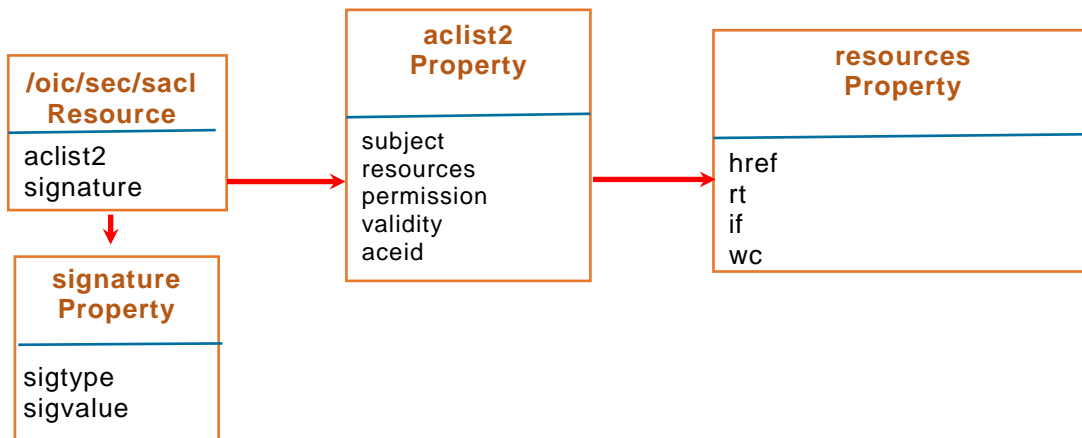
Figure 36 – "/oic/sec/acl2" Resource and Properties

3135



3136

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



3137

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

3138 13.2 Device Owner Transfer Resource

3139 13.2.1 Device Owner Transfer Resource General

3140 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

3141 Resource discovery processing respects the CRUDN constraints supplied as part of the security
3142 Resource definitions contained in this document.

3143 "/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

3145 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.

3147 Table 26 defines the Properties of the "oic.sec.didtype".

3148 **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

3149 The oxms Property contains a list of OTM where the entries appear in the order of preference.
3150 This Property contains the higher priority methods appearing before the lower priority methods.
3151 The DOTS queries this list at the time of onboarding and selects the most appropriate method.

3152 The DOTS shall update the oxmsel Property of the "/oic/sec/doxm" Resource with the OTM that
3153 was used to onboard the Device.

