

1           **OCF “Essen” – Security Specification with no Cloud content - Security WG CR 2925**

2  
3  
4  
5

Legal Disclaimer

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

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

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

25 Copying or other form of reproduction and/or distribution of these works are strictly prohibited.  
26

# OCF Security Specification

VERSION 2 | June 20, 2019



# CONTENTS

44			
45	1	Scope .....	1
46	2	Normative References .....	1
47	3	Terms, definitions, and abbreviated terms .....	3
48	3.1	Terms and definitions.....	3
49	3.2	Abbreviated terms.....	6
50	4	Document Conventions and Organization .....	9
51	4.1	Conventions.....	9
52	4.2	Notation.....	10
53	4.3	Data types .....	11
54	4.4	Document structure.....	11
55	5	Security Overview.....	12
56	5.1	Preamble .....	12
57	5.2	Access Control.....	14
58	5.2.1	ACL Architecture .....	15
59	5.2.2	Access Control Scoping Levels.....	19
60	5.3	Onboarding Overview .....	21
61	5.3.1	Onboarding General .....	21
62	5.3.2	Onboarding Steps.....	23
63	5.3.3	Establishing a Device Owner .....	24
64	5.3.4	Provisioning for Normal Operation .....	25
65	5.3.5	Device Provisioning for OCF Cloud and Device Registration Overview .....	25
66	5.3.6	OCF Compliance Management System.....	25
67	5.4	Provisioning.....	26
68	5.4.1	Provisioning General .....	26
69	5.4.2	Provisioning other services.....	26
70	5.4.3	Provisioning Credentials for Normal Operation .....	26
71	5.4.4	Role Assignment and Provisioning for Normal Operation .....	27
72	5.4.5	ACL provisioning .....	27
73	5.5	Secure Resource Manager (SRM).....	27
74	5.6	Credential Overview.....	28
75	6	Security for the Discovery Process .....	29
76	6.1	Preamble .....	29
77	6.2	Security Considerations for Discovery.....	29
78	7	Security Provisioning.....	32
79	7.1	Device Identity.....	32
80	7.1.1	General Device Identity .....	32
81	7.1.2	Device Identity for Devices with UAID [Deprecated].....	32
82	7.2	Device Ownership.....	32
83	7.3	Device Ownership Transfer Methods.....	33
84	7.3.1	OTM implementation requirements .....	33
85	7.3.2	SharedKey Credential Calculation .....	35
86	7.3.3	Certificate Credential Generation.....	35

87	7.3.4	Just-Works OTM.....	35
88	7.3.5	Random PIN Based OTM.....	37
89	7.3.6	Manufacturer Certificate Based OTM.....	39
90	7.3.7	Vendor Specific OTMs.....	42
91	7.3.8	Establishing Owner Credentials.....	43
92	7.3.9	Security considerations regarding selecting an Ownership Transfer Method ..	51
93	7.3.10	Security Profile Assignment.....	51
94	7.4	Provisioning.....	52
95	7.4.1	Provisioning Flows.....	52
96	7.5	Device Provisioning for OCF Cloud.....	57
97	7.5.1	Cloud Provisioning General.....	<b>Error! Bookmark not defined.</b>
98	7.5.2	Device Provisioning by Mediator.....	<b>Error! Bookmark not defined.</b>
99	8	Device Onboarding State Definitions.....	57
100	8.1	Device Onboarding General.....	57
101	8.2	Device Onboarding-Reset State Definition.....	59
102	8.3	Device Ready-for-OTM State Definition.....	59
103	8.4	Device Ready-for-Provisioning State Definition.....	60
104	8.5	Device Ready-for-Normal-Operation State Definition.....	60
105	8.6	Device Soft Reset State Definition.....	60
106	9	Security Credential Management.....	63
107	9.1	Preamble.....	63
108	9.2	Credential Lifecycle.....	63
109	9.2.1	Credential Lifecycle General.....	63
110	9.2.2	Creation.....	63
111	9.2.3	Deletion.....	63
112	9.2.4	Refresh.....	63
113	9.2.5	Revocation.....	63
114	9.3	Credential Types.....	64
115	9.3.1	Preamble.....	64
116	9.3.2	Pair-wise Symmetric Key Credentials.....	64
117	9.3.3	Group Symmetric Key Credentials.....	64
118	9.3.4	Asymmetric Authentication Key Credentials.....	65
119	9.3.5	Asymmetric Key Encryption Key Credentials.....	65
120	9.3.6	Certificate Credentials.....	66
121	9.3.7	Password Credentials.....	66
122	9.4	Certificate Based Key Management.....	66
123	9.4.1	Overview.....	66
124	9.4.2	X.509 Digital Certificate Profiles.....	67
125	9.4.3	Certificate Revocation List (CRL) Profile.....	76
126	9.4.4	Resource Model.....	77
127	9.4.5	Certificate Provisioning.....	77
128	9.4.6	CRL Provisioning.....	78
129	10	Device Authentication.....	80
130	10.1	Device Authentication General.....	80

131	10.2	Device Authentication with Symmetric Key Credentials .....	80
132	10.3	Device Authentication with Raw Asymmetric Key Credentials.....	80
133	10.4	Device Authentication with Certificates .....	80
134	10.4.1	Device Authentication with Certificates General.....	80
135	10.4.2	Role Assertion with Certificates .....	81
136	10.4.3	OCF PKI Roots .....	82
137	10.4.4	PKI Trust Store.....	82
138	10.4.5	Path Validation and extension processing.....	83
139	10.5	Device Authentication with OCF Cloud.....	84
140	10.5.1	Device Authentication with OCF Cloud General <b>Error! Bookmark not defined.</b>	
141	10.5.2	Device Connection with the OCF Cloud .....	<b>Error! Bookmark not defined.</b>
142	10.5.3	Security Considerations.....	<b>Error! Bookmark not defined.</b>
143	11	Message Integrity and Confidentiality .....	85
144	11.1	Preamble .....	85
145	11.2	Session Protection with DTLS.....	85
146	11.2.1	DTLS Protection General.....	85
147	11.2.2	Unicast Session Semantics.....	85
148	11.2.3	Cloud Session Semantics .....	85
149	11.3	Cipher Suites .....	85
150	11.3.1	Cipher Suites General .....	85
151	11.3.2	Cipher Suites for Device Ownership Transfer .....	85
152	11.3.3	Cipher Suites for Symmetric Keys.....	86
153	11.3.4	Cipher Suites for Asymmetric Credentials.....	87
154	11.3.5	Cipher suites for OCF Cloud Credentials .....	87
155	12	Access Control .....	88
156	12.1	ACL Generation and Management .....	88
157	12.2	ACL Evaluation and Enforcement.....	88
158	12.2.1	ACL Evaluation and Enforcement General .....	88
159	12.2.2	Host Reference Matching .....	88
160	12.2.3	Resource Wildcard Matching .....	88
161	12.2.4	Multiple Criteria Matching .....	89
162	12.2.5	Subject Matching using Wildcards .....	89
163	12.2.6	Subject Matching using Roles.....	89
164	12.2.7	ACL Evaluation.....	90
165	13	Security Resources .....	92
166	13.1	Security Resources General .....	92
167	13.2	Device Owner Transfer Resource .....	94
168	13.2.1	Device Owner Transfer Resource General.....	94
169	13.2.2	Persistent and Semi-Persistent Device Identifiers.....	97
170	13.2.3	Onboarding Considerations for Device Identifier .....	97
171	13.2.4	OCF defined OTMs.....	98
172	13.3	Credential Resource .....	99
173	13.3.1	Credential Resource General.....	99
174	13.3.2	Properties of the Credential Resource .....	103

175	13.3.3	Key Formatting .....	106
176	13.3.4	Credential Refresh Method Details .....	106
177	13.4	Certificate Revocation List .....	108
178	13.4.1	CRL Resource Definition .....	108
179	13.5	ACL Resources .....	108
180	13.5.1	ACL Resources General .....	108
181	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures. ....	108
182	13.5.3	ACL Resource .....	109
183	13.6	Access Manager ACL Resource .....	115
184	13.7	Signed ACL Resource .....	115
185	13.8	Provisioning Status Resource .....	117
186	13.9	Certificate Signing Request Resource .....	122
187	13.10	Roles Resource .....	123
188	13.11	Account Resource .....	124
189	13.12	Account Session Resource .....	124
190	13.13	Account Token Refresh Resource .....	124
191	13.14	Security Virtual Resources (SVRs) and Access Policy .....	124
192	13.15	SVRs, Discoverability and OCF Endpoints .....	125
193	13.16	Additional Privacy Consideration for Core and SVRs Resources .....	125
194	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ....	125
195	13.16.2	Privacy Protecting the Device Identifiers .....	127
196	13.16.3	Privacy Protecting the Protocol Independent Device Identifier .....	128
197	13.16.4	Privacy Protecting the Platform Identifier .....	128
198	13.17	Easy Setup Resource Device State .....	128
199	14	Security Hardening Guidelines/ Execution Environment Security .....	131
200	14.1	Preamble .....	131
201	14.2	Execution Environment Elements .....	131
202	14.2.1	Execution Environment Elements General .....	131
203	14.2.2	Secure Storage .....	131
204	14.2.3	Secure execution engine .....	134
205	14.2.4	Trusted input/output paths .....	134
206	14.2.5	Secure clock .....	134
207	14.2.6	Approved algorithms .....	134
208	14.2.7	Hardware tamper protection .....	135
209	14.3	Secure Boot .....	135
210	14.3.1	Concept of software module authentication .....	135
211	14.3.2	Secure Boot process .....	137
212	14.3.3	Robustness Requirements .....	137
213	14.4	Attestation .....	137
214	14.5	Software Update .....	137
215	14.5.1	Overview: .....	137
216	14.5.2	Recognition of Current Differences .....	138
217	14.5.3	Software Version Validation .....	139
218	14.5.4	Software Update .....	139

219	14.5.5	Recommended Usage.....	139
220	14.6	Non-OCF Endpoint interoperability.....	140
221	14.7	Security Levels .....	140
222	14.8	Security Profiles.....	141
223	14.8.1	Security Profiles General.....	141
224	14.8.2	Identification of Security Profiles (Normative) .....	141
225	14.8.3	Security Profiles .....	143
226	15	Device Type Specific Requirements.....	148
227	15.1	Bridging Security .....	148
228	15.1.1	Universal Requirements for Bridging to another Ecosystem .....	148
229	15.1.2	Additional Security Requirements specific to Bridged Protocols .....	149
230	Annex A	(informative) Access Control Examples.....	151
231	A.1	Example OCF ACL Resource .....	151
232	A.2	Example AMS .....	151
233	Annex B	(Informative) Execution Environment Security Profiles .....	153
234	Annex C	(normative) Resource Type definitions.....	154
235	C.1	List of Resource Type definitions .....	154
236	C.2	Account Token.....	154
237	C.2.1	Introduction .....	<b>Error! Bookmark not defined.</b>
238	C.2.2	Well-known URI.....	<b>Error! Bookmark not defined.</b>
239	C.2.3	Resource type .....	<b>Error! Bookmark not defined.</b>
240	C.2.4	OpenAPI 2.0 definition.....	<b>Error! Bookmark not defined.</b>
241	C.2.5	Property definition .....	<b>Error! Bookmark not defined.</b>
242	C.2.6	CRUDN behaviour .....	<b>Error! Bookmark not defined.</b>
243	C.3	Access Control List [DEPRECATED].....	154
244	C.4	Access Control List-2.....	154
245	C.4.1	Introduction .....	154
246	C.4.2	Well-known URI.....	154
247	C.4.3	Resource type .....	154
248	C.4.4	OpenAPI 2.0 definition.....	155
249	C.4.5	Property definition .....	164
250	C.4.6	CRUDN behaviour .....	164
251	C.5	Managed Access Control .....	164
252	C.5.1	Introduction .....	164
253	C.5.2	Well-known URI.....	164
254	C.5.3	Resource type .....	165
255	C.5.4	OpenAPI 2.0 definition.....	165
256	C.5.5	Property definition .....	168
257	C.5.6	CRUDN behaviour .....	168
258	C.6	Credential.....	169
259	C.6.1	Introduction .....	169
260	C.6.2	Well-known URI.....	169
261	C.6.3	Resource type .....	169
262	C.6.4	OpenAPI 2.0 definition.....	169