3154 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3155 <DoxmType> ::= <NSS>
3156 <NSS> ::= <Identifier> | { {<NID> "."} <NameSpaceQualifier> "."} <Method>
3157 <NID> ::= <Vendor-or-Organization>
3158 <Identifier> ::= INTEGER
3159 <NameSpaceQualifier> ::= String
3160 <Method> ::= String
3161 <Vendor-Organization> ::= String

```

3162 When an OTM successfully completes, the "owned" Property is set to "1" (TRUE). Consequently,
3163 subsequent attempts to take ownership of the Device will fail.

3164 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in
3165 the "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource
3166 is UPDATED to non-nil UUID value.

3167 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource
3168 after it has updated the devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-
3169 nil-UUID value.

3170 The Device vendor shall determine that the Device identifier ("deviceuuid") is persistent (not
3171 updatable) or that it is non-persistent (updatable by the owner transfer service – aka. DOTS).

3172 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall
3173 fail with the error PROPERTY_NOT_FOUND.

3174 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is non-persistent, the request to
3175 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is
3176 RESET. If the UPDATE to deviceuuid Property of the "/oic/sec/doxm" Resource fails while in the
3177 RFOTM Device state the device state shall transition to RESET where the Server shall set the
3178 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g.
3179 "00000000-0000-0000-0000-000000000000").

3180 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the
3181 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the
3182 "deviceuuid" Property of the "/oic/sec/doxm" Resource each time the device enters RESET
3183 Device state. The temporary deviceuuid value is used while the device state is in the RESET
3184 state and while in the RFOTM device state until the DOTS establishes a secure OTM connection.
3185 The DOTS should RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm"
3186 Resource after it has updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a
3187 non-nil-UUID value.

3188 The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall expose a persistent value (i.e. is
3189 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTS via
3190 an OCF Interface to the deviceuuid Property of the "/oic/sec/doxm" Resource during RFOTM
3191 Device state.).

3192 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a
3193 secure OTM connection and UPDATES the "devowneruuid" Property to a non-nil UUID value.
3194 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the "deviceuuid" Property
3195 of the "/oic/sec/doxm" Resource shall reveal the persistent or semi-persistent value to
3196 authenticated requestors and shall reveal the temporary non-repeated value to unauthenticated
3197 requestors.

3198 See 13.16 for additional details related to privacy sensitive considerations.

3199 **13.2.2 Persistent and Semi-Persistent Device Identifiers**

3200 The Device vendor determines whether a device identifier can be set by a configuration tool or
3201 whether it is immutable. If it is an immutable value this document refers to it as a persistent
3202 device identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four
3203 device identifiers that could be considered persistent or semi-persistent:

- 3204 1) "deviceuuid" Property of "/oic/sec/doxm"
- 3205 2) "di" Property of "/oic/d"
- 3206 3) "piid" Property of "/oic/d"
- 3207 4) "pi" Property of "/oic/p"

3208 **13.2.3 Onboarding Considerations for Device Identifier**

3209 The "deviceuuid" is used to onboard the Device. The other identifiers ("di", "piid" and "pi") are not
3210 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the
3211 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security
3212 Domain owner may have a preference for persistent or semi-persistent device identifiers.

3213 Detecting whether the Device is using persistent or semi-persistent deviceuuid can be achieved
 3214 by attempting to update it.

3215 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is persistent, then an UPDATE
 3216 request, at the appropriate time during onboarding shall fail with an appropriate error response.

3217 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device
 3218 state is RFOTM and when devowneruuid Property value of the "/oic/sec/doxm" Resource has a
 3219 non-nil UUID value.

3220 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is semi-persistent, subsequent to a
 3221 successful UPDATE request to change it; the Device shall remember the semi-persistent value
 3222 until the next successful UPDATE request or until the Device state transitions to RESET.

3223 See 13.16 for addition behaviour regarding "deviceuuid".

3224

3225 **13.2.4 OCF defined OTMs**

3226 Table 27 defines the Properties of the "oic.sec.doxmtype".

3227 **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 ^a .
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a 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.doxm.self	4	The manufacturer shall set the "/doxm.oxmsel" value to (4). The Server shall reset this value to (4) upon entering RESET Device state.
OCF Reserved	<Reserved>	5~0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	Reserved for vendor-specific OTM use

^a The just-works method is subject to a man-in-the-middle attacker. Precautions should be taken to provide physical security when this method is used.

3228 **13.3 Credential Resource**

3229 **13.3.1 Credential Resource General**

3230 The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and
 3231 support services as well as credentials used to verify Clients and support services.

3232 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared
 3233 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to
 3234 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3235 In order to provide an interface which allows management of the "creds" Array Property, the
 3236 RETRIEVE, UPDATE and DELETE operations on the "oic.r.cred" Resource shall behave as
 3237 follows:

- 3238 1) A RETRIEVE shall return the full Resource representation, except that any write-only
 3239 Properties shall be omitted (e.g. private key data).
- 3240 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
 3241 UPDATE request, as follows:
 - 3242 a) If an UPDATE representation includes the "creds" array Property, then:
 - 3243 i) Supplied "creds" with a "credid" that matches an existing "credid" shall replace
 3244 completely the corresponding "cred" in the existing "creds" array.
 - 3245 ii) Supplied "creds" without a "credid" shall be appended to the existing "creds" array,
 3246 and a unique (to the cred Resource) "credid" shall be created and assigned to the new
 3247 "cred" by the Server. The "credid" of a deleted "cred" should not be reused, to
 3248 improve the determinism of the interface and reduce opportunity for race conditions.
 - 3249 iii) Supplied "creds" with a "credid" that does not match an existing "credid" shall be
 3250 appended to the existing "creds" array, using the supplied "credid".
 - 3251 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device
 3252 States in which an UPDATE of the "creds" array Property is always rejected. If OCF
 3253 Device is in a Device State where the Access Mode in this row contains "R", then the
 3254 OCF Device shall reject all UPDATES of the "creds" array Property.
- 3255 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not
 3256 remove the "oic.r.cred" Resource.
- 3257 4) A DELETE with one or more "credid" query parameters shall remove the "cred"(s) with the
 3258 corresponding "credid"(s) from the "creds" array.
- 3259 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in
 3260 which a DELETE is always rejected. If OCF Device is in a Device State where the Access
 3261 Mode in this row contains "R", then the OCF Device shall reject all DELETES.

3262 NOTE The "oic.r.cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined
 3263 in ISO/IEC 30118-1:2018.

3264 "oic.r.cred" Resource is defined in Table 28.

3265 **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

3266 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.

3268 All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end
3269 interaction.

3270 The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid
3271 Property.

3272 ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access
3273 modes.

3274 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".

3276 Table 31 defines the Properties of "oic.sec.credusagetype".

3277 **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

3278 Table 32 defines the Properties of "oic.sec.pubdatatype".

3279 **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

3280 Table 33 defines the Properties of "oic.sec.privdatatype".

3281 **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

3282 Table 34 defines the Properties of "oic.sec.optdatatype".

3283

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

3284 Table 35 defines the Properties of "oic.sec.roletype".

3285

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.

3286 **13.3.2 Properties of the Credential Resource**

3287 **13.3.2.1 Credential ID**

3288 Credential ID ("credid") is a local reference to an entry in a "creds" Property array of the
3289 "/oic/sec/cred" Resource. The SRM generates it. The "credid" Property shall be used to
3290 disambiguate array elements of the "creds" Property.

3291 **13.3.2.2 Subject UUID**

3292 The "subjectuid" Property identifies the Device to which an entry in a "creds" Property array of
3293 the "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication
3294 challenge-response or to authenticate an authentication challenge.

3295 A "subjectuid" Property that matches the Server's own "deviceuid" Property, distinguishes the
3296 array entries in the "creds" Property that pertain to this Device.

3297 The "subjectuid" Property shall be used to identify a group to which a group key is used to
3298 protect shared data.

3299 When certificate chain is used during secure connection establishment, the "subjectuuid"
3300 Property shall also be used to verify the identity of the responder. The presented certificate chain
3301 shall be accepted, if there is a matching Credential entry on the Device that satisfies all of the
3302 following:

- 3303 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3304 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity
3305 certificate in the presented chain. If Subject UUID of the entry is set as a wildcard "*", this
3306 condition is automatically satisfied.
- 3307 – Credential Usage of the entry is "oic.sec.cred.trustca".

3308 **13.3.2.3 Role ID**

3309 The roleid Property identifies a role that has been granted to the credential.

3310 **13.3.2.4 Credential Type**

3311 The "credtype" Property is used to interpret several of the other Property values whose contents
3312 can differ depending on credential type. These Properties include "publicdata", "privatedata" and
3313 "optionaldata". The "credtype" Property value of "0" ("no security mode") is reserved for testing
3314 and debugging circumstances. Production deployments shall not allow provisioning of credentials
3315 of type "0". The SRM should introduce checking code that prevents its use in production
3316 deployments.

3317 **13.3.2.5 Public Data**

3318 The "publicdata" Property contains information that provides additional context surrounding the
3319 issuance of the credential. For example, it might contain information included in a certificate or
3320 response data from a CMS. It might contain wrapped data.

3321 **13.3.2.6 Private Data**

3322 The "privatedata" Property contains secret information that is used to authenticate a Device,
3323 protect data or verify an authentication challenge-response.

3324 The "privatedata" Property shall not be disclosed outside of the SRM's trusted computing
3325 perimeter. A secure element (SE) or trusted execution environment (TEE) should be used to
3326 implement the SRM's trusted computing perimeter. The privatedata contents may be referenced
3327 using a handle; for example, if used with a secure storage sub-system.

3328 **13.3.2.7 Optional Data**

3329 The "optionaldata" Property contains information that is optionally supplied, but facilitates key
3330 management, scalability or performance optimization.

3331 **13.3.2.8 Period**

3332 The "period" Property identifies the validity period for the credential. If no validity period is
3333 specified, the credential lifetime is undetermined. Constrained devices that do not implement a
3334 date-time capability shall obtain current date-time information from its CMS.

3335 **13.3.2.9 Credential Refresh Method Type Definition**

3336 The CMS shall implement the credential refresh methods specified in the "crms" Property of the
3337 "oic.sec.creds" array in the "/oic/sec/cred" Resource.

3338 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.

3340 13.3.2.10 Credential Usage

3341 Credential Usage indicates to the Device the circumstances in which a credential should be used.
3342 Five values are defined:

- 3343 – "oic.sec.cred.trustca": This certificate is a trust anchor for the purposes of certificate chain
3344 validation, as defined in 10.4. OCF Server SHALL remove any "/oic/sec/cred" entries with an
3345 "oic.sec.cred.trustca" credusage upon transitioning to RFOTM. OCF Servers SHALL use
3346 "/oic/sec/cred" entries that have an "oic.sec.cred.trustca" Value of "credusage" Property only
3347 as trust anchors for post-onboarding (D)TLS session establishment in RFNOP state; these
3348 entries are not to be used for onboarding (D)TLS sessions.
- 3349 – "oic.sec.cred.cert": This "credusage" is used for certificates for which the Device possesses
3350 the private key and uses it for identity authentication in a secure session, as defined in clause
3351 10.4.
- 3352 – "oic.sec.cred.rolecert": This "credusage" is used for certificates for which the Device
3353 possesses the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3354 – "oic.sec.cred.mfgtrustca": This certificate is a trust anchor for the purposes of the
3355 Manufacturer Certificate Based OTM as defined in clause 7.3.6. OCF Servers SHALL use
3356 "/oic/sec/cred" entries that have an "oic.sec.cred.mfgtrustca" Value of "credusage" Property
3357 only as trust anchors for onboarding (D)TLS session establishment; these entries are not to
3358 be used for post-onboarding (D)TLS sessions.

3359 – "oic.sec.cred.mfgcert": This certificate is used for certificates for which the Device possesses
3360 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as
3361 defined in clause 7.3.6.

3362 13.3.3 Key Formatting

3363 13.3.3.1 Symmetric Key Formatting

3364 Symmetric keys shall have the format described in Table 37 and Table 38.

3365 **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.

3366

3367 **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.

3368 13.3.3.2 Asymmetric Keys

3369 Asymmetric key formatting is not available in this revision of the document.

3370 13.3.3.3 Asymmetric Keys with Certificate

3371 Key formatting is defined by certificate definition.

3372 13.3.3.4 Passwords

3373 Password formatting is not available in this revision of the document.

3374 13.3.4 Credential Refresh Method Details

3375 13.3.4.1 Provisioning Service

3376 The resource owner identifies the provisioning service. If the Server determines a credential
3377 requires refresh and the other methods do not apply or fail, the Server will request re-
3378 provisioning of the credential before expiration. If the credential is allowed to expire, the Server
3379 should delete the Resource.

3380 13.3.4.2 Pre-Shared Key

3381 13.3.4.2.1 Pre-Shared Key General

3382 Using this mode, the current PSK is used to establish a Diffie-Hellman session key in DTLS. The
3383 TLS_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3384 $PSK = TLS_PRF(\text{MasterSecret}, \text{Message}, \text{length});$

3385 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of
3386 the above ciphersuites.

3387 – Message is the concatenation of the following values:

3388 – RM - Refresh method – I.e. "oic.sec.crm.psk"

3389 – Device ID_A is the string representation of the Device ID that supplied the DTLS
3390 ClientHello.

3391 – Device ID_B is the Device responding to the DTLS ClientHello message

3392 – Length of Message in bytes.

3393 Both Server and Client use the PSK to update the "/oic/sec/cred" Resource's "privatedata"
3394 Property. If Server initiated the credential refresh, it selects the new validity period. The Server
3395 sends the chosen validity period to the Client over the newly established DTLS session so it can
3396 update the corresponding credential Resource for the Server.

3397 **13.3.4.2.2 Random PIN**

3398 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.
3399 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session
3400 key should be used to switch from PIN to PSK mode.

3401 The PIN is randomly generated by the Server and communicated to the Client through an out-of-
3402 band method. The OOB method used is out-of-scope.

3403 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used
3404 to generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a
3405 DTLS ciphersuite that accepts a PSK.

3406 PPSK = PBKDF2(PRF, PIN, RM, Device ID, c, dkLen)

3407 The PBKDF2 function has the following parameters:

3408 – PRF – Uses the DTLS PRF.

3409 – PIN – Shared between Devices.

3410 – RM - Refresh method – I.e. "oic.sec.crm.rdp"

3411 – Device ID – UUID of the new Device.

3412 – c – Iteration count initialized to 1000, incremented upon each use.

3413 – dkLen – Desired length of the derived PSK in octets.

3414 Both Server and Client use the PPSK to update the "/oic/sec/cred" Resource's PrivateData
3415 Property. If Server initiated the credential refresh, it selects the new validity period. The Server
3416 sends the chosen validity period to the Client over the newly established DTLS session so it can
3417 update its corresponding credential Resource for the Server.

3418 **13.3.4.2.3 SKDC**

3419 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid
3420 Property value that matches a CMS that implements SKDC functionality and where the Client
3421 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request
3422 message is delivered to the CMS and in response returns the ticket request. The Server updates
3423 or instantiates a "/oic/sec/cred" Resource guided by the ticket response contents.

3424 **13.3.4.2.4 PKCS10**

3425 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid
3426 Property value that matches a CMS that supports the "oic.sec.crm.pk10" credential refresh
3427 method. A PKCS10 formatted message is delivered to the service. After the refreshed certificate
3428 is issued, the CMS pushes the certificate to the Server. The Server updates or instantiates an
3429 "/oic/sec/cred" Resource guided by the certificate contents.

3430 **13.3.4.3 Resource Owner**

3431 The Resource Owner Property allows credential provisioning to occur soon after Device
 3432 onboarding before access to support services has been established. It identifies the entity
 3433 authorized to manage the "/oic/sec/cred" Resource in response to Device recovery situations.

3434 **13.4 Certificate Revocation List**

3435 **13.4.1 CRL Resource Definition**

3436 Device certificates and private keys are kept in "cred" Resource. CRL is maintained and updated
 3437 with a separate "crl" Resource that is newly defined for maintaining the revocation list.

3438 "oic.r.crl" Resource is defined in Table 39.

3439 **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

3440 Table 40 defines the Properties of "oic.r.crl".

3441 **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

3442 **13.5 ACL Resources**

3443 **13.5.1 ACL Resources General**

3444 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be
 3445 expressed using three ACL Resource Types: "/oic/sec/acl2", "/oic/sec/amacl" and "/oic/sec/sacl".
 3446 The subject (e.g. "deviceuuid" of the Client) requesting access to a Resource shall be
 3447 authenticated prior to applying the ACL check. Resources that are available to multiple Clients
 3448 can be matched using a wildcard subject. All Resources accessible via the unsecured
 3449 communication endpoint shall be matched using a wildcard subject.

3450 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3451 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3452 **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>

3453 The <Deviceld> token means the requestor must possess a credential that uses <UUID> as its
3454 identity in order to match the requestor to the <ACE> policy.

3455 The <RoleID> token means the requestor must possess a role credential with <Character> as its
3456 role in order to match the requestor to the <ACE> policy.

3457 The <Wildcard> token "*" means any requestor is matched to the <ACE> policy, with or without
3458 authentication.

3459 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the
3460 <ACE> policy to Resources.

3461 The <OIC_LINK> token contains values used to query existence of hosted Resources.

3462 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>
3463 and <ResourceRef> matching does not produce the empty set match.

3464 Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE
3465 ("D"), NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the
3466 respective permission is not granted.

3467 The empty set match result defaults to a condition where no access rights are granted.

3468 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.
3469 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively
3470 be granted and rescinded according to the pattern.

3471 13.5.3 ACL Resource

3472 An "acl2" is a list of type "ace2".

3473 In order to provide an interface which allows management of array elements of the "aclist2"
3474 Property associated with a "/oic/sec/acl2" Resource. The RETRIEVE, UPDATE and DELETE
3475 operations on the "/oic/sec/acl2" Resource SHALL behave as follows:

- 3476 1) A RETRIEVE shall return the full Resource representation.
- 3477 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
3478 UPDATE request, as follows:
 - 3479 a) If an UPDATE representation includes the array Property, then:
 - 3480 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace
3481 completely the corresponding ACE in the existing "aces2" array.

3482 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array,
3483 and a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new
3484 ACE by the Server. The "aceid" of a deleted ACE should not be reused, to improve
3485 the determinism of the interface and reduce opportunity for race conditions.

3486 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be
3487 appended to the existing "aces2" array, using the supplied "aceid".

3488 The rows in Table 44 defines the Properties of "oic.sec.acl2".

3489 iv) Table 44 corresponding to the "aclist2" array Property dictate the Device States in
3490 which an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is
3491 in a Device State where the Access Mode in this row contains "R", then the OCF
3492 Device shall reject all UPDATES of the "aclist2" array Property.

3493 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not
3494 remove the "oic.r.ace2" Resource.

3495 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the
3496 corresponding "aceid"(s) from the "aces2" array.

3497 The rows in Table 44 defines the Properties of "oic.sec.acl2".

3498 5) Table 44 corresponding to the "aclist2" array Property dictate the Device States in which a
3499 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this
3500 row contains "R", then the OCF Device shall reject all DELETES.

3501 NOTE The "oic.r.acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces
3502 defined in ISO/IEC 30118-1:2018.

3503 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no
3504 entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl2",
3505 "/oic/sec/sacl" and "/oic/sec/amacl" do not match the subject and the requested Resource.

3506 It is possible the AMS has an ACL policy that satisfies a resource access request, but the
3507 necessary ACE has not been provisioned to Server. The Server may open a secure connection to
3508 the AMS to request ACL provisioning. The Server may use filter criteria that returns a subset of
3509 the AMS ACL policy. The AMS shall obtain the Server Device ID using the secure connection
3510 context.

3511 The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS
3512 to process an "/oic/sec/amacl" Resource. The AMS shall match the AMACL policy and return the
3513 Permission Property or an error if no match is found.

3514 If the requested Resource is still not matched, the Server returns an error. The requester should
3515 query the Server to discover the configured AMS services. The Client should contact the AMS to
3516 request a sacl ("/oic/sec/sacl") Resource. Performing the following operations implement this type
3517 of request:

3518 1) Client: Open secure connection to AMS.

3519 2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3520 3) AMS: constructs a "/oic/sec/sacl" Resource that is signed by the AMS and returns it in
3521 response to the RETRIEVE command.

3522 4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3523 5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3524 6) Client: retries original Resource access request. This time the new ACL is included in the
3525 local ACL evaluation.

3526 The ACL contained in the "/oic/sec/sacl" Resource should grant longer term access that satisfies
 3527 repeated Resource requests.

3528 Table 42 defines the values of "oic.sec.crudntype".

3529 **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

3530 "oic.sec.acl2" Resource is defined in Table 28.

3531 **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

3532 Table 44 defines the Properties of "oic.sec.acl2".

3533 **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.

3534

3535 Table 45 defines the Properties of "oic.sec.ace2".

3536

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.

3537 Table 46 defines the Properties of "oic.sec.ace2.resource-ref".

3538

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.

3539 Table 47 defines the values of "oic.sec.ace2.resource-ref".

3540

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

3541 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack
3542 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's
3543 request using policies contained in ACL resources.

3544 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified
3545 Resource references include the device identifier in the href Property that identifies the remote
3546 Resource Server that hosts the Resource. Partially qualified references mean that the local
3547 Resource Server hosts the Resource. If a fully qualified resource reference is given, the
3548 Intermediary enforcing access shall have a secure channel to the Resource Server and the
3549 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource
3550 access enforcement point.

3551 Resource Servers should include references to Device and ACL Resources where access
3552 enforcement is to be applied. However, access enforcement logic shall not depend on these
3553 references for access control processing as access to Server Resources will have already been
3554 granted.

3555 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and
3556 modify this Resource. This prevents non-terminating dependency on some other ACL Resource.
3557 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL
3558 Resource.

3559 An ACE2 entry is considered "currently valid" if the validity period of the ACE2 entry includes the
3560 time of the request. The validity period in the ACE2 may be a recurring time period (e.g., daily
3561 from 1:00-2:00). Matching the resource(s) specified in a request to the resource Property of the
3562 ACE2 is defined in clause 12.2. For example, one way they can match is if the Resource URI in
3563 the request exactly matches one of the resource references in the ACE2 entries.

3564 A request will match an ACE2 if any of the following are true:

- 3565 1) The ACE2 "subject" Property is of type "oic.sec.didtype" has a UUID value that matches the
3566 "deviceuuid" Property associated with the secure session;
3567 AND the Resource of the request matches one of the resources Property of the ACE2
3568 "oic.sec.ace2.resource-ref";
3569 AND the ACE2 is currently valid.
- 3570 2) The ACE2 "subject" Property is of type "oic.sec.conntype" and has the wildcard value that
3571 matches the currently established connection type;
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 3) When Client authentication uses a certificate credential;
3576 AND one of the "roleid" values contained in the role certificate matches the "roleid" Property
3577 of the ACE2 "oic.sec.roletype";
3578 AND the role certificate public key matches the public key of the certificate used to establish
3579 the current secure session;
3580 AND the resource of the request matches one of the array elements of the "resources"
3581 Property of the ACE2 "oic.sec.ace2.resource-ref";
3582 AND the ACE2 is currently valid.

3583 4) When Client authentication uses a certificate credential;
3584 AND the CoAP payload query string of the request specifies a role, which is member of the
3585 set of roles contained in the role certificate;
3586 AND the roleid values contained in the role certificate matches the "roleid" Property of the
3587 ACE2 "oic.sec.roletype";
3588 AND the role certificate public key matches the public key of the certificate used to establish
3589 the current secure session;
3590 AND the resource of the request matches one of the resources Property of the ACE2
3591 "oic.sec.ace2.resource-ref";
3592 AND the ACE2 is currently valid.

3593 5) When Client authentication uses a symmetric key credential;
3594 AND one of the "roleid" values associated with the symmetric key credential used in the
3595 secure session, matches the "roleid" Property of the ACE2 "oic.sec.roletype";
3596 AND the resource of the request matches one of the array elements of the "resources"
3597 Property of the ACE2 "oic.sec.ace2.resource-ref";
3598 AND the ACE2 is currently valid.

3599 6) When Client authentication uses a symmetric key credential;
3600 AND the CoAP payload query string of the request specifies a role, which is contained in the
3601 "oic.r.cred.creds.roleid" Property of the current secure session;
3602 AND CoAP payload query string of the request specifies a role that matches the "roleid"
3603 Property of the ACE2 "oic.sec.roletype";
3604 AND the resource of the request matches one of the array elements of the "resources"
3605 Property of the ACE2 "oic.sec.ace2.resource-ref";
3606 AND the ACE2 is currently valid.

3607 A request is granted if ANY of the 'matching' ACE2 entries contain the permission to allow the
3608 request. Otherwise, the request is denied.

3609 There is no way for an ACE2 entry to explicitly deny permission to a resource. Therefore, if one
3610 Device with a given role should have slightly different permissions than another Device with the
3611 same role, they must be provisioned with different roles.

3612 The Server is required to verify that any hosted Resource has authorized access by the Client
3613 requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the
3614 Resource request processing should be applied securely and efficiently. See Annex A for
3615 example.

3616 **13.6 Access Manager ACL Resource**

3617 "oic.r.amacl" Resource is defined in Table 48.

3618

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

3619 Table 49 defines the Properties of "oic.r.amacl".

3620

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

3621 The AMS should be used to centralize management of access policy, but requires Servers to
 3622 open a connection to the AMS whenever the named Resources are accessed. See A.2 for
 3623 example.

3624 **13.7 Signed ACL Resource**

3625 "oic.r.sacl" Resource is defined in Table 50.

3626

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

3627 Table 51 defines the Properties of "oic.r.sacl".

3628

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

3629 Table 52 defines the Properties of "oic.sec.sigtype".

3630 **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

3631 **13.8 Provisioning Status Resource**

3632 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning
3633 should be Client-directed or Server-directed. Client-directed provisioning relies on a Client device
3634 to determine what, how and when Server Resources should be instantiated and updated. Server-
3635 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-
3636 directed provisioning depends on configuration of the rowneruuid Property of the "/oic/sec/doxm",
3637 "/oic/sec/cred" and "/oic/sec/acl2" Resources to identify the device ID of the trusted DOTS, CMS
3638 and AMS services respectively. Furthermore, the "/oic/sec/cred" Resource should be provisioned
3639 at ownership transfer with credentials necessary to open a secure connection with appropriate
3640 support service.

3641 "oic.r.pstat" Resource is defined in Table 53.

3642 **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

3643 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	rowneruid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOTS should configure the rowneruid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruid does not refer to a valid DOTS the Server shall transition to RESET Device state.

3645 The provisioning status Resource "/oic/sec/pstat" is used to enable Devices to perform self-
3646 directed provisioning. Devices are aware of their current configuration status and a target
3647 configuration objective. When there is a difference between current and target status, the Device

3648 should consult the rowneruid Property of "/oic/sec/cred" Resource to discover whether any
 3649 suitable provisioning services exist. The Device should request provisioning if configured to do so.
 3650 The om Property of "/oic/sec/pstat" Resource will specify expected Device behaviour under these
 3651 circumstances.

3652 Self-directed provisioning enables Devices to function with greater autonomy to minimize
 3653 dependence on a central provisioning authority that should be a single point of failure in the OCF
 3654 Security Domain.

3655 Table 55 defines the Properties of "/oic/sec/dostype".

3656 **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

3657 In all Device states:

3658 – An authenticated and authorised Client may change the Device state of a Device by updating
 3659 pstat.dos.s to the desired value. The allowed Device state transitions are defined in
 3660 Figure 27.

3661 – Prior to updating "pstat.dos.s", the Client configures the Device to meet entry conditions for
 3662 the new Device state. The SVR definitions define the entity (Client or Server) expected to
 3663 perform the specific SVR configuration change to meet the entry conditions. Once the Client
 3664 has configured the aspects for which the Client is responsible, it may update "pstat.dos.s".
 3665 The Server then makes any changes for which the Server is responsible, including updating
 3666 required SVR values, and set pstat.dos.s to the new value.

3667 – The "pstat.dos.p" Property is read-only by all Clients.

3668 – The Server sets "pstat.dos.p" to TRUE before beginning the process of updating "pstat.dos.s",
 3669 and sets it back to FALSE when the "pstat.dos.s" change is completed.

3670 Any requests to update "pstat.dos.s" while "pstat.dos.p" is TRUE are denied.

3671 When Device state is RESET:

3672 – All SVR content is removed and reset to manufacturer default values.

3673 – The default manufacturer Device state is RESET.

3674 – NCRs are reset to manufacturer default values.

3675 – NCRs are inaccessible.

3676 – After successfully processing RESET the SRM transitions to RFOTM by setting "s" Property of
3677 "/oic/sec/dostype" Resource to RFOTM.

3678 When Device state is RFOTM:

3679 – NCRs are inaccessible.

3680 – Before OTM is successful, the deviceuuid Property of "/oic/sec/doxm" Resource shall be set
3681 to a temporary non-repeated value as defined in clauses 13.2 and 13.16.

3682 – Before OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-only by
3683 unauthenticated requestors

3684 – After the OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-write by
3685 authorized requestors.

3686 – The negotiated Device OC is used to create an authenticated session over which the DOTS
3687 directs the Device state to transition to RFPRO.

3688 – If an authenticated session cannot be established the ownership transfer session should be
3689 disconnected and SRM sets back the Device state to RESET state.

3690 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the
3691 SRM asserts the OTM failed, should be disconnected, and transitions to RESET
3692 ("/pstat.dos.s"=RESET).

3693 – The DOTS UPDATES the "devowneruuid" Property in the "/doxm" Resource to a non-nil UUID
3694 value. The DOTS (or other authorized client) may update it multiple times while in RFOTM. It
3695 is not updatable while in other device states except when the Device state returns to RFOTM
3696 through RESET.

3697 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done,
3698 the DOTS UPDATES the "owned" Property in the "/doxm" Resource to "true".

3699 When Device state is RFPRO:

3700 – The s Property of "/oic/sec/dostype" Resource is read-only by unauthorized requestors and
3701 read-write by authorized requestors.

3702 – NCRs are inaccessible, except for Easy Setup Resources, if supported.

3703 – The OCF Server may re-create NCRs.

3704 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.

3705 – An authorized Client may perform consistency checks on SVRs to determine which shall be
3706 re-provisioned.

3707 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if
3708 the Device has already transitioned from SRESET but consistency checks continue to fail.

3709 – The authorized Client sets the "/pstat.dos.s"=RFNOP.

3710 When Device state is RFNOP:

3711 – The "/pstat.dos.s" Property is read-only by unauthorized requestors and read-write by
3712 authorized requestors.

3713 – NCRs, SVRs and core Resources are accessible following normal access processing.

3714 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or
3715 RESET.

3716 When Device state is SRESET:

3717 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to
3718 access or reference them.

- 3719 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
3720 include devowneruuid Property of the "/oic/sec/doxm" Resource,
3721 "creds":[{"...","subjectuuid":<devowneruuid>},...] Property of the "/oic/sec/cred" Resource and
3722 s Property of the "/oic/sec/dostype" Resource of "/oic/sec/pstat" Resource.
- 3723 – The certificates that identify and authorize the Device owner are sufficient to re-create
3724 minimalist "/cred" and "/doxm" resources enabling Device owner control of SRESET. If the
3725 SRM can't establish these Resources, then it will transition to RESET state.
- 3726 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as
3727 needed to ensure they are available for continued provisioning in RFPRO or for normal
3728 functioning in RFNOP.
- 3729 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING
3730 "dos.s" Property of the "/pstat" Resource with RFPRO or RFNOP values
- 3731 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to
3732 Device owner privileges.
- 3733 – The SRM asserts a Client-directed operational mode (e.g. "/pstat.om"=CLIENT_DIRECTED).
- 3734 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.
3735 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning
3736 mode without selecting any particular value.
- 3737 "oic.sec.dpmttype" is defined in Table 56.

3738 **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

3739 Table 57 and Table 58 define the values of "oic.sec.dpmttype".

3740 **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)

3741 **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
----------	------------	------------------------

3742 The *provisioning operation mode* type is an 8-bit mask enumerating the various provisioning
3743 operation modes.

3744 "oic.sec.pomtype" is defined in Table 59.

3745 **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

3746 Table 60 defines the values of "oic.sec.pomtype".

3747 **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

3748 13.9 Certificate Signing Request Resource

3749 The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be
3750 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC
3751 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of
3752 "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr"
3753 Resource.

3754 "oic.r.csr" Resource is defined in Table 61.

3755 **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

3756 Table 62 defines the Properties of "oic.r.csr".

3757

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

3758 The Device chooses which public key to use, and may optionally generate a new key pair for this
3759 purpose.

3760 In the CSR, the Common Name component of the Subject Name shall contain a string of the
3761 format "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122.
3762 The Common Name, and other components of the Subject Name, may contain other data. If the
3763 Device chooses to include additional information in the Common Name component, it shall delimit
3764 it from the UUID field by white space, a comma, or a semicolon.

3765 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to
3766 generate a new key pair, the Device may begin generation of a key pair as a result of a
3767 RETRIEVE of this resource. If the Device cannot immediately respond to the RETRIEVE request
3768 due to time required to generate a key pair, the Device shall return an "operation pending" error.
3769 This indicates to the Client that the Device is not yet ready to respond, but will be able at a later
3770 time. The Client should retry the request after a short delay.

3771 13.10 Roles Resource

3772 The roles Resource maintains roles that have been asserted with role certificates, as described in
3773 clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role
3774 certificate. Servers shall only grant access to the roles information associated with the public key
3775 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.
3776 See 10.4.2 for how role certificates are validated.

3777 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS
3778 session with a Client, if is not already created. The roles Resource shall only expose a secured
3779 OCF Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long
3780 as the (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least
3781 until the certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of
3782 clause 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A
3783 Server should regularly inspect the contents of the roles resource and purge contents based on a
3784 policy it determines based on its resource constraints. For example, expired certificates, and
3785 certificates from Clients that have not been heard from for some arbitrary period of time could be
3786 candidates for purging.

3787 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In
3788 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall
3789 behave as follows. Unlisted operations are implementation specific and not reliable.

3790 1) A RETRIEVE request shall return all previously asserted roles associated with the currently
3791 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query
3792 parameter is not supported; all previously asserted roles associated with the currently
3793 connected and authenticated Client's identity are returned.

3794 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties
3795 included in the array as follows:

3796 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the
3797 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid"
3798 value.

3799 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,
3800 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed
3801 response and a duplicate entry shall not be added to the array.

3802 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored
3803 by the Server. The Server shall assign a unique "credid" value for every entry of the
3804 "roles" array.

3805 3) A DELETE request without a "credid" query parameter shall remove all entries from the
3806 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated
3807 Client's identity.

3808 4) A DELETE request with a "credid" query parameter shall remove only the entries of the
3809 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated
3810 Client's identity and where the corresponding "credid" matches the entry.

3811 NOTE The "oic.r.roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined
3812 in ISO/IEC 30118-1:2018.

3813 "oic.r.roles" Resource is defined in Table 63.

3814 **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

3815 Table 64 defines the Properties of "oic.r.roles".

3816 **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

3817 Because "oic.r.roles" shares the "oic.sec.cred" schema with "oic.r.cred", "subjectuud" is a required Property. However,
3818 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property
3819 is contained in an UPDATE request to the "/oic/sec/roles" Resource.

3820 13.11 Account Resource

3821 The Account Resource specifies the Properties based on IETF RFC 6749 Access Token based
3822 account creation. The mechanism to obtain credentials is described in clause 7.5. The Account
3823 Resource is used for Device Registration. The Account Resource is instantiated on the OCF
3824 Cloud as "oic/sec/account" SVR and is used by cloud-enabled Devices to register with the OCF
3825 Cloud. It should be only accessible on a secure channel; non-secure channel should not be able
3826 access this Resource.

3827 During the Device Registration process, an OCF Cloud can provide a distinct URI of another OCF
3828 Cloud ("redirected-to" OCF Cloud). Both initial and redirected-to OCF Clouds are expected to
3829 belong to the same Vendor; they are assumed to have the same UUID and are assumed to have
3830 an out-of-band communication mechanism established. Device does not have to perform the
3831 Device Registration on the redirected-to OCF Cloud and the OCF Cloud may ignore such

3832 attempts. Redirected-to OCF Cloud is expected to accept the Access Token, provided to the
 3833 Device by the initial OCF Cloud.

3834 The "di", "uid", "refresh token" and "access token" Properties of the Account Resource should be
 3835 securely stored as described in clause 15.

3836 The RETRIEVE operation on OCF Cloud's "/oic/sec/account" Resource is not allowed and the
 3837 OCF Cloud is expected to reject all attempts to perform such operation.

3838 The UPDATE operation on the OCF Cloud's "/oic/sec/account" Resource behaves as follows:

3839 – A Device intending to register with the OCF Cloud shall send UPDATE with following
 3840 Properties "di" ("di" Property Value of "/oic/d" Resource), and "access token" as configured by
 3841 the Mediator ("at" Property Value of "oic.r.coapcloudconf" Resource). The OCF Cloud verifies
 3842 it is the same "access token" which was assigned to the Mediator for the corresponding "di"
 3843 Property Value. The "access token" is the permission for the Device to access the OCF Cloud.
 3844 If the "apn" was included when the Mediator UPDATED the "oic.r.coapcloudconf" Resource,
 3845 the Device shall also include "auth provider" Property when registering with the OCF Cloud. If
 3846 no "apn" is specified, then the "auth provider" Property shall not be included in the UPDATE
 3847 request.

3848 – OCF Cloud returns "access token", "uid", "refresh token", "expires in" It may also return
 3849 "redirect uri". Received "access token" is to be treated by Device as an Access Token with
 3850 "Bearer" token type as defined in IETF RFC 6750. This "access token" shall be used for the
 3851 following Account Session start using "oic/sec/session" SVR. Received "refresh token" is to be
 3852 treated by Device as a Refresh Token as defined in IETF RFC 6749. The Device stores the
 3853 OCF Cloud's Response values. If "redirect uri" is received, Device shall use received value as
 3854 a new OCF Cloud URI instead of "cis" Property Value of "oic.r.coapcloudconf" Resource for
 3855 further connections.

3856 The DELETE operation on the OCF Cloud's "/oic/sec/account" Resource should behave as
 3857 follows:

3858 – To deregister with the OCF Cloud, a DELETE operation shall be sent with the "access token"
 3859 and either "uid", or "di" to be deregistered with the OCF Cloud. On DELETE with the OCF
 3860 Cloud, the Device should also delete values internally stored. Once deregister with an OCF
 3861 Cloud, Device can connect to any other OCF Cloud. Device deregistered need to go through
 3862 the steps in 7.5 again to be registered with the OCF Cloud.

3863 "oic.r.account" Resource is defined in Table 65.

3864 **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

3865 Table 66 defines the Properties of "oic.r.account".

3866 **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.

3867 **13.12 Account Session Resource**

3868 The "/oic/sec/session" Resource hosted on the OCF Cloud is used for creating connections with
3869 the OCF Cloud subsequent to Device registration though "/oic/sec/account" Resource. The
3870 "/oic/sec/session" Resource requires the device ID, User ID and Access Token which are stored
3871 securely on the Device.

3872 The "/oic/sec/session" Resource is exposed by the OCF Cloud. It should be only accessible on a
3873 secure channel; non-secure channel cannot access this Resource.

3874 The RETRIEVE operation on OCF Cloud's "/oic/sec/session" Resource is not allowed and the
3875 OCF Cloud is expected to reject all attempts to perform such operation.

3876 The UPDATE operation is defined as follows for OCF Cloud's "/oic/sec/session" Resource:

- 3877 – The Device connecting to the OCF Cloud shall send an UPDATE request message to the OCF
3878 Cloud's "/oic/sec/session" Resource. The message shall include the "di" Property Value of
3879 "/oic/d" Resource and "uid", "login" Value ("true" to establish connection; "false" to disconnect)
3880 and "accesstoken" as returned by OCF Cloud during Device Registration. The OCF Cloud
3881 verifies it is the same Access Token which was returned to the Device during Device
3882 Registration process. If Device was attempting to establish the connection and provided
3883 values were verified as correct by the OCF Cloud, OCF Cloud sends a response with
3884 remaining lifetime of the associated Access Token ("expiresin" Property Value).

3885 "/oic.r.session" Resource is defined in Table 67.

3886 **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

3887 Table 68 defines the Properties of "oic.r.session".

3888 **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

3889 **13.13 Account Token Refresh Resource**

3890 The "/oic/sec/tokenrefresh" Resource is used by the Device for refreshing the Access Token.

3891 The "/oic/sec/tokenrefresh" Resource is hosted by the OCF Cloud. It should be only accessible
3892 on a secure channel; non-secure channel cannot access this Resource.

3893 The Device should use "/oic/sec/tokenrefresh" to refresh the Access Token with the OCF Cloud,
3894 when the time specified in "expiresin" is near.

3895 The RETRIEVE operation on OCF Cloud's "/oic/sec/ tokenrefresh" Resource is not allowed and
3896 the OCF Cloud is expected to reject all attempts to perform such operation.

3897 The UPDATE operation is defined as follows for "/oic/sec/tokenrefresh" Resource

- 3898 – The Device attempting to refresh the Access Token shall send an UPDATE request message
3899 to the OCF Cloud's "/oic/sec/tokenrefresh" Resource. The message shall include the "di"
3900 Property Value of "/oic/d" Resource, "uid" and "refreshtoken", as returned by OCF Cloud.
- 3901 – OCF Cloud response is expected to include a "refreshtoken", new "acesstoken", and
3902 "expiresin". Received "acesstoken" is to be treated by Device as an Access Token with
3903 "Bearer" token type as defined in IETF RFC 6750. This Access Token is the permission for
3904 the Device to access the OCF Cloud. Received "refreshtoken" is to be treated by Device as a
3905 Refresh Token as defined in IETF RFC 6749. Received "refreshtoken" may be the new
3906 Refresh Token or the same one as provided by the Device in the UPDATE request. In case
3907 when new distinct "refreshtoken" is provided by the OCF Cloud, the Device shall discard the
3908 old value. The OCF Cloud's response values "refreshtoken", "acesstoken" and "expiresin" are
3909 securely stored on the Device.

3910 "/oic.r.tokenrefresh" Resource is defined in Table 69.

3911 **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

3912 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)

3914 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3915 The SVRs expose the security-related Properties of the Device.

3916 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to
3917 unauthenticated (anonymous) Clients could create privacy or security concerns.

3918 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for
3919 the "oic.r.doxm" Resource to anonymous requesters, so that the Device can be discovered and
3920 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the "oic.r.doxm"
3921 Resource to anonymous requesters, to preserve privacy.

3922 **13.15 SVRs, Discoverability and OCF Endpoints**

3923 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy
3924 Parameter clause 7.8.2.1.2).

3925 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS)
3926 (reference ISO/IEC 30118-1:2018, clause 10).

3927 The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM
3928 (reference ISO/IEC 30118-1:2018, clause 10).

3929 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

3930 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

3931 Unique identifiers are a privacy consideration due to their potential for being used as a tracking
3932 mechanism. These include the following Resources and Properties:

- 3933 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3934 – "/oic/p" Resource containing the "pi" Property.
- 3935 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3936 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous
3937 requestors. This implies any Client Device, including those with malicious intent, are able to
3938 reliably obtain identifiers useful for building a log of activity correlated with a specific Platform
3939 and Device.

3940 There are two strategies for privacy protection of Devices:

- 3941 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers
3942 2) Limit identifier persistence to make it impractical for tracking use.
3943 Both techniques can be used effectively together to limit exposure to privacy attacks.

- 3944 1) A Platform / Device manufacturer should specify a default ACL policy that restricts
3945 anonymous requestors from accessing unique identifiers. An OCF Security Domain owner
3946 should modify the ACL policy to grant access to authenticated Devices who, presumably, do
3947 not present a privacy threat.
- 3948 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the
3949 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and
3950 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated
3951 identifiers shall be:
- 3952 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
 - 3953 b) Generated by a function that is pre-image resistant, second pre-image resistant and
3954 collision resistant

3955 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier
3956 used to begin the onboarding process. However, attackers could obtain the value too and use it
3957 for Device tracking throughout the Device's lifetime.

3958 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the
3959 deviceuuid Property of the "/oic/sec/doxm" Resource to unauthenticated "/oic/res" and
3960 "/oic/sec/doxm" Resource RETRIEVE requests when the devowneruuid Property of
3961 "/oic/sec/doxm" Resource is the nil-UUID. The Server shall expose a new temporary non-
3962 repeated deviceuuid Property of the "/oic/sec/doxm" Resource when the device state transitions
3963 to RESET. This ensures the deviceuuid Property of the "/oic/sec/doxm" cannot be used to track
3964 across multiple owners.

3965 The devowneruuid Property of "/oic/sec/doxm" Resource is initialized to the nil-UUID upon
3966 entering RESET; which is retained until being set to a non-nil-UUID value during RFOTM device
3967 state. The device shall supply a temporary, non-repeated deviceuuid Property of "/oic/sec/doxm"
3968 Resource to RETRIEVE requests on "/oic/sec/doxm" and "/oic/res" Resources while
3969 devowneruuid Property of "/oic/sec/doxm" Resource is the nil-UUID. During the OTM process the
3970 DOTS shall UPDATE devowneruuid Property of the "/oic/sec/doxm" Resource to a non-nil UUID
3971 value which is the trigger for the Device to expose its persistent or semi-persistent device
3972 identifier. Therefore, the Device shall supply deviceuuid Property of "/oic/sec/doxm" Resource in
3973 response to RETRIEVE requests while the devowneruuid Property of the "/oic/sec/doxm"
3974 Resource is a non-nil-UUID value.

3975 The DOTS or AMS may also provision an ACL policy that restricts access to the "/oic/sec/doxm"
3976 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent
3977 device identifier via the deviceuuid Property value of the "/oic/sec/doxm" Resource.

3978 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent
3979 or semi-persistent identifier using the "/oic/sec/cred" Resource to first establish an authenticated
3980 connection. This is achieved by first provisioning a "/oic/sec/cred" Resource entry that contains
3981 the Server's deviceuuid Property value of the "/oic/sec/doxm" Resource.

3982 The "di" Property in the "/oic/d" Resource shall mirror that of the deviceuuid Property of the
3983 "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the
3984 "/oic/d" resource such that only authenticated Clients are able to obtain the "di" Property of
3985 "/oic/d" Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

3986 Servers should expose a temporary, non-repeated, piid Property of "/oic/p" Resource Value upon
3987 entering RESET Device state. Servers shall expose a persistent value via the "piid" Property of

3988 "/oic/p" Property when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. An ACL
 3989 policy on the "/oic/d" Resource should protect the "piid" Property of "/oic/p" Resource from being
 3990 disclosed to unauthenticated requestors.

3991 Servers shall expose a temporary, non-repeated, "pi" Property value upon entering RESET
 3992 Device state. Servers shall expose a persistent or semi-persistent platform identifier value via the
 3993 "pi" Property of the "/oic/p" Resource when onboarding sets "devowneruuid" Property to a non-
 3994 nil-UUID value. An ACL policy on the "/oic/p" Resource should protect the "pi" Property from
 3995 being disclosed to unauthenticated requestors.

3996 Table 71 depicts Core Resource Properties Access Modes given various Device States.

3997 **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.

3998 Four identifiers are thought to be privacy sensitive:

- 3999 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 4000 – "/oic/p" Resource containing the "pi" Property.
- 4001 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

4002 There are three strategies for privacy protection of Devices:

- 4003 1) Apply access control to restrict read access to Resources containing unique identifiers. This
 4004 ensures privacy sensitive identifiers do not leave the Device.
- 4005 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy
 4006 sensitive identifiers are less effective for tracking and correlation.
- 4007 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can
 4008 do so.

4009 These techniques can be used to limit exposure to privacy attacks. For example:

- 4010 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent
 4011 identifiers can be created.
- 4012 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to
 4013 facilitate onboarding.
- 4014 – Persistent and semi-persistent identifiers can be encrypted before sending them to another
 4015 Device.

4016 A temporary, non-repeated identifier shall be:

- 4017 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 4018 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision
4019 resistant

4020 NOTE This requirement is met through a vendor attestation certification mechanism.

4021 **13.16.2 Privacy Protecting the Device Identifiers**

4022 The "di" Property Value of the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of
4023 the "/oic/sec/doxm" Resource. The Device should use a new, temporary non-repeated identifier in
4024 place of the "deviceuuid" Property Value of "/oic/sec/doxm" Resource upon entering the RESET
4025 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID
4026 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value
4027 of the "/oic/sec/doxm" Resource after the DOTS sets the "devowneruuid" Property to a non-nil-
4028 UUID value. The temporary identifier should not change more frequently than once per Device
4029 state transition to RESET.

4030 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4031 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"
4032 Property values:
 - 4033 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di")
4034 Property value only if responding to an authenticated requestor and the "deviceuuid" (or
4035 "di") value is confidentiality protected .
 - 4036 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if
4037 responding to an unauthenticated requestor.
- 4038 – The AMS should provision an ACL policy on the "/oic/sec/doxm" and "/oic/d" resources to
4039 further protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

4040 See 13.2 for deviceuuid Property lifecycle requirements.

4041 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary
4042 discovery requests. This is achieved by provisioning a "/oic/sec/cred" Resource entry that contains the Server's
4043 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

4044 **13.16.3 Privacy Protecting the Protocol Independent Device Identifier**

4045 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property
4046 Value of "/oic/d" Resource upon entering the RESET Device state. If a temporary, non-repeated
4047 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID
4048 value. The Device should use its persistent "piid" Property value after the DOTS sets the
4049 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change
4050 more frequently than once per Device state transition to RESET.

4051 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4052 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
 - 4053 – The Device should include its persistent "piid" Property value only if responding to an
4054 authenticated requestor and the "piid" value is confidentiality protected.
 - 4055 – The Device should include a temporary non-repeated "piid" Property value if responding to
4056 an unauthenticated requestor.
- 4057 – The AMS should provision an ACL policy on the "/oic/d" Resource to further protect the piid
4058 Property of "/oic/p" Resource from being disclosed unnecessarily.

4059 **13.16.4 Privacy Protecting the Platform Identifier**

4060 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property
 4061 Value of the "/oic/p" Resource upon entering the RESET Device state. This value should be
 4062 exposed while the "devowneruid" Property has a nil UUID value. The Device should use its
 4063 persistent (or semi-persistent) "pi" Property value after the DOTS sets the "devowneruid"
 4064 Property to a non-nil-UUID value. The temporary identifier should not change more frequently
 4065 than once per Device state transition to RESET.

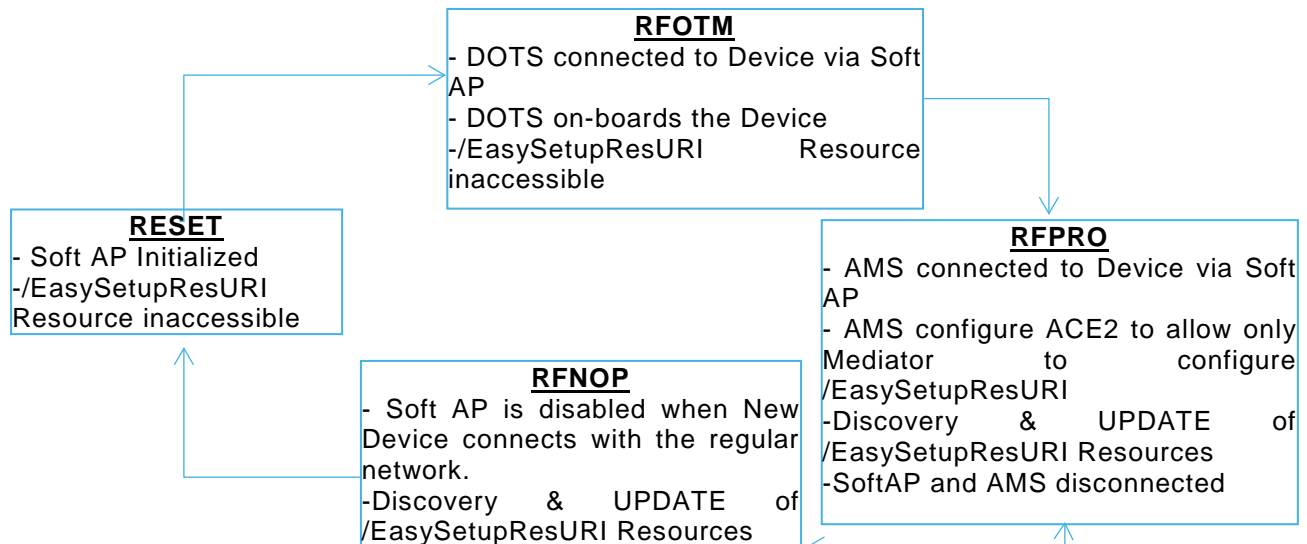
4066 Subsequent to the "devowneruid" being UPDATED to a non-nil UUID:

- 4067 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
- 4068 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
 4069 responding to an authenticated requestor and the "pi" value is confidentiality protected.
 - 4070 – The Device should include a temporary non-repeated "pi" Property value if responding to
 4071 an unauthenticated requestor.
- 4072 – The AMS should provision an ACL policy on the "/oic/p" Resource to protect the pi Property
 4073 from being disclosed unnecessarily.

4074 **13.17 Easy Setup Resource Device State**

4075 This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined
 4076 in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of
 4077 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup
 4078 Devices.

4079 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



4097 **Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states**

4098 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's
 4099 state.

4100 While it is reasonable for a user to expect that power cycling a new Device will turn on the Soft
 4101 AP for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it
 4102 is a security risk to make this the default behaviour of a device that remains unenrolled beyond a
 4103 reasonable period after first boot.

4104 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

- 4105 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes
4106 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for
4107 Easy Setup.
- 4108 – If a new Device tried and failed to complete Easy Setup Enrolment immediately following the
4109 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically
4110 for another 30 minutes upon being power cycled, provided that the power cycle occurs within
4111 3 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup
4112 Soft AP directly without a factory reset, it is not necessary to turn it back on if it was on
4113 immediately prior to power cycle, because the user obviously knows how to initiate the
4114 process manually.
- 4115 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft
4116 AP should not turn back on for Easy Setup until another factory reset occurs, or the user
4117 initiates the Easy Setup Soft AP directly.
- 4118 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new
4119 Device to connect to the Enroller.
- 4120 – The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the
4121 Enroller.
- 4122 – Once a new Device has successfully connected to the Enroller, it shall not turn the Easy
4123 Setup Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or
4124 the user initiates the Easy Setup Soft AP directly.
- 4125 – Just Works OTM shall not be enabled on Devices which support Easy Setup.
- 4126 – The Soft AP shall be secured (e.g. shall not expose an open AP).
- 4127 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase
4128 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on
4129 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.
- 4130 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the
4131 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by
4132 an attacker with knowledge of the Device type, model, manufacturer, or any other information
4133 discoverable through Device's exposed interfaces.
- 4134 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the
4135 "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the
4136 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller,
4137 for use in connecting the Enrollee to the Enroller.
- 4138 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the
4139 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.
- 4140 The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET
4141 state. After ownership transfer process is completed with the DOTS, and the Device enters in
4142 RFPRO Device state, the "/example/EasySetupResURI" may be Discoverable. The DOTS may be
4143 hosted on the Mediator Device.
- 4144 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership
4145 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used
4146 by AMS for "/oic/sec/acl2" Resource provisioning in RFPRO state. The CoAPS session
4147 authentication and encryption is already defined in the Security spec.
- 4148 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following
4149 Resources to be only configurable by the Mediator Device with permission to UPDATE or
4150 RETRIEVE access:
- 4151 – "/example/EasySetupResURI"

4152 – "/example/WifiConfResURI"
4153 – "/example/DevConfResURI"
4154 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
4155 {  
4156     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
4157     "resources": [  
4158         { "href": "/example/EasySetupResURI" },  
4159         { "href": "/example/WiFiConfResURI" },  
4160         { "href": "/example/DevConfResURI" },  
4161     ],  
4162     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
4163 }
```

4164 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to
4165 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

4166 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE
4167 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

4168 **14 Security Hardening Guidelines/ Execution Environment Security**

4169 **14.1 Preamble**

4170 This is an informative clause. Many TGs in OCF have security considerations for their protocols
4171 and environments. These security considerations are addressed through security mechanisms
4172 specified in the security documents for OCF. However, effectiveness of these mechanisms
4173 depends on security robustness of the underlying hardware and software Platform. This clause
4174 defines the components required for execution environment security.

4175 **14.2 Execution Environment Elements**

4176 **14.2.1 Execution Environment Elements General**

4177 Execution environment within a computing Device has many components. To perform security
4178 functions in a robustness manner, each of these components has to be secured as a separate
4179 dimension. For instance, an execution environment performing AES cannot be considered secure
4180 if the input path entering keys into the execution engine is not secured, even though the
4181 partitions of the CPU, performing the AES encryption, operate in isolation from other processes.
4182 Different dimensions referred to as elements of the execution environment are listed below. To
4183 qualify as a secure execution environment (SEE), the corresponding SEE element must qualify as
4184 secure.

- 4185 – (Secure) Storage
- 4186 – (Secure) Execution engine
- 4187 – (Trusted) Input/output paths
- 4188 – (Secure) Time Source/clock
- 4189 – (Random) number generator
- 4190 – (Approved) cryptographic algorithms
- 4191 – Hardware Tamper (protection)

4192 **NOTE** Software security practices (such as those covered by OWASP) are outside scope of this document, as
4193 development of secure code is a practice to be followed by the open source development community. This document
4194 will however address the underlying Platform assistance required for executing software. Examples are secure boot
4195 and secure software upgrade.

4196 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,
4197 14.2.7.

4198 **14.2.2 Secure Storage**

4199 **14.2.2.1 Secure Storage General**

4200 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive
4201 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,
4202 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive
4203 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both
4204 its integrity and confidentiality be maintained.

4205 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive
4206 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either
4207 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication
4208 and encryption, it must maintain its integrity against intentional or accidental alteration.

4209 A partial list of Sensitive Data is outlined in Table 72:

Table 72 – Examples of Sensitive Data

Data	Integrity protection	Confidentiality protection
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
OCF Cloud URL	Yes	Not required
OCF Cloud Identity	Yes	Not required
Access Token	Yes	Yes

4211 Exact method of protection for secure storage is implementation specific, but typically
4212 combinations of hardware and software methods are used.

4213 **14.2.2.2 Hardware Secure Storage**

4214 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric
4215 and asymmetric private keys, access credentials, and personal private data. Hardware secure
4216 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes
4217 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

4218 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides
4219 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or
4220 electronic attacks. It is not necessary to prevent the attacks themselves, but an attempted attack
4221 should not result in an unauthorized entity successfully retrieving Sensitive Data.

4222 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data
4223 from attacks that include but are not limited to:

- 4224 1) Physical decapping of chip packages to optically read NVRAM contents
- 4225 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 4226 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit
4227 patterns of Critical Sensitive Data
- 4228 4) Use of malicious software or firmware to read memory contents at rest or in transit within a
4229 microcontroller
- 4230 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

4231 **14.2.2.3 Software Storage**

4232 It is generally NOT recommended to rely solely on software and unsecured memory to store
4233 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and
4234 encryption keys should be housed in hardware secure storage whenever possible.

4235 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable
4236 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

4237 **14.2.2.4 Additional Security Guidelines and Best Practices**

4238 Some general practices that can help ensure that Sensitive Data is not compromised by various
4239 forms of security attacks:

- 4240 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG
4241 used for authentication challenges can substantially degrade security strength. For this
4242 reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source
4243 be used for all authentication challenges.
- 4244 2) Secure download and boot – To prevent the loading and execution of malicious software,
4245 where it is practical, it is recommended that Secure Download and Secure Boot methods that
4246 authenticate a binary's source as well as its contents be used.
- 4247 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered
4248 unsecure and shall not be used for any security-related function:
 - 4249 a) SHA-1
 - 4250 b) MD5
 - 4251 c) RC4
 - 4252 d) RSA 1024
- 4253 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is
4254 stored in Secure Storage, any use of that data that requires its transmission out of that
4255 Secure Storage should be encrypted to prevent eavesdropping by malicious software within
4256 an MCU/MPU.
- 4257 5) It is recommended to avoid using wildcard in Subject Id ("*"), when setting up "oic.r.cred"
4258 Resource entries, since this opens up an identity spoofing opportunity.
- 4259 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device
4260 meets security requirements for its intended uses. As an example, IoTivity is a reference
4261 implementation intended to be used as a basis for a product, but IoTivity has not undergone
4262 3rd party security review, penetration testing, etc. Any Device based on IoTivity should
4263 undergo appropriate penetration testing and security review prior to sale or deployment.
- 4264 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to
4265 consumers. Changes should be made to a public and accessible website. Expectations
4266 should be clear as to what will be supported and for how long the Device vendor expects to
4267 support security updates to the software, operating system, drivers, networking, firmware and
4268 hardware of the device.
- 4269 8) Device vendor has not implemented test or debug interfaces on the Device which are
4270 operable or which can be enabled which might present an attack vector on the Device which
4271 circumvents the interface-level security or access policies of the Device.
- 4272 9) Device vendor understands that if an application running on the Device has access to
4273 cryptographic elements such as the private keys or Ownership Credential, then those
4274 elements have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or
4275 a Device with access to the Internet beyond the local network, the execution of critical
4276 functions should take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4277 10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management
4278 or access, or other security-critical function, should be sufficiently unique (do not duplicate
4279 passphrases. The creation of these passphrases or PINS should not be algorithmically
4280 deterministic nor should they use insufficient entropy in their creation.
- 4281 11) Ensure that there are no remaining "VENDOR_TODO" items in the source code.

4282 12) If the implementation of this document uses the "Just Works" onboarding method, understand
4283 that there is a man-in-the-middle vulnerability during the onboarding process where a
4284 malicious party could intercept messages between the device being onboarded and the OBT
4285 and could persist, acting as an intermediary with access to message traffic, during the lifetime
4286 of that onboarded device. The recommended best practice would be to use an alternate
4287 ownership transfer method (OTM) instead of "Just Works".

4288 13) It is recommended that at least one static and dynamic analysis tool¹ be applied to any
4289 proposed major production release of the software before its release, and any vulnerabilities
4290 resolved.

4291 14) To avoid a malicious device being able to covertly join an OCF Security Domain,
4292 implementers of any OBT may eliminate completely autonomous sequences where a device
4293 is brought into the OCF Security Domain without any authorization by the owner. Consider
4294 either including a confirmation with the OCF Security Domain owner/operator (e.g. "Do you
4295 want to add 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'? Yes/No/Cancel?") or a
4296 confirmation with a security policy (e.g. an enterprise policy where the OCF Security Domain
4297 admin can bulk-onboard devices).

4298 **14.2.3 Secure execution engine**

4299 Execution engine is the part of computing Platform that processes security functions, such as
4300 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine
4301 requires the following

4302 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This
4303 includes isolation of CPU caches, and all of execution elements that needed to be considered
4304 as part of trusted (crypto) boundary.

4305 – Isolation of data paths into and out of execution engine. For instance, both unencrypted but
4306 sensitive data prior to encryption or after decryption, or cryptographic keys used for
4307 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

4308 **14.2.4 Trusted input/output paths**

4309 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be
4310 protected. This includes paths into and out secure execution engine and secure memory.

4311 Path protection can be both hardware based (e.g. use of a privileged bus) or software based
4312 (using encryption over an untrusted bus).

4313 **14.2.5 Secure clock**

4314 Many security functions depend on time-sensitive credentials. Examples are time stamped
4315 Kerberos tickets, OAUTH tokens, X.509 certificates, OSCP response, software upgrades, etc.
4316 Lack of secure source of clock can mean an attacker can modify the system clock and fool the
4317 validation mechanism. Thus an SEE needs to provide a secure source of time that is protected
4318 from tampering. Trustworthiness from security robustness standpoint is not the same as accuracy.
4319 Protocols such as NTP can provide rather accurate time sources from the network, but are not
4320 immune to attacks. A secure time source on the other hand can be off by seconds or minutes
4321 depending on the time-sensitivity of the corresponding security mechanism. Secure time source
4322 can be external as long as it is signed by a trusted source and the signature validation in the
4323 local Device is a trusted process (e.g. backed by secure boot).

4324 **14.2.6 Approved algorithms**

4325 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and
4326 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by
4327 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only

¹ A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4328 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic
4329 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are
4330 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms
4331 (even if they deemed stronger by some parties) must be considered non-approved.

4332 The set of algorithms to be considered for approval are algorithms for

- 4333 – Hash functions
- 4334 – Signature algorithms
- 4335 – Encryption algorithms
- 4336 – Key exchange algorithms
- 4337 – Pseudo Random functions (PRF) used for key derivation

4338 This list will be included in this or a separate security robustness rules document and must be
4339 followed for all security specifications within OCF.

4340 **14.2.7 Hardware tamper protection**

4341 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not
4342 requirements) regarding tamper protection for cryptographic module

- 4343 – Production-grade (lowest level): this means components that include conformal sealing
4344 coating applied over the module's circuitry to protect against environmental or other physical
4345 damage. This does not however require zeroization of secret material during physical
4346 maintenance. This definition is borrowed from FIPS 140-2 security level 1.
- 4347 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,
4348 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from
4349 FIPS 140-2 security level 2.
- 4350 – Tamper resistance (highest level), this means there is a response to physical tempering that
4351 typically includes zeroization of sensitive material on the module. This definition is borrowed
4352 from FIPS 140-2 security level 3.

4353 It is difficult of specify quantitative certification test cases for accreditation of these levels.
4354 Content protection regimes usually talk about different tools (widely available, specialized and
4355 professional tools) used to circumvent the hardware protections put in place by manufacturing. If
4356 needed, OCF can follow that model, if and when OCF engage in distributing sensitive key
4357 material (e.g. PKI) to its members.

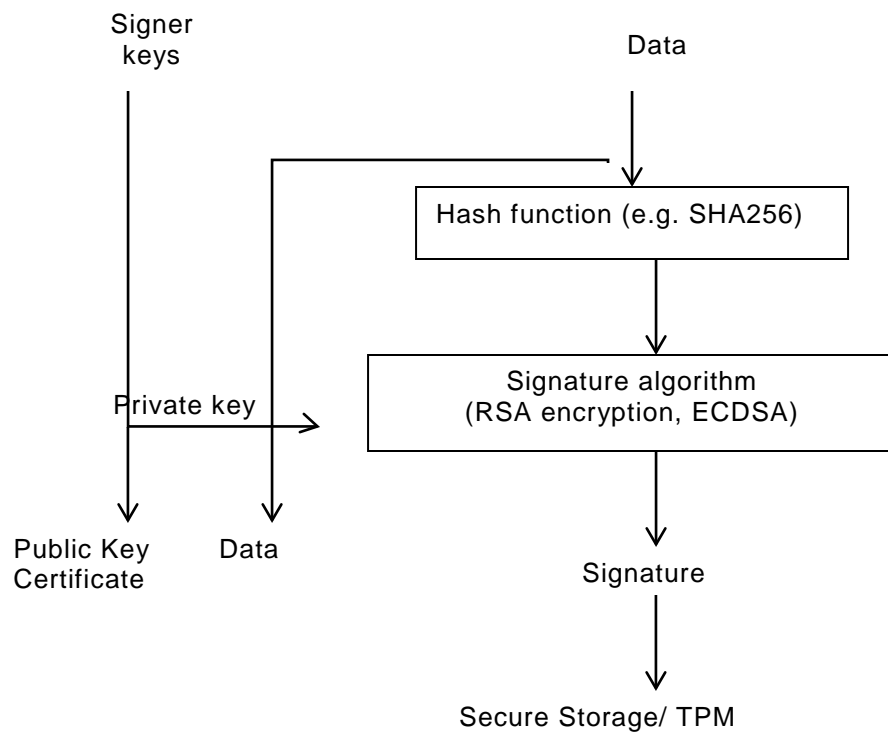
4358 **14.3 Secure Boot**

4359 **14.3.1 Concept of software module authentication**

4360 In order to ensure that all components of a Device are operating properly and have not been
4361 tampered with, it is best to ensure that the Device is booted properly. There may be multiple
4362 stages of boot. The end result is an application running on top an operating system that takes
4363 advantage of memory, CPU and peripherals through drivers.

4364 The general concept is that each software module is invoked only after cryptographic integrity
4365 verification is complete. The integrity verification relies on the software module having been
4366 hashed (e.g. SHA_1, SHA_256) and then signed with a cryptographic signature algorithm with
4367 (e.g. RSA), with a key that only a signing authority has access to.

4368 Figure 40 depicts software module authentication.

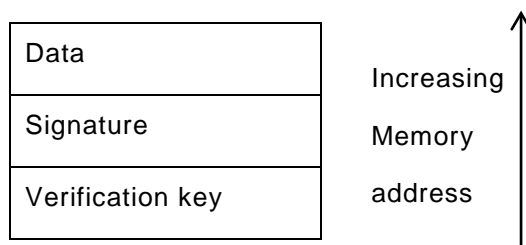


4369

Figure 40 – Software Module Authentication

4370 After the data is signed with the signer’s signing key (a private key), the verification key (the
 4371 public key corresponding to the private signing key) is provided for later verification. For lower
 4372 level software modules, such as bootloaders, the signatures and verification keys are inserted
 4373 inside tamper proof memory, such as one-time programmable memory or TPM. For higher level
 4374 software modules, such as application software, the signing is typically performed according to
 4375 the PKCS#7 format IETF RFC 2315, where the signedData format includes both indications for
 4376 signature algorithm, hash algorithm as well as the signature verification key (or certificate).
 4377 Secure boot does not require use of PKCS#7 format.

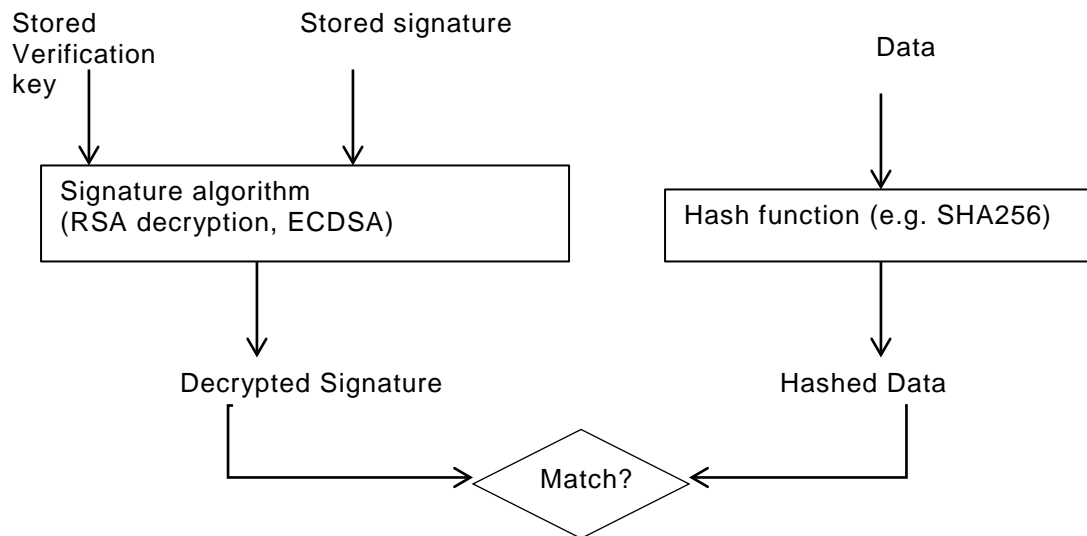
4378 Figure 41 depicts verification software module.



4379

Figure 41 – Verification Software Module

4380 As shown in Figure 42. the verification module first decrypts the signature with the verification
 4381 key (public key of the signer). The verification module also calculates a hash of the data and then
 4382 compares the decrypted signature (the original) with the hash of data (actual) and if the two
 4383 values match, the software module is authentic.



4384 **Figure 42 – Software Module Authenticity**

4384

4385 **14.3.2 Secure Boot process**

4386 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/
 4387 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to
 4388 find out where the boot code is and then run the boot code (second-stage boot loader). The
 4389 second stage bootloader is typically the process that loads the operating system (Kernel) and
 4390 transfers the execution to the where the Kernel code is. Once the Kernel starts, it may load
 4391 external Kernel modules and drivers.

4392 When performing a secure boot, it is required that the integrity of each boot loader is verified
 4393 before executing the boot loader stage. As mentioned, while the signature and verification key for
 4394 the lowest level bootloader is typically stored in tamper-proof memory, the signature and
 4395 verification key for higher levels should be embedded (but attached in an easily accessible
 4396 manner) in the data structures software.

4397 **14.3.3 Robustness Requirements**

4398 **14.3.3.1 Robustness General**

4399 To qualify as high robustness secure boot process, the signature and hash algorithms shall be
 4400 one of the approved algorithms, the signature values and the keys used for verification shall be
 4401 stored in secure storage and the algorithms shall run inside a secure execution environment and
 4402 the keys shall be provided the SEE over trusted path.

4403 **14.3.3.2 Next steps**

4404 Develop a list of approved algorithms and data formats

4405 **14.4 Attestation**

4406 **14.5 Software Update**

4407 **14.5.1 Overview:**

4408 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;
 4409 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and
 4410 end-of-life stages for the Device remain outstanding. It is possible for the Device to require

4411 update during any of these stages, although the most likely times are during onboarding, regular
 4412 operation and maintenance. The aspects of the software include, but are not limited to, firmware,
 4413 operating system, networking stack, application code, drivers, etc.

4414 **14.5.2 Recognition of Current Differences**

4415 Different manufacturers approach software update utilizing a collection of tools and strategies:
 4416 over-the-air or wired USB connections, full or partial replacement of existing software, signed and
 4417 verified code, attestation of the delivery package, verification of the source of the code, package
 4418 structures for the software, etc.

4419 It is recommended that manufacturers review their processes and technologies for compliance
 4420 with industry best-practices that a thorough security review of these takes place and that periodic
 4421 review continue after the initial architecture has been established.

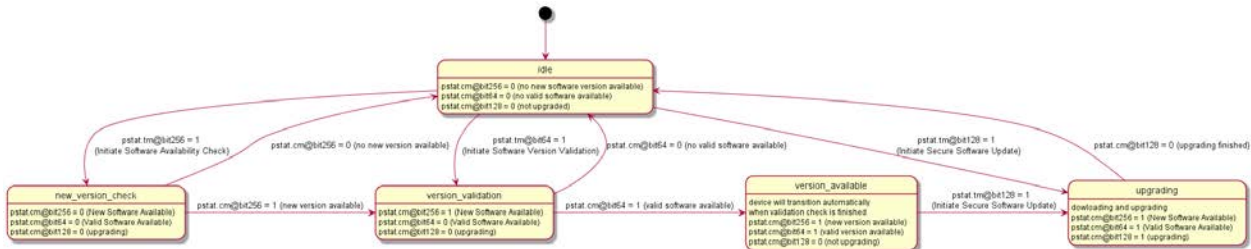
4422 This document applies to software updates as recommended to be implemented by OCF Devices;
 4423 it does not have any bearing on the above-mentioned alternative proprietary software update
 4424 mechanisms. The described steps are being triggered by an OCF Client, the actual
 4425 implementation of the steps and how the software package is downloaded and upgraded is
 4426 vendor specific.

4427 The triggers that can be invoked from OCF clients can perform:

- 4428 1) Check if new software is available
- 4429 2) Download and verify the integrity of the software package
- 4430 3) Install the verified software package

4431 The triggers are not sequenced, each trigger can be invoked individually.

4432 The state of the transitions of software update is in Figure 43.



4433
 4434 **Figure 43 – State transitioning diagram for software download**

4435
 4436 **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

4437
 4438 **14.5.2.1 Checking availability of new software**

4439 Setting the Initiate Software Availability Check bit in the "/oic/sec/pstat.tm" Property (see
 4440 Table 54 of clause 13.8) indicates a request to initiate the process to check if new software is

4441 available, e.g. the process whereby the Device checks if a newer software version is available on
4442 the external endpoint. Once the Device has determined if an newer software version is available,
4443 it sets the Initiate Software Availability Check bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE),
4444 indicating that new software is available or to 0 (FALSE) if no newer software version is available,
4445 See also Table 73 where the bits in property TM indicates that the action is initiated and the CM
4446 bits are indicating the result of the action. The Device receiving this trigger is not downloading
4447 and not validating the software to determine if new software is available. The version check is
4448 determined by the current software version and the software version on the external endpoint.
4449 The determination if a software package is newer is vendor defined.

4450 **14.5.3 Software Version Validation**

4451 Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see
4452 Table 54 defines the Properties of "oic.r.pstat".

4453 Table 54 of 13.8) indicates a request to initiate the software version validation process, the
4454 process whereby the Device validates the software (including firmware, operating system, Device
4455 drivers, networking stack, etc.) against a trusted source to see if, at the conclusion of the check,
4456 the software update process will need to be triggered (see clause 14.5.4). When the Initiate
4457 Software Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged
4458 Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and
4459 initiates a software version check. Once the Device has determined if a valid software is available,
4460 it sets the Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE)
4461 if an update is available or 0 (FALSE) if no update is available. To signal completion of the
4462 Software Version Validation process, the Device sets the Initiate Software Version Validation bit
4463 in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation
4464 bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation
4465 process. The Software Version Validation process can download the software from the external
4466 endpoint to verify the integrity of the software package.

4467 **14.5.4 Software Update**

4468 Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 54 of
4469 clause 13.8) indicates a request to initiate the software update process. When the Initiate Secure
4470 Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the
4471 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a
4472 software update process. Once the Device has completed the software update process, it sets
4473 the Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if/when
4474 the software was successfully updated or 0 (FALSE) if no update was performed. To signal
4475 completion of the Secure Software Update process, the Device sets the Initiate Secure Software
4476 Update bit in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure Software
4477 Update bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the update
4478 process.

4479 **14.5.4.1 State of Device after software update**

4480 The state of all resources implemented in the Device should be the same as after boot, meaning
4481 that the software update is not resetting user data and retaining a correct state.

4482 User data of a Device is defined as:

- 4483 – Retain the SVR states, e.g. the on boarded state, registered clients.
- 4484 – Retain all created resources
- 4485 – Retain all stored data of a resource
- 4486 – For example the preferences stored for the brewing resource ("oic.r.brewing").

4487 **14.5.5 Recommended Usage**

4488 The Initiate Secure Software Update bit of "/oic/sec/pstat.tm" should only be set by a Client after
4489 the Initiate Software Version Validation check is complete.

4490 The process of updating Device software may involve state changes that affect the Device
4491 Operational State ("/oic/sec/pstat.dos"). Devices with an interest in the Device(s) being updated
4492 should monitor "/oic/sec/pstat.dos" and be prepared for pending software update(s) to affect
4493 Device state(s) prior to completion of the update.

4494 The Device itself may indicate that it is autonomously initiating a software version check/update
4495 or that a check/update is complete by setting the "pstat.tm" and "pstat.cm" Initiate Software
4496 Version Validation and Secure Software Update bits when starting or completing the version
4497 check or update process. As is the case with a Client-initiated update, Clients can be notified that
4498 an autonomous version check or software update is pending and/or complete by observing pstat
4499 resource changes.

4500 The "oic.r.softwareupdate" Resource Type specifies additional features to control the software
4501 update process see core specification.

4502 **14.6 Non-OCF Endpoint interoperability**

4503 **14.7 Security Levels**

4504 Security Levels are a way to differentiate Devices based on their security criteria. This need for
4505 differentiation is based on the requirements from different verticals such as industrial and health
4506 care and may extend into smart home. This differentiation is distinct from Device classification
4507 (e.g. IETF RFC 7228)

4508 These categories of security differentiation may include, but is not limited to:

- 4509 1) Security Hardening
- 4510 2) Identity Attestation
- 4511 3) Certificate/Trust
- 4512 4) Onboarding Technique
- 4513 5) Regulatory Compliance
 - 4514 a) Data at rest
 - 4515 b) Data in transit
- 4516 6) Cipher Suites – Crypto Algorithms & Curves
- 4517 7) Key Length
- 4518 8) Secure Boot/Update

4519 In the future security levels can be used to define interoperability.

4520 The following applies to the OCF Security Specification 1.1:

4521 The current document does not define any other level beyond Security Level 0. All Devices will
4522 be designated as Level 0. Future versions may define additional levels.

4523 Additional comments:

- 4524 – The definition of a given security level will remain unchanged between versions of the
4525 document.
- 4526 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.

4527 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of
4528 the higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and
4529 a later document version defines a security level 1, the Device could be evaluated and
4530 classified as level 1 if it meets level 1 requirements).

4531 – The security levels may need to be visible to the end user.

4532 **14.8 Security Profiles**

4533 **14.8.1 Security Profiles General**

4534 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This
4535 need for differentiation is based on the requirements from different verticals such as industrial
4536 and health care and may extend into smart home. This differentiation is distinct from device
4537 classification (e.g. IETF RFC 7228)

4538 These categories of security differentiation may include, but is not limited to:

- 4539 1) Security Hardening and assurances criteria
- 4540 2) Identity Attestation
- 4541 3) Certificate/Trust
- 4542 4) Onboarding Technique
- 4543 5) Regulatory Compliance
 - 4544 a) Data at rest
 - 4545 b) Data in transit
- 4546 6) Cipher Suites – Crypto Algorithms & Curves
- 4547 7) Key Length
- 4548 8) Secure Boot/Update

4549 Each Security Profile definition must specify the version or versions of the OCF Security
4550 Specification(s) that form a baseline set of normative requirements. The profile definition may
4551 include security requirements that supersede baseline requirements (not to relax security
4552 requirements).

4553 Security Profiles have the following properties:

- 4554 – A given profile definition is not specific to the version of the document that defines it. For
4555 example, the profile may remain constant for subsequent OCF Security Specification versions.
- 4556 – A specific OCF Device and platform combination may be used to satisfy the security profile.
- 4557 – Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles
4558 simultaneously.
- 4559 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found
4560 to satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the
4561 document, and a later document version defines a security profile Black, the device could be
4562 evaluated and classified as profile Black if it meets profile Black requirements).
- 4563 – A machine-readable representation of compliance results specifically describing profiles
4564 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or
4565 manifest may contain security profiles attributes).

4566 **14.8.2 Identification of Security Profiles (Normative)**

4567 **14.8.2.1 Security Profiles in Prior Documents**

4568 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles
4569 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in

4570 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use
 4571 the OCF Security Specification version to characterize expected security behaviour.

4572 **14.8.2.2 Security Profile Resource Definition**

4573 The "oic.sec.sp" Resource is used by the OCF Device to show which OCF Security Profiles the
 4574 OCF Device is capable of supporting and which are authorized for use by the OCF Security
 4575 Domain owner. Properties of the Resource identify which OCF Security Profile is currently
 4576 operational. The ocfSecurityProfileOID value type shall represent OID values and may reference
 4577 an entry in the form of strings (UTF-8).

4578 "oic.sec.sp" Resource is defined in Table 74.

4579 **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

4580 Table 75 defines the Properties of "oic.sec.sp".

4581 **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")

4582 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or
 4583 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4584 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
 4585 private(4) enterprise(1) OCF(51414) }

4586
 4587 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4588
 4589 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4590
 4591 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }

4592 --The Security Profile is not specified

4593 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }

4594 --This specifies the OCF Baseline Security Profile(s)

4595 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }

4596 --This specifies the OCF Black Security Profile(s)

4597 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }

4598 --This specified the OCF Blue Security Profile(s)

4599 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }

4600 --This specifies the OCF Purple Security Profile(s)

4601
 4602 --versioned Security Profiles

4603 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)

4604 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4605 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}

4606 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4607 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}

```
4608     --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"
4609     sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
4610     --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"
4611     sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
4612     --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"
4613
4614     ocfSecurityProfileOID ::= UTF8String
4615
```

4616 **14.8.3 Security Profiles**

4617 **14.8.3.1 Security Profiles General**

4618 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to
4619 the Security Profile clauses for additional details).

4620 The OCF Conformance criteria may require vendor attestation that establishes the expected
4621 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific
4622 requirements).

4623 **14.8.3.2 Security Profile Unspecified (sp-undefined-v0)**

4624 The Security Profile "sp-undefined-v0" is reserved for future use.

4625 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4626 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where
4627 the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the
4628 "supportedprofiles" Property of the "/oic/sec/sp" Resource.

4629 It indicates the OCF Device satisfies the normative security requirements for this document.

4630 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-
4631 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other
4632 profiles.

4633 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to
4634 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

4635 **14.8.3.4 Security Profile Black (sp-black-v0)**

4636 **14.8.3.4.1 Black Profile General**

4637 The need for Security Profile Black v0 is to support devices and manufacturers who wish to
4638 certify their devices meeting this specific set of security criteria. A Device may satisfy the Black
4639 requirements as well as requirements of other profiles, the Black Security Profile is not
4640 necessarily mutually exclusive with other Security Profiles unless those requirements conflict with
4641 the explicit requirements of the Black Security Profile.