263	C.6.5	Property definition .....	179
264	C.6.6	CRUDN behaviour .....	180
265	C.7	Certificate Revocation.....	180
266	C.7.1	Introduction .....	180
267	C.7.2	Well-known URI .....	180
268	C.7.3	Resource type .....	180
269	C.7.4	OpenAPI 2.0 definition.....	180
270	C.7.5	Property definition .....	182
271	C.7.6	CRUDN behaviour .....	183
272	C.8	Certificate Signing Request.....	183
273	C.8.1	Introduction .....	183
274	C.8.2	Well-known URI .....	183
275	C.8.3	Resource type .....	183
276	C.8.4	OpenAPI 2.0 definition.....	183
277	C.8.5	Property definition .....	185
278	C.8.6	CRUDN behaviour .....	185
279	C.9	Device Owner Transfer Method.....	185
280	C.9.1	Introduction .....	185
281	C.9.2	Well-known URI .....	185
282	C.9.3	Resource type .....	185
283	C.9.4	OpenAPI 2.0 definition.....	185
284	C.9.5	Property definition .....	189
285	C.9.6	CRUDN behaviour .....	191
286	C.10	Device Provisioning Status .....	191
287	C.10.1	Introduction .....	191
288	C.10.2	Well-known URI .....	191
289	C.10.3	Resource type .....	191
290	C.10.4	OpenAPI 2.0 definition.....	191
291	C.10.5	Property definition .....	195
292	C.10.6	CRUDN behaviour .....	199
293	C.11	Asserted Roles .....	199
294	C.11.1	Introduction .....	199
295	C.11.2	Well-known URI .....	199
296	C.11.3	Resource type .....	200
297	C.11.4	OpenAPI 2.0 definition.....	200
298	C.11.5	Property definition .....	209
299	C.11.6	CRUDN behaviour .....	209
300	C.12	Signed Access Control List .....	209
301	C.12.1	Introduction .....	209
302	C.12.2	Well-known URI .....	209
303	C.12.3	Resource type .....	209
304	C.12.4	OpenAPI 2.0 definition.....	209
305	C.12.5	Property definition .....	217
306	C.12.6	CRUDN behaviour .....	217

307	C.13	Session.....	217
308	C.13.1	Introduction .....	<b>Error! Bookmark not defined.</b>
309	C.13.2	Well-known URI .....	<b>Error! Bookmark not defined.</b>
310	C.13.3	Resource type .....	<b>Error! Bookmark not defined.</b>
311	C.13.4	OpenAPI 2.0 definition.....	<b>Error! Bookmark not defined.</b>
312	C.13.5	Property definition .....	<b>Error! Bookmark not defined.</b>
313	C.13.6	CRUDN behaviour .....	<b>Error! Bookmark not defined.</b>
314	C.14	Security Profile .....	217
315	C.14.1	Introduction .....	217
316	C.14.2	Well-known URI .....	218
317	C.14.3	Resource type .....	218
318	C.14.4	OpenAPI 2.0 definition.....	218
319	C.14.5	Property definition .....	220
320	C.14.6	CRUDN behaviour .....	220
321	C.15	Token Refresh .....	220
322	C.15.1	Introduction .....	<b>Error! Bookmark not defined.</b>
323	C.15.2	Well-known URI .....	<b>Error! Bookmark not defined.</b>
324	C.15.3	Resource type .....	<b>Error! Bookmark not defined.</b>
325	C.15.4	OpenAPI 2.0 definition.....	<b>Error! Bookmark not defined.</b>
326	C.15.5	Property definition .....	<b>Error! Bookmark not defined.</b>
327	C.15.6	CRUDN behaviour .....	<b>Error! Bookmark not defined.</b>
328	Annex D (informative)	OID definitions .....	221
329	Annex E (informative)	Security considerations specific to Bridged Protocols .....	223
330	E.1	Security Considerations specific to the AllJoyn Protocol .....	223
331	E.2	Security Considerations specific to the Bluetooth LE Protocol.....	223
332	E.3	Security Considerations specific to the oneM2M Protocol .....	223
333	E.4	Security Considerations specific to the U+ Protocol .....	224
334	E.5	Security Considerations specific to the Z-Wave Protocol.....	224
335	E.6	Security Considerations specific to the Zigbee Protocol .....	225
336			

## FIGURES

337		
338	Figure 1 – OCF Interaction.....	10
339	Figure 2 – OCF Layers .....	12
340	Figure 3 – OCF Security Enforcement Points .....	14
341	Figure 4 – Use case-1 showing simple ACL enforcement.....	16
342	Figure 5 – Use case 2: A policy for the requested Resource is missing.....	17
343	Figure 6 – Use case-3 showing AMS supported ACL .....	18
344	Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS.....	19
345	Figure 8 – Example Resource definition with opaque Properties .....	20
346	Figure 9 – Property Level Access Control .....	20
347	Figure 10 – Onboarding Overview.....	22
348	Figure 11 – OCF Onboarding Process .....	24
349	Figure 12 – OCF's SRM Architecture .....	28
350	Figure 13 – Discover New Device Sequence.....	34
351	Figure 14 – A Just Works OTM .....	36
352	Figure 15 – Random PIN-based OTM .....	38
353	Figure 16 – Manufacturer Certificate Based OTM Sequence .....	41
354	Figure 17 – Vendor-specific Owner Transfer Sequence.....	42
355	Figure 18 – Establish Device Identity Flow.....	44
356	Figure 19 – Owner Credential Selection Provisioning Sequence .....	45
357	Figure 20 – Symmetric Owner Credential Provisioning Sequence .....	46
358	Figure 21 – Asymmetric Owner Credential Provisioning Sequence.....	47
359	Figure 22 – Configure Device Services .....	49
360	Figure 23 – Provision New Device for Peer to Peer Interaction Sequence.....	50
361	Figure 24 – Example of Client-directed provisioning.....	53
362	Figure 25 – Example of Server-directed provisioning using a single provisioning service .....	54
363	Figure 26 – Example of Server-directed provisioning involving multiple support services .....	57
364	Figure 27 – Device state model.....	58
365	Figure 28 – OBT Sanity Check Sequence in SRESET .....	61
366	Figure 29 – Client-directed Certificate Transfer.....	78
367	Figure 30 – Client-directed CRL Transfer.....	79
368	Figure 32 – Asserting a role with a certificate role credential. ....	82
369	Figure 33 – Device connection with OCF Cloud .....	<b>Error! Bookmark not defined.</b>
370	Figure 34 – OCF Security Resources .....	92
371	Figure 35 – "/oic/sec/cred" Resource and Properties.....	93
372	Figure 36 – "/oic/sec/acl2" Resource and Properties.....	93
373	Figure 37 – "/oic/sec/amacl" Resource and Properties .....	94
374	Figure 38 – "/oic/sec/sacl" Resource and Properties .....	94
375	Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states .....	129

376 Figure 40 – Software Module Authentication ..... 136  
377 Figure 41 – Verification Software Module ..... 136  
378 Figure 42 – Software Module Authenticity ..... 137  
379 Figure 43 – State transitioning diagram for firmware download ..... 138  
380 Figure A-1 – Example "/oic/sec/acl2" Resource ..... 151  
381 Figure A-2 Example "/oic/sec/amacl" Resource ..... 152  
382 Figure E-1 Security Considerations for BLE Bridge ..... 223  
383 Figure E-2 Security Considerations for Z-Wave Bridge ..... 225  
384 Figure E-3 Security Considerations for Zigbee Bridge ..... 226  
385

DRAFT

386	<b>Tables</b>	
387	Table 1 – Discover New Device Details.....	34
388	Table 2 – A Just Works OTM Details.....	36
389	Table 3 – Random PIN-based OTM Details.....	38
390	Table 4 – Manufacturer Certificate Based OTM Details.....	41
391	Table 5 – Vendor-specific Owner Transfer Details.....	43
392	Table 6 – Establish Device Identity Details.....	44
393	Table 7 – Owner Credential Selection Details.....	45
394	Table 8 – Symmetric Owner Credential Assignment Details.....	46
395	Table 9 – Asymmetric Owner Credential Assignment Details.....	48
396	Table 10 – Configure Device Services Detail.....	49
397	Table 11 – Provision New Device for Peer to Peer Details.....	50
398	Table 12 – Steps describing Client -directed provisioning.....	53
399	Table 13 – Steps for Server-directed provisioning using a single provisioning service.....	54
400	Table 14 – Steps for Server-directed provisioning involving multiple support services.....	57
401	Table 15 – Mapping of Properties of the "oic.r.account" and "oic.r.coapcloudconf"	
402	Resources.....	<b>Error! Bookmark not defined.</b>
403	Table 16 – X.509 v1 fields for Root CA Certificates.....	67
404	Table 17 - X.509 v3 extensions for Root CA Certificates.....	68
405	Table 18 - X.509 v1 fields for Intermediate CA Certificates.....	68
406	Table 19 – X.509 v3 extensions for Intermediate CA Certificates.....	69
407	Table 20 – X.509 v1 fields for End-Entity Certificates.....	69
408	Table 21 – X.509 v3 extensions for End-Entity Certificates.....	70
409	Table 22 – Device connection with the OCF Cloud flow.....	<b>Error! Bookmark not defined.</b>
410	Table 23 – ACE2 Wildcard Matching Strings Description.....	88
411	Table 24 – Definition of the "/oic/sec/doxm" Resource.....	95
412	Table 25 – Properties of the "/oic/sec/doxm" Resource.....	95
413	Table 26 – Properties of the "oic.sec.didtype" type.....	96
414	Table 27 – Properties of the "oic.sec.doxmtype" type.....	98
415	Table 28 – Definition of the "oic.r.cred" Resource.....	99
416	Table 29 – Properties of the "/oic/sec/cred" Resource.....	100
417	Table 30 – Properties of the "oic.sec.cred" Property.....	101
418	Table 31: Properties of the "oic.sec.credusagetype" Property.....	102
419	Table 32 – Properties of the "oic.sec.pubdatatype" Property.....	102
420	Table 33 – Properties of the "oic.sec.privdatatype" Property.....	102
421	Table 34 – Properties of the "oic.sec.optdatatype" Property.....	103
422	Table 35 – Definition of the "oic.sec.roletype" type.....	103
423	Table 36 – Value Definition of the "oic.sec.crmttype" Property.....	105
424	Table 37 – 128-bit symmetric key.....	106

425	Table 38 – 256-bit symmetric key .....	106
426	Table 39 – Definition of the "oic.r.crl" Resource .....	108
427	Table 40 – Properties of the "oic.r.crl" Resource .....	108
428	Table 41 – BNF Definition of OCF ACL .....	108
429	Table 42 – Value Definition of the "oic.sec.crudntype" Property .....	111
430	Table 43 – Definition of the "oic.sec.acl2" Resource .....	111
431	Table 44 – Properties of the "oic.sec.acl2" Resource .....	112
432	Table 45 – "oic.sec.ace2" data type definition. ....	113
433	Table 46 – "oic.sec.ace2.resource-ref" data type definition. ....	113
434	Table 47 – Value definition "oic.sec.conntype" Property.....	113
435	Table 48 – Definition of the "oic.r.amacl" Resource.....	115
436	Table 49 – Properties of the "oic.r.amacl" Resource .....	115
437	Table 50 – Definition of the "oic.r.sacl" Resource.....	116
438	Table 51 – Properties of the "oic.r.sacl" Resource .....	116
439	Table 52 – Properties of the "oic.sec.sigtype" Property .....	116
440	Table 53 – Definition of the "oic.r.pstat" Resource .....	117
441	Table 54 – Properties of the "oic.r.pstat" Resource .....	118
442	Table 55 – Properties of the "/oic/sec/dostype" Property.....	119
443	Table 56 – Definition of the "oic.sec.dpmttype" Property .....	121
444	Table 57 – Value Definition of the "oic.sec.dpmttype" Property (Low-Byte) .....	121
445	Table 58 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte).....	121
446	Table 59 – Definition of the "oic.sec.pomtype" Property .....	122
447	Table 60 – Value Definition of the "oic.sec.pomtype" Property .....	122
448	Table 61 – Definition of the "oic.r.csr" Resource .....	122
449	Table 62 – Properties of the "oic.r.csr" Resource .....	123
450	Table 63 – Definition of the "oic.r.roles" Resource .....	124
451	Table 64 – Properties of the "oic.r.roles" Resource .....	124
452	Table 65 – Definition of the "oic.r.account" Resource.....	<b>Error! Bookmark not defined.</b>
453	Table 66 – Properties of the "oic.r.account" Resource .....	<b>Error! Bookmark not defined.</b>
454	Table 67 – Definition of the "oic.r.session" Resource .....	<b>Error! Bookmark not defined.</b>
455	Table 68 – Properties of the "oic.r.session" Resource.....	<b>Error! Bookmark not defined.</b>
456	Table 69 – Definition of the "oic.r.tokenrefresh" Resource .....	<b>Error! Bookmark not defined.</b>
457	Table 70 – Properties of the "oic.r.tokenrefresh" Resource .....	<b>Error! Bookmark not defined.</b>
458	Table 71 – Core Resource Properties Access Modes given various Device States.....	126
459	Table 72 – Examples of Sensitive Data.....	132
460	Table 73 – Description of the software update bits.....	138
461	Table 73 – Definition of the "oic.sec.sp" Resource .....	142
462	Table 74 – Properties of the "oic.sec.sp" Resource.....	142

463	Table 75 – Dependencies of VOD Behaviour on Bridge state, as clarification of	
464	accompanying text.....	149
465	Table B.1 – OCF Security Profile .....	153
466	Table C.1 – Alphabetized list of security resources .....	154
467	Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account". .....	<b>Error!</b>
468	<b>Bookmark not defined.</b>	
469	Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account".....	<b>Error!</b>
470	<b>Bookmark not defined.</b>	
471	Table C.4 – The Property definitions of the Resource with type "rt" = "oic.r.acl2". .....	164
472	Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".....	164
473	Table C.6 – The Property definitions of the Resource with type "rt" = "oic.r.amacl". .....	168
474	Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl".....	169
475	Table C.8 – The Property definitions of the Resource with type "rt" = "oic.r.cred". .....	179
476	Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred". .....	180
477	Table C.10 – The Property definitions of the Resource with type "rt" = "oic.r.crl".....	182
478	Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl". .....	183
479	Table C.12 – The Property definitions of the Resource with type "rt" = "oic.r.csr". .....	185
480	Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr". .....	185
481	Table C.14 – The Property definitions of the Resource with type "rt" = "oic.r.doxm". .....	189
482	Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".....	191
483	Table C.16 – The Property definitions of the Resource with type "rt" = "oic.r.pstat". .....	195
484	Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat". .....	199
485	Table C.18 – The Property definitions of the Resource with type "rt" = "oic.r.roles". .....	209
486	Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles". .....	209
487	Table C.20 – The Property definitions of the Resource with type "rt" = "oic.r.sacl". .....	217
488	Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl".....	217
489	Table C.22 – The Property definitions of the Resource with type "rt" = "oic.r.session".....	<b>Error!</b>
490	<b>Bookmark not defined.</b>	
491	Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.session". .....	<b>Error!</b>
492	<b>Bookmark not defined.</b>	
493	Table C.24 – The Property definitions of the Resource with type "rt" = "oic.r.sp". .....	220
494	Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp". .....	220
495	Table C.26 – The Property definitions of the Resource with type "rt" = "oic.r.tokenrefresh". .....	<b>Error!</b>
496	<b>Bookmark not defined.</b>	
497	Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh". .....	<b>Error!</b>
498	<b>Bookmark not defined.</b>	
499	Table E.1 GAP security mode .....	223
500	Table E.2 TLS 1.2 Cipher Suites used by U+ .....	224
501	Table E.3 Z-Wave Security Class.....	225
502	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers .....	226
503		

DRAFT

505

## Change History

506

**(DELETE BEFORE PUBLISHING)**

Release	Date	Description
		(Baseline – OCF Security Specification v2.0.2)
01	May 23, 2019	Bug 2661 - post-onboarding D2D connections and local credentials Bug 2662 - Review and populate list of DCRs [Unmerged in r5] Bug 2692 - Clarifying Owned state behaviour
02	May 23, 2019	Bug 2719 - Remove OIC 1.1 security features
03	May 23, 2019	Bug 2629 - RBSTG: Relationship between Device States of Bridge & Virtual OCF Devices Some Editorial cleanup
04	May 24, 2019	Bug 2708 - A few inconsistencies and errata in OCF 2.0 Security Specification Some Editorial cleanup
05	May 28, 2019	Unmerge Bug 2692 - Clarifying Owned state behaviour
06	May 28, 2019	Bug 2764 - This CR aims to clarify the behaviour of rowneruid Property during onboarding process.
07	May 28, 2019	Bug 2715 - Remove 'Subject property shall refer to the Device"
08	May 28, 2019	Bug 2453 - firmware update (SVR part)
09	May 28, 2019	Bug 2332 - Post-Bangkok access control for batch interface requests to a OCF Collection Resource - local references
10	May 30, 2019	Resolve inline markup comments related to Bug 2453, Bug 2332 Change firmware -> software in title for figure 43 and few other related places.

507

508

509

## 510 **1 Scope**

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

## 515 **2 Normative References**

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

519 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)  
520 Specification -- Part 1: Core specification

521 <https://www.iso.org/standard/53238.html>

522 Latest version available at:

523 [https://openconnectivity.org/specs/OCF\\_Core\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Core_Specification.pdf)

524 ISO/IEC 30118-3:2018 Information technology -- Open Connectivity Foundation (OCF)  
525 Specification -- Part 3: Bridging specification

526 <https://www.iso.org/standard/74240.html>

527 Latest version available at:

528 [https://openconnectivity.org/specs/OCF\\_Bridging\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf)

529 OCF Wi-Fi Easy Setup, Information technology – Open Connectivity Foundation (OCF)  
530 Specification – Part 7: Wi-Fi Easy Setup specification

531 Latest version available at:

532 [https://openconnectivity.org/specs/OCF\\_Wi-Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)

533 OCF Cloud Specification, Information technology – Open Connectivity Foundation (OCF)  
534 Specification – Part 8: Cloud Specification

535 Latest version available at:

536 [https://openconnectivity.org/specs/OCF\\_Cloud\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf)

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

538 IETF RFC 2315, *PKCS #7: Cryptographic Message Syntax Version 1.5*, March 1998,

539 <https://tools.ietf.org/html/rfc2315>

540 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September

541 2000, <https://tools.ietf.org/html/rfc2898>

542 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November

543 2000, <https://tools.ietf.org/html/rfc2986>

544 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December

545 2005, <https://tools.ietf.org/html/rfc4279>

546 IETF RFC 4492, *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security*

547 *(TLS)*, May 2006, <https://tools.ietf.org/html/rfc4492>

548 IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, August 2008,

549 <https://tools.ietf.org/html/rfc5246>

550 IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation*

551 *List (CRL) Profile*, May 2008, <https://tools.ietf.org/html/rfc5280>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

580

581 **3 Terms, definitions, and abbreviated terms**

582 **3.1 Terms and definitions**

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

585 ISO and IEC maintain terminological databases for use in standardization at the following  
586 addresses:

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

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

589 **3.1.1**

590 **Access Management Service (AMS)**

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

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

594 **3.1.2**

595 **Access Token – moved to OCF Cloud Security document**

596 **3.1.3**

597 **Authorization Provider – moved to OCF Cloud Security document**

598 **3.1.4**

599 **Client**

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

601 **3.1.5**

602 **Credential Management Service (CMS)**

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

605 **3.1.6**

606 **Device**

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

608 **3.1.7**

609 **Device Class**

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

612 **3.1.8**

613 **Device ID**

614 a stack instance identifier.

615 **3.1.9**

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

617 a logical entity that establishes device ownership

618 **3.1.10**

619 **3.1.11 Device Registration – moved to OCF Cloud Security document**

620 **End-Entity**

621 any certificate holder which is not a Root or Intermediate Certificate Authority.

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

623 **3.1.12**

624 **Entity**

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

626 **3.1.13**  
627 **OCF Interface**  
628 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

629 **3.1.14**  
630 **Intermediary**  
631 a Device that implements both Client and Server roles and may perform protocol translation, virtual  
632 device to physical device mapping or Resource translation

633 **3.1.15**  
634 **OCF Cipher Suite**  
635 a set of algorithms and parameters that define the cryptographic functionality of a Device. The OCF  
636 Cipher Suite includes the definition of the public key group operations, signatures, and specific  
637 hashing and encoding used to support the public key.

638 **3.1.16**  
639 **OCF Cloud User – moved to OCF Cloud Security spec**

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

645 **3.1.18**  
646 **Onboarding Tool (OBT)**  
647 a tool that implements DOTS(3.1.9), AMS(3.1.1) and CMS(3.1.5) functionality

648 **3.1.19**  
649 **Out of Band Method**  
650 any mechanism for delivery of a secret from one party to another, not specified by OCF

651 **3.1.20**  
652 **Owner Credential (OC)**  
653 credential, provisioned by an OBT(3.1.18) to a Device during onboarding, for the purposes of  
654 mutual authentication of the Device and OBT(3.1.18) during subsequent interactions

655 **3.1.21**  
656 **Platform ID**  
657 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

658 **3.1.22**  
659 **Property**  
660 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

661 **3.1.23**  
662 **Resource**  
663 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

664 **3.1.24**  
665 **Role (Network context)**  
666 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

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

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

676 **3.1.27**  
677 **Security Virtual Resource (SVR)**  
678 a resource supporting security features.

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

680 **3.1.28**  
681 **Server**

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

683 **3.1.29**  
684 **Trust Anchor**  
685 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.  
686 a Device and an OBT(3.1.18)) can assume trust

687 **3.1.30**  
688 **Unique Authenticable Identifier**  
689 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is  
690 used to create the Device ID.

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

692 **3.1.31**  
693 **Device Configuration Resource (DCR)**  
694 a Resource that is any of the following:

- 695 a) a Discovery Core Resource, or  
696 b) a Security Virtual Resource, or  
697 c) a Wi-Fi Easy Setup Resource ("oic.r.easyssetup", "oic.r.wificonf", "oic.r.devconf"), or  
698 d) a CoAP Cloud Configuration Resource ("oic.r.coapcloudconf"), or  
699 e) a Software Update Resource ("oic.r.softwareupdate"), or  
700 f) a Maintenance Resource ("oic.wk.mnt").

701 **3.1.32**  
702 **Non-Configuration Resource (NCR)**  
703 a Resource that is not a Device Configuration Resource (3.1.31).

704 **3.1.33**  
705 **Bridged Device**

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

707 **3.1.34**  
708 **Bridged Protocol**

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

710 **3.1.35**  
711 **Bridge**

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

713 **3.1.36**  
714 **Bridging Platform**

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

716 **3.1.37**  
717 **Virtual Bridged Device**  
718 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

719 **3.1.38**  
720 **Virtual OCF Device**  
721 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

722 **3.1.39**  
723 **OCF Security Domain**  
724 set of onboarded OCF Devices that are provisioned with credentialing information for confidential  
725 communication with one another

726 **3.1.40**  
727 **Owned (or "in Owned State")**  
728 having the "owned" Property of the "/oic/sec/doxm" resource equal to "TRUE"

729 **3.1.41**  
730 **Unowned (or "in Unowned State")**  
731 having the "owned" Property of the "/oic/sec/doxm" resource equal to "FALSE"

732 **3.2 Abbreviated terms**

733 **3.2.1**  
734 **AC**  
735 Access Control

736 **3.2.2**  
737 **ACE**  
738 Access Control Entry

739 **3.2.3**  
740 **ACL**  
741 Access Control List

742 **3.2.4**  
743 **AES**  
744 Advanced Encryption Standard  
745 Note 1 to entry: See NIST FIPS 197, "Advanced Encryption Standard (AES)"

746 **3.2.5**  
747 **AMS**  
748 Access Management Service

749 **3.2.6**  
750 **CMS**  
751 Credential Management Service

752 **3.2.7**  
753 **CRUDN**  
754 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

755 **3.2.8**  
756 **CSR**  
757 Certificate Signing Request

758 **3.2.9**  
759 **CVC**  
760 Code Verification Certificate

761 **3.2.10**  
762 **ECC**  
763 Elliptic Curve Cryptography

764 **3.2.11**  
765 **ECDSA**  
766 Elliptic Curve Digital Signature Algorithm

767 **3.2.12**  
768 **EKU**  
769 Extended Key Usage

770 **3.2.13**  
771 **EPC**  
772 Embedded Platform Credential

773 **3.2.14**  
774 **EPK**  
775 Embedded Public Key

776 **3.2.15**  
777 **DOTS**  
778 Device Ownership Transfer Service

779 **3.2.16**  
780 **DPKP**  
781 Dynamic Public Key Pair

782 **3.2.17**  
783 **ID**  
784 Identity/Identifier

785 **3.2.18**  
786 **JSON**  
787 JavaScript Object Notation.

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

789 **3.2.19**  
790 **JWS**  
791 JSON Web Signature.

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

793 **3.2.20**  
794 **KDF**  
795 Key Derivation Function

796 **3.2.21**  
797 **MAC**  
798 Message Authentication Code

799 **3.2.22**  
800 **MITM**  
801 Man-in-the-Middle

802 **3.2.23**  
803 **NVRAM**  
804 Non-Volatile Random-Access Memory

805 **3.2.24**  
806 **OC**  
807 Owner Credential

808 **3.2.25**  
809 **OCSP**  
810 Online Certificate Status Protocol

811 **3.2.26**  
812 **OBT**  
813 Onboarding Tool

814 **3.2.27**  
815 **OID**  
816 Object Identifier

817 **3.2.28**  
818 **OTM**  
819 Owner Transfer Method

820 **3.2.29**  
821 **OOB**  
822 Out of Band

823 **3.2.30**  
824 **OWASP**  
825 Open Web Application Security Project.