4642 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4643 Security Profile Black devices could include any device a manufacturer wishes to certify at this
4644 profile, but healthcare devices and industrial devices with additional security requirements are
4645 the initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or
4646 devices with exceptional profiles of trust bestowed upon them, may wish to certify at this profile;
4647 these types of devices may include, but are not limited to the following:

- 4648 – Bridges (Mapping devices between ecosystems handling virtual devices from different
4649 ecosystems)
- 4650 – Resource Directories (Devices trusted to manage OCF Security Domain resources)
- 4651 – Remote Access (Devices which have external access but can also act within the OCF
4652 Security Domain)

- 4653 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)
- 4654 – Industrial Devices (Devices with advanced management, security and attestation
- 4655 requirements)

4656 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4657 Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security
4658 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4659 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4660 – Support for AES 128 encryption for data at rest and in transit.
- 4661 – Hardening minimums: manufacturer assertion of secure credential storage
- 4662 – In 13) in enumerated item #10 "The "/oic/sec/cred" Resource should contain credential(s) if
4663 required by the selected OTM" is changed to require the credential be stored: "The
4664 "/oic/sec/cred" Resource shall contain credential(s)."
- 4665 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4666 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by
4667 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4668 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-
4669 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4670 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to
4671 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4672 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework
4673 described in the supporting documents:

- 4674 – Certificate Profile (See 9.4.2)
- 4675 – Certificate Policy (see Certificate Policy document:
4676 <https://openconnectivity.org/specs/OCF%20Certificate%20Policy.pdf>)

4677 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4678 **14.8.3.5.1 Blue Profile General**

4679 The Security Profile Blue is used when manufacturers issue platform certificates for platforms
4680 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is
4681 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF
4682 Security Domain owners evaluate manufacturer supplied certificates and attributed data to
4683 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.
4684 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may
4685 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4686 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting
4687 Criteria defined by OCF.

4688 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4689 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from
4690 OCF Device vendor and where platform vendors may implement trusted platforms that may
4691 conform to industry standards defining trusted platforms. The OCF Security Profile Blue specifies
4692 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance
4693 criteria produced by OCF conformance operations. The OCF Security Domain owner evaluates
4694 these data when an OCF Device is onboarded into the OCF Security Domain. Based on this
4695 evaluation the OCF Security Domain owner determines which Security Profile may be applied
4696 during OCF Device operation. All OCF Device types may be considered for evaluation using the
4697 OCF Security Profile Blue.

4698 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4699 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for
4700 this document version are satisfied and the following additional criteria are satisfied.

4701 OCF Blue profile defines the following OCF Device quality assurances:

- 4702 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF
4703 Device was hosted on one or more platforms that satisfies OCF Blue platform security
4704 assurances and platform security and privacy functionality requirements.
- 4705 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF
4706 and published by OCF in a machine readable format.
- 4707 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned
4708 signing key.
- 4709 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4710 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by
4711 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4712 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in
4713 its certificate.
- 4714 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF
4715 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's
4716 "securityprofiles" field.

4717 OCF Blue profile defines the following OCF Device security functionality:

- 4718 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage
4719 functions are hardened by the platform.
- 4720 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials
4721 using the "/oic/sec/cred" Resource where the "credusage" Property contains the value
4722 "oic.sec.cred.mfgcert".
- 4723 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an
4724 IEEE802.1AR IDevID and should verify the "TCG Endorsement Key Credential". All TCG-
4725 defined manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the
4726 "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to
4727 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4728 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See
4729 NIST SP 800-57).
- 4730 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST
4731 SP 800-57).
- 4732 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST
4733 SP 800-57).
- 4734 – OCF Device(s) should protect the "/oic/sec/cred" resource using the platform provided secure
4735 storage.
- 4736 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned
4737 certificates) using platform provided secure storage.
- 4738 – OCF Device(s) should check certificate revocation status for locally issued certificates.
- 4739 – OCF OBTs (aka DOTS) shall check certificate revocation status for all certificates in
4740 manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation
4741 fails and the connection is refused. The DOTS may disregard revocation status results if
4742 unavailable.

4743 OCF Blue profile defines the following platform security assurances:

4744 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage
4745 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL
4746 Level 2.

4747 – Platforms implementing trusted platform functionality should be evaluated with a minimum
4748 Common Criteria EAL Level 1.

4749 OCF Blue profile defines the following platform security and privacy functionality:

4750 – The Platform shall implement cryptographic service provider (CSP) functionality.

4751 – Platform CSP functionality shall include cryptographic algorithms, random number generation,
4752 secure time.

4753 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST
4754 SP 800-57).

4755 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See
4756 NIST SP 800-57).

4757 – Platforms hosting OCF Device(s) should implement a platform identifier following
4758 IEEE802.1AR or Trusted Computing Group(TCG) specifications.

4759 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF
4760 Device(s) should supply TCG-defined manufacture certificates; also known as "TCG
4761 Endorsement Key Credential" (which complies with IETF RFC 5280) and "TCG Platform
4762 Credential" (which complies with IETF RFC 5755).

4763 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,
4764 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

4765 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to
4766 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

4767 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"
4768 Property to one of the other values found in the "supportedprofiles" Property.

4769 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**

4770 Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause
4771 14.8.2 must support following minimum requirements

4772 – Hardening minimums: secure credential storage, software integrity validation, secure update.

4773 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension
4774 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-
4775 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"

4776 – The OCF Device shall include a X.509v3 OCF CPLAttributes Extension (clause 9.4.2.2.7) in its
4777 End-Entity Certificate when manufacturer certificate is used.

4778 Security Profile Purple has following optional security hardening requirements that the device can
4779 additionally support.

4780 – Hardening additions: secure boot, hardware backed secure storage

4781 – The OCF Device shall include a X.509v3 OCF SecurityClaims Extension (clause 9.4.2.2.6) in
4782 its End-Entity Certificate and it shall include corresponding OIDs to the hardening additions
4783 implemented and attested by the vendor. If there is no additional support for hardening
4784 requirements, X.509v3 OCF SecurityClaims Extension shall be omitted.

4785 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism
4786 for security critical executables such as cryptographic modules or secure service applications,
4787 and they should be validated before the execution. The key used for validating the integrity must
4788 be pinned at the least to the validating software module.

4789 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4790 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM)
4791 to be executed by the processor on power-on, and secure boot parameters to be provisioned by
4792 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the
4793 security critical executables and stop the boot process if any integrity of them is compromised.

4794 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile
4795 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic
4796 attacks.

4797 More details on security hardening guidelines for software integrity validation, secure boot,
4798 secure update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4799 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA
4800 Vetting Criteria defined by OCF.

4801 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-
4802 purple-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other
4803 profiles.

4804 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to
4805 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

4806 **15 Device Type Specific Requirements**

4807 **15.1 Bridging Security**

4808 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4809 The Bridge shall go through OCF ownership transfer as any other onboarder would.

4810 The software of an Bridge shall be field updatable. (This requirement need not be tested but can
4811 be certified via a vendor declaration.)

4812 Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be
4813 provisioned as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged
4814 Devices are treated the same way as physical Devices. They are entities that have to be
4815 provisioned in their network.

4816 Each VOD shall implement the behaviour required by ISO/IEC 30118-1:2018 and this document.
4817 Each VOD shall perform authentication, access control, and encryption according to the security
4818 settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security
4819 requirements of the Bridged Protocol.

4820 In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall
4821 use appropriate ecosystem-specific security options for communication between the Virtual
4822 Bridged Devices instantiated by the Bridge and Bridged Devices. This security shall include
4823 mutual authentication, and encryption and integrity protection of messages in the bridged
4824 ecosystem.

4825 A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see
4826 clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which
4827 instantiated that VOD.

4828 A VOD may authenticate itself to the OCF Cloud (see clause 10.5.2) using the Manufacturer
4829 Certificate and corresponding private key of the Bridge which instantiated that VOD.

4830 A Bridge and the VODs created by that Bridge shall operate as independent Devices, with the
4831 following exceptions:

4832 – If a Bridge creates a VOD while the Bridge is in an Unowned State, then the VOD shall be
4833 created in an Unowned State.

4834 – An Unowned VOD shall not accept DTLS connection attempts nor TLS connection attempts
4835 nor any other requests, including discovery requests, while the Bridge (that created that VOD)
4836 is Unowned.

4837 – At any time when a Bridge is transitioning from Owned to Unowned State, all Unowned VODs
4838 (created by that Bridge prior to the transition) shall drop any existing TLS and/or DTLS
4839 connections.

4840 – At any time when a Bridge is transitioning from Unowned to Owned State, the Bridge shall
4841 trigger all Unowned VODs (created by that Bridge prior to the transition) to become
4842 accessible in RFOTM state, with internal state as if the VOD has just transitioned from
4843 RESET to RFOTM.

4844 – If a Bridge creates a VOD while the Bridge is in an Owned State, then the VOD shall become
4845 accessible in RFOTM state, with internal state as if the VOD has just transitioned from
4846 RESET to RFOTM.

4847 Table 76 intends to clarify this behaviour.

4848
4849

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

4850 The "vods" Property of the "oic.r.vodlist" Resource on a Bridge reflects the details of all currently
 4851 Owned VODs which have been created by that Bridge since the most recent hardware reset (if
 4852 any) of the Bridge Platform (which removes all the created VODs), regardless of whether the
 4853 VODs have the same owner as the Bridge or not. The entries in the "vods" Property are added
 4854 and removed according to the following criteria:

- 4855 – Whenever a VOD created by a Bridge transitions from being Unowned to being Owned, then
 4856 an entry for that VOD shall be added to the "vods" Property of the "oic.r.vodlist" Resource of
 4857 that Bridge.
- 4858 – Whenever a VOD created by a Bridge transitions from being Owned to being Unowned, then
 4859 entry for that VOD shall be removed from the "vods" Property of the "oic.r.vodlist" Resource of
 4860 that Bridge. If that Bridge is currently in Unowned state, then the "oic.r.vodlist" Resource is
 4861 not accessible, and the entry for that VOD shall be removed from the "vods" Property before
 4862 or during the transition of that Bridge to the Owned state.
- 4863 – All other modifications of the list are not allowed.

4864 A Bridge shall only expose a secure OCF Endpoint for the "oic.r.vodlist" Resource.

4865 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4866 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4867 For AllJoyn translator, an OCF OBT shall be able to block the communication of all OCF Devices
 4868 with all Bridged Devices that don't communicate securely with the Bridge, by using the Bridge
 4869 Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

4870 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4871 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
 4872 communicate securely with the Bridge.

4873 **15.1.2.3 Additional Security Requirements specific to the oneM2M Protocols**

4874 The Bridge shall implement oneM2M application access control as defined in the oneM2M
 4875 Release 3 Specifications.

4876 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
 4877 communicate securely with the Bridge.

4878 **15.1.2.4 Additional Security Requirements specific to the U+ Protocol**

4879 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4880 communicate securely with the Bridge.

4881 **15.1.2.5 Additional Security Requirements specific to the Z-Wave Protocol**

4882 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4883 communicate securely with the Bridge.

4884 **15.1.2.6 Additional Security Requirements specific to the Zigbee Protocol**

4885 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4886 communicate securely with the Bridge.

4887

4888

4889

4890

4891

4892

4893

4894

4895

4896

4897

4898

4899

4900

4901

4902

4903

4904

4905

4906

4907

4908 .

4909
4910
4911

Annex A (informative) Access Control Examples

4912

Example OCF ACL Resource

4913 Figure A-1 shows how a "/oic/sec/acl2" Resource could be configured to enforce an example
4914 access policy on the Server.

```
4915 {
4916   "aclist2": [
4917     {
4918       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve,
4919       Update, Delete and Notify)
4920       "subject": {"uuid": "XXX-...-XX01"},
4921       "resources": [
4922         {"href": "/oic/sh/light/1"},
4923         {"href": "/oic/sh/temp/0"}
4924       ],
4925       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC
4926       "validity": [
4927         // The period starting at 18:00:00 UTC, on January 1, 2015 and
4928         // ending at 07:00:00 UTC on January 2, 2015
4929         "period": ["20150101T180000Z/20150102T070000Z"],
4930         // Repeats the {period} every week until the last day of Jan. 2015.
4931         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
4932       ],
4933       "aceid": 1
4934     }
4935   ],
4936   // An ACL provisioning and management service should be identified as
4937   // the resource owner
4938   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
4939 }
```

4940

Figure A-1 – Example "/oic/sec/acl2" Resource

4941

Example AMS

4942 Figure A-2 demonstrates how the "/oic/sec/amacl" Resource should be configured to achieve this
4943 objective.

```
4944 {
4945   "resources": [
4946     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
4947     // supplied then use the sacl validation credential to enforce access.
4948     {"href": "/oic/sh/light/1"},
4949     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
4950     // supplied then use the sacl validation credential to enforce access.
4951     {"href": "/oma/3"},

```

```
4952 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4953 // the sacl validation credential to enforce access.
4954 {"wc": ""}]
4955 }
4956
```

Figure A-2 Example "/oic/sec/amacl" Resource

4957
4958
4959

Annex B (Informative) Execution Environment Security Profiles

4960 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security
4961 robustness requirements meeting all IOT applications and services will not serve the needs of
4962 OCF, and security profiles of varying degree of robustness (trustworthiness), cost and complexity
4963 have to be defined. To address a large ecosystem of vendors, the profiles can only be defined as
4964 requirements and the exact solutions meeting those requirements are specific to the vendors'
4965 open or proprietary implementations, and thus in most part outside scope of this document.

4966 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228
4967 (Terminology for constrained node networks) methodology, we limit the number of security
4968 profiles to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities
4969 criteria for each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4970 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are
4971 either capable of no security functionality or easily breakable security that depend on
4972 environmental (e.g. availability of human) factors to perform security functions. This means the
4973 class 0 will not be equipped with an SEE.

4974

Table B.1 – OCF Security Profile

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4975 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of
4976 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,
4977 memory, storage).

4978
4979
4980

Annex C (normative) Resource Type definitions

4981 C.1 List of Resource Type definitions

4982 Table C.1 contains the list of defined security resources in this document.

4983 **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

4984

4985 C.2 Account Token

4986 C.2.1 Introduction

4987 Sign-up using generic account provider.

4988 C.2.2 Well-known URI

4989 /oic/sec/account

4990 C.2.3 Resource type

4991 The Resource Type is defined as: "oic.r.account".

4992 C.2.4 OpenAPI 2.0 definition

```
4993 {  
4994   "swagger": "2.0",  
4995   "info": {  
4996     "title": "Account Token",  
4997     "version": "20190111",  
4998     "license": {  
4999       "name": "OCF Data Model License",  
5000       "url":  
5001       "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
5002       CENSE.md",  
5003       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
5004       reserved."  
5005     }  
5006     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
```

```

5007     },
5008     "schemes": ["http"],
5009     "consumes": ["application/json"],
5010     "produces": ["application/json"],
5011     "paths": {
5012         "/oic/sec/account" : {
5013             "post": {
5014                 "description": "Sign-up using generic account provider.\n",
5015                 "parameters": [
5016                     { "$ref": "#/parameters/interface" },
5017                     {
5018                         "name": "body",
5019                         "in": "body",
5020                         "required": true,
5021                         "schema": { "$ref": "#/definitions/Account-request" },
5022                         "x-example":
5023                             {
5024                                 "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
5025                                 "authprovider" : "github",
5026                                 "accesstoken" : "8802f2eaf8b5e147a936"
5027                             }
5028                     }
5029                 ],
5030                 "responses": {
5031                     "204": {
5032                         "description": "2.04 Changed respond with required and optional information\n",
5033                         "x-example":
5034                             {
5035                                 "rt": ["oic.r.account"],
5036                                 "accesstoken" : "0f3d9f7fe5491d54077d",
5037                                 "refreshToken" : "00fe4644a6fbe5324eec",
5038                                 "expiresin" : 3600,
5039                                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
5040                                 "redirecturi" : "coaps+tcp://example.com:443"
5041                             },
5042                         "schema": { "$ref": "#/definitions/Account-response" }
5043                     }
5044                 }
5045             },
5046             "delete": {
5047                 "description": "Delete a device. This also removes all resources in the device on cloud
5048 side.\nexample: /oic/account?di=9cfbeb8e-5ale-4dlc-9d01-
5049 00c04fd430c8&accesstoken=0f3d9f7fe5491d54077d\n",
5050                 "parameters": [
5051                     { "$ref": "#/parameters/interface" }
5052                 ],
5053                 "responses": {
5054                     "202": {
5055                         "description": "2.02 Deleted response informing the device is successfully
5056 deleted.\n"
5057                     }
5058                 }
5059             }
5060         },
5061     },
5062     "parameters": {
5063         "interface" : {
5064             "in" : "query",
5065             "name" : "if",
5066             "type" : "string",
5067             "enum" : ["oic.if.baseline"]
5068         }
5069     },
5070     "definitions": {
5071         "Account-request" : {
5072             "properties": {
5073                 "authprovider": {
5074                     "description": "The name of Authorization Provider through which Access Token was
5075 obtained",
5076                     "type": "string"
5077                 }

```

```

5078     "accesstoken" : {
5079         "description": "Access-Token used for communication with OCF Cloud after account creation",
5080         "pattern": "(?!$|\\s+).*",
5081         "type": "string"
5082     },
5083     "di": {
5084         "description": "Format pattern according to IETF RFC 4122.",
5085         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5086         "type": "string"
5087     }
5088 },
5089 "type" : "object",
5090 "required": ["di", "accesstoken"]
5091 },
5092 "Account-response": {
5093     "properties": {
5094         "expiresin" : {
5095             "description": "Access-Token remaining life time in seconds (-1 if permanent)",
5096             "readOnly": true,
5097             "type": "integer"
5098         },
5099         "rt": {
5100             "description": "Resource Type of the Resource",
5101             "items": {
5102                 "maxLength": 64,
5103                 "type": "string",
5104                 "enum" : ["oic.r.account"]
5105             },
5106             "minItems": 1,
5107             "maxItems": 1,
5108             "readOnly": true,
5109             "type": "array"
5110         },
5111         "refreshtoken" : {
5112             "description": "Refresh token can be used to refresh the Access Token before getting
5113 expired",
5114             "pattern": "(?!$|\\s+).*",
5115             "readOnly": true,
5116             "type": "string"
5117         },
5118         "uid" : {
5119             "description": "Format pattern according to IETF RFC 4122.",
5120             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5121             "type": "string"
5122         },
5123         "accesstoken" : {
5124             "description": "Access-Token used for communication with cloud after account creation",
5125             "pattern": "(?!$|\\s+).*",
5126             "type": "string"
5127         },
5128         "n": {
5129             "$ref":
5130 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5131 schema.json#/definitions/n"
5132         },
5133         "id": {
5134             "$ref":
5135 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5136 schema.json#/definitions/id"
5137         },
5138         "redirecturi" : {
5139             "description": "Using this URI, the Client needs to reconnect to a redirected OCF Cloud.
5140 If provided, this value shall be used by the Device instead of Mediator-provided URI during the
5141 Device Registration.",
5142             "readOnly": true,
5143             "type": "string"
5144         },
5145         "if": {
5146             "description": "The interface set supported by this resource",
5147             "items": {
5148                 "enum": [

```

```

5149         "oic.if.baseline"
5150     ],
5151     "type": "string"
5152 },
5153     "minItems": 1,
5154     "maxItems": 1,
5155     "uniqueItems": true,
5156     "readOnly": true,
5157     "type": "array"
5158 }
5159 },
5160     "type" : "object",
5161     "required": ["accesstoken", "refreshtoken", "expiresin", "uid"]
5162 }
5163 }
5164 }
5165

```

5166 **C.2.5 Property definition**

5167 Table C.2 defines the Properties that are part of the "oic.r.account" Resource Type.

5168 **Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account".**

Property name	Value type	Mandatory	Access mode	Description
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
authprovider	string	No	Read Write	The name of Authorization Provider through which Access Token was obtained
accesstoken	string	Yes	Read Write	Access-Token used for communication with OCF Cloud after account creation
id	multiple types: see schema	No	Read Write	
refreshtoken	string	Yes	Read Only	Refresh token can be used to refresh the Access Token before getting expired
rt	array: see schema	No	Read Only	Resource Type of the Resource
accesstoken	string	Yes	Read Write	Access-Token used for communication with cloud after account creation
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds

				(-1 if permanent)
if	array: schema	see	No	Read Only
redirecturi	string		No	Read Only
				Using this URI, the Client needs to reconnect to a redirected OCF Cloud. If provided, this value shall be used by the Device instead of Mediator-provided URI during the Device Registration.
n	multiple types: see schema		No	Read Write

5169 **C.2.6 CRUDN behaviour**

5170 Table C.3 defines the CRUDN operations that are supported on the "oic.r.account" Resource
5171 Type.

5172 **Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account".**

Create	Read	Update	Delete	Notify
		post	delete	

5173 **C.3 Access Control List [DEPRECATED]**

5174 This clause intentionally left empty.

5175 **C.4 Access Control List-2**

5176 **C.4.1 Introduction**

5177 This Resource specifies the local access control list.
5178 When used without query parameters, all the ACE entries are returned.
5179 When used with a query parameter, only the ACEs matching the specified
5180 parameter are returned.
5181

5182 **C.4.2 Well-known URI**

5183 /oic/sec/acl2

5184 **C.4.3 Resource type**

5185 The Resource Type is defined as: "oic.r.acl2".

5186 **C.4.4 OpenAPI 2.0 definition**

```
5187 {
5188   "swagger": "2.0",
5189   "info": {
5190     "title": "Access Control List-2",
5191     "version": "20190111",
5192     "license": {
5193       "name": "OCF Data Model License",
5194       "url":
5195       "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
```

```

5196 CENSE.md",
5197     "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5198 reserved."
5199 },
5200     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5201 },
5202     "schemes": ["http"],
5203     "consumes": ["application/json"],
5204     "produces": ["application/json"],
5205     "paths": {
5206         "/oic/sec/acl2" : {
5207             "get": {
5208                 "description": "This Resource specifies the local access control list.\nWhen used without
5209 query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
5210 matching the specified\nparameter are returned.\n",
5211                 "parameters": [
5212                     {"$ref": "#/parameters/interface"},
5213                     {"$ref": "#/parameters/ace-filtered"}
5214 ],
5215                 "responses": {
5216                     "200": {
5217                         "description": "",
5218                         "x-example":
5219                             {
5220                                 "rt" : ["oic.r.acl2"],
5221                                 "aclist2": [
5222                                     {
5223                                         "aceid": 1,
5224                                         "subject": {
5225                                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5226                                             "role": "SOME_STRING"
5227                                         },
5228                                         "resources": [
5229                                             {
5230                                                 "href": "/light",
5231                                                 "rt": ["oic.r.light"],
5232                                                 "if": ["oic.if.baseline", "oic.if.a"]
5233                                             },
5234                                             {
5235                                                 "href": "/door",
5236                                                 "rt": ["oic.r.door"],
5237                                                 "if": ["oic.if.baseline", "oic.if.a"]
5238                                             }
5239                                         ],
5240                                         "permission": 24
5241                                     },
5242                                     {
5243                                         "aceid": 2,
5244                                         "subject": {
5245                                             "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5246                                         },
5247                                         "resources": [
5248                                             {
5249                                                 "href": "/light",
5250                                                 "rt": ["oic.r.light"],
5251                                                 "if": ["oic.if.baseline", "oic.if.a"]
5252                                             },
5253                                             {
5254                                                 "href": "/door",
5255                                                 "rt": ["oic.r.door"],
5256                                                 "if": ["oic.if.baseline", "oic.if.a"]
5257                                             }
5258                                         ],
5259                                         "permission": 24
5260                                     },
5261                                     {
5262                                         "aceid": 3,
5263                                         "subject": {"conntype": "anon-clear"},
5264                                         "resources": [
5265                                             {
5266                                                 "href": "/light",

```

```

5267         "rt": ["oic.r.light"],
5268         "if": ["oic.if.baseline", "oic.if.a"]
5269     },
5270     {
5271         "href": "/door",
5272         "rt": ["oic.r.door"],
5273         "if": ["oic.if.baseline", "oic.if.a"]
5274     }
5275 ],
5276 "permission": 16,
5277 "validity": [
5278     {
5279         "period": "20160101T180000Z/20170102T070000Z",
5280         "recurrence": [ "DSTART:XXXXX",
5281 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5282     },
5283     {
5284         "period": "20160101T180000Z/PT5H30M",
5285         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5286     }
5287 ]
5288 },
5289 ],
5290 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5291 },
5292 "schema": { "$ref": "#/definitions/Acl2" }
5293 },
5294 "400": {
5295     "description": "The request is invalid."
5296 }
5297 },
5298 },
5299 "post": {
5300     "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
5301 update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
5302 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
5303 the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
5304     "parameters": [
5305         { "$ref": "#/parameters/interface" },
5306         { "$ref": "#/parameters/ace-filtered" },
5307         {
5308             "name": "body",
5309             "in": "body",
5310             "required": true,
5311             "schema": { "$ref": "#/definitions/Acl2-Update" },
5312             "x-example":
5313                 {
5314                     "aclist2": [
5315                         {
5316                             "aceid": 1,
5317                             "subject": {
5318                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5319                                 "role": "SOME_STRING"
5320                             },
5321                             "resources": [
5322                                 {
5323                                     "href": "/light",
5324                                     "rt": ["oic.r.light"],
5325                                     "if": ["oic.if.baseline", "oic.if.a"]
5326                                 },
5327                                 {
5328                                     "href": "/door",
5329                                     "rt": ["oic.r.door"],
5330                                     "if": ["oic.if.baseline", "oic.if.a"]
5331                                 }
5332                             ],
5333                             "permission": 24
5334                         },
5335                         {
5336                             "aceid": 3,
5337                             "subject": {

```

```

5338         "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5339     },
5340     "resources": [
5341         {
5342             "href": "/light",
5343             "rt": ["oic.r.light"],
5344             "if": ["oic.if.baseline", "oic.if.a"]
5345         },
5346         {
5347             "href": "/door",
5348             "rt": ["oic.r.door"],
5349             "if": ["oic.if.baseline", "oic.if.a"]
5350         }
5351     ],
5352     "permission": 24
5353 }
5354 ],
5355 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5356 }
5357 },
5358 ],
5359 "responses": {
5360     "400": {
5361         "description": "The request is invalid."
5362     },
5363     "201": {
5364         "description": "The ACL entry is created."
5365     },
5366     "204": {
5367         "description": "The ACL entry is updated."
5368     }
5369 },
5370 },
5371 "delete": {
5372     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5373 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
5374 the\nspecified parameter are deleted.\n",
5375     "parameters": [
5376         {"$ref": "#/parameters/interface"},
5377         {"$ref": "#/parameters/ace-filtered"}
5378     ],
5379     "responses": {
5380         "200": {
5381             "description": "The matching ACEs or the entire ACL Resource has been successfully
5382 deleted."
5383         },
5384         "400": {
5385             "description": "The request is invalid."
5386         }
5387     }
5388 },
5389 },
5390 },
5391 "parameters": {
5392     "interface": {
5393         "in": "query",
5394         "name": "if",
5395         "type": "string",
5396         "enum": ["oic.if.baseline"]
5397     },
5398     "ace-filtered": {
5399         "in": "query",
5400         "name": "aceid",
5401         "required": false,
5402         "type": "integer",
5403         "description": "Only applies to the ACE with the specified aceid.",
5404         "x-example": 2112
5405     }
5406 },
5407 "definitions": {
5408     "Acl2": {

```



```

5409     "properties": {
5410         "rowneruuid" : {
5411             "description": "The value identifies the unique Resource owner\nFormat pattern according
5412 to IETF RFC 4122.",
5413             "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}$",
5414             "type": "string"
5415         },
5416         "rt" : {
5417             "description": "Resource Type of the Resource.",
5418             "items": {
5419                 "maxLength": 64,
5420                 "type": "string",
5421                 "enum": ["oic.r.acl2"]
5422             },
5423             "minItems": 1,
5424             "maxItems": 1,
5425             "readOnly": true,
5426             "type": "array"
5427         },
5428         "aclist2" : {
5429             "description": "Access Control Entries in the ACL Resource.",
5430             "items": {
5431                 "properties": {
5432                     "aceid": {
5433                         "description": "An identifier for the ACE that is unique within the ACL. In cases
5434 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5435                         "minimum": 1,
5436                         "type": "integer"
5437                     },
5438                     "permission": {
5439                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5440 permissions.",
5441                         "x-detail-desc": [
5442                             "0 - No permissions",
5443                             "1 - Create permission is granted",
5444                             "2 - Read, observe, discover permission is granted",
5445                             "4 - Write, update permission is granted",
5446                             "8 - Delete permission is granted",
5447                             "16 - Notify permission is granted"
5448                         ],
5449                         "maximum": 31,
5450                         "minimum": 0,
5451                         "type": "integer"
5452                     },
5453                     "resources": {
5454                         "description": "References the application's Resources to which a security policy
5455 applies.",
5456                         "items": {
5457                             "description": "Each Resource must have at least one of these properties set.",
5458                             "properties": {
5459                                 "href": {
5460                                     "description": "When present, the ACE only applies when the href matches\nThis
5461 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5462                                     "format": "uri",
5463                                     "maxLength": 256,
5464                                     "type": "string"
5465                                 },
5466                                 "if": {
5467                                     "description": "When present, the ACE only applies when the if (interface)
5468 matches\nThe interface set supported by this Resource.",
5469                                     "items": {
5470                                         "enum": [
5471                                             "oic.if.baseline",
5472                                             "oic.if.ll",
5473                                             "oic.if.b",
5474                                             "oic.if.rw",
5475                                             "oic.if.r",
5476                                             "oic.if.a",
5477                                             "oic.if.s"
5478                                         ],
5479                                         "type": "string"

```

```

5480         },
5481         "minItems": 1,
5482         "type": "array"
5483     },
5484     "rt": {
5485         "description": "When present, the ACE only applies when the rt (Resource type)
5486 matches\nResource Type of the Resource.",
5487         "items": {
5488             "maxLength": 64,
5489             "type": "string"
5490         },
5491         "minItems": 1,
5492         "type": "array"
5493     },
5494     "wc": {
5495         "description": "A wildcard matching policy.",
5496         "pattern": "^[+*]$",
5497         "type": "string"
5498     }
5499 },
5500 "type": "object"
5501 },
5502 "type": "array"
5503 },
5504 "subject": {
5505     "anyOf": [
5506     {
5507         "description": "This is the Device identifier.",
5508         "properties": {
5509             "uuid": {
5510                 "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5511 4122.",
5512                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5513 fA-F0-9]{12}$",
5514                 "type": "string"
5515             }
5516         },
5517         "required": [
5518             "uuid"
5519         ],
5520         "type": "object"
5521     },
5522     {
5523         "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5524 <Authority> refers to the local entity or Device.",
5525         "properties": {
5526             "authority": {
5527                 "description": "The Authority component of the entity being identified. A
5528 NULL <Authority> refers to the local entity or Device.",
5529                 "type": "string"
5530             },
5531             "role": {
5532                 "description": "The ID of the role being identified.",
5533                 "type": "string"
5534             }
5535         },
5536         "required": [
5537             "role"
5538         ],
5539         "type": "object"
5540     },
5541     {
5542         "properties": {
5543             "conntype": {
5544                 "description": "This property allows an ACE to be matched based on the
5545 connection or message type.",
5546                 "x-detail-desc": [
5547                     "auth-crypt - ACE applies if the Client is authenticated and the data
5548 channel or message is encrypted and integrity protected",
5549                     "anon-clear - ACE applies if the Client is not authenticated and the data
5550 channel or message is not encrypted but may be integrity protected"

```

```

5551         ],
5552         "enum": [
5553             "auth-crypt",
5554             "anon-clear"
5555         ],
5556         "type": "string"
5557     }
5558 },
5559     "required": [
5560         "conntype"
5561     ],
5562     "type": "object"
5563 }
5564 ]
5565 },
5566 "validity": {
5567     "description": "validity is an array of time-pattern objects.",
5568     "items": {
5569         "description": "The time-pattern contains a period and recurrence expressed in
5570 RFC5545 syntax.",
5571         "properties": {
5572             "period": {
5573                 "description": "String represents a period using the RFC5545 Period.",
5574                 "type": "string"
5575             },
5576             "recurrence": {
5577                 "description": "String array represents a recurrence rule using the RFC5545
5578 Recurrence.",
5579                 "items": {
5580                     "type": "string"
5581                 },
5582                 "type": "array"
5583             }
5584         },
5585         "required": [
5586             "period"
5587         ],
5588         "type": "object"
5589     },
5590     "type": "array"
5591 }
5592 },
5593 "required": [
5594     "aceid",
5595     "resources",
5596     "permission",
5597     "subject"
5598 ],
5599 "type": "object"
5600 },
5601 "type": "array"
5602 },
5603 "n": {
5604     "$ref":
5605 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5606 schema.json#/definitions/n"
5607 },
5608 "id": {
5609     "$ref":
5610 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5611 schema.json#/definitions/id"
5612 },
5613 "if" : {
5614     "description": "The interface set supported by this Resource.",
5615     "items": {
5616         "enum": [
5617             "oic.if.baseline"
5618         ],
5619         "type": "string"
5620     },
5621     "minItems": 1,

```

```

5622         "maxItems": 1,
5623         "readOnly": true,
5624         "type": "array"
5625     }
5626 },
5627 "type" : "object",
5628 "required": ["acllist2", "rowneruuid"]
5629 },
5630 "Acl2-Update" : {
5631     "properties": {
5632         "rowneruuid" : {
5633             "description": "The value identifies the unique Resource owner\n Format pattern according
5634 to IETF RFC 4122.",
5635             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5636 9]{12}$",
5637             "type": "string"
5638         },
5639         "acllist2" : {
5640             "description": "Access Control Entries in the ACL Resource.",
5641             "items": {
5642                 "properties": {
5643                     "aceid": {
5644                         "description": "An identifier for the ACE that is unique within the ACL. In cases
5645 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5646                         "minimum": 1,
5647                         "type": "integer"
5648                     },
5649                     "permission": {
5650                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5651 permissions.",
5652                         "x-detail-desc": [
5653                             "0 - No permissions",
5654                             "1 - Create permission is granted",
5655                             "2 - Read, observe, discover permission is granted",
5656                             "4 - Write, update permission is granted",
5657                             "8 - Delete permission is granted",
5658                             "16 - Notify permission is granted"
5659                         ],
5660                         "maximum": 31,
5661                         "minimum": 0,
5662                         "type": "integer"
5663                     },
5664                     "resources": {
5665                         "description": "References the application's Resources to which a security policy
5666 applies.",
5667                         "items": {
5668                             "description": "Each Resource must have at least one of these properties set.",
5669                             "properties": {
5670                                 "href": {
5671                                     "description": "When present, the ACE only applies when the href matches\nThis
5672 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5673                                     "format": "uri",
5674                                     "maxLength": 256,
5675                                     "type": "string"
5676                                 },
5677                                 "if": {
5678                                     "description": "When present, the ACE only applies when the if (interface)
5679 matches\nThe interface set supported by this Resource.",
5680                                     "items": {
5681                                         "enum": [
5682                                             "oic.if.baseline",
5683                                             "oic.if.ll",
5684                                             "oic.if.b",
5685                                             "oic.if.rw",
5686                                             "oic.if.r",
5687                                             "oic.if.a",
5688                                             "oic.if.s"
5689                                         ],
5690                                         "type": "string"
5691                                     },
5692                                     "minItems": 1,

```

```

5693         "type": "array"
5694     },
5695     "rt": {
5696         "description": "When present, the ACE only applies when the rt (Resource type)
5697 matches\nResource Type of the Resource.",
5698         "items": {
5699             "maxLength": 64,
5700             "type": "string"
5701         },
5702         "minItems": 1,
5703         "type": "array"
5704     },
5705     "wc": {
5706         "description": "A wildcard matching policy.",
5707         "x-detail-desc": [
5708             "+ - Matches all discoverable Resources",
5709             "- - Matches all non-discoverable Resources",
5710             "* - Matches all Resources"
5711         ],
5712         "enum": [
5713             "+",
5714             "-",
5715             "*"
5716         ],
5717         "type": "string"
5718     }
5719 },
5720 "type": "object"
5721 },
5722 "type": "array"
5723 },
5724 "subject": {
5725     "anyOf": [
5726         {
5727             "description": "This is the Device identifier.",
5728             "properties": {
5729                 "uuid": {
5730                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5731 4122.",
5732                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5733 fA-F0-9]{12}$",
5734                     "type": "string"
5735                 }
5736             },
5737             "required": [
5738                 "uuid"
5739             ],
5740             "type": "object"
5741         },
5742         {
5743             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5744 <Authority> refers to the local entity or Device.",
5745             "properties": {
5746                 "authority": {
5747                     "description": "The Authority component of the entity being identified. A
5748 NULL <Authority> refers to the local entity or Device.",
5749                     "type": "string"
5750                 },
5751                 "role": {
5752                     "description": "The ID of the role being identified.",
5753                     "type": "string"
5754                 }
5755             },
5756             "required": [
5757                 "role"
5758             ],
5759             "type": "object"
5760         },
5761         {
5762             "properties": {
5763                 "conntype": {

```

```

5764         "description": "This property allows an ACE to be matched based on the
5765 connection or message type.",
5766         "x-detail-desc": [
5767           "auth-crypt - ACE applies if the Client is authenticated and the data
5768 channel or message is encrypted and integrity protected",
5769           "anon-clear - ACE applies if the Client is not authenticated and the data
5770 channel or message is not encrypted but may be integrity protected"
5771         ],
5772         "enum": [
5773           "auth-crypt",
5774           "anon-clear"
5775         ],
5776         "type": "string"
5777       }
5778     },
5779     "required": [
5780       "conntype"
5781     ],
5782     "type": "object"
5783   }
5784 ]
5785 },
5786 "validity": {
5787   "description": "validity is an array of time-pattern objects.",
5788   "items": {
5789     "description": "The time-pattern contains a period and recurrence expressed in
5790 RFC5545 syntax.",
5791     "properties": {
5792       "period": {
5793         "description": "String represents a period using the RFC5545 Period.",
5794         "type": "string"
5795       },
5796       "recurrence": {
5797         "description": "String array represents a recurrence rule using the RFC5545
5798 Recurrence.",
5799         "items": {
5800           "type": "string"
5801         },
5802         "type": "array"
5803       }
5804     },
5805     "required": [
5806       "period"
5807     ],
5808     "type": "object"
5809   },
5810   "type": "array"
5811 }
5812 },
5813 "required": [
5814   "resources",
5815   "permission",
5816   "subject"
5817 ],
5818 "type": "object"
5819 },
5820 "type": "array"
5821 }
5822 },
5823 "type": "object"
5824 }
5825 }
5826 }
5827

```

5828 C.4.5 Property definition

5829 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.

5831 C.4.6 CRUDN behaviour

5832 Table C.5 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

5833 Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5834 C.5 Managed Access Control**5835 C.5.1 Introduction**

5836 This Resource specifies the host Resources with access permission that is managed by an AMS.

5837 C.5.2 Well-known URI

5838 /oic/sec/amacl

5839 C.5.3 Resource type

5840 The Resource Type is defined as: "oic.r.amacl".

5841 C.5.4 OpenAPI 2.0 definition

```
5842 {
5843   "swagger": "2.0",
5844   "info": {
5845     "title": "Managed Access Control",
```

```

5846     "version": "20190111",
5847     "license": {
5848         "name": "OCF Data Model License",
5849         "url":
5850 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5851 CENSE.md",
5852         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5853 reserved."
5854     },
5855     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5856 },
5857     "schemes": ["http"],
5858     "consumes": ["application/json"],
5859     "produces": ["application/json"],
5860     "paths": {
5861         "/oic/sec/amacl" : {
5862             "get": {
5863                 "description": "This Resource specifies the host Resources with access permission that is
5864 managed by an AMS.\n",
5865                 "parameters": [
5866                     {"$ref": "#/parameters/interface"}
5867                 ],
5868                 "responses": {
5869                     "200": {
5870                         "description": "",
5871                         "x-example":
5872                         {
5873                             "rt" : ["oic.r.amacl"],
5874                             "resources": [
5875                                 {
5876                                     "href": "/temp",
5877                                     "rt": ["oic.r.temperature"],
5878                                     "if": ["oic.if.baseline", "oic.if.a"]
5879                                 },
5880                                 {
5881                                     "href": "/temp",
5882                                     "rt": ["oic.r.temperature"],
5883                                     "if": ["oic.if.baseline", "oic.if.s"]
5884                                 }
5885                             ]
5886                         },
5887                     "schema": { "$ref": "#/definitions/Amacl" }
5888                 }
5889             }
5890         },
5891         "post": {
5892             "description": "Sets the new amacl data.\n",
5893             "parameters": [
5894                 {"$ref": "#/parameters/interface"},
5895                 {
5896                     "name": "body",
5897                     "in": "body",
5898                     "required": true,
5899                     "schema": { "$ref": "#/definitions/Amacl" },
5900                     "x-example":
5901                     {
5902                         "resources": [
5903                             {
5904                                 "href": "/temp",
5905                                 "rt": ["oic.r.temperature"],
5906                                 "if": ["oic.if.baseline", "oic.if.a"]
5907                             },
5908                             {
5909                                 "href": "/temp",
5910                                 "rt": ["oic.r.temperature"],
5911                                 "if": ["oic.if.baseline", "oic.if.s"]
5912                             }
5913                         ]
5914                     }
5915                 }
5916             ],

```



```

5917     "responses": {
5918         "400": {
5919             "description": "The request is invalid."
5920         },
5921         "201": {
5922             "description": "The AMACL entry is created."
5923         },
5924         "204": {
5925             "description": "The AMACL entry is updated."
5926         }
5927     },
5928 },
5929 "put": {
5930     "description": "Creates the new acl data.\n",
5931     "parameters": [
5932         {"$ref": "#/parameters/interface"},
5933         {
5934             "name": "body",
5935             "in": "body",
5936             "required": true,
5937             "schema": { "$ref": "#/definitions/Amacl" },
5938             "x-example":
5939                 {
5940                     "resources": [
5941                         {
5942                             "href": "/temp",
5943                             "rt": ["oic.r.temperature"],
5944                             "if": ["oic.if.baseline", "oic.if.a"]
5945                         },
5946                         {
5947                             "href": "/temp",
5948                             "rt": ["oic.r.temperature"],
5949                             "if": ["oic.if.baseline", "oic.if.s"]
5950                         }
5951                     ]
5952                 }
5953         }
5954     ],
5955     "responses": {
5956         "400": {
5957             "description": "The request is invalid."
5958         },
5959         "201": {
5960             "description": "The AMACL entry is created."
5961         }
5962     }
5963 },
5964 "delete": {
5965     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
5966     entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
5967     matched entry is deleted.\n",
5968     "parameters": [
5969         {"$ref": "#/parameters/interface"},
5970         {
5971             "in": "query",
5972             "description": "Delete the ACE identified by the string matching the subject value.\n",
5973             "type": "string",
5974             "name": "subject"
5975         }
5976     ],
5977     "responses": {
5978         "200": {
5979             "description": "The ACE instance or the the entire AMACL Resource has been
5980     successfully deleted."
5981         },
5982         "400": {
5983             "description": "The request is invalid."
5984         }
5985     }
5986 }
5987 }

```

```

5988 },
5989 "parameters": {
5990   "interface" : {
5991     "in" : "query",
5992     "name" : "if",
5993     "type" : "string",
5994     "enum" : ["oic.if.baseline"]
5995   }
5996 },
5997 "definitions": {
5998   "Amacl" : {
5999     "properties": {
6000       "rt" : {
6001         "description": "Resource Type of the Resource.",
6002         "items": {
6003           "maxLength": 64,
6004           "type": "string",
6005           "enum": ["oic.r.amacl"]
6006         },
6007         "minItems": 1,
6008         "maxItems": 1,
6009         "readOnly": true,
6010         "type": "array"
6011       },
6012       "n": {
6013         "$ref":
6014 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6015 schema.json#/definitions/n"
6016       },
6017       "id": {
6018         "$ref":
6019 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6020 schema.json#/definitions/id"
6021       },
6022       "resources" : {
6023         "description": "Multiple links to this host's Resources.",
6024         "items": {
6025           "description": "Each Resource must have at least one of these properties set.",
6026           "properties": {
6027             "href": {
6028               "description": "When present, the ACE only applies when the href matches\nThis is
6029 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6030               "format": "uri",
6031               "maxLength": 256,
6032               "type": "string"
6033             },
6034             "if": {
6035               "description": "When present, the ACE only applies when the if (interface)
6036 matches\nThe interface set supported by this Resource.",
6037               "items": {
6038                 "enum": [
6039                   "oic.if.baseline",
6040                   "oic.if.ll",
6041                   "oic.if.b",
6042                   "oic.if.rw",
6043                   "oic.if.r",
6044                   "oic.if.a",
6045                   "oic.if.s"
6046                 ],
6047                 "type": "string"
6048               },
6049               "minItems": 1,
6050               "type": "array"
6051             },
6052             "rt": {
6053               "description": "When present, the ACE only applies when the rt (Resource type)
6054 matches\nResource Type of the Resource.",
6055               "items": {
6056                 "maxLength": 64,
6057                 "type": "string"
6058               },

```

```

6059         "minItems": 1,
6060         "type": "array"
6061     },
6062     "wc": {
6063         "description": "A wildcard matching policy.",
6064         "pattern": "^[--+]*$",
6065         "type": "string"
6066     }
6067 },
6068     "type": "object"
6069 },
6070     "type": "array"
6071 },
6072     "if" : {
6073         "description": "The interface set supported by this Resource.",
6074         "items": {
6075             "enum": [
6076                 "oic.if.baseline"
6077             ],
6078             "type": "string"
6079         },
6080         "minItems": 1,
6081         "maxItems": 1,
6082         "readOnly": true,
6083         "type": "array"
6084     }
6085 },
6086     "type" : "object",
6087     "required": ["resources"]
6088 }
6089 }
6090 }
6091

```

6092 **C.5.5 Property definition**

6093 Table C.6 defines the Properties that are part of the "oic.r.amacl" Resource Type.

6094 **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	

6095 **C.5.6 CRUDN behaviour**

6096 Table C.7 defines the CRUDN operations that are supported on the "oic.r.amacl" Resource Type.

6097 **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

6098 C.6 Credential

6099 C.6.1 Introduction

6100 This Resource specifies credentials a Device may use to establish secure communication.

6101 Retrieves the credential data.

6102 When used without query parameters, all the credential entries are returned.

6103 When used with a query parameter, only the credentials matching the specified
6104 parameter are returned.

6105

6106 Note that write-only credential data will not be returned.

6107

6108 C.6.2 Well-known URI

6109 /oic/sec/cred

6110 C.6.3 Resource type

6111 The Resource Type is defined as: "oic.r.cred".

6112 C.6.4 OpenAPI 2.0 definition

```
6113 {
6114   "swagger": "2.0",
6115   "info": {
6116     "title": "Credential",
6117     "version": "v1.0-20181031",
6118     "license": {
6119       "name": "OCF Data Model License",
6120       "url":
6121         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6122         CENSE.md",
6123       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6124         reserved."
6125     },
6126     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6127   },
6128   "schemes": ["http"],
6129   "consumes": ["application/json"],
6130   "produces": ["application/json"],
6131   "paths": {
6132     "/oic/sec/cred" : {
6133       "get": {
6134         "description": "This Resource specifies credentials a Device may use to establish secure
6135         communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
6136         credential entries are returned.\nWhen used with a query parameter, only the credentials matching
6137         the specified\nparameter are returned.\n\nNote that write-only credential data will not be
6138         returned.\n",
6139         "parameters": [
6140           {"$ref": "#/parameters/interface"}
6141           ,{"$ref": "#/parameters/cred-filtered-credid"}
6142           ,{"$ref": "#/parameters/cred-filtered-subjectuoid"}
6143         ],
6144         "responses": {
6145           "200": {
6146             "description": "",
6147             "x-example":
6148               {
6149                 "xt": ["oic.r.cred"],
6150                 "creds": [
6151                   {
6152                     "credid": 55,
6153                     "subjectuoid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6154                     "roleid": {
6155                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6156                       "role": "SOME_STRING"
6157                     },
6158                     "credtype": 32,
6159                     "publicdata": {
```

```

6160         "encoding": "oic.sec.encoding.base64",
6161         "data": "BASE-64-ENCODED-VALUE"
6162     },
6163     "privatedata": {
6164         "encoding": "oic.sec.encoding.base64",
6165         "data": "BASE-64-ENCODED-VALUE",
6166         "handle": 4
6167     },
6168     "optionaldata": {
6169         "revstat": false,
6170         "encoding": "oic.sec.encoding.base64",
6171         "data": "BASE-64-ENCODED-VALUE"
6172     },
6173     "period": "20160101T180000Z/20170102T070000Z",
6174     "crms": [ "oic.sec.crm.pk10" ]
6175 },
6176 {
6177     "credid": 56,
6178     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6179     "roleid": {
6180         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6181         "role": "SOME_STRING"
6182     },
6183     "credtype": 1,
6184     "publicdata": {
6185         "encoding": "oic.sec.encoding.base64",
6186         "data": "BASE-64-ENCODED-VALUE"
6187     },
6188     "privatedata": {
6189         "encoding": "oic.sec.encoding.base64",
6190         "data": "BASE-64-ENCODED-VALUE",
6191         "handle": 4
6192     },
6193     "optionaldata": {
6194         "revstat": false,
6195         "encoding": "oic.sec.encoding.base64",
6196         "data": "BASE-64-ENCODED-VALUE"
6197     },
6198     "period": "20160101T180000Z/20170102T070000Z",
6199     "crms": [ "oic.sec.crm.pk10" ]
6200 }
6201 ],
6202 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6203 }
6204 ,
6205 "schema": { "$ref": "#/definitions/Cred" }
6206 },
6207 "400": {
6208     "description": "The request is invalid."
6209 }
6210 }
6211 },
6212 "post": {
6213     "description": "Updates the credential Resource with the provided
6214 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
6215 Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
6216 Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
6217 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
6218 Resource.\n",
6219     "parameters": [
6220         { "$ref": "#/parameters/interface" },
6221         {
6222             "name": "body",
6223             "in": "body",
6224             "required": true,
6225             "schema": { "$ref": "#/definitions/Cred-Update" },
6226             "x-example":
6227                 {
6228                     "creds": [
6229                         {
6230                             "credid": 55,

```

```

6231         "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6232         "roleid": {
6233             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6234             "role": "SOME_STRING"
6235         },
6236         "credtype": 32,
6237         "publicdata": {
6238             "encoding": "oic.sec.encoding.base64",
6239             "data": "BASE-64-ENCODED-VALUE"
6240         },
6241         "privatedata": {
6242             "encoding": "oic.sec.encoding.base64",
6243             "data": "BASE-64-ENCODED-VALUE",
6244             "handle": 4
6245         },
6246         "optionaldata": {
6247             "revstat": false,
6248             "encoding": "oic.sec.encoding.base64",
6249             "data": "BASE-64-ENCODED-VALUE"
6250         },
6251         "period": "20160101T180000Z/20170102T070000Z",
6252         "crms": [ "oic.sec.crm.pk10" ]
6253     },
6254     {
6255         "credid": 56,
6256         "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6257         "roleid": {
6258             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6259             "role": "SOME_STRING"
6260         },
6261         "credtype": 1,
6262         "publicdata": {
6263             "encoding": "oic.sec.encoding.base64",
6264             "data": "BASE-64-ENCODED-VALUE"
6265         },
6266         "privatedata": {
6267             "encoding": "oic.sec.encoding.base64",
6268             "data": "BASE-64-ENCODED-VALUE",
6269             "handle": 4
6270         },
6271         "optionaldata": {
6272             "revstat": false,
6273             "encoding": "oic.sec.encoding.base64",
6274             "data": "BASE-64-ENCODED-VALUE"
6275         },
6276         "period": "20160101T180000Z/20170102T070000Z",
6277         "crms": [ "oic.sec.crm.pk10" ]
6278     }
6279 ],
6280     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6281 }
6282 }
6283 ],
6284 "responses": {
6285     "400": {
6286         "description": "The request is invalid."
6287     },
6288     "201": {
6289         "description": "The credential entry is created."
6290     },
6291     "204": {
6292         "description": "The credential entry is updated."
6293     }
6294 }
6295 },
6296 "delete": {
6297     "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
6298 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
6299 matching\nthe query parameter are deleted.\n",
6300     "parameters": [
6301         {"$ref": "#/parameters/interface"},

```

```

6302         {"$ref": "#/parameters/cred-filtered-credid"},
6303         {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6304     ],
6305     "responses": {
6306         "400": {
6307             "description": "The request is invalid."
6308         },
6309         "204": {
6310             "description": "The specific credential(s) or the the entire credential Resource has
6311 been successfully deleted."
6312         }
6313     }
6314 }
6315 }
6316 },
6317 "parameters": {
6318     "interface" : {
6319         "in" : "query",
6320         "name" : "if",
6321         "type" : "string",
6322         "enum" : ["oic.if.baseline"]
6323     },
6324     "cred-filtered-credid" : {
6325         "in" : "query",
6326         "name" : "credid",
6327         "required" : false,
6328         "type" : "integer",
6329         "description" : "Only applies to the credential with the specified credid.",
6330         "x-example" : 2112
6331     },
6332     "cred-filtered-subjectuuid" : {
6333         "in" : "query",
6334         "name" : "subjectuuid",
6335         "required" : false,
6336         "type" : "string",
6337         "description" : "Only applies to credentials with the specified subject UUID.",
6338         "x-example" : "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6339     }
6340 },
6341 "definitions": {
6342     "Cred" : {
6343         "properties": {
6344             "rowneruuid" : {
6345                 "description": "Format pattern according to IETF RFC 4122.",
6346                 "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}$",
6347                 "type": "string"
6348             },
6349             "rt" : {
6350                 "description": "Resource Type of the Resource.",
6351                 "items": {
6352                     "maxLength": 64,
6353                     "type": "string",
6354                     "enum": ["oic.r.cred"]
6355                 },
6356                 "minItems": 1,
6357                 "readOnly": true,
6358                 "type": "array",
6359                 "uniqueItems": true
6360             },
6361             "n": {
6362                 "$ref":
6363 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6364 schema.json#/definitions/n"
6365             },
6366             "id": {
6367                 "$ref":
6368 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6369 schema.json#/definitions/id"
6370             },
6371             "creds" : {
6372                 "description": "List of credentials available at this Resource.",

```

```

6373     "items": {
6374         "properties": {
6375             "credid": {
6376                 "description": "Local reference to a credential Resource.",
6377                 "type": "integer"
6378             },
6379             "credtype": {
6380                 "description": "Representation of this credential's type\nCredential Types - Cred
6381 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6382 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6383 password32 - Asymmetric encryption key.",
6384                 "maximum": 63,
6385                 "minimum": 0,
6386                 "type": "integer"
6387             },
6388             "credusage": {
6389                 "description": "A string that provides hints about how/where the cred is used\nThe
6390 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6391 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6392 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6393                 "enum": [
6394                     "oic.sec.cred.trustca",
6395                     "oic.sec.cred.cert",
6396                     "oic.sec.cred.rolecert",
6397                     "oic.sec.cred.mfgtrustca",
6398                     "oic.sec.cred.mfgcert"
6399                 ],
6400                 "type": "string"
6401             },
6402             "crms": {
6403                 "description": "The refresh methods that may be used to update this credential.",
6404                 "items": {
6405                     "description": "Each enum represents a method by which the credentials are
6406 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6407 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6408 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6409 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6410                     "enum": [
6411                         "oic.sec.crm.pro",
6412                         "oic.sec.crm.psk",
6413                         "oic.sec.crm.rdp",
6414                         "oic.sec.crm.skdc",
6415                         "oic.sec.crm.pk10"
6416                     ],
6417                     "type": "string"
6418                 },
6419                 "type": "array",
6420                 "uniqueItems": true
6421             },
6422             "optionaldata": {
6423                 "description": "Credential revocation status information\nOptional credential
6424 contents describes revocation status for this credential.",
6425                 "properties": {
6426                     "data": {
6427                         "description": "The encoded structure.",
6428                         "type": "string"
6429                     },
6430                     "encoding": {
6431                         "description": "A string specifying the encoding format of the data contained in
6432 the optdata.",
6433                         "x-detail-desc": [
6434                             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6435                             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6436                             "oic.sec.encoding.base64 - Base64 encoded object.",
6437                             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6438                             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6439                             "oic.sec.encoding.raw - Raw hex encoded data."
6440                         ],
6441                         "enum": [
6442                             "oic.sec.encoding.jwt",
6443                             "oic.sec.encoding.cwt",

```



```

6444         "oic.sec.encoding.base64",
6445         "oic.sec.encoding.pem",
6446         "oic.sec.encoding.der",
6447         "oic.sec.encoding.raw"
6448     ],
6449     "type": "string"
6450 },
6451 "revstat": {
6452     "description": "Revocation status flag - true = revoked.",
6453     "type": "boolean"
6454 }
6455 },
6456 "required": [
6457     "revstat"
6458 ],
6459 "type": "object"
6460 },
6461 "period": {
6462     "description": "String with RFC5545 Period.",
6463     "type": "string"
6464 },
6465 "privatedata": {
6466     "description": "Private credential information\nCredential Resource non-public
6467 contents.",
6468     "properties": {
6469         "data": {
6470             "description": "The encoded value.",
6471             "maxLength": 3072,
6472             "type": "string"
6473         },
6474         "encoding": {
6475             "description": "A string specifying the encoding format of the data contained in
6476 the privdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6477 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6478 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6479 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data.",
6480             "enum": [
6481                 "oic.sec.encoding.jwt",
6482                 "oic.sec.encoding.cwt",
6483                 "oic.sec.encoding.base64",
6484                 "oic.sec.encoding.uri",
6485                 "oic.sec.encoding.handle",
6486                 "oic.sec.encoding.raw"
6487             ],
6488             "type": "string"
6489         },
6490         "handle": {
6491             "description": "Handle to a key storage Resource.",
6492             "type": "integer"
6493         }
6494     },
6495     "required": [
6496         "encoding"
6497     ],
6498     "type": "object"
6499 },
6500 "publicdata": {
6501     "description": "Public credential information.",
6502     "properties": {
6503         "data": {
6504             "description": "The encoded value.",
6505             "maxLength": 3072,
6506             "type": "string"
6507         },
6508         "encoding": {
6509             "description": "A string specifying the encoding format of the data contained in
6510 the pubdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6511 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6512 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.pem - Encoding for PEM encoded
6513 certificate or chain\noic.sec.encoding.der - Encoding for DER encoded
6514 certificate\noic.sec.encoding.raw - Raw hex encoded data.",