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

827 **3.2.31**  
828 **PE**  
829 Policy Engine

830 **3.2.32**  
831 **PIN**  
832 Personal Identification Number

833 **3.2.33**  
834 **PPSK**  
835 PIN-authenticated pre-shared key

836 **3.2.34**  
837 **PRF**  
838 Pseudo Random Function

839 **3.2.35**  
840 **PSI**  
841 Persistent Storage Interface

842 **3.2.36**  
843 **PSK**  
844 Pre Shared Key

845 **3.2.37**  
846 **RBAC**  
847 Role Based Access Control

848 **3.2.38**  
849 **RM**  
850 Resource Manager

851 **3.2.39**  
852 **RNG**  
853 Random Number Generator

854 **3.2.40**  
855 **SACL**  
856 Signed Access Control List

857 **3.2.41**  
858 **SBAC**  
859 Subject Based Access Control

860 **3.2.42**  
861 **SEE**  
862 Secure Execution Environment

863 **3.2.43**  
864 **SRM**  
865 Secure Resource Manager

866 **3.2.44**  
867 **SVR**  
868 Security Virtual Resource

869 **3.2.45**  
870 **SW**  
871 Software

872 **3.2.46**  
873 **UAID**  
874 Unique Authenticable Identifier

875 **3.2.47**  
876 **URI**  
877 Uniform Resource Identifier

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

879 **3.2.48**  
880 **VOD**  
881 Virtual OCF Device

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

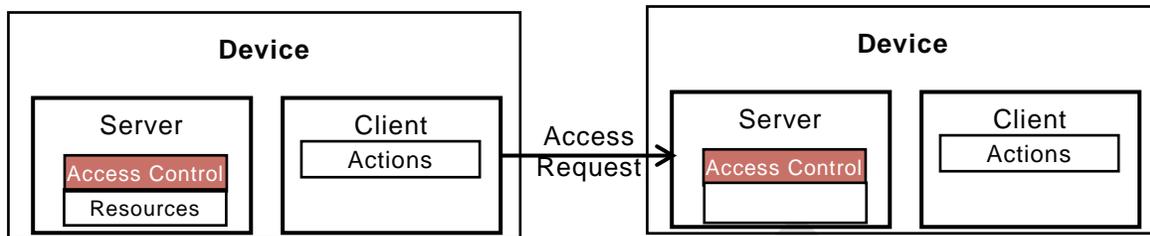
## 883 **4 Document Conventions and Organization**

### 884 **4.1 Conventions**

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

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

888 Figure 1 depicts interaction between OCF Devices.



889

890

**Figure 1 – OCF Interaction**

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

#### 895 **4.2 Notation**

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

##### 898 **Required (or shall or mandatory).**

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

##### 902 **Recommended (or should).**

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

##### 909 **Allowed (may or allowed).**

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

##### 912 **Conditionally allowed (CA)**

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

##### 915 **Conditionally required (CR)**

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

919 **DEPRECATED**

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

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

927 Words that are emphasized are printed in *italic*.

928 **4.3 Data types**

929 See ISO/IEC 30118-1:2018.

930 **4.4 Document structure**

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

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

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

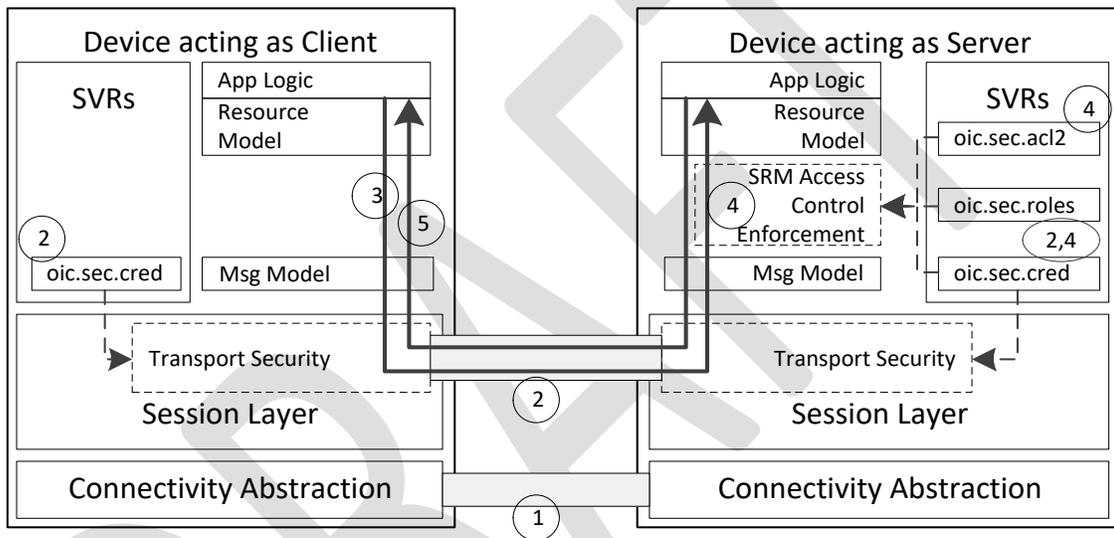
937

938 **5 Security Overview**

939 **5.1 Preamble**

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

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



948

949

**Figure 2 – OCF Layers**

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

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

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

968 4) The Server receives the request.

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

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

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

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

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

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

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

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

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

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

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

1000 Figure 3 depicts OCF Security Enforcement Points.

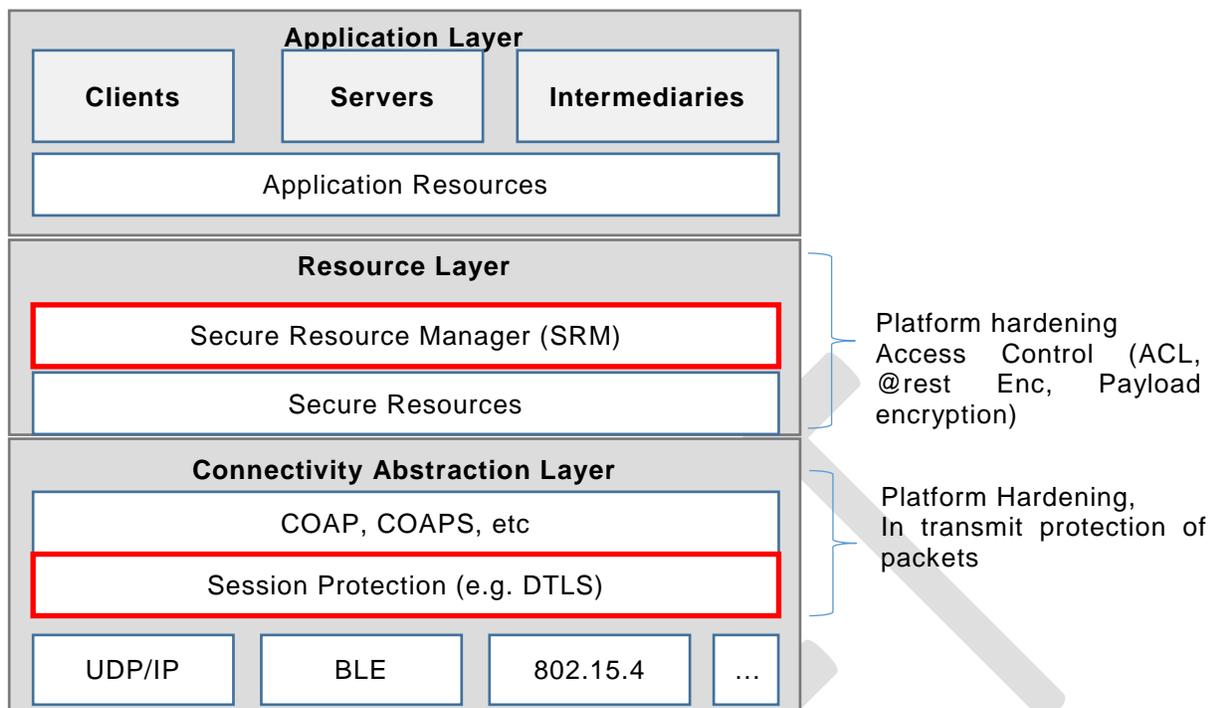


Figure 3 – OCF Security Enforcement Points

## 5.2 Access Control

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

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

- Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of requestor) of the requesting entity against the subject included in the policy defined for Resource. Asserting the identity of the requestor requires an authentication process.
- Role-based Access Control (RBAC), where each ACE will match a role identifier included in the policy for the Resource to a role identifier associated with the requestor.

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

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

The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier or (3) wildcard. The way in which the client connects to the server may be relevant context for making

1028 access control decisions. Wildcard matching on authenticated vs. unauthenticated and encrypted  
1029 vs. unencrypted connection allows an access policy to be broadly applied to subject classes.

1030 Example Wildcard Matching Policy:

```
1031 "aclist2": [  
1032 {  
1033   "subject": {"conntype": "anon-clear" },  
1034   "resources": [  
1035     { "wc": "*" }  
1036   ],  
1037   "permission": 31  
1038 },  
1039 {  
1040   "subject": {"conntype": "auth-crypt" },  
1041   "resources": [  
1042     { "wc": "*" }  
1043   ],  
1044   "permission": 31  
1045 },  
1046 ]
```

1047 Details of the format for ACL are defined in clause 12. The ACL is composed of one or more ACEs.  
1048 The ACL defines the access control policy for the Devices.

1049 ACL Resource requires the same security protection as other sensitive Resources, when it comes  
1050 to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW and  
1051 SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may have  
1052 different scoping levels and thus hardening needs to be specially considered for each scoping level.  
1053 For instance, a physical device may host multiple Device implementations and thus secure storage,  
1054 usage and isolation of ACLs for different Servers on the same Device needs to be considered.

## 1055 **5.2.1 ACL Architecture**

### 1056 **5.2.1.1 ACL Architecture General**

1057 The Server examines the Resource(s) requested by the client before processing the request. The  
1058 access control resource is searched to find one or more ACE entries that match the requestor and  
1059 the requested Resources. If a match is found, then permission and period constraints are applied.  
1060 If more than one match is found, then the logical UNION of permissions is applied to the overlapping  
1061 periods.

1062 The server uses the connection context to determine whether the subject has authenticated or not  
1063 and whether data confidentiality has been applied or not. Subject matching wildcard policies can  
1064 match on each aspect. If the user has authenticated, then subject matching may happen at  
1065 increased granularity based on role or device identity.

1066 Each ACE contains the permission set that will be applied for a given Resource requestor.  
1067 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY  
1068 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more  
1069 roles. Devices may acquire elevated access permissions when asserting a role. For example, an  
1070 ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally  
1071 accessible.

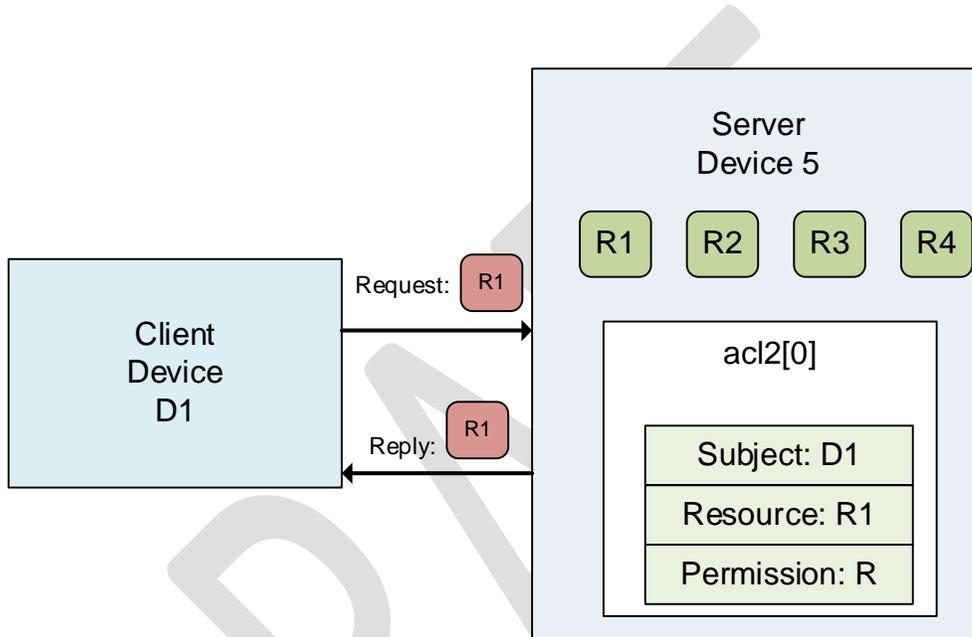
1072 **5.2.1.2 Use of local ACLs**

1073 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control  
1074 processing than remote ACL processing by an AMS.

1075 The following use cases describe the operation of access control

1076 Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4). Client  
1077 Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0] corresponds to  
1078 Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives access to  
1079 Resource R1 because the local ACL "/oic/sec/acl2/0" matches the request.

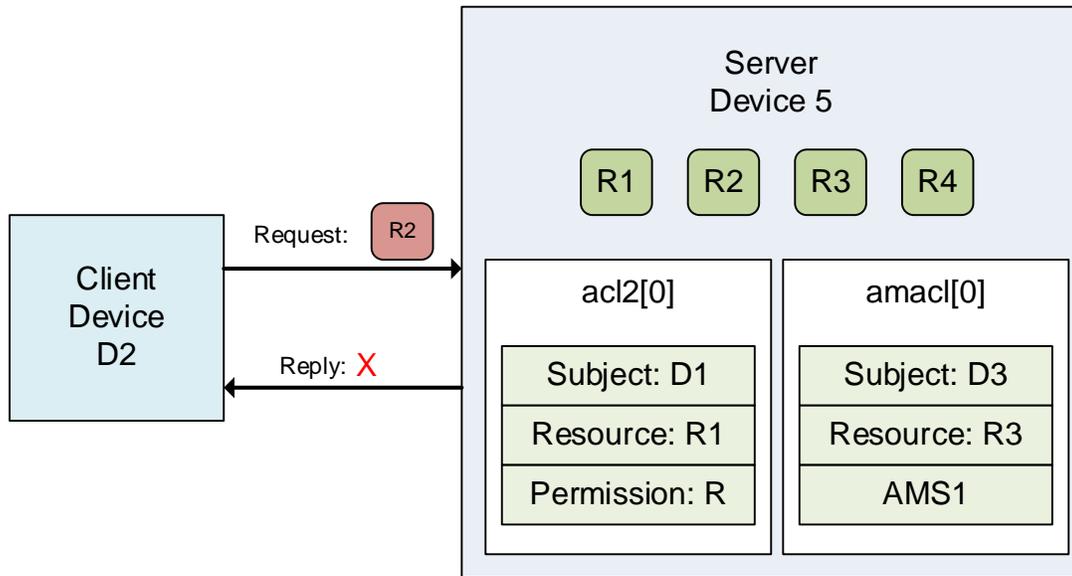
1080



1081

1082 **Figure 4 – Use case-1 showing simple ACL enforcement**

1083 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL  
1084 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.