```

```

6515         "enum": [
6516             "oic.sec.encoding.jwt",
6517             "oic.sec.encoding.cwt",
6518             "oic.sec.encoding.base64",
6519             "oic.sec.encoding.uri",
6520             "oic.sec.encoding.pem",
6521             "oic.sec.encoding.der",
6522             "oic.sec.encoding.raw"
6523         ],
6524         "type": "string"
6525     },
6526 },
6527 "type": "object"
6528 },
6529 "roleid": {
6530     "description": "The role this credential possesses\nSecurity role specified as an
6531 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6532     "properties": {
6533         "authority": {
6534             "description": "The Authority component of the entity being identified. A NULL
6535 <Authority> refers to the local entity or Device.",
6536             "type": "string"
6537         },
6538         "role": {
6539             "description": "The ID of the role being identified.",
6540             "type": "string"
6541         }
6542     },
6543     "required": [
6544         "role"
6545     ],
6546     "type": "object"
6547 },
6548 "subjectuuid": {
6549     "anyOf": [
6550         {
6551             "description": "The id of the Device, which the cred entry applies to or \"*\n
6552 for wildcard identity.",
6553             "pattern": "^[\\*]*$",
6554             "type": "string"
6555         },
6556         {
6557             "description": "Format pattern according to IETF RFC 4122.",
6558             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6559 F0-9]{12}$",
6560             "type": "string"
6561         }
6562     ]
6563 },
6564 },
6565 "type": "object"
6566 },
6567 "type": "array"
6568 },
6569 "if" : {
6570     "description": "The interface set supported by this Resource.",
6571     "items": {
6572         "enum": [
6573             "oic.if.baseline"
6574         ],
6575         "type": "string"
6576     },
6577     "minItems": 1,
6578     "readOnly": true,
6579     "type": "array"
6580 }
6581 },
6582 "type" : "object",
6583 "required": ["creds", "rowneruuid"]
6584 },
6585 "Cred-Update" : {

```

```

6586     "properties": {
6587         "rowneruuid" : {
6588             "description": "Format pattern according to IETF RFC 4122.",
6589             "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}$",
6590             "type": "string"
6591         },
6592         "creds" : {
6593             "description": "List of credentials available at this Resource.",
6594             "items": {
6595                 "properties": {
6596                     "credid": {
6597                         "description": "Local reference to a credential Resource.",
6598                         "type": "integer"
6599                     },
6600                     "credtype": {
6601                         "description": "Representation of this credential's type\nCredential Types - Cred
6602 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6603 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6604 password32 - Asymmetric encryption key.",
6605                         "maximum": 63,
6606                         "minimum": 0,
6607                         "type": "integer"
6608                     },
6609                     "credusage": {
6610                         "description": "A string that provides hints about how/where the cred is used\nThe
6611 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6612 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6613 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6614                         "enum": [
6615                             "oic.sec.cred.trustca",
6616                             "oic.sec.cred.cert",
6617                             "oic.sec.cred.rolecert",
6618                             "oic.sec.cred.mfgtrustca",
6619                             "oic.sec.cred.mfgcert"
6620                         ],
6621                         "type": "string"
6622                     },
6623                     "crms": {
6624                         "description": "The refresh methods that may be used to update this credential.",
6625                         "items": {
6626                             "description": "Each enum represents a method by which the credentials are
6627 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6628 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6629 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6630 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6631                             "enum": [
6632                                 "oic.sec.crm.pro",
6633                                 "oic.sec.crm.psk",
6634                                 "oic.sec.crm.rdp",
6635                                 "oic.sec.crm.skdc",
6636                                 "oic.sec.crm.pk10"
6637                             ],
6638                             "type": "string"
6639                         },
6640                         "type": "array"
6641                     },
6642                     "optionaldata": {
6643                         "description": "Credential revocation status information\nOptional credential
6644 contents describes revocation status for this credential.",
6645                         "properties": {
6646                             "data": {
6647                                 "description": "The encoded structure.",
6648                                 "type": "string"
6649                             },
6650                             "encoding": {
6651                                 "description": "A string specifying the encoding format of the data contained in
6652 the optdata.",
6653                                 "x-detail-desc": [
6654                                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6655                                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6656                                     "oic.sec.encoding.base64 - Base64 encoded object.",

```

```

6657         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6658         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6659         "oic.sec.encoding.raw - Raw hex encoded data."
6660     ],
6661     "enum": [
6662         "oic.sec.encoding.jwt",
6663         "oic.sec.encoding.cwt",
6664         "oic.sec.encoding.base64",
6665         "oic.sec.encoding.pem",
6666         "oic.sec.encoding.der",
6667         "oic.sec.encoding.raw"
6668     ],
6669     "type": "string"
6670 },
6671 "revstat": {
6672     "description": "Revocation status flag - true = revoked.",
6673     "type": "boolean"
6674 },
6675 },
6676 "required": [
6677     "revstat"
6678 ],
6679 "type": "object"
6680 },
6681 "period": {
6682     "description": "String with RFC5545 Period.",
6683     "type": "string"
6684 },
6685 "privatedata": {
6686     "description": "Private credential information\nCredential Resource non-public
6687 contents.",
6688     "properties": {
6689         "data": {
6690             "description": "The encoded value.",
6691             "maxLength": 3072,
6692             "type": "string"
6693         },
6694         "encoding": {
6695             "description": "A string specifying the encoding format of the data contained in
6696 the privdata.",
6697             "x-detail-desc": [
6698                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6699                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6700                 "oic.sec.encoding.base64 - Base64 encoded object.",
6701                 "oic.sec.encoding.uri - URI reference.",
6702                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
6703 referenced using a handle.",
6704                 "oic.sec.encoding.raw - Raw hex encoded data."
6705             ],
6706             "enum": [
6707                 "oic.sec.encoding.jwt",
6708                 "oic.sec.encoding.cwt",
6709                 "oic.sec.encoding.base64",
6710                 "oic.sec.encoding.uri",
6711                 "oic.sec.encoding.handle",
6712                 "oic.sec.encoding.raw"
6713             ],
6714             "type": "string"
6715         },
6716         "handle": {
6717             "description": "Handle to a key storage Resource.",
6718             "type": "integer"
6719         }
6720     },
6721     "required": [
6722         "encoding"
6723     ],
6724     "type": "object"
6725 },
6726 "publicdata": {
6727     "properties": {

```

```

6728         "data": {
6729             "description": "The encoded value.",
6730             "maxLength": 3072,
6731             "type": "string"
6732         },
6733         "encoding": {
6734             "description": "Public credential information\nA string specifying the encoding
6735 format of the data contained in the pubdata.",
6736             "x-detail-desc": [
6737                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6738                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6739                 "oic.sec.encoding.base64 - Base64 encoded object.",
6740                 "oic.sec.encoding.uri - URI reference.",
6741                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6742                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6743                 "oic.sec.encoding.raw - Raw hex encoded data."
6744             ],
6745             "enum": [
6746                 "oic.sec.encoding.jwt",
6747                 "oic.sec.encoding.cwt",
6748                 "oic.sec.encoding.base64",
6749                 "oic.sec.encoding.uri",
6750                 "oic.sec.encoding.pem",
6751                 "oic.sec.encoding.der",
6752                 "oic.sec.encoding.raw"
6753             ],
6754             "type": "string"
6755         },
6756     },
6757     "type": "object"
6758 },
6759 "roleid": {
6760     "description": "The role this credential possesses\nSecurity role specified as an
6761 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6762     "properties": {
6763         "authority": {
6764             "description": "The Authority component of the entity being identified. A NULL
6765 <Authority> refers to the local entity or Device.",
6766             "type": "string"
6767         },
6768         "role": {
6769             "description": "The ID of the role being identified.",
6770             "type": "string"
6771         }
6772     },
6773     "required": [
6774         "role"
6775     ],
6776     "type": "object"
6777 },
6778 "subjectuuid": {
6779     "anyOf": [
6780         {
6781             "description": "The id of the Device, which the cred entry applies to or \"*\n
6782 for wildcard identity.",
6783             "pattern": "^\\*$",
6784             "type": "string"
6785         },
6786         {
6787             "description": "Format pattern according to IETF RFC 4122.",
6788             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6789 F0-9]{12}$",
6790             "type": "string"
6791         }
6792     ]
6793 },
6794 },
6795 "type": "object"
6796 },
6797 "type": "array"
6798 },

```

```

6799     "if" :
6800     {
6801         "description": "The interface set supported by this Resource.",
6802         "items": {
6803             "enum": [
6804                 "oic.if.baseline"
6805             ],
6806             "type": "string"
6807         },
6808         "minItems": 1,
6809         "readOnly": true,
6810         "type": "array"
6811     },
6812 },
6813 "type" : "object"
6814 }
6815 }
6816 }
6817

```

6818 **C.6.5 Property definition**

6819 Table C.8 defines the Properties that are part of the "oic.r.cred" Resource Type.

6820 **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.

6821 **C.6.6 CRUDN behaviour**

6822 Table C.9 defines the CRUDN operations that are supported on the "oic.r.cred" Resource Type.

6823 **Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

6824 C.7 Certificate Revocation

6825 C.7.1 Introduction

6826 This Resource specifies certificate revocation lists as X.509 objects.

6827 C.7.2 Well-known URI

6828 /oic/sec/crl

6829 C.7.3 Resource type

6830 The Resource Type is defined as: "oic.r.crl".

6831 C.7.4 OpenAPI 2.0 definition

```
6832 {
6833   "swagger": "2.0",
6834   "info": {
6835     "title": "Certificate Revocation",
6836     "version": "v1.0-20150819",
6837     "license": {
6838       "name": "OCF Data Model License",
6839       "url":
6840         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6841         CENSE.md",
6842       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6843       reserved."
6844     },
6845     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6846   },
6847   "schemes": ["http"],
6848   "consumes": ["application/json"],
6849   "produces": ["application/json"],
6850   "paths": {
6851     "/oic/sec/crl" : {
6852       "get": {
6853         "description": "This Resource specifies certificate revocation lists as X.509 objects.\n",
6854         "parameters": [
6855           { "$ref": "#/parameters/interface" }
6856         ],
6857         "responses": {
6858           "200": {
6859             "description": "",
6860             "x-example":
6861               {
6862                 "rt": ["oic.r.crl"],
6863                 "crlid": 1,
6864                 "thisupdate": "2016-04-12T23:20:50.52Z",
6865                 "crldata": "Base64ENCODEDCRL"
6866               },
6867             "schema": { "$ref": "#/definitions/Crl" }
6868           }
6869         }
6870       },
6871       "post": {
6872         "description": "Updates the CRL data.\n",
6873         "parameters": [
6874           { "$ref": "#/parameters/interface" },
6875           {
6876             "name": "body",
6877             "in": "body",
6878             "required": true,
6879             "schema": { "$ref": "#/definitions/Crl-Update" },
6880             "x-example":
6881               {
6882                 "crlid": 1,
6883                 "thisupdate": "2016-04-12T23:20:50.52Z",
6884                 "crldata": "Base64ENCODEDCRL"
6885               }
6886           }
6887         ]
6888       }
6889     }
6890   }
6891 }
```

```

6888     "responses": {
6889         "400": {
6890             "description": "The request is invalid."
6891         },
6892         "204": {
6893             "description": "The CRL entry is updated."
6894         }
6895     }
6896 }
6897 },
6898 },
6899 "parameters": {
6900     "interface" : {
6901         "in" : "query",
6902         "name" : "if",
6903         "type" : "string",
6904         "enum" : ["oic.if.baseline"]
6905     }
6906 },
6907 "definitions": {
6908     "Crl" : {
6909         "properties": {
6910             "rt" : {
6911                 "description": "Resource Type of the Resource.",
6912                 "items": {
6913                     "maxLength": 64,
6914                     "type": "string",
6915                     "enum": ["oic.r.crl"]
6916                 },
6917                 "minItems": 1,
6918                 "readOnly": true,
6919                 "type": "array"
6920             },
6921             "n": {
6922                 "$ref":
6923 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6924 schema.json#/definitions/n"
6925             },
6926             "id": {
6927                 "$ref":
6928 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6929 schema.json#/definitions/id"
6930             },
6931             "crlldata" : {
6932                 "description": "Base64 BER encoded CRL data.",
6933                 "type": "string"
6934             },
6935             "crlid" : {
6936                 "description": "Local reference to a CRL Resource.",
6937                 "type": "integer"
6938             },
6939             "thisupdate" : {
6940                 "description": "UTC time of last CRL update.",
6941                 "type": "string"
6942             },
6943             "if" : {
6944                 "description": "The interface set supported by this Resource.",
6945                 "items": {
6946                     "enum": [
6947                         "oic.if.baseline"
6948                     ],
6949                     "type": "string"
6950                 },
6951                 "minItems": 1,
6952                 "readOnly": true,
6953                 "type": "array"
6954             }
6955         },
6956         "type": "object",
6957         "required": ["crlid", "thisupdate", "crlldata"]
6958     }

```



```

6959     ,
6960     "Crl-Update": {
6961         "properties": {
6962             "crldata": {
6963                 "description": "Base64 BER encoded CRL data.",
6964                 "type": "string"
6965             },
6966             "crlid": {
6967                 "description": "Local reference to a CRL Resource.",
6968                 "type": "integer"
6969             },
6970             "thisupdate": {
6971                 "description": "UTC time of last CRL update.",
6972                 "type": "string"
6973             }
6974         },
6975         "type": "object"
6976     }
6977 }
6978 }
6979

```

6980 C.7.5 Property definition

6981 Table C.10 defines the Properties that are part of the "oic.r.crl" Resource Type.

6982 **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.

6983 C.7.6 CRUDN behaviour

6984 Table C.11 defines the CRUDN operations that are supported on the "oic.r.crl" Resource Type.

6985 **Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".**

Create	Read	Update	Delete	Notify
	get	post		observe

6986 C.8 Certificate Signing Request

6987 C.8.1 Introduction

6988 This Resource specifies a Certificate Signing Request.

6989 C.8.2 Well-known URI

6990 /oic/sec/csr

6991 C.8.3 Resource type

6992 The Resource Type is defined as: "oic.r.csr".

6993 C.8.4 OpenAPI 2.0 definition

```
6994 {
6995   "swagger": "2.0",
6996   "info": {
6997     "title": "Certificate Signing Request",
6998     "version": "v1.0-20150819",
6999     "license": {
7000       "name": "OCF Data Model License",
7001       "url":
7002         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7003         CENSE.md",
7004       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7005         reserved."
7006     },
7007     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7008   },
7009   "schemes": ["http"],
7010   "consumes": ["application/json"],
7011   "produces": ["application/json"],
7012   "paths": {
7013     "/oic/sec/csr" : {
7014       "get": {
7015         "description": "This Resource specifies a Certificate Signing Request.\n",
7016         "parameters": [
7017           {"$ref": "#/parameters/interface"}
7018         ],
7019         "responses": {
7020           "200": {
7021             "description": "",
7022             "x-example":
7023               {
7024                 "rt": ["oic.r.csr"],
7025                 "encoding" : "oic.sec.encoding.pem",
7026                 "csr": "PEMENCODEDCSR"
7027               },
7028             "schema": { "$ref": "#/definitions/Csr" }
7029           },
7030           "404": {
7031             "description" : "The Device does not support certificates and generating CSRs."
7032           },
7033           "503": {
7034             "description" : "The Device is not yet ready to return a response. Try again later."
7035           }
7036         }
7037       }
7038     }
7039   },
7040   "parameters": {
7041     "interface" : {
7042       "in" : "query",
7043       "name" : "if",
7044       "type" : "string",
7045       "enum" : ["oic.if.baseline"]
7046     }
7047   },
7048   "definitions": {
7049     "Csr" : {
```

```

7050     "properties": {
7051       "rt" : {
7052         "description": "Resource Type of the Resource.",
7053         "items": {
7054           "maxLength": 64,
7055           "type": "string",
7056           "enum": ["oic.r.csr"]
7057         },
7058         "minItems": 1,
7059         "readOnly": true,
7060         "type": "array"
7061       },
7062       "encoding": {
7063         "description": "A string specifying the encoding format of the data contained in CSR.",
7064         "x-detail-desc": [
7065           "oic.sec.encoding.pem - Encoding for PEM encoded CSR.",
7066           "oic.sec.encoding.der - Encoding for DER encoded CSR."
7067         ],
7068         "enum": [
7069           "oic.sec.encoding.pem",
7070           "oic.sec.encoding.der"
7071         ],
7072         "readOnly": true,
7073         "type": "string"
7074       },
7075       "n": {
7076         "$ref":
7077         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7078         schema.json#/definitions/n"
7079       },
7080       "id": {
7081         "$ref":
7082         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7083         schema.json#/definitions/id"
7084       },
7085       "csr": {
7086         "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
7087         "maxLength": 3072,
7088         "readOnly": true,
7089         "type": "string"
7090       },
7091       "if": {
7092         "description": "The interface set supported by this Resource.",
7093         "items": {
7094           "enum": [
7095             "oic.if.baseline"
7096           ],
7097           "type": "string"
7098         },
7099         "minItems": 1,
7100         "readOnly": true,
7101         "type": "array"
7102       }
7103     },
7104     "type" : "object",
7105     "required": ["csr", "encoding"]
7106   }
7107 }
7108 }
7109

```

7110 C.8.5 Property definition

7111 Table C.12 defines the Properties that are part of the "oic.r.csr" Resource Type.

7112 **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.

7113 **C.8.6 CRUDN behaviour**

7114 Table C.13 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

7115 **Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".**

Create	Read	Update	Delete	Notify
	get			observe

7116 **C.9 Device Owner Transfer Method**

7117 **C.9.1 Introduction**

7118 This Resource specifies properties needed to establish a Device owner.

7119

7120 **C.9.2 Well-known URI**

7121 /oic/sec/doxm

7122 **C.9.3 Resource type**

7123 The Resource Type is defined as: "oic.r.doxm".

7124 **C.9.4 OpenAPI 2.0 definition**

```

7125 {
7126   "swagger": "2.0",
7127   "info": {
7128     "title": "Device Owner Transfer Method",
7129     "version": "v1.0-20181001",
7130     "license": {
7131       "name": "OCF Data Model License",
7132       "url":
7133         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7134         CENSE.md",
7135       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7136         reserved."
7137     },
7138     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7139   },
7140   "schemes": ["http"],
7141   "consumes": ["application/json"],
7142   "produces": ["application/json"],
7143   "paths": {
7144     "/oic/sec/doxm" : {

```

```

7145     "get": {
7146         "description": "This Resource specifies properties needed to establish a Device owner.\n",
7147         "parameters": [
7148             {"$ref": "#/parameters/interface"}
7149         ],
7150         "responses": {
7151             "200": {
7152                 "description": "",
7153                 "x-example":
7154                 {
7155                     "rt": ["oic.r.doxm"],
7156                     "oxms": [ 0, 2, 3 ],
7157                     "oxmsel": 0,
7158                     "sct": 16,
7159                     "owned": true,
7160                     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7161                     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7162                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7163                 }
7164             },
7165             "schema": { "$ref": "#/definitions/Doxm" }
7166         },
7167         "400": {
7168             "description": "The request is invalid."
7169         }
7170     },
7171 },
7172 "post": {
7173     "description": "Updates the DOXM Resource data.\n",
7174     "parameters": [
7175         {"$ref": "#/parameters/interface"},
7176         {
7177             "name": "body",
7178             "in": "body",
7179             "required": true,
7180             "schema": { "$ref": "#/definitions/Doxm-Update" },
7181             "x-example":
7182             {
7183                 "oxmsel": 0,
7184                 "owned": true,
7185                 "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7186                 "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7187                 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7188             }
7189         }
7190     ],
7191     "responses": {
7192         "400": {
7193             "description": "The request is invalid."
7194         },
7195         "204": {
7196             "description": "The DOXM entry is updated."
7197         }
7198     }
7199 },
7200 },
7201 },
7202 "parameters": {
7203     "interface": {
7204         "in": "query",
7205         "name": "if",
7206         "type": "string",
7207         "enum": ["oic.if.baseline"]
7208     }
7209 },
7210 "definitions": {
7211     "Doxm": {
7212         "properties": {
7213             "rowneruuid": {
7214                 "description": "Format pattern according to IETF RFC 4122.",
7215                 "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}$",

```

```

7216         "type": "string"
7217     },
7218     "oxms": {
7219         "description": "List of supported owner transfer methods.",
7220         "items": {
7221             "description": "The Device owner transfer methods that may be selected at Device on-
7222 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
7223 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
7224 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
7225 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
7226 (deprecated).",
7227             "type": "integer"
7228         },
7229         "readOnly": true,
7230         "type": "array"
7231     },
7232     "devowneruuid": {
7233         "description": "Format pattern according to IETF RFC 4122.",
7234         "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}$",
7235         "type": "string"
7236     },
7237     "deviceuuid": {
7238         "description": "The uuid formatted identity of the Device\nFormat pattern according to
7239 IETF RFC 4122.",
7240         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7241         "type": "string"
7242     },
7243     "owned": {
7244         "description": "Ownership status flag.",
7245         "type": "boolean"
7246     },
7247     "n": {
7248         "$ref":
7249 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7250 schema.json#/definitions/n"
7251     },
7252     "id": {
7253         "$ref":
7254 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7255 schema.json#/definitions/id"
7256     },
7257     "oxmsel": {
7258         "description": "The selected owner transfer method used during on-boarding\nThe Device
7259 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7260 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7261 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7262 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7263 method (oic.sec.doxm.dcap) (deprecated).",
7264         "type": "integer"
7265     },
7266     "sct": {
7267         "description": "Bitmask encoding of supported credential types\nCredential Types -
7268 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7269 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7270 password32 - Asymmetric encryption key.",
7271         "maximum": 63,
7272         "minimum": 0,
7273         "type": "integer",
7274         "readOnly": true
7275     },
7276     "rt" : {
7277         "description": "Resource Type of the Resource.",
7278         "items": {
7279             "maxLength": 64,
7280             "type": "string",
7281             "enum": ["oic.r.doxm"]
7282         },
7283         "minItems": 1,
7284         "readOnly": true,
7285         "type": "array"
7286     },

```

```

7287     "if": {
7288         "description": "The interface set supported by this Resource.",
7289         "items": {
7290             "enum": [
7291                 "oic.if.baseline"
7292             ],
7293             "type": "string"
7294         },
7295         "minItems": 1,
7296         "readOnly": true,
7297         "type": "array"
7298     }
7299 },
7300 "type" : "object",
7301 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
7302 },
7303 "Doxm-Update" : {
7304     "properties": {
7305         "rowneruuid": {
7306             "description": "Format pattern according to IETF RFC 4122.",
7307             "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}$",
7308             "type": "string"
7309         },
7310         "devowneruuid": {
7311             "description": "Format pattern according to IETF RFC 4122.",
7312             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7313             "type": "string"
7314         },
7315         "deviceuuid": {
7316             "description": "The uuid formatted identity of the Device\nFormat pattern according to
7317 IETF RFC 4122.",
7318             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7319 9]{12}$",
7320             "type": "string"
7321         },
7322         "owned": {
7323             "description": "Ownership status flag.",
7324             "type": "boolean"
7325         },
7326         "oxmsel": {
7327             "description": "The selected owner transfer method used during on-boarding\nThe Device
7328 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7329 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7330 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7331 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7332 method (oic.sec.doxm.dcap) (deprecated).",
7333             "type": "integer"
7334         }
7335     },
7336     "type" : "object"
7337 }
7338 }
7339 }
7340

```

7341 C.9.5 Property definition

7342 Table C.14 defines the Properties that are part of the "oic.r.doxm" Resource Type.

7343 **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.

7344 **C.9.6 CRUDN behaviour**

7345 Table C.15 defines the CRUDN operations that are supported on the "oic.r.doxm" Resource Type.

7346 **Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".**

Create	Read	Update	Delete	Notify
	get	post		observe

7347 **C.10 Device Provisioning Status**

7348 **C.10.1 Introduction**

7349 This Resource specifies Device provisioning status.

7350

7351 **C.10.2 Well-known URI**

7352 /oic/sec/pstat

7353 **C.10.3 Resource type**

7354 The Resource Type is defined as: "oic.r.pstat".

7355 **C.10.4 OpenAPI 2.0 definition**

```
7356 {
7357   "swagger": "2.0",
7358   "info": {
7359     "title": "Device Provisioning Status",
7360     "version": "v1.0-20191001",
7361     "license": {
7362       "name": "OCF Data Model License",
7363       "url":
7364         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7365         CENSE.md",
7366       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7367         reserved."
7368     },
7369     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7370   },
7371   "schemes": ["http"],
7372   "consumes": ["application/json"],
7373   "produces": ["application/json"],
7374   "paths": {
7375     "/oic/sec/pstat" : {
7376       "get": {
7377         "description": "This Resource specifies Device provisioning status.\n",
7378         "parameters": [
7379           { "$ref": "#/parameters/interface" }
7380         ],
7381         "responses": {
7382           "200": {
7383             "description": "",
7384             "x-example":
7385               {
7386                 "rt": ["oic.r.pstat"],
7387                 "dos": {"s": 3, "p": true},
7388                 "isop": true,
7389                 "cm": 8,
7390                 "tm": 60,
7391                 "om": 2,
7392                 "sm": 7,
7393                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7394               },
7395             "schema": { "$ref": "#/definitions/Pstat" }
7396           },
7397           "400": {
7398             "description": "The request is invalid."
7399           }
7400         }
7401       },
7402       "post": {
7403         "description": "Sets or updates Device provisioning status data.\n",
7404         "parameters": [
7405           { "$ref": "#/parameters/interface" },
7406           {
7407             "name": "body",
7408             "in": "body",
7409             "required": true,
7410             "schema": { "$ref": "#/definitions/Pstat-Update" },
7411             "x-example":
7412               {
7413                 "dos": {"s": 3},
7414                 "tm": 60,
7415                 "om": 2,
7416                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7417               }
7418           }
7419         ]
7420       }
7421     }
7422   }
7423 }
```

```

7418     }
7419   ],
7420   "responses": {
7421     "400": {
7422       "description": "The request is invalid."
7423     },
7424     "204": {
7425       "description": "The PSTAT entry is updated."
7426     }
7427   }
7428 }
7429 }
7430 },
7431 "parameters": {
7432   "interface": {
7433     "in": "query",
7434     "name": "if",
7435     "type": "string",
7436     "enum": ["oic.if.baseline"]
7437   }
7438 },
7439 "definitions": {
7440   "Pstat": {
7441     "properties": {
7442       "rowneruuid": {
7443         "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7444 according to IETF RFC 4122.",
7445         "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}$",
7446         "type": "string"
7447       },
7448       "rt": {
7449         "description": "Resource Type of the Resource.",
7450         "items": {
7451           "maxLength": 64,
7452           "type": "string",
7453           "enum": ["oic.r.pstat"]
7454         },
7455         "minItems": 1,
7456         "readOnly": true,
7457         "type": "array"
7458       },
7459       "om": {
7460         "description": "Current operational mode\nDevice provisioning operation may be server
7461 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7462 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7463 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7464 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7465         "maximum": 7,
7466         "minimum": 1,
7467         "type": "integer"
7468       },
7469       "cm": {
7470         "description": "Current Device provisioning mode\nDevice provisioning mode maintains a
7471 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7472 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7473 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7474 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7475 Software Version Validation128 - Initiate Secure Software Update.",
7476         "maximum": 255,
7477         "minimum": 0,
7478         "type": "integer",
7479         "readOnly": true
7480       },
7481       "n": {
7482         "$ref":
7483 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7484 schema.json#/definitions/n"
7485       },
7486       "id": {
7487         "$ref":
7488 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-

```

```

7489 schema.json#/definitions/id"
7490     },
7491     "isop": {
7492         "description": "true indicates Device is operational.",
7493         "readOnly": true,
7494         "type": "boolean"
7495     },
7496     "tm": {
7497         "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7498         bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7499         in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7500         - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7501         services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7502         Software Version Validation128 - Initiate Secure Software Update.",
7503         "maximum": 255,
7504         "minimum": 0,
7505         "type": "integer"
7506     },
7507     "sm": {
7508         "description": "Supported operational modes\nDevice provisioning operation may be server
7509         directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7510         and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7511         services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7512         - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7513         "maximum": 7,
7514         "minimum": 1,
7515         "type": "integer",
7516         "readOnly": true
7517     },
7518     "dos": {
7519         "description": "Device on-boarding state\nDevice operation state machine.",
7520         "properties": {
7521             "p": {
7522                 "default": true,
7523                 "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7524                 to Device Resources are complete.",
7525                 "readOnly": true,
7526                 "type": "boolean"
7527             },
7528             "s": {
7529                 "description": "The current or pending operational state.",
7530                 "x-detail-desc": [
7531                     "0 - RESET - Device reset state.",
7532                     "1 - RFOTM - Ready for Device owner transfer method state.",
7533                     "2 - RFPPO - Ready for Device provisioning state.",
7534                     "3 - RFNOP - Ready for Device normal operation state.",
7535                     "4 - SRESET - The Device is in a soft reset state."
7536                 ],
7537                 "maximum": 4,
7538                 "minimum": 0,
7539                 "type": "integer"
7540             }
7541         },
7542         "required": [
7543             "s"
7544         ],
7545         "type": "object"
7546     },
7547     "if" : {
7548         "description": "The interface set supported by this Resource.",
7549         "items": {
7550             "enum": [
7551                 "oic.if.baseline"
7552             ],
7553             "type": "string"
7554         },
7555         "minItems": 1,
7556         "readOnly": true,
7557         "type": "array"
7558     }
7559 },

```

```

7560     "type" : "object",
7561     "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
7562 },
7563     "Pstat-Update" : {
7564     "properties": {
7565     "rowneruuid": {
7566         "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7567 according to IETF RFC 4122.",
7568         "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}$",
7569         "type": "string"
7570     },
7571     "om": {
7572         "description": "Current operational mode\nDevice provisioning operation may be server
7573 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7574 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7575 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7576 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7577         "maximum": 7,
7578         "minimum": 1,
7579         "type": "integer"
7580     },
7581     "tm": {
7582         "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7583 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7584 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7585 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7586 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7587 Software Version Validation128 - Initiate Secure Software Update.",
7588         "maximum": 255,
7589         "minimum": 0,
7590         "type": "integer"
7591     },
7592     "dos": {
7593         "description": "Device on-boarding state\nDevice operation state machine.",
7594         "properties": {
7595         "p": {
7596             "default": true,
7597             "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7598 to Device Resources are complete.",
7599             "readOnly": true,
7600             "type": "boolean"
7601         },
7602         "s": {
7603             "description": "The current or pending operational state.",
7604             "x-detail-desc": [
7605             "0 - RESET - Device reset state.",
7606             "1 - RFOTM - Ready for Device owner transfer method state.",
7607             "2 - RFPPO - Ready for Device provisioning state.",
7608             "3 - RFNOP - Ready for Device normal operation state.",
7609             "4 - SRESET - The Device is in a soft reset state."
7610             ],
7611             "maximum": 4,
7612             "minimum": 0,
7613             "type": "integer"
7614         }
7615         },
7616         "required": [
7617         "s"
7618         ],
7619         "type": "object"
7620     }
7621     },
7622     "type" : "object"
7623 }
7624 }
7625 }
7626

```

7627 C.10.5 Property definition

7628 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.
owneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.

7630 **C.10.6 CRUDN behaviour**

7631 Table C.17 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

7632 **Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".**

Create	Read	Update	Delete	Notify
	get	post		observe

7633 **C.11 Asserted Roles**

7634 **C.11.1 Introduction**

7635 This Resource specifies roles that have been asserted.

7636

7637 **C.11.2 Well-known URI**

7638 /oic/sec/roles

7639 **C.11.3 Resource type**

7640 The Resource Type is defined as: "oic.r.roles".

7641 C.11.4 OpenAPI 2.0 definition

```
7642 {
7643   "swagger": "2.0",
7644   "info": {
7645     "title": "Asserted Roles",
7646     "version": "v1.0-20170323",
7647     "license": {
7648       "name": "OCF Data Model License",
7649       "url":
7650         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7651 CENSE.md",
7652       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7653 reserved."
7654     },
7655     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7656   },
7657   "schemes": ["http"],
7658   "consumes": ["application/json"],
7659   "produces": ["application/json"],
7660   "paths": {
7661     "/oic/sec/roles" : {
7662       "get": {
7663         "description": "This Resource specifies roles that have been asserted.\n",
7664         "parameters": [
7665           {"$ref": "#/parameters/interface"}
7666         ],
7667         "responses": {
7668           "200": {
7669             "description": "",
7670             "x-example":
7671               {
7672                 "roles" :[
7673                   {
7674                     "credid":1,
7675                     "credtype":8,
7676                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7677                     "publicdata":
7678                       {
7679                         "encoding":"oic.sec.encoding.pem",
7680                         "data":"PEMENCODEDROLECERT"
7681                       },
7682                     "optionaldata":
7683                       {
7684                         "revstat": false,
7685                         "encoding":"oic.sec.encoding.pem",
7686                         "data":"PEMENCODEDISSUERCERT"
7687                       }
7688                   },
7689                   {
7690                     "credid":2,
7691                     "credtype":8,
7692                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7693                     "publicdata":
7694                       {
7695                         "encoding":"oic.sec.encoding.pem",
7696                         "data":"PEMENCODEDROLECERT"
7697                       },
7698                     "optionaldata":
7699                       {
7700                         "revstat": false,
7701                         "encoding":"oic.sec.encoding.pem",
7702                         "data":"PEMENCODEDISSUERCERT"
7703                       }
7704                   }
7705                 ],
7706                 "rt":["oic.r.roles"],
7707                 "if":["oic.if.baseline"]
7708               }
7709           },
7710           "schema": { "$ref": "#/definitions/Roles" }
7711         }
7712       }
7713     }
7714   }
7715 }
```

```

7711     },
7712     "400": {
7713         "description" : "The request is invalid."
7714     }
7715 },
7716 ],
7717 "post": {
7718     "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
7719 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
7720 same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
7721 ignored, the Resource assigns its own.\n",
7722     "parameters": [
7723         { "$ref": "#/parameters/interface" },
7724         {
7725             "name": "body",
7726             "in": "body",
7727             "required": true,
7728             "schema": { "$ref": "#/definitions/Roles-update" },
7729             "x-example":
7730             {
7731                 "roles" :[
7732                 {
7733                     "credid":1,
7734                     "credtype":8,
7735                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7736                     "publicdata":
7737                     {
7738                         "encoding":"oic.sec.encoding.pem",
7739                         "data":"PEMENCODEDROLECERT"
7740                     },
7741                     "optionaldata":
7742                     {
7743                         "revstat": false,
7744                         "encoding":"oic.sec.encoding.pem",
7745                         "data":"PEMENCODEDISSUERCERT"
7746                     }
7747                 },
7748                 {
7749                     "credid":2,
7750                     "credtype":8,
7751                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7752                     "publicdata":
7753                     {
7754                         "encoding":"oic.sec.encoding.pem",
7755                         "data":"PEMENCODEDROLECERT"
7756                     },
7757                     "optionaldata":
7758                     {
7759                         "revstat": false,
7760                         "encoding":"oic.sec.encoding.pem",
7761                         "data":"PEMENCODEDISSUERCERT"
7762                     }
7763                 }
7764             ]
7765         }
7766     ]
7767 },
7768 "responses": {
7769     "400": {
7770         "description" : "The request is invalid."
7771     },
7772     "204": {
7773         "description" : "The roles entry is updated."
7774     }
7775 },
7776 ],
7777 "delete": {
7778     "description": "Deletes roles Resource entries.\n\nWhen DELETE is used without query
7779 parameters, all the roles entries are deleted.\n\nWhen DELETE is used with a query parameter, only the
7780 entries matching\nthe query parameter are deleted.\n",
7781     "parameters": [

```

```

7782         {"$ref": "#/parameters/interface"},
7783         {"$ref": "#/parameters/roles-filtered"}
7784     ],
7785     "responses": {
7786         "200": {
7787             "description": "The specified or all roles Resource entries have been successfully
7788 deleted."
7789         },
7790         "400": {
7791             "description": "The request is invalid."
7792         }
7793     }
7794 }
7795 }
7796 },
7797 "parameters": {
7798     "interface": {
7799         "in": "query",
7800         "name": "if",
7801         "type": "string",
7802         "enum": ["oic.if.baseline"]
7803     },
7804     "roles-filtered": {
7805         "in": "query",
7806         "name": "credid",
7807         "required": false,
7808         "type": "integer",
7809         "description": "Only applies to the credential with the specified credid.",
7810         "x-example": 2112
7811     }
7812 },
7813 "definitions": {
7814     "Roles": {
7815         "properties": {
7816             "rt": {
7817                 "description": "Resource Type of the Resource.",
7818                 "items": {
7819                     "maxLength": 64,
7820                     "type": "string",
7821                     "enum": ["oic.r.roles"]
7822                 },
7823                 "minItems": 1,
7824                 "readOnly": true,
7825                 "type": "array"
7826             },
7827             "n": {
7828                 "$ref":
7829 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7830 schema.json#/definitions/n"
7831             },
7832             "id": {
7833                 "$ref":
7834 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7835 schema.json#/definitions/id"
7836             },
7837             "roles": {
7838                 "description": "List of role certificates.",
7839                 "items": {
7840                     "properties": {
7841                         "credid": {
7842                             "description": "Local reference to a credential Resource.",
7843                             "type": "integer"
7844                         },
7845                         "credtype": {
7846                             "description": "Representation of this credential's type\nCredential Types - Cred
7847 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7848 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7849 password32 - Asymmetric encryption key.",
7850                             "maximum": 63,
7851                             "minimum": 0,
7852                             "type": "integer"

```

```

7853     },
7854     "credusage": {
7855         "description": "A string that provides hints about how/where the cred is used\nThe
7856 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7857 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7858 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7859         "enum": [
7860             "oic.sec.cred.trustca",
7861             "oic.sec.cred.cert",
7862             "oic.sec.cred.rolecert",
7863             "oic.sec.cred.mfgtrustca",
7864             "oic.sec.cred.mfgcert"
7865         ],
7866         "type": "string"
7867     },
7868     "crms": {
7869         "description": "The refresh methods that may be used to update this credential.",
7870         "items": {
7871             "description": "Each enum represents a method by which the credentials are
7872 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7873 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7874 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7875 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7876             "enum": [
7877                 "oic.sec.crm.pro",
7878                 "oic.sec.crm.psk",
7879                 "oic.sec.crm.rdp",
7880                 "oic.sec.crm.skdc",
7881                 "oic.sec.crm.pk10"
7882             ],
7883             "type": "string"
7884         },
7885         "type": "array"
7886     },
7887     "optionaldata": {
7888         "description": "Credential revocation status information\nOptional credential
7889 contents describes revocation status for this credential.",
7890         "properties": {
7891             "data": {
7892                 "description": "This is the encoded structure.",
7893                 "type": "string"
7894             },
7895             "encoding": {
7896                 "description": "A string specifying the encoding format of the data contained in
7897 the optdata.",
7898                 "x-detail-desc": [
7899                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7900                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7901                     "oic.sec.encoding.base64 - Base64 encoded object.",
7902                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7903                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7904                     "oic.sec.encoding.raw - Raw hex encoded data."
7905                 ],
7906                 "enum": [
7907                     "oic.sec.encoding.jwt",
7908                     "oic.sec.encoding.cwt",
7909                     "oic.sec.encoding.base64",
7910                     "oic.sec.encoding.pem",
7911                     "oic.sec.encoding.der",
7912                     "oic.sec.encoding.raw"
7913                 ],
7914                 "type": "string"
7915             },
7916             "revstat": {
7917                 "description": "Revocation status flag - true = revoked.",
7918                 "type": "boolean"
7919             }
7920         },
7921         "required": [
7922             "revstat"
7923         ],

```

```

7924         "type": "object"
7925     },
7926     "period": {
7927         "description": "String with RFC5545 Period.",
7928         "type": "string"
7929     },
7930     "privatedata": {
7931         "description": "Private credential information\nCredential Resource non-public
7932 contents.",
7933         "properties": {
7934             "data": {
7935                 "description": "The encoded value.",
7936                 "maxLength": 3072,
7937                 "type": "string"
7938             },
7939             "encoding": {
7940                 "description": "A string specifying the encoding format of the data contained in
7941 the privdata.",
7942                 "x-detail-desc": [
7943                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7944                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7945                     "oic.sec.encoding.base64 - Base64 encoded object.",
7946                     "oic.sec.encoding.uri - URI reference.",
7947                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
7948 referenced using a handle.",
7949                     "oic.sec.encoding.raw - Raw hex encoded data."
7950                 ],
7951                 "enum": [
7952                     "oic.sec.encoding.jwt",
7953                     "oic.sec.encoding.cwt",
7954                     "oic.sec.encoding.base64",
7955                     "oic.sec.encoding.uri",
7956                     "oic.sec.encoding.handle",
7957                     "oic.sec.encoding.raw"
7958                 ],
7959                 "type": "string"
7960             },
7961             "handle": {
7962                 "description": "Handle to a key storage Resource.",
7963                 "type": "integer"
7964             }
7965         },
7966         "required": [
7967             "encoding"
7968         ],
7969         "type": "object"
7970     },
7971     "publicdata": {
7972         "description": "Public credential information.",
7973         "properties": {
7974             "data": {
7975                 "description": "This is the encoded value.",
7976                 "maxLength": 3072,
7977                 "type": "string"
7978             },
7979             "encoding": {
7980                 "description": "A string specifying the encoding format of the data contained in
7981 the pubdata.",
7982                 "x-detail-desc": [
7983                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7984                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7985                     "oic.sec.encoding.base64 - Base64 encoded object.",
7986                     "oic.sec.encoding.uri - URI reference.",
7987                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7988                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7989                     "oic.sec.encoding.raw - Raw hex encoded data."
7990                 ],
7991                 "enum": [
7992                     "oic.sec.encoding.jwt",
7993                     "oic.sec.encoding.cwt",
7994                     "oic.sec.encoding.base64",

```

```

7995         "oic.sec.encoding.uri",
7996         "oic.sec.encoding.pem",
7997         "oic.sec.encoding.der",
7998         "oic.sec.encoding.raw"
7999     ],
8000     "type": "string"
8001 }
8002 },
8003 "type": "object"
8004 },
8005 "roleid": {
8006     "description": "The role this credential possesses\nSecurity role specified as an
8007 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8008     "properties": {
8009         "authority": {
8010             "description": "The Authority component of the entity being identified. A NULL
8011 <Authority> refers to the local entity or Device.",
8012             "type": "string"
8013         },
8014         "role": {
8015             "description": "The ID of the role being identified.",
8016             "type": "string"
8017         }
8018     },
8019     "required": [
8020         "role"
8021     ],
8022     "type": "object"
8023 },
8024 "subjectuuid": {
8025     "anyOf": [
8026         {
8027             "description": "The id of the Device, which the cred entry applies to or \"*\n
8028 for wildcard identity.",
8029             "pattern": "^[^\\*$]",
8030             "type": "string"
8031         },
8032         {
8033             "description": "Format pattern according to IETF RFC 4122.",
8034             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8035 F0-9]{12}$",
8036             "type": "string"
8037         }
8038     ]
8039 }
8040 },
8041 "type": "object"
8042 },
8043 "type": "array"
8044 },
8045 "if": {
8046     "description": "The interface set supported by this Resource.",
8047     "items": {
8048         "enum": [
8049             "oic.if.baseline"
8050         ],
8051         "type": "string"
8052     },
8053     "minItems": 1,
8054     "readOnly": true,
8055     "type": "array"
8056 }
8057 },
8058 "type": "object",
8059 "required": ["roles"]
8060 },
8061 "Roles-update" : {
8062     "properties": {
8063         "roles": {
8064             "description": "List of role certificates.",
8065             "items": {

```



```

8066     "properties": {
8067         "credid": {
8068             "description": "Local reference to a credential Resource.",
8069             "type": "integer"
8070         },
8071         "credtype": {
8072             "description": "Representation of this credential's type\nCredential Types - Cred
8073 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8074 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
8075 password32 - Asymmetric encryption key.",
8076             "maximum": 63,
8077             "minimum": 0,
8078             "type": "integer"
8079         },
8080         "credusage": {
8081             "description": "A string that provides hints about how/where the cred is used\nThe
8082 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8083 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8084 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
8085             "enum": [
8086                 "oic.sec.cred.trustca",
8087                 "oic.sec.cred.cert",
8088                 "oic.sec.cred.rolecert",
8089                 "oic.sec.cred.mfgtrustca",
8090                 "oic.sec.cred.mfgcert"
8091             ],
8092             "type": "string"
8093         },
8094         "crms": {
8095             "description": "The refresh methods that may be used to update this credential.",
8096             "items": {
8097                 "description": "Each enum represents a method by which the credentials are
8098 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8099 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8100 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8101 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
8102                 "enum": [
8103                     "oic.sec.crm.pro",
8104                     "oic.sec.crm.psk",
8105                     "oic.sec.crm.rdp",
8106                     "oic.sec.crm.skdc",
8107                     "oic.sec.crm.pk10"
8108                 ],
8109                 "type": "string"
8110             },
8111             "type": "array"
8112         },
8113         "optionaldata": {
8114             "description": "Credential revocation status information\nOptional credential
8115 contents describes revocation status for this credential.",
8116             "properties": {
8117                 "data": {
8118                     "description": "This is the encoded structure.",
8119                     "type": "string"
8120                 },
8121                 "encoding": {
8122                     "description": "A string specifying the encoding format of the data contained in
8123 the optdata.",
8124                     "x-detail-desc": [
8125                         "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8126                         "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8127                         "oic.sec.encoding.base64 - Base64 encoded object.",
8128                         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8129                         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8130                         "oic.sec.encoding.raw - Raw hex encoded data."
8131                     ],
8132                     "enum": [
8133                         "oic.sec.encoding.jwt",
8134                         "oic.sec.encoding.cwt",
8135                         "oic.sec.encoding.base64",
8136                         "oic.sec.encoding.pem",

```

```

8137         "oic.sec.encoding.der",
8138         "oic.sec.encoding.raw"
8139     ],
8140     "type": "string"
8141 },
8142     "revstat": {
8143         "description": "Revocation status flag - true = revoked.",
8144         "type": "boolean"
8145     }
8146 },
8147     "required": [
8148         "revstat"
8149     ],
8150     "type": "object"
8151 },
8152     "period": {
8153         "description": "String with RFC5545 Period.",
8154         "type": "string"
8155     },
8156     "privatedata": {
8157         "description": "Private credential information\nCredential Resource non-public
8158 contents.",
8159         "properties": {
8160             "data": {
8161                 "description": "The encoded value.",
8162                 "maxLength": 3072,
8163                 "type": "string"
8164             },
8165             "encoding": {
8166                 "description": "A string specifying the encoding format of the data contained in
8167 the privdata.",
8168                 "x-detail-desc": [
8169                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8170                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8171                     "oic.sec.encoding.base64 - Base64 encoded object.",
8172                     "oic.sec.encoding.uri - URI reference.",
8173                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
8174 referenced using a handle.",
8175                     "oic.sec.encoding.raw - Raw hex encoded data."
8176                 ],
8177                 "enum": [
8178                     "oic.sec.encoding.jwt",
8179                     "oic.sec.encoding.cwt",
8180                     "oic.sec.encoding.base64",
8181                     "oic.sec.encoding.uri",
8182                     "oic.sec.encoding.handle",
8183                     "oic.sec.encoding.raw"
8184                 ],
8185                 "type": "string"
8186             },
8187             "handle": {
8188                 "description": "Handle to a key storage Resource.",
8189                 "type": "integer"
8190             }
8191         },
8192         "required": [
8193             "encoding"
8194         ],
8195         "type": "object"
8196     },
8197     "publicdata": {
8198         "description": "Public credential information.",
8199         "properties": {
8200             "data": {
8201                 "description": "The encoded value.",
8202                 "maxLength": 3072,
8203                 "type": "string"
8204             },
8205             "encoding": {
8206                 "description": "A string specifying the encoding format of the data contained in
8207 the pubdata.",

```

```

8208         "x-detail-desc": [
8209             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8210             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8211             "oic.sec.encoding.base64 - Base64 encoded object.",
8212             "oic.sec.encoding.uri - URI reference.",
8213             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8214             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8215             "oic.sec.encoding.raw - Raw hex encoded data."
8216         ],
8217         "enum": [
8218             "oic.sec.encoding.jwt",
8219             "oic.sec.encoding.cwt",
8220             "oic.sec.encoding.base64",
8221             "oic.sec.encoding.uri",
8222             "oic.sec.encoding.pem",
8223             "oic.sec.encoding.der",
8224             "oic.sec.encoding.raw"
8225         ],
8226         "type": "string"
8227     }
8228 },
8229 "type": "object"
8230 },
8231 "roleid": {
8232     "description": "The role this credential possesses\nSecurity role specified as an
8233 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8234     "properties": {
8235         "authority": {
8236             "description": "The Authority component of the entity being identified. A NULL
8237 <Authority> refers to the local entity or Device.",
8238             "type": "string"
8239         },
8240         "role": {
8241             "description": "The ID of the role being identified.",
8242             "type": "string"
8243         }
8244     },
8245     "required": [
8246         "role"
8247     ],
8248     "type": "object"
8249 },
8250 "subjectuuid": {
8251     "anyOf": [
8252         {
8253             "description": "The id of the Device, which the cred entry applies to or \"*\n
8254 for wildcard identity.",
8255             "pattern": "^[\\*$]",
8256             "type": "string"
8257         },
8258         {
8259             "description": "Format pattern according to IETF RFC 4122.",
8260             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8261 F0-9]{12}$",
8262             "type": "string"
8263         }
8264     ]
8265 },
8266 },
8267 "type": "object"
8268 },
8269 "type": "array"
8270 }
8271 },
8272 "type": "object",
8273 "required": ["roles"]
8274 }
8275 }
8276 }
8277 }

```

8278 **C.11.5 Property definition**

8279 Table C.18 defines the Properties that are part of the "oic.r.roles" Resource Type.

8280 **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.

8281 **C.11.6 CRUDN behaviour**

8282 Table C.19 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

8283 **Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

8284 **C.12 Signed Access Control List**

8285 **C.12.1 Introduction**

8286 This Resource specifies a signed ACL object.

8287

8288 **C.12.2 Well-known URI**

8289 /oic/sec/sacl

8290 **C.12.3 Resource type**

8291 The Resource Type is defined as: "oic.r.sacl".

8292 **C.12.4 OpenAPI 2.0 definition**

```

8293 {
8294   "swagger": "2.0",
8295   "info": {
8296     "title": "Signed Access Control List",
8297     "version": "v1.0-20150819",
8298     "license": {
8299       "name": "OCF Data Model License",
8300       "url":
8301         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8302         CENSE.md",
8303       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8304         reserved."
8305     },
8306     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8307   },
8308   "schemes": ["http"],
8309   "consumes": ["application/json"],
8310   "produces": ["application/json"],
8311   "paths": {
8312     "/oic/sec/sacl" : {

```

```

8313 "get": {
8314   "description": "This Resource specifies a signed ACL object.\n",
8315   "parameters": [
8316     {"$ref": "#/parameters/interface"}
8317   ],
8318   "responses": {
8319     "200": {
8320       "description": "",
8321       "x-example":
8322         {
8323           "rt": ["oic.r.sacl"],
8324           "aclist2": [
8325             {
8326               "aceid": 1,
8327               "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8328               "resources": [
8329                 {
8330                   "href": "/temp",
8331                   "rt": ["oic.r.temperature"],
8332                   "if": ["oic.if.baseline", "oic.if.a"]
8333                 },
8334                 {
8335                   "href": "/temp",
8336                   "rt": ["oic.r.temperature"],
8337                   "if": ["oic.if.baseline", "oic.if.s"]
8338                 }
8339               ],
8340               "permission": 31,
8341               "validity": [
8342                 {
8343                   "period": "20160101T180000Z/20170102T070000Z",
8344                   "recurrence": [ "DSTART:XXXXX",
8345 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8346                 },
8347                 {
8348                   "period": "20160101T180000Z/PT5H30M",
8349                   "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8350                 }
8351               ]
8352             },
8353             {
8354               "aceid": 2,
8355               "subject": {
8356                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8357                 "role": "SOME_STRING"
8358               },
8359               "resources": [
8360                 {
8361                   "href": "/light",
8362                   "rt": ["oic.r.light"],
8363                   "if": ["oic.if.baseline", "oic.if.a"]
8364                 },
8365                 {
8366                   "href": "/door",
8367                   "rt": ["oic.r.door"],
8368                   "if": ["oic.if.baseline", "oic.if.a"]
8369                 }
8370               ],
8371               "permission": 15
8372             }
8373           ],
8374           "signature": {
8375             "sigtype": "oic.sec.sigtype.pk7",
8376             "sigvalue": "ENCODED-SIGNATURE-VALUE"
8377           }
8378         },
8379         "schema": { "$ref": "#/definitions/Sacl" }
8380       }
8381     },
8382   },
8383   "post": {

```

```

8384     "description": "Sets the sacl Resource data.\n",
8385     "parameters": [
8386         {"$ref": "#/parameters/interface"},
8387         {
8388             "name": "body",
8389             "in": "body",
8390             "required": true,
8391             "schema": { "$ref": "#/definitions/Sacl" },
8392             "x-example":
8393                 {
8394                     "aclist2": [
8395                         {
8396                             "aceid": 1,
8397                             "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8398                             "resources": [
8399                                 {
8400                                     "href": "/temp",
8401                                     "rt": ["oic.r.temperature"],
8402                                     "if": ["oic.if.baseline", "oic.if.a"]
8403                                 },
8404                                 {
8405                                     "href": "/temp",
8406                                     "rt": ["oic.r.temperature"],
8407                                     "if": ["oic.if.baseline", "oic.if.s"]
8408                                 }
8409                             ],
8410                             "permission": 31,
8411                             "validity": [
8412                                 {
8413                                     "period": "20160101T180000Z/20170102T070000Z",
8414                                     "recurrence": [ "DSTART:XXXXX",
8415 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8416                                 },
8417                                 {
8418                                     "period": "20160101T180000Z/PT5H30M",
8419                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8420                                 }
8421                             ]
8422                         },
8423                         {
8424                             "aceid": 2,
8425                             "subject": {
8426                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8427                                 "role": "SOME_STRING"
8428                             },
8429                             "resources": [
8430                                 {
8431                                     "href": "/light",
8432                                     "rt": ["oic.r.light"],
8433                                     "if": ["oic.if.baseline", "oic.if.a"]
8434                                 },
8435                                 {
8436                                     "href": "/door",
8437                                     "rt": ["oic.r.door"],
8438                                     "if": ["oic.if.baseline", "oic.if.a"]
8439                                 }
8440                             ],
8441                             "permission": 15
8442                         }
8443                     ],
8444                     "signature": {
8445                         "sigtype": "oic.sec.sigtype.pk7",
8446                         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8447                     }
8448                 }
8449             ],
8450             "responses": {
8451                 "400": {
8452                     "description": "The request is invalid."
8453                 }
8454             }

```

```

8455         "201": {
8456             "description": "The ACL entry is created."
8457         },
8458         "204": {
8459             "description": "The ACL entry is updated."
8460         }
8461     },
8462 },
8463 "put": {
8464     "description": "Sets the sacl Resource data\n",
8465     "parameters": [
8466         { "$ref": "#/parameters/interface" },
8467         {
8468             "name": "body",
8469             "in": "body",
8470             "required": true,
8471             "schema": { "$ref": "#/definitions/Sacl" },
8472             "x-example":
8473                 {
8474                     "aclist2": [
8475                         {
8476                             "aceid": 1,
8477                             "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8478                             "resources": [
8479                                 {
8480                                     "href": "/temp",
8481                                     "rt": ["oic.r.temperature"],
8482                                     "if": ["oic.if.baseline", "oic.if.a"]
8483                                 },
8484                                 {
8485                                     "href": "/temp",
8486                                     "rt": ["oic.r.temperature"],
8487                                     "if": ["oic.if.baseline", "oic.if.s"]
8488                                 }
8489                             ],
8490                             "permission": 31,
8491                             "validity": [
8492                                 {
8493                                     "period": "20160101T180000Z/20170102T070000Z",
8494                                     "recurrence": [ "DSTART:XXXXX",
8495 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8496                                 },
8497                                 {
8498                                     "period": "20160101T180000Z/PT5H30M",
8499                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8500                                 }
8501                             ]
8502                         },
8503                     {
8504                         "aceid": 2,
8505                         "subject": {
8506                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8507                             "role": "SOME_STRING"
8508                         },
8509                         "resources": [
8510                             {
8511                                 "href": "/light",
8512                                 "rt": ["oic.r.light"],
8513                                 "if": ["oic.if.baseline", "oic.if.a"]
8514                             },
8515                             {
8516                                 "href": "/door",
8517                                 "rt": ["oic.r.door"],
8518                                 "if": ["oic.if.baseline", "oic.if.a"]
8519                             }
8520                         ],
8521                         "permission": 15
8522                     }
8523                 ],
8524                 "signature": {
8525                     "sigtype": "oic.sec.sigtype.pk7",

```

```

8526         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8527     }
8528 }
8529 }
8530 ],
8531 "responses": {
8532     "400": {
8533         "description": "The request is invalid."
8534     },
8535     "201": {
8536         "description": "The signed ACL entry is created."
8537     }
8538 }
8539 },
8540 "delete": {
8541     "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
8542 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
8543 specified, only the matched entry is deleted.\n",
8544     "parameters": [
8545         { "$ref": "#/parameters/interface" },
8546         {
8547             "in": "query",
8548             "description": "Delete the signed ACL identified by the string containing subject
8549 UUID.\n",
8550             "type": "string",
8551             "name": "subject"
8552         }
8553     ],
8554     "responses": {
8555         "200": {
8556             "description": "The signed ACL instance or the the entire signed ACL Resource has
8557 been successfully deleted."
8558         },
8559         "400": {
8560             "description": "The request is invalid."
8561         }
8562     }
8563 }
8564 }
8565 },
8566 "parameters": {
8567     "interface": {
8568         "in": "query",
8569         "name": "if",
8570         "type": "string",
8571         "enum": ["oic.if.baseline"]
8572     }
8573 },
8574 "definitions": {
8575     "Sacl": {
8576         "properties": {
8577             "rt": {
8578                 "description": "Resource Type of the Resource.",
8579                 "items": {
8580                     "maxLength": 64,
8581                     "type": "string",
8582                     "enum": ["oic.r.sacl"]
8583                 },
8584                 "minItems": 1,
8585                 "readOnly": true,
8586                 "type": "array"
8587             },
8588             "aclist2": {
8589                 "description": "Access Control Entries in the ACL Resource.",
8590                 "items": {
8591                     "properties": {
8592                         "aceid": {
8593                             "description": "An identifier for the ACE that is unique within the ACL. In cases
8594 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
8595                             "minimum": 1,
8596                             "type": "integer"