1086

1087

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

### 1088 5.2.1.3 Use of AMS

1089 AMS improves ACL policy management. However, they can become a central point of failure. Due to  
1090 network latency overhead, ACL processing may be slower through an AMS.

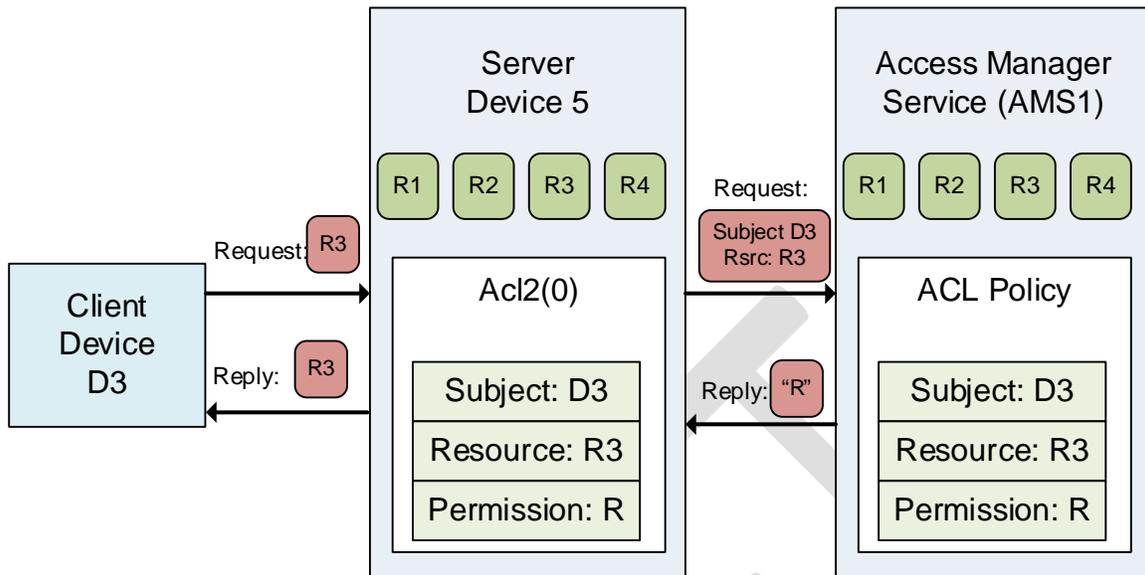
1091 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The  
1092 Server shall determine which ACL mechanism to use for which Resource set. The "/oic/sec/amacl"  
1093 Resource is an ACL structure that specifies which Resources will use an AMS to resolve access  
1094 decisions. The "/oic/sec/amacl" may be used in concert with local ACLs ("/oic/sec/acl2").

1095 The AMS is authenticated by referencing a credential issued to the device identifier contained in  
1096 "/oic/sec/acl2.rowneruuid".

1097 The Server Device may proactively open a connection to the AMS using the Device ID found in  
1098 "/oic/sec/acl2.rowneruuid". Alternatively, the Server may reject the Resource access request with  
1099 an error, ACCESS\_DENIED\_REQUIRES\_SACL that instructs the requestor to obtain a suitable  
1100 ACE policy using a SACL Resource "/oic/sec/sacl". The "/oic/sec/sacl" signature may be validated  
1101 using the credential Resource associated with the "/oic/sec/acl2.rowneruuid".

1102 The following use cases describe access control using the AMS:

1103 Use Case 3: As depicted in Figure 6, Device D3 requests and receives access to Resource R3 with  
1104 permission Perm1 because the "/oic/sec/amacl/0" matches a policy to consult the Access Manager  
1105 Server AMS1 service



1106

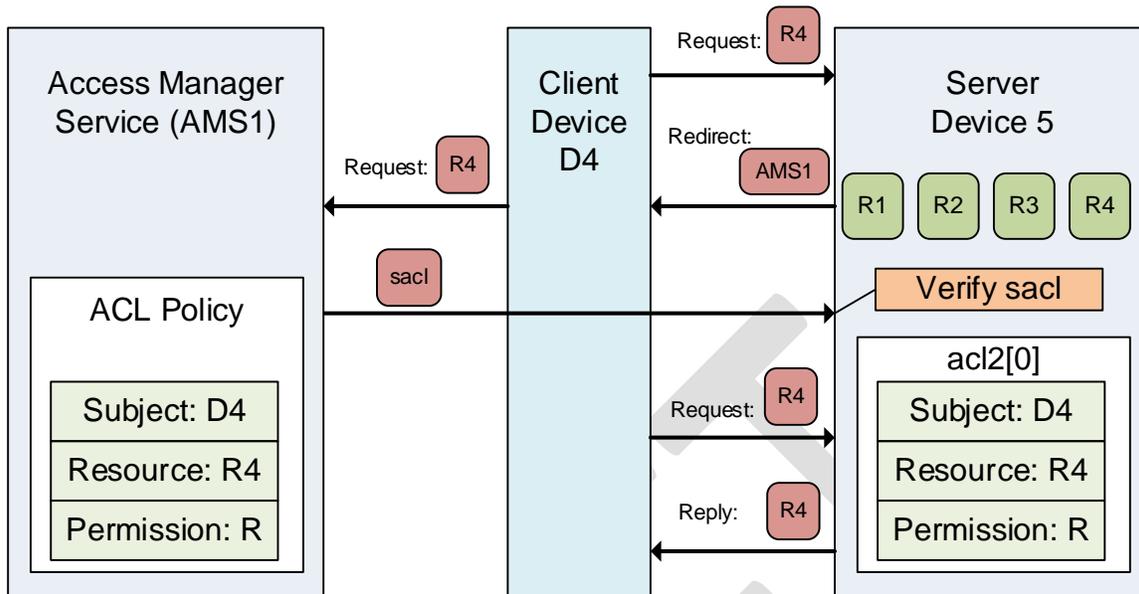
1107

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

1108 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from Server  
 1109 Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1 by  
 1110 returning an error identifying AMS1 as a "/oic/sec/sacl" Resource issuer. Device D4 obtains Sacl1  
 1111 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the "/oic/sec/sacl"  
 1112 Resource and evaluates the ACE policy that grants Perm2 access.

1113 ACE redirection may occur when D4 receives an error result with reason code indicating no match  
 1114 exists (i.e. ACCESS\_DENIED\_NO\_ACE). D4 reads the "/oic/sec/acl2" Resource to find the  
 1115 "rowneruid" which identifies the AMS and then submits a request to be provisioned, in this  
 1116 example the AMS chooses to supply a SACL Resource, however it may choose to re-provision the  
 1117 local ACL Resource "/oic/sec/acl2". The request is reissued subsequently. D4 is presumed to have  
 1118 been introduced to the AMS as part of Device onboarding or through subsequent credential  
 1119 provisioning actions.

1120 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.



1121

1122

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

1123

### 5.2.2 Access Control Scoping Levels

1124

1125

1126

1127

**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

1128

1129

1130

1131

**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.

1132

1133

1134

**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.

1135

1136

1137

**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.

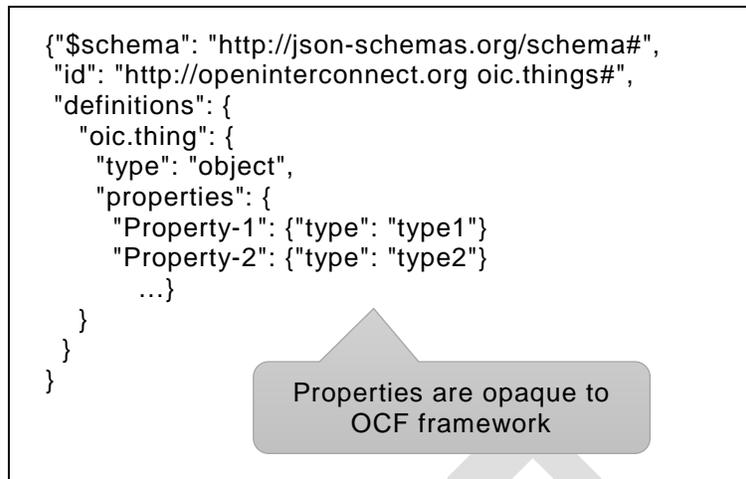
1138

1139

1140

1141

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.

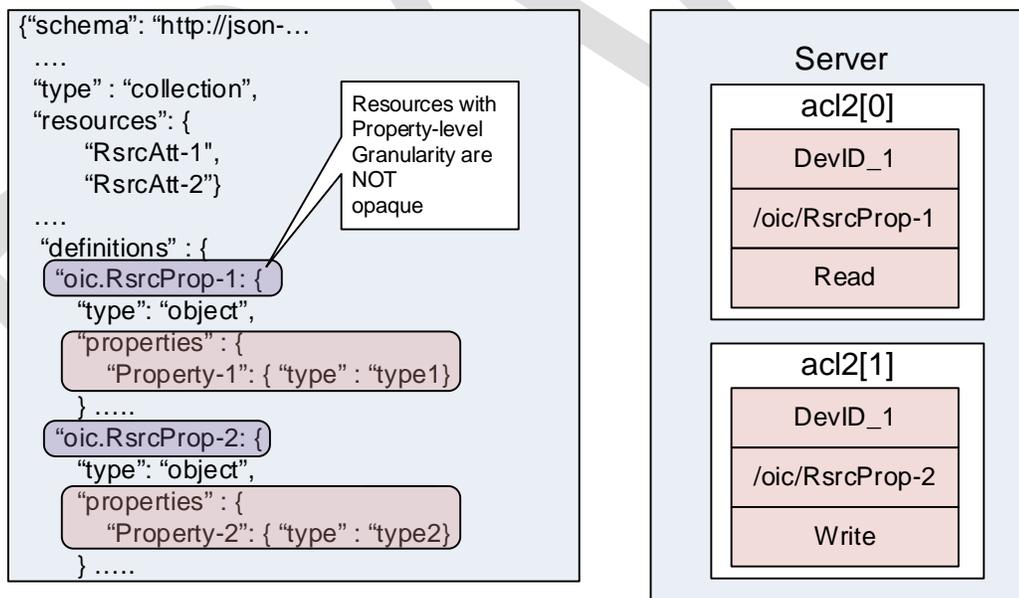


1142

1143

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

1144 Currently, OCF framework treats property level information as opaque; therefore, different  
 1145 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1  
 1146 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into  
 1147 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be  
 1148 achieved through use of Resource-level ACLs.