```



```

8597     },
8598     "permission": {
8599         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
8600 permissions.",
8601         "x-detail-desc": [
8602             "0 - No permissions.",
8603             "1 - Create permission is granted.",
8604             "2 - Read, observe, discover permission is granted.",
8605             "4 - Write, update permission is granted.",
8606             "8 - Delete permission is granted.",
8607             "16 - Notify permission is granted."
8608         ],
8609         "maximum": 31,
8610         "minimum": 0,
8611         "type": "integer"
8612     },
8613     "resources": {
8614         "description": "References the application's Resources to which a security policy
8615 applies.",
8616         "items": {
8617             "description": "Each Resource must have at least one of these properties set.",
8618             "properties": {
8619                 "href": {
8620                     "allOf": [
8621                         {
8622                             "description": "When present, the ACE only applies when the href matches."
8623                         },
8624                         {
8625                             "description": "This is the target URI, it can be specified as a Relative
8626 Reference or fully-qualified URI.",
8627                             "format": "uri",
8628                             "maxLength": 256,
8629                             "type": "string"
8630                         }
8631                     ]
8632                 },
8633                 "if": {
8634                     "description": "When present, the ACE only applies when the if (interface)
8635 matches\nThe interface set supported by this Resource.",
8636                     "items": {
8637                         "enum": [
8638                             "oic.if.baseline",
8639                             "oic.if.ll",
8640                             "oic.if.b",
8641                             "oic.if.rw",
8642                             "oic.if.r",
8643                             "oic.if.a",
8644                             "oic.if.s"
8645                         ],
8646                         "type": "string"
8647                     },
8648                     "minItems": 1,
8649                     "type": "array"
8650                 },
8651                 "rt": {
8652                     "description": "When present, the ACE only applies when the rt (resource type)
8653 matches\nResource Type of the Resource.",
8654                     "items": {
8655                         "maxLength": 64,
8656                         "type": "string"
8657                     },
8658                     "minItems": 1,
8659                     "type": "array"
8660                 },
8661                 "wc": {
8662                     "description": "A wildcard matching policy.",
8663                     "pattern": "^[-*]*$",
8664                     "type": "string"
8665                 }
8666             },
8667             "type": "object"

```

```

8668     },
8669     "type": "array"
8670 },
8671 "subject": {
8672   "anyOf": [
8673     {
8674       "description": "Device identifier.",
8675       "properties": {
8676         "uuid": {
8677           "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4122.",
8679           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
8680 fA-F0-9]{12}$",
8681           "type": "string"
8682         }
8683       },
8684       "required": [
8685         "uuid"
8686       ],
8687       "type": "object"
8688     },
8689     {
8690       "description": "Security role specified as an <Authority> & <Rolename>. A NULL
8691 <Authority> refers to the local entity or Device.",
8692       "properties": {
8693         "authority": {
8694           "description": "The Authority component of the entity being identified. A
8695 NULL <Authority> refers to the local entity or Device.",
8696           "type": "string"
8697         },
8698         "role": {
8699           "description": "The ID of the role being identified.",
8700           "type": "string"
8701         }
8702       },
8703       "required": [
8704         "role"
8705       ],
8706       "type": "object"
8707     },
8708     {
8709       "properties": {
8710         "conntype": {
8711           "description": "This property allows an ACE to be matched based on the
8712 connection or message type.",
8713           "x-detail-desc": [
8714             "auth-crypt - ACE applies if the Client is authenticated and the data
8715 channel or message is encrypted and integrity protected.",
8716             "anon-clear - ACE applies if the Client is not authenticated and the data
8717 channel or message is not encrypted but may be integrity protected."
8718           ],
8719           "enum": [
8720             "auth-crypt",
8721             "anon-clear"
8722           ],
8723           "type": "string"
8724         }
8725       },
8726       "required": [
8727         "conntype"
8728       ],
8729       "type": "object"
8730     }
8731   ]
8732 },
8733 "validity": {
8734   "description": "validity is an array of time-pattern objects.",
8735   "items": {
8736     "description": "The time-pattern contains a period and recurrence expressed in
8737 RFC5545 syntax.",
8738     "properties": {

```

```

8739         "period": {
8740             "description": "String represents a period using the RFC5545 Period.",
8741             "type": "string"
8742         },
8743         "recurrence": {
8744             "description": "String array represents a recurrence rule using the RFC5545
Recurrence.",
8745             "items": {
8746                 "type": "string"
8747             },
8748             "type": "array"
8749         }
8750     },
8751     "required": [
8752         "period"
8753     ],
8754     "type": "object"
8755 },
8756 "type": "array"
8757 },
8758 },
8759 },
8760 "required": [
8761     "aceid",
8762     "resources",
8763     "permission",
8764     "subject"
8765 ],
8766 "type": "object"
8767 },
8768 "type": "array"
8769 },
8770 "n": {
8771     "$ref":
8772     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8773     schema.json#/definitions/n"
8774 },
8775 "id": {
8776     "$ref":
8777     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8778     schema.json#/definitions/id"
8779 },
8780 "signature": {
8781     "description": "The signature over the ACL Resource\nEncoded signature data.",
8782     "properties": {
8783         "sigtype": {
8784             "description": "The string specifies the predefined signature format.",
8785             "x-detail-desc": [
8786                 "RFC7515 JSON web signature (JWS) object.",
8787                 "RFC2315 base64 encoded object.",
8788                 "CBOR encoded JWS object."
8789             ],
8790             "enum": [
8791                 "oic.sec.sigtype.jws",
8792                 "oic.sec.sigtype.pk7",
8793                 "oic.sec.sigtype.cws"
8794             ],
8795             "type": "string"
8796         },
8797         "sigvalue": {
8798             "description": "The encoded signature.",
8799             "type": "string"
8800         }
8801     },
8802     "required": [
8803         "sigtype",
8804         "sigvalue"
8805     ],
8806     "type": "object"
8807 },
8808 "if": {
8809     "description": "The interface set supported by this Resource.",

```

```

8810         "items": {
8811             "enum": [
8812                 "oic.if.baseline"
8813             ],
8814             "type": "string"
8815         },
8816         "minItems": 1,
8817         "readOnly": true,
8818         "type": "array"
8819     }
8820 },
8821 "type" : "object",
8822 "required": ["aclist2", "signature"]
8823 }
8824 }
8825 }
8826

```

8827 **C.12.5 Property definition**

8828 Table C.20 defines the Properties that are part of the "oic.r.sacl" Resource Type.

8829 **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	

8830 **C.12.6 CRUDN behaviour**

8831 Table C.21 defines the CRUDN operations that are supported on the "oic.r.sacl" Resource Type.

8832 **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

8833 **C.13 Session**

8834 **C.13.1 Introduction**

8835 Resource that manages the persistent session between a Device and OCF Cloud.

8836 **C.13.2 Well-known URI**

8837 /oic/sec/session

8838 **C.13.3 Resource type**

8839 The Resource Type is defined as: "oic.r.session".

8840 C.13.4 OpenAPI 2.0 definition

```
8841 {
8842   "swagger": "2.0",
8843   "info": {
8844     "title": "Session",
8845     "version": "v1.0-20181001",
8846     "license": {
8847       "name": "OCF Data Model License",
8848       "url":
8849 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8850 CENSE.md",
8851     "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8852 reserved."
8853   },
8854   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8855 },
8856 "schemes": ["http"],
8857 "consumes": ["application/json"],
8858 "produces": ["application/json"],
8859 "paths": {
8860   "/oic/sec/session" : {
8861     "post": {
8862       "description": "Resource that manages the persistent session between a Device and OCF
8863 Cloud.",
8864       "parameters": [
8865         {"$ref": "#/parameters/interface"},
8866         {
8867           "name": "body",
8868           "in": "body",
8869           "required": true,
8870           "schema": { "$ref": "#/definitions/Account-Session-Request" },
8871           "x-example":
8872             {
8873               "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
8874               "di" : "9cfbeb8e-5a1e-4d1c-9d01-00c04fd430c8",
8875               "accesstoken" : "0f3d9f7fe5491d54077d",
8876               "login" : true
8877             }
8878         }
8879       ],
8880       "responses": {
8881         "204": {
8882           "description" : "",
8883           "x-example":
8884             {
8885               "rt": ["oic.r.session"],
8886               "expiresin" : 3600
8887             },
8888           "schema": { "$ref": "#/definitions/Account-Session-Response" }
8889         }
8890       }
8891     }
8892   },
8893 },
8894 "parameters": {
8895   "interface" : {
8896     "in" : "query",
8897     "name" : "if",
8898     "type" : "string",
8899     "enum" : ["oic.if.baseline"]
8900   }
8901 },
8902 "definitions": {
8903   "Account-Session-Request" : {
8904     "properties": {
8905       "uid": {
8906         "description": "Format pattern according to IETF RFC 4122.",
8907         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
8908         "type": "string"
8909       },

```

```

8910     "di": {
8911         "description": "The Device ID\nFormat 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}$",
8913         "type": "string"
8914     },
8915     "accesstoken": {
8916         "description": "Access-Token used to grant access right for the Device to sign-in.",
8917         "pattern": "(?!$|\\s+).*",
8918         "type": "string"
8919     },
8920     "login": {
8921         "description": "Action for the request: true = login, false = logout.",
8922         "type": "boolean"
8923     }
8924 },
8925 "type" : "object",
8926 "required": ["uid", "di", "accesstoken", "login"]
8927 },
8928 "Account-Session-Response" : {
8929     "properties": {
8930         "expiresin": {
8931             "description": "Access-Token remaining life time in seconds (-1 if permanent).",
8932             "readOnly": true,
8933             "type": "integer"
8934         },
8935         "rt": {
8936             "description": "Resource Type of the Resource.",
8937             "items": {
8938                 "maxLength": 64,
8939                 "type": "string",
8940                 "enum": ["oic.r.session"]
8941             },
8942             "minItems": 1,
8943             "readOnly": true,
8944             "type": "array"
8945         },
8946         "n": {
8947             "$ref":
8948 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8949 schema.json#/definitions/n"
8950         },
8951         "id": {
8952             "$ref":
8953 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8954 schema.json#/definitions/id"
8955         },
8956         "if": {
8957             "description": "The interface set supported by this Resource.",
8958             "items": {
8959                 "enum": [
8960                     "oic.if.baseline"
8961                 ],
8962                 "type": "string"
8963             },
8964             "minItems": 1,
8965             "readOnly": true,
8966             "type": "array"
8967         }
8968     },
8969     "type" : "object",
8970     "required" : ["expiresin"]
8971 }
8972 }
8973 }
8974

```

8975 C.13.5 Property definition

8976 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.

8978 **C.13.6 CRUDN behaviour**8979 Table C.23 defines the CRUDN operations that are supported on the "oic.r.session" Resource
8980 Type.8981 **Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.session".**

Create	Read	Update	Delete	Notify
		post		

8982 **C.14 Security Profile**8983 **C.14.1 Introduction**

8984 Resource specifying supported and active security profile(s).

8985

8986 **C.14.2 Well-known URI**

8987 /oic/sec/sp

8988 **C.14.3 Resource type**

8989 The Resource Type is defined as: "oic.r.sp".

8990 C.14.4 OpenAPI 2.0 definition

```
8991 {
8992   "swagger": "2.0",
8993   "info": {
8994     "title": "Security Profile",
8995     "version": "v1.0-20190208",
8996     "license": {
8997       "name": "OCF Data Model License",
8998       "url":
8999         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9000 CENSE.md",
9001       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9002 reserved."
9003     },
9004     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9005   },
9006   "schemes": ["http"],
9007   "consumes": ["application/json"],
9008   "produces": ["application/json"],
9009   "paths": {
9010     "/oic/sec/sp" : {
9011       "get": {
9012         "description": "Resource specifying supported and active security profile(s).\n",
9013         "parameters": [
9014           { "$ref": "#/parameters/interface" }
9015         ],
9016         "responses": {
9017           "200": {
9018             "description": "",
9019             "x-example":
9020               {
9021                 "rt": ["oic.r.sp"],
9022                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9023                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9024               },
9025             "schema": { "$ref": "#/definitions/SP" }
9026           },
9027           "400": {
9028             "description": "The request is invalid."
9029           }
9030         }
9031       },
9032       "post": {
9033         "description": "Sets or updates Device provisioning status data.\n",
9034         "parameters": [
9035           { "$ref": "#/parameters/interface" },
9036           {
9037             "name": "body",
9038             "in": "body",
9039             "required": true,
9040             "schema": { "$ref": "#/definitions/SP-Update" },
9041             "x-example":
9042               {
9043                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9044                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9045               }
9046           }
9047         ],
9048         "responses": {
9049           "200": {
9050             "description": "",
9051             "x-example":
9052               {
9053                 "rt": ["oic.r.sp"],
9054                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9055                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9056               },
9057             "schema": { "$ref": "#/definitions/SP" }
9058           },
9059           "400": {
```



```

9060         "description" : "The request is invalid."
9061     }
9062 }
9063 }
9064 },
9065 },
9066 "parameters": {
9067     "interface" : {
9068         "in" : "query",
9069         "name" : "if",
9070         "type" : "string",
9071         "enum" : ["oic.if.baseline"]
9072     }
9073 },
9074 "definitions": {
9075     "SP" : {
9076         "properties": {
9077             "rt": {
9078                 "description": "Resource Type of the Resource.",
9079                 "items": {
9080                     "maxLength": 64,
9081                     "type": "string",
9082                     "enum": ["oic.r.sp"]
9083                 },
9084                 "minItems": 1,
9085                 "readOnly": true,
9086                 "type": "array"
9087             },
9088             "n": {
9089                 "$ref":
9090 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9091 schema.json#/definitions/n"
9092             },
9093             "id": {
9094                 "$ref":
9095 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9096 schema.json#/definitions/id"
9097             },
9098             "currentprofile": {
9099                 "description": "Security Profile currently active.",
9100                 "type": "string"
9101             },
9102             "supportedprofiles": {
9103                 "description": "Array of supported Security Profiles.",
9104                 "items": {
9105                     "type": "string"
9106                 },
9107                 "type": "array"
9108             },
9109             "if": {
9110                 "description": "The interface set supported by this Resource.",
9111                 "items": {
9112                     "enum": [
9113                         "oic.if.baseline"
9114                     ],
9115                     "type": "string"
9116                 },
9117                 "minItems": 1,
9118                 "readOnly": true,
9119                 "type": "array"
9120             }
9121         },
9122         "type" : "object",
9123         "required": ["supportedprofiles", "currentprofile"]
9124     },
9125     "SP-Update" : {
9126         "properties": {
9127             "currentprofile": {
9128                 "description": "Security Profile currently active.",
9129                 "type": "string"
9130             },

```

```

9131     "supportedprofiles": {
9132       "description": "Array of supported Security Profiles.",
9133       "items": {
9134         "type": "string"
9135       },
9136       "type": "array"
9137     },
9138   },
9139   "type" : "object"
9140 }
9141 }
9142 }
9143

```

9144 **C.14.5 Property definition**

9145 Table C.24 defines the Properties that are part of the "oic.r.sp" Resource Type.

9146 **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.

9147 **C.14.6 CRUDN behaviour**

9148 Table C.25 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

9149 **Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".**

Create	Read	Update	Delete	Notify
	get	post		observe

9150 **C.15 Token Refresh**

9151 **C.15.1 Introduction**

9152 Obtain fresh access-token using the refresh token, client should refresh access-token before it expires.

9154 **C.15.2 Well-known URI**

9155 /oic/sec/tokenrefresh

9156 **C.15.3 Resource type**

9157 The Resource Type is defined as: "oic.r.tokenrefresh".

9158 C.15.4 OpenAPI 2.0 definition

```
9159 {
9160   "swagger": "2.0",
9161   "info": {
9162     "title": "Token Refresh",
9163     "version": "v1.0-20181001",
9164     "license": {
9165       "name": "OCF Data Model License",
9166       "url":
9167         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9168         CENSE.md",
9169       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9170       reserved."
9171     },
9172     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9173   },
9174   "schemes": ["http"],
9175   "consumes": ["application/json"],
9176   "produces": ["application/json"],
9177   "paths": {
9178     "/oic/sec/tokenrefresh" : {
9179       "post": {
9180         "description": "Obtain fresh access-token using the refresh token, client should refresh
9181         access-token before it expires.\n",
9182         "parameters": [
9183           { "$ref": "#/parameters/interface" },
9184           {
9185             "name": "body",
9186             "in": "body",
9187             "required": true,
9188             "schema": { "$ref": "#/definitions/TokenRefresh-Request" },
9189             "x-example":
9190               {
9191                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9192                 "di" : "9cfbeb8e-5a1e-4d1c-9d01-00c04fd430c8",
9193                 "refreshtoken" : "00fe4644a6f5324eec"
9194               }
9195           }
9196         ],
9197         "responses": {
9198           "204": {
9199             "description": "2.04 Changed respond with new access-token.\n",
9200             "x-example":
9201               {
9202                 "rt": ["oic.r.tokenrefresh"],
9203                 "accesstoken" : "8ce598980761869837be",
9204                 "refreshtoken" : "d4922312b6df0518e146",
9205                 "expiresin" : 3600
9206               }
9207             ,
9208             "schema": { "$ref": "#/definitions/TokenRefresh-Response" }
9209           }
9210         }
9211       }
9212     }
9213   },
9214   "parameters": {
9215     "interface" : {
9216       "in" : "query",
9217       "name" : "if",
9218       "type" : "string",
9219       "enum" : ["oic.if.baseline"]
9220     }
9221   },
9222   "definitions": {
9223     "TokenRefresh-Request" : {
9224       "properties": {
9225         "refreshtoken": {
9226           "description": "Refresh token received by account management or during token refresh
9227           procedure.",

```

```

9228         "pattern": "(?!$|\\s+).*",
9229         "type": "string"
9230     },
9231     "uid": {
9232         "description": "Format pattern according to IETF RFC 4122.",
9233         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9234         "type": "string"
9235     },
9236     "di": {
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}$",
9239         "type": "string"
9240     }
9241 },
9242 "type": "object",
9243 "required": ["uid", "di", "refreshToken"]
9244 },
9245 "TokenRefresh-Response" : {
9246     "properties": {
9247         "expiresin": {
9248             "description": "Access-Token life time in seconds (-1 if permanent).",
9249             "readOnly": true,
9250             "type": "integer"
9251         },
9252         "rt": {
9253             "description": "Resource Type of the Resource.",
9254             "items": {
9255                 "maxLength": 64,
9256                 "type": "string",
9257                 "enum": ["oic.r.tokenrefresh"]
9258             },
9259             "minItems": 1,
9260             "readOnly": true,
9261             "type": "array"
9262         },
9263         "refreshToken": {
9264             "description": "Refresh token received by account management or during token refresh
9265 procedure.",
9266             "pattern": "(?!$|\\s+).*",
9267             "type": "string"
9268         },
9269         "accessToken": {
9270             "description": "Granted Access-Token.",
9271             "pattern": "(?!$|\\s+).*",
9272             "readOnly": true,
9273             "type": "string"
9274         },
9275         "n": {
9276             "$ref":
9277 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9278 schema.json#/definitions/n"
9279         },
9280         "id": {
9281             "$ref":
9282 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9283 schema.json#/definitions/id"
9284         },
9285         "if" :
9286         {
9287             "description": "The interface set supported by this Resource.",
9288             "items": {
9289                 "enum": [
9290                     "oic.if.baseline"
9291                 ],
9292                 "type": "string"
9293             },
9294             "minItems": 1,
9295             "readOnly": true,
9296             "type": "array"
9297         }
9298     },

```

```

9299     "type" : "object",
9300     "required": ["accesstoken", "refreshtoken", "expiresin"]
9301   }
9302 }
9303 }
9304

```

9305 **C.15.5 Property definition**

9306 Table C.26 defines the Properties that are part of the "oic.r.tokenrefresh" Resource Type.

9307 **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	

9308 **C.15.6 CRUDN behaviour**

9309 Table C.27 defines the CRUDN operations that are supported on the "oic.r.tokenrefresh"
 9310 Resource Type.

9311 **Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh".**

Create	Read	Update	Delete	Notify
		post		

9312
9313
9314
9315

Annex D (informative)

OID definitions

9316 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to
9317 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of
9318 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
9319 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
9320     private(4) enterprise(1) OCF(51414) }
9321
9322 -- OCF Security specific OIDs
9323
9324 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
9325 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9326
9327 -- OCF Security Categories
9328
9329 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
9330 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
9331
9332 -- OCF Security Profiles
9333
9334 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
9335 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
9336 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
9337 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
9338 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
9339
9340 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)
9341 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
9342 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
9343 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
9344 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
9345
9346 ocfSecurityProfileOID ::= UTF8String
9347
9348 -- OCF Security Certificate Policies
9349
9350 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
9351
9352 -- OCF X.509v3 Extensions
9353
9354 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9355 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
9356 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
9357 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
9358
9359 ocfVersion ::= SEQUENCE {
9360     major    INTEGER,
9361     minor    INTEGER,
9362     build    INTEGER}
9363
9364 ocfCompliance ::= SEQUENCE {
9365     version      ocfVersion,
9366     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
9367     deviceName   UTF8String,
9368     deviceManufacturer UTF8String}
9369
9370 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
9371 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
9372
9373 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
9374
9375 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
9376
9377 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
9378 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
9379 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
9380
9381 ocfCPLAttributes ::= SEQUENCE {
9382     cpl-at-IANAPen UTF8String,
9383     cpl-at-model UTF8String,
9384     cpl-at-version UTF8String}
```

**Annex E
(informative)**

Security considerations specific to Bridged Protocols

9385
9386
9387
9388

9389 The text in this Annex is provided for information only. This Annex has no normative impact. This
9390 information is applicable at the time of initial publication and may become out of date.

9391 **E.1 Security Considerations specific to the AllJoyn Protocol**

9392 This clause intentionally left empty.

9393 **E.2 Security Considerations specific to the Bluetooth LE Protocol**

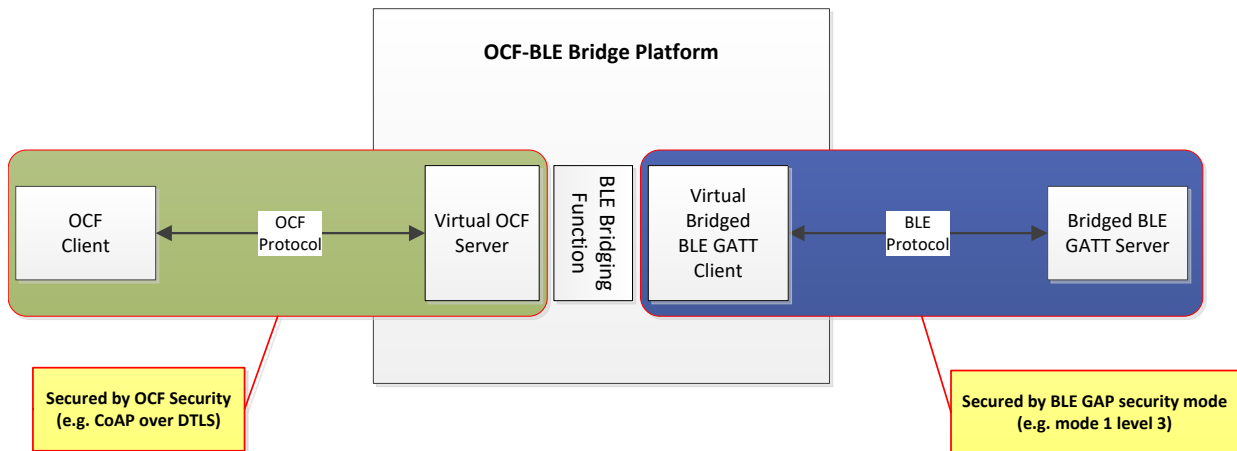
9394 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security
9395 mode has several security levels (see Table E.1)

9396 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF
9397 perspective. The appropriate selection of security mode and level is left to the vendor.

9398 **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)

9399 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are
9400 secured by their own security.



9401
9402

Figure E-1 Security Considerations for BLE Bridge

9403 **E.3 Security Considerations specific to the oneM2M Protocol**

9404 This clause intentionally left empty.

9406 **E.4 Security Considerations specific to the U+ Protocol**

9407 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

9408 **Table E.2 TLS 1.2 Cipher Suites used by U+**

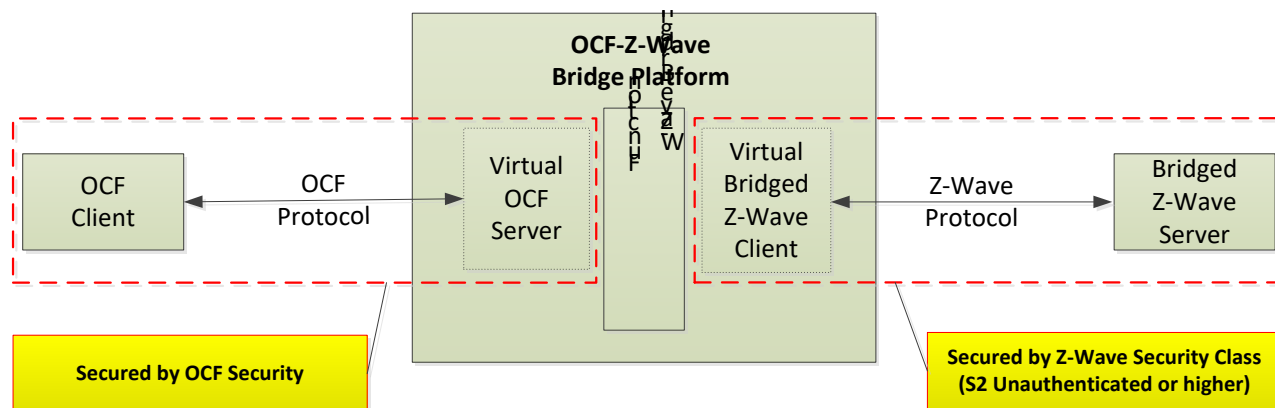
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

9409 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

9410 **E.5 Security Considerations specific to the Z-Wave Protocol**

9411 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2
 9412 Security Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for
 9413 communication with a Virtual Bridged Client would typically be considered secure from an OCF
 9414 perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

9415 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their
 9416 own security.



9417
9418

9419

Figure E-2 Security Considerations for Z-Wave Bridge

9420 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2
9421 Unauthenticated provides the following advantages from the security perspective;

- 9422 – The unique device specific key for every secure device enables validation of device identity
9423 and prevents man-in-the-middle compromises to security
- 9424 – The Secure cryptographic key exchange methods during inclusion achieves high level of
9425 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 9426 – Out of band key exchange for product authentication which is combined with device specific
9427 key prevents eavesdropping and man-in-the-middle attack vectors.

9428 See Table E.3 for a summary of Z-Wave Security Classes.

9429

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

9430 On the other hand, S0 Security Class has the vulnerability of security during inclusion by
9431 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the
9432 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices
9433 might be no longer secure in that case.

9434 E.6 Security Considerations specific to the Zigbee Protocol

9435 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the
9436 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0
9437 stack, "nwkSecurityLevel", represents the security level of a device.

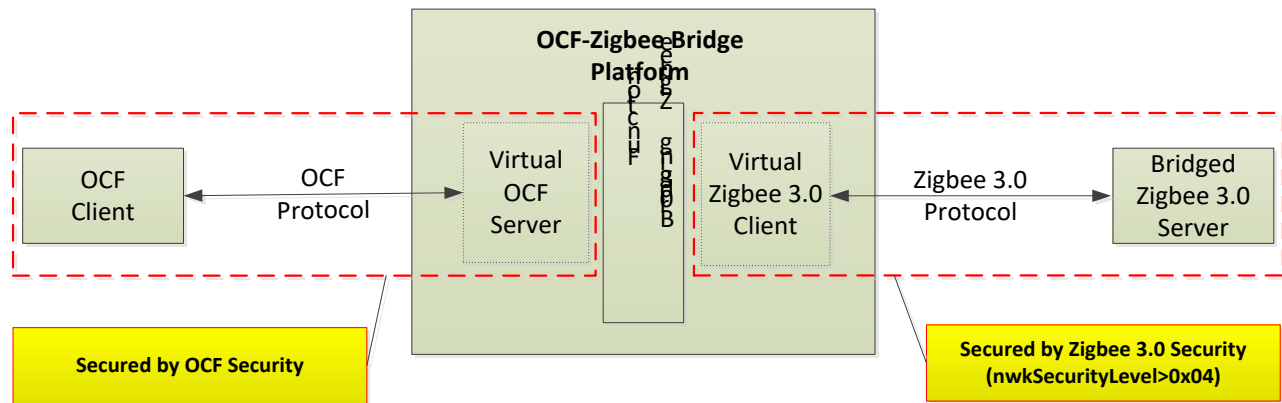
9438 The security level `nwkSecurityLevel > 0x04` provides message integrity code (MIC) and/or
 9439 AES128-CCM encryption (ENC). Zigbee Servers using `nwkSecurityLevel > 0x04` would typically
 9440 be considered secure from an OCF perspective. The appropriate selection for `nwkSecurityLevel`
 9441 is left to the vendor.

9442 See Table E.4 for a summary of the Zigbee Security Levels.

9443 **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)

9444 Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are
 9445 secured by their own security.



9446
 9447

9448 **Figure E-3 Security Considerations for Zigbee Bridge**