1149

1150

**Figure 9 – Property Level Access Control**

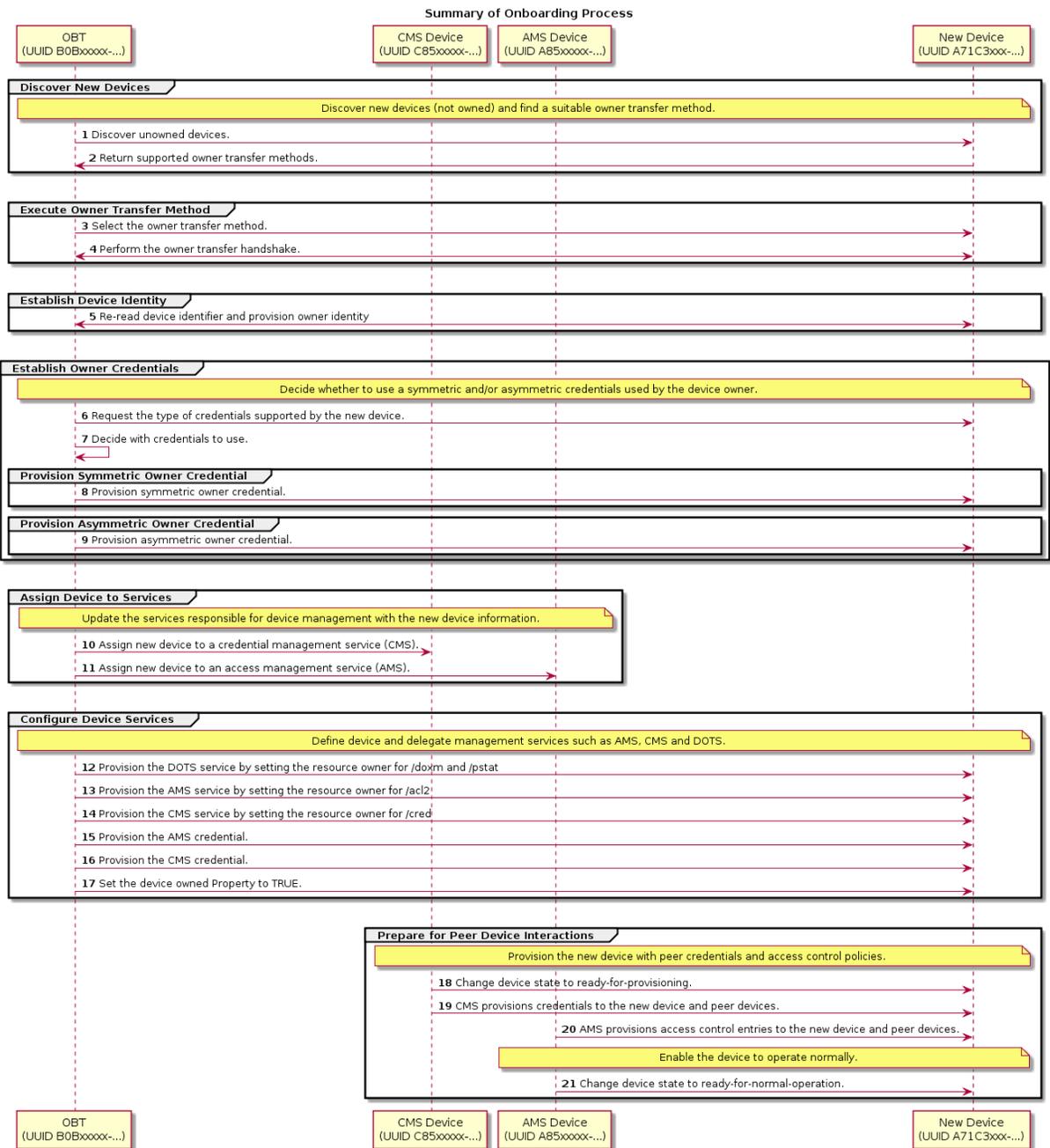
1151 **5.3 Onboarding Overview**

1152 **5.3.1 Onboarding General**

1153 Before a Device becomes operational in an OCF environment and is able to interact with other  
1154 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to  
1155 configure the ownership where the legitimate user that owns/purchases the Device uses an  
1156 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to  
1157 establish ownership. Once ownership is established, the OBT becomes the mechanism through  
1158 which the Device can then be provisioned, at the end of which the Device becomes operational  
1159 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an  
1160 OCF Device.

1161 Figure 10 depicts Onboarding Overview.

DRAFT



**Figure 10 – Onboarding Overview**

1162  
1163

1164 This clause explains the onboarding and security provisioning process but leaves the provisioning  
 1165 of non-security aspects to other OCF documents. In the context of security, all Devices are required  
 1166 to be provisioned with minimal security configuration that allows the Device to securely  
 1167 interact/communicate with other Devices in an OCF environment. This minimal security  
 1168 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified  
 1169 in 7.5.

1170 Onboarding and provisioning implementations could utilize services defined outside this document,  
 1171 it is expected that in using other services, trust between the device being onboarded and the  
 1172 various tools is not transitive. This implies that the device being onboarded will individually

1173 authenticate the credentials of each and every tool used during the onboarding process; that the  
1174 tools not share credentials or imply a trust relationship where one has not been established.

### 1175 **5.3.2 Onboarding Steps**

1176 The flowchart in Figure 11 shows the typical steps that are involved during onboarding. Although  
1177 onboarding may include a variety of non-security related steps, the diagram focus is mainly on the  
1178 security related configuration to allow a new Device to function within an OCF environment.  
1179 Onboarding typically begins with the Device becoming an Owned Device followed by configuring  
1180 the Device for the environment that it will operate in. This would include setting information such  
1181 as who can access the Device and what actions can be performed as well as what permissions the  
1182 Device has for interacting with other Devices.

DRAFT

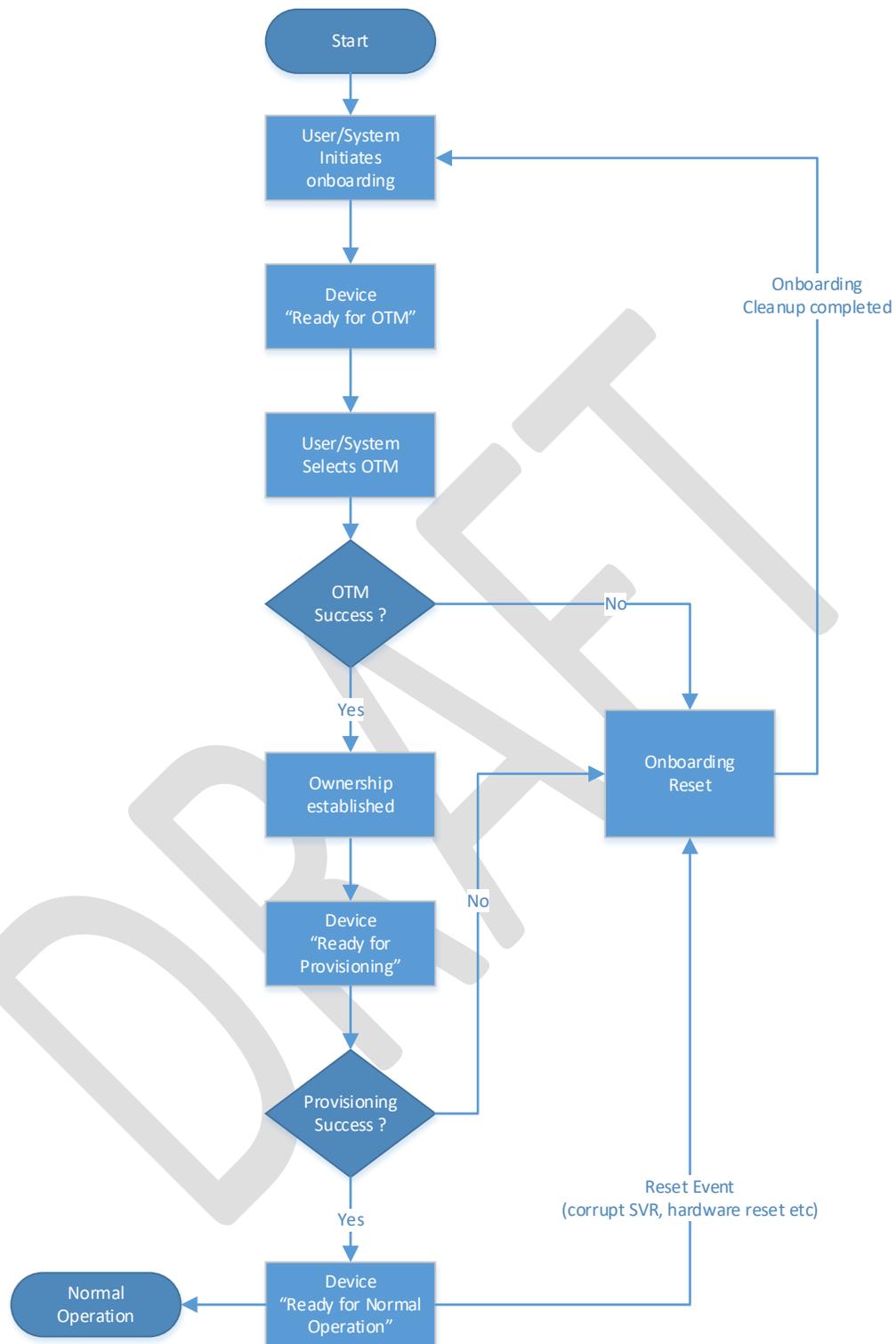


Figure 11 – OCF Onboarding Process

1183

1184

### 1185 5.3.3 Establishing a Device Owner

1186 The objective behind establishing Device ownership is to allow the legitimate user that  
 1187 owns/purchased the Device to assert itself as the owner and manager of the Device. This is done  
 Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 24

1188 through the use of an OBT that includes the creation of an ownership context between the new  
1189 Device and the OBT tool and asserts operational control and management of the Device. The OBT  
1190 can be considered a logical entity hosted by tools/ Servers such as a network management console,  
1191 a device management tool, a network-authoring tool, a network provisioning tool, a home gateway  
1192 device, or a home automation controller. A physical device hosting the OBT will be subject to some  
1193 security hardening requirements, thus preserving integrity and confidentiality of any credentials  
1194 being stored. The tool/Server that establishes Device ownership is referred to as 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 transferred  
1197 from the manufacturer/provider of the Device to the buyer/legitimate user of the new Device.

1198 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.  
1199 Owner transfer establishes the following

- 1200 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"  
1201 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during  
1202 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property  
1203 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the  
1204 identity of Device as part of ownership transfer.
- 1205 – The Device owner establishes trust in the Device through the OTM.
- 1206 – Preparing the Device for provisioning by providing credentials that may be needed.

#### 1207 **5.3.4 Provisioning for Normal Operation**

1208 Once the Device has the necessary information to initiate provisioning, the next step is to provision  
1209 additional security configuration that allows the Device to become operational. This can include  
1210 setting various parameters and may also involve multiple steps. Also provisioning of ACL's for the  
1211 various Resources hosted by the Server on the Device is done at this time. The provisioning step  
1212 is not limited to this stage only. Device provisioning can happen at multiple stages in the Device's  
1213 operational lifecycle. However specific security related provisioning of Resource and Property state  
1214 would likely happen at this stage at the end of which, each Device reaches the Onboarded Device  
1215 "Ready for Normal Operation" State. The "Ready for Normal Operation" State is expected to be  
1216 consistent and well defined regardless of the specific OTM used or regardless of the variability in  
1217 what gets provisioned. However individual OTM mechanisms and provisioning steps may specify  
1218 additional configuration of Resources and Property states. The minimal mandatory configuration  
1219 required for a Device to be in "Ready for Normal Operation" state is specified in 8.

#### 1220 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview – moved to** 1221 **OCF Cloud Security document**

#### 1222 **5.3.6 OCF Compliance Management System**

1223 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that  
1224 provides Certification status and information for OCF Devices.

1225 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:  
1226 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1227 The OBT shall possess the Root Certificate needed to enable https connection to the URI  
1228 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1229 The OBT should periodically refresh its copy of the CPL via the URI  
1230 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security  
1231 Domain owner policy requirements.

## 1232 **5.4 Provisioning**

### 1233 **5.4.1 Provisioning General**

1234 In general, provisioning may include processes during manufacturing and distribution of the Device  
1235 as well as processes after the Device has been brought into its intended environment (parts of  
1236 onboarding process). In this document, security provisioning includes, processes after ownership  
1237 transfer (even though some activities during ownership transfer and onboarding may lead to  
1238 provisioning of some data in the Device) configuration of credentials for interacting with  
1239 provisioning services, configuration of any security related Resources and credentials for dealing  
1240 with any services that the Device need to contact later on.

1241 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to  
1242 be provisioned with proper security credentials and parameters for regular operation. These  
1243 parameters can include:

- 1244 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1245 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the  
1246 same OBT, but may be part of AMS in future.

1247 As mentioned, to accommodate a scalable and modular design, these functions are considered as  
1248 services that in future could be deployed as separate servers. Currently, the deployment assumes  
1249 that these services are all deployed as part of a OBT. Regardless of physical deployment scenario,  
1250 the same security-hardening requirement) applies to any physical server that hosts the tools and  
1251 security provisioning services discussed here.

1252 Devices are *aware* of their security provisioning status. Self-awareness allows them to be proactive  
1253 about provisioning or re-provisioning security Resources as needed to achieve the devices  
1254 operational goals.

### 1255 **5.4.2 Provisioning other services**

1256 To be able to support the use of potentially different device management service hosts, each Device  
1257 Secure Virtual Resource (SVR) has an associated Resource owner identified in the Resource's  
1258 rowneruuid Property.

1259 The DOTS shall update the rowneruuid Property of the "/oic/sec/doxm" and "/oic/sec/pstat"  
1260 resources with the DOTS resource owner identifier.

1261 The DOTS shall update the rowneruuid Property of the "/oic/sec/cred" resource with the CMS  
1262 resource owner identifier.

1263 The DOTS shall update the rowneruuid Property of the "/oic/sec/acl2" resource with the AMS  
1264 resource owner identifier

1265 When these OCF Services are configured, the Device may proactively request provisioning and  
1266 verify provisioning requests are authorized. The DOTS shall provision credentials that enable  
1267 secure connections between OCF Services and the new Device. The DOTS may initiate client-  
1268 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed  
1269 provisioning by setting tm Property of the "/oic/sec/pstat" Resource.

### 1270 **5.4.3 Provisioning Credentials for Normal Operation**

1271 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1272 – Pairwise symmetric keys
- 1273 – Group symmetric keys
- 1274 – Certificates

1275 – Raw asymmetric keys

1276 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS  
1277 credential provisioned by the DOTS.

1278 The following example describes how a Device updates a symmetric key credential involving a peer  
1279 Device. The Device discovers the credential to be updated; for example, a secure connection  
1280 attempt fails. The Device requests its CMS to supply the updated credential. The CMS returns an  
1281 updated symmetric key credential. The CMS updates the corresponding symmetric key credential  
1282 on the peer Device.

#### 1283 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1284 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)  
1285 asserted by the Client requesting the Resource and compare that role identifier(s) with the  
1286 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned with  
1287 one or more role credentials.

1288 Each Device holds the role information as a Property within the credential Resource.

1289 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a  
1290 certificate role credential.

1291 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned  
1292 for a client, access to a Resource is granted if it would be granted under any of the roles.

#### 1293 **5.4.5 ACL provisioning**

1294 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.  
1295 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL  
1296 policy by updating the Device's ACL Resources.

1297 The AMS shall digitally sign an ACL as part of issuing a "/oic/sec/sacl" Resource if the Device  
1298 supports the "/oic/sec/sacl" Resource. The public key used by the Device to verify the signature  
1299 shall be provisioned by the CMS as needed. A "/oic/sec/cred" Resource with an asymmetric key  
1300 type or signed asymmetric key type is used. The "PublicData" Property contains the AMS's public  
1301 key.

### 1302 **5.5 Secure Resource Manager (SRM)**

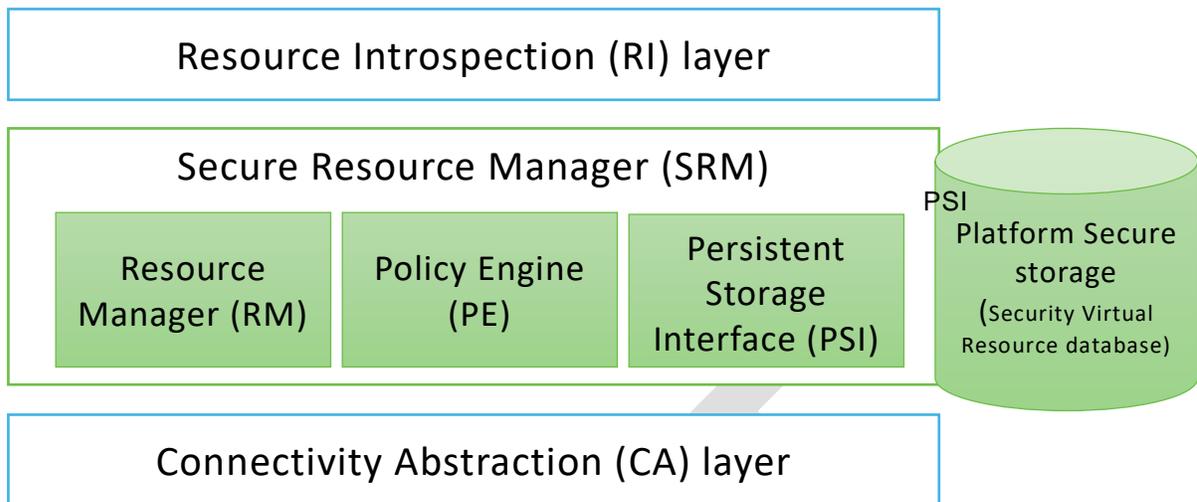
1303 SRM plays a key role in the overall security operation. In short, SRM performs both management  
1304 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3  
1305 main functional elements:

1306 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using PSI)  
1307 as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3) Responding  
1308 to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a format that is  
1309 consistent with device-specific data store format. However, the RM will use JSON format to  
1310 marshal SVR data structures before being passed to PSI for storage, or travel off-device.

1311 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control  
1312 policies responds to the requests with either "ACCESS\_GRANTED" or "ACCESS\_DENIED". To  
1313 make the access decisions, the PE consults the appropriate ACL and looks for best Access  
1314 Control Entry (ACE) that can serve the request given the subject (Device or role) that was  
1315 authenticated by DTLS.

1316 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in  
1317 its own memory and storage. The SRM design is modular such that it may be implemented in  
1318 the Platform's secure execution environment; if available.

1319 Figure 12 depicts OCF's SRM Architecture.



1320

1321

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

1322

**5.6 Credential Overview**

1323 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional  
 1324 communication. Credentials can be symmetric or asymmetric. Each device stores secret and public  
 1325 parts of its own credentials where applicable, as well as credentials for other devices that have  
 1326 been provided by the DOTS or a CMS. These credentials are then used in the establishment of  
 1327 secure communication sessions (e.g. using DTLS) to validate the identities of the participating  
 1328 parties. Role credentials are used once an authenticated session is established, to assert one or  
 1329 more roles for a device.

1330

## 1331 **6 Security for the Discovery Process**

### 1332 **6.1 Preamble**

1333 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,  
1334 called links) for the Resources hosted by the Server, complemented by attributes about those  
1335 Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-1:2018)

### 1336 **6.2 Security Considerations for Discovery**

1337 When defining discovery process, care must be taken that only a minimum set of Resources are  
1338 exposed to the discovering entity without violating security of sensitive information or privacy  
1339 requirements of the application at hand. This includes both data included in the Resources, as well  
1340 as the corresponding metadata.

1341 To achieve extensibility and scalability, this document does not provide a mandate on  
1342 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on  
1343 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any  
1344 of the Resources.

1345 The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted  
1346 Resources. (See 13.5)

1347 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery  
1348 process itself needs to be secured. This document sets the following requirements for the discovery  
1349 process:

- 1350 1) Providing integrity protection for discovered Resources.
- 1351 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1352 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)  
1353 on the known "/oic/res" Resource.

1354 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a  
1355 Server cannot determine the identity of the requester. In such cases, a Server that wants to  
1356 authenticate the Client before responding can list the secure discovery URI (e.g.  
1357 coaps://IP:PORT/oic/res ) in the unsecured "/oic/res" Resource response. This means the secure  
1358 discovery URI is by default discoverable by any Client. The Client will then be required to send a  
1359 separate unicast request using DTLS to the secure discovery URI.

1360 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to  
1361 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN  
1362 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1363 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource  
1364 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1365 {  
1366   "aclist2": [  
1367     {  
1368       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1369       "resources": [{"href": "/door"}],  
1370       "permission": 2, // RETRIEVE  
1371       "aceid": 1  
1372     }  
1373   ],
```

```

1374     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1375 }
1376 {
1377     "aclist2": [
1378     {
1379         "subject": {"authority": "owner", "role": "owner"}
1380         "resources": [{"href": "/door"}],
1381         "permission": 2, // RETRIEVE
1382         "aceid": 2
1383     }
1384     ],
1385     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1386 }
1387 {
1388     "aclist2": [
1389     {
1390         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1391         "resources": [{"href": "/door/lock"}],
1392         "permission": 4, // UPDATE
1393         "aceid": 3
1394     }
1395     ],
1396     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1397 }
1398 {
1399     "aclist2": [
1400     {
1401         "subject": {"conntype": "anon-clear"},
1402         "resources": [{"href": "/light"}],
1403         "permission": 2, // RETRIEVE
1404         "aceid": 4
1405     }
1406     ],
1407     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1408 }

```

1409 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when  
1410 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will include  
1411 the URI of the "/door" Resource metadata. Client "d2" will have access to both the Resources.  
1412 ACE2 will prevent "d4" from update.

1413 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1414 [
1415 {
1416     "href": "/door",
1417     "rt": ["oic.r.door"],
1418     "if": ["oic.if.b", "oic.if.ll"],

```

```
1419     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1420   }
1421 ]
```

1422 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1423 [
1424   {
1425     "href": "/door",
1426     "rt": ["oic.r.door"],
1427     "if": ["oic.if.b", "oic.if.ll"],
1428     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1429   },
1430   {
1431     "href": "/door/lock",
1432     "rt": ["oic.r.lock"],
1433     "if": ["oic.if.b"],
1434     "type": ["application/json", "application/exi+xml"]
1435   }
1436 ]
```

1437 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

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

1447 Discovery results delivered to any device regarding d3's "/oic/res" Resource from the unsecure interface:

```
1449 [
1450   {
1451     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1452     "href": "/light",
1453     "rt": ["oic.r.light"],
1454     "if": ["oic.if.s"]
1455   }
1456 ]
1457
```

## 1458 **7 Security Provisioning**

### 1459 **7.1 Device Identity**

#### 1460 **7.1.1 General Device Identity**

1461 Each Device, which is a logical device, is identified with a Device ID.

1462 Devices shall be identified by a Device ID value that is established as part of device onboarding.  
1463 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. "urn:uuid"). Device IDs shall be  
1464 unique within the scope of operation of the corresponding OCF Security Domain, and should be  
1465 universally unique. The DOTS shall ensure Device ID of the new Device is unique within the scope  
1466 of the owner's OCF Security Domain. The DOTS shall verify the chosen new device identifier does  
1467 not conflict with Device IDs previously introduced into the OCF Security Domain.

1468 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"  
1469 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying  
1470 authentication credentials of a peer device.

1471 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of credentials,  
1472 both its own and other Device credentials, in the "/oic/sec/cred" Resource. The device ID can be  
1473 used to distinguish between a device's own credential, and credentials for other devices.  
1474 Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the device.

1475 Device ID shall be:

- 1476 – Unique
- 1477 – Immutable
- 1478 – Verifiable

1479 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the  
1480 device as described later in this clause.

1481 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The  
1482 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in the  
1483 device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1484 An OCF Platform may have a secure execution environment, which shall be used to secure unique  
1485 identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to provide  
1486 each Device with the appropriate and separate security.

#### 1487 **7.1.2 Device Identity for Devices with UAID [Deprecated]**

1488 This clause is intentionally left blank.

### 1489 **7.2 Device Ownership**

1490 This is an informative clause. Devices are logical entities that are security endpoints that have an  
1491 identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first  
1492 initialized. Establishing device ownership is a process by which the device asserts its identity to  
1493 the DOTS and the DOTS provisions an owner identity. This exchange results in the device changing  
1494 its ownership state, thereby preventing a different DOTS from asserting administrative control over  
1495 the device.

1496 The ownership transfer process starts with the OBT discovering a new device that is in Unowned  
1497 state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new  
1498 device. At the end of ownership transfer, the following is accomplished:

- 1499 1) The DOTS shall establish a secure session with new device.

- 1500 2) Optionally asserts any of the following:
- 1501 a) Proximity (using PIN) of the OBT to the Platform.
  - 1502 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific
  - 1503 attributes.
- 1504 3) Determines the device identifier.
- 1505 4) Determines the device owner.
- 1506 5) Specifies the device owner (e.g. Device ID of the OBT).
- 1507 6) Provisions the device with owner's credentials.
- 1508 7) Sets the "Owned" state of the new device to TRUE.
- 1509 .

### 1510 **7.3 Device Ownership Transfer Methods**

#### 1511 **7.3.1 OTM implementation requirements**

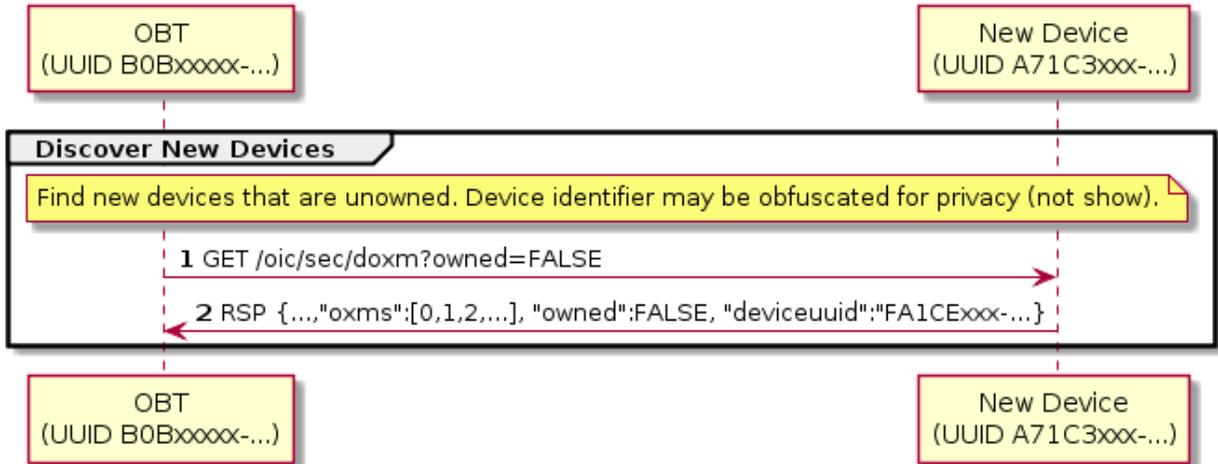
1512 This document provides specifications for several methods for ownership transfer. Implementation  
1513 of each individual ownership transfer method is considered optional. However, each device shall  
1514 implement at least one of the ownership transfer methods not including vendor specific methods.

1515 All OTMs included in this document are considered optional. Each vendor is required to choose  
1516 and implement at least one of the OTMs specified in this document. The OCF, does however,  
1517 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability  
1518 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with  
1519 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the preferred  
1520 approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in designing  
1521 vendor-specific OTMs.

1522 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer  
1523 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.  
1524 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the  
1525 "/oic/sec/doxm" Resource. The DOTS shall query the Device's supported credential types using  
1526 the "credtype" Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision  
1527 credentials according to the credential types supported.

1528 Figure 13 depicts new Device discovery sequence.

### Discover New Devices Sequence



1529

1530

Figure 13 – Discover New Device Sequence

1531

1532

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.

1533 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCS  
 1534 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for  
 1535 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the  
 1536 new Device.

1537 The new device may have to perform some initialization steps at the beginning of an OTM. For  
 1538 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN  
 1539 value. The OBT shall POST to the oxmsel property of "/oic/sec/doxm" the value corresponding to  
 1540 the OTM being used, before performing other OTM steps. This POST notifies the new device that  
 1541 ownership transfer is starting.

1542 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT and  
 1543 the OBT to authenticate to the new device.

1544 The DOTS may perform additional provisioning steps subsequent to owner transfer success  
 1545 leveraging the established OTM session.

1546 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall  
 1547 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array  
 1548 includes a SVR.

### 1549 7.3.2 SharedKey Credential Calculation

1550 The SharedKey credential is derived using a PRF that accepts the key\_block value resulting from  
1551 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following  
1552 calculation to ensure interoperability across vendor products:

1553 SharedKey = PRF(Secret, Message);

1554 Where:

- 1555 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1556 - Secret is the key\_block resulting from the DTLS handshake
  - 1557 ▪ See IETF RFC 5246 clause 6.3
  - 1558 ▪ The length of key\_block depends on cipher suite.
    - 1559 • (e.g. 96 bytes for TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
    - 1560 40 bytes for TLS\_PSK\_WITH\_AES\_128\_CCM\_8)
- 1561 - Message is a concatenation of the following:
  - 1562 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
    - 1563 • See clause 13.2.4 for specific DoxmTypes
  - 1564 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
    - 1565 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
  - 1566 ▪ Device ID is new device's UUID Device ID
    - 1567 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1568 - SharedKey Length will be 32 octets.
  - 1569 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used.
  - 1570 DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

### 1571 7.3.3 Certificate Credential Generation

1572 The Certificate Credential will be used by Devices for secure bidirectional communication. The  
1573 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used  
1574 to mutually establish the authenticity of the Device. The onboarding details for certificate generation  
1575 will be specified in a later version of this document.

### 1576 7.3.4 Just-Works OTM

#### 1577 7.3.4.1 Just-Works OTM General

1578 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a  
1579 secure connection through which a device should be provisioned for use within the owner's OCF  
1580 Security Domain. Provisioning additional credentials and Resources is a typical step following  
1581 ownership establishment. The pre-shared key is called SharedKey.

1582 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite  
1583 defined for the Just-works OTM.

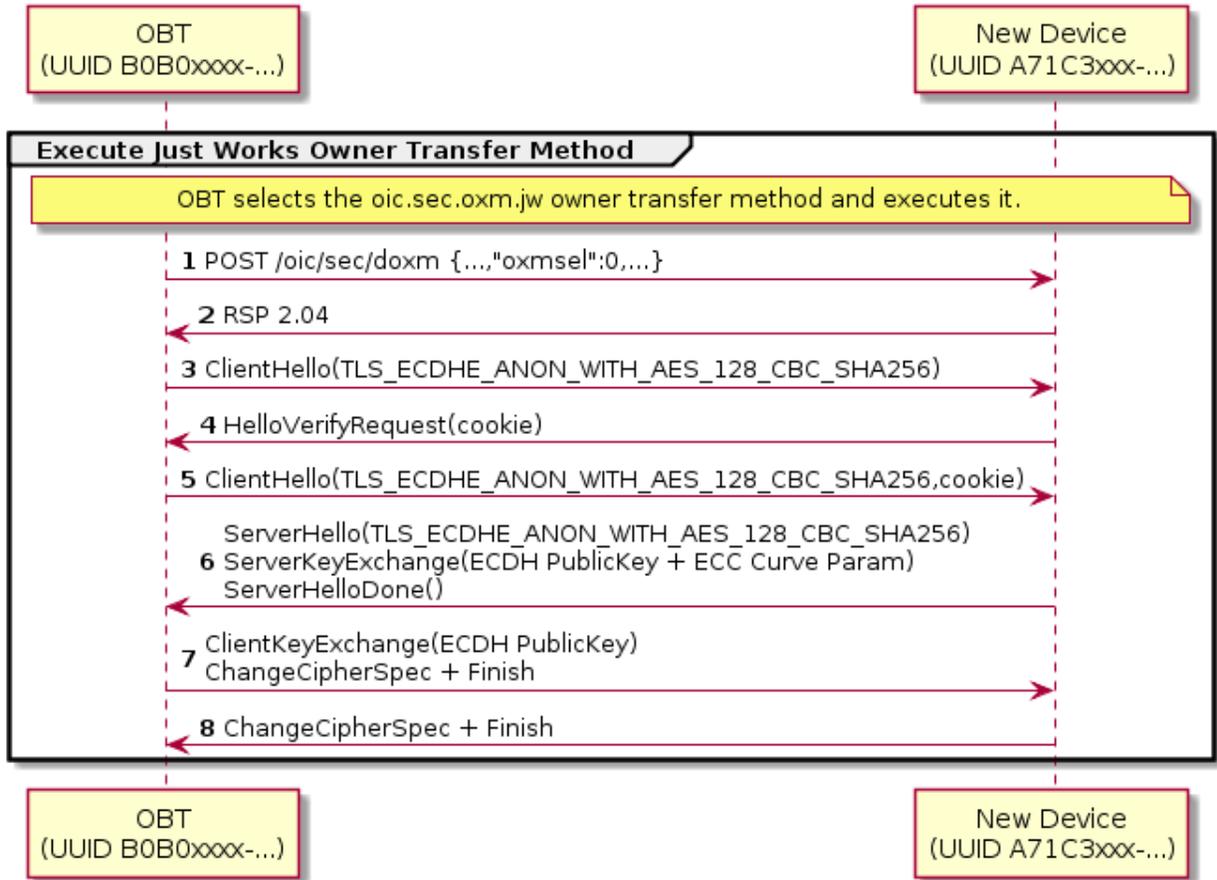
1584 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1585 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,  
1586 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

1587 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for  
1588 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1589 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.

### Perform Just-Works Owner Transfer Method



1590

1591

1592

1593

Figure 14 – A Just Works OTM

Table 2 – A Just Works OTM Details

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

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

1594

#### 7.3.4.2 Security Considerations

1595

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

1596

1597

1598

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

1599

1600

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

1601

1602 temporal device ID that could differ from the temporal value during the secure session in which  
1603 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not conflict  
1604 with a Device ID already in use. If it is already in use the existing credentials are used to establish  
1605 a secure session.

1606 An un-owned Device that also has established device credentials might be an indication of a  
1607 corrupted or compromised device.

### 1608 **7.3.5 Random PIN Based OTM**

#### 1609 **7.3.5.1 Random PIN OTM General**

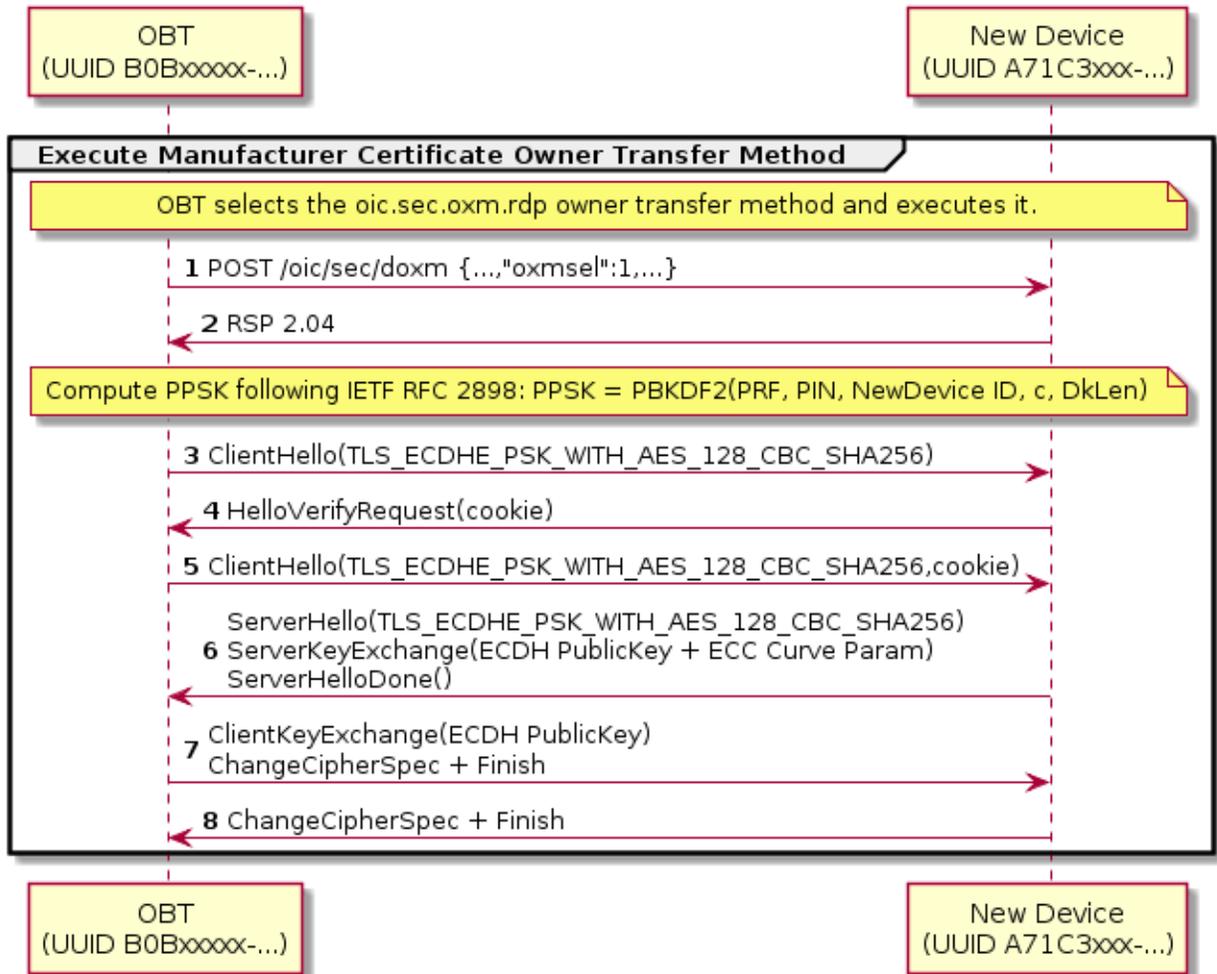
1610 The Random PIN method establishes physical proximity between the new device and the OBT can  
1611 prevent man-in-the-middle attacks. The Device generates a random number that is communicated  
1612 to the OBT over an out-of-band channel. The definition of out-of-band communications channel is  
1613 outside the scope of the definition of device OTMs. The OBT and new Device use the PIN in a key  
1614 exchange as evidence that someone authorized the transfer of ownership by having physical  
1615 access to the new Device via the out-of-band-channel.

#### 1616 **7.3.5.2 Random PIN Owner Transfer Sequence**

1617 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.

DRAFT

## Perform Random PIN Device Owner Transfer Method



1618  
1619  
1620  
1621

**Figure 15 – Random PIN-based OTM**

**Table 3 – Random PIN-based OTM Details**

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

1622 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF  
 1623 RFC 2898 and a PIN exchanged via an out-of-band method to generate a pre-shared key. The PIN-  
 1624 authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a PSK.

1625 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)  
1626 The PBKDF2 function has the following parameters:  
1627 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.  
1628 - PIN – obtain via out-of-band channel.  
1629 - Device ID – UUID of the new device.  
1630 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2  
1631 - c – Iteration count initialized to 1000  
1632 - dkLen – Desired length of the derived PSK in octets.

### 1633 7.3.5.3 Security Considerations

1634 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with  
1635 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials  
1636 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,  
1637 allows an attacker to masquerade as the onboarded device.

1638 It is recommended that the entropy of the PIN be enough to withstand an online brute-force attack,  
1639 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-9a-z), or  
1640 a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack (MITM) is  
1641 when the attacker is active on the network and can intercept and modify messages between the  
1642 OBT and device. In the MITM attack, the attacker must recover the PIN from the key exchange  
1643 messages in "real time", i.e., before the peer's time out and abort the connection attempt. Having  
1644 recovered the PIN, he can complete the authentication step of key exchange. The guidance given  
1645 here calls for a minimum of 40 bits of entropy, however, the assurance this provides depends on  
1646 the resources available to the attacker. Given the parallelizable nature of a brute force guessing  
1647 attack, the attack enjoys a linear speedup as more cores/threads are added. A more conservative  
1648 amount of entropy would be 64 bits. Since the Random PIN OTM requires using a DTLS ciphersuite  
1649 that includes an ECDHE key exchange, the security of the Random PIN OTM is always at least  
1650 equivalent to the security of the JustWorks OTM.

1651 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN. The  
1652 rationale is to increase the cost of a brute force attack, by increasing the cost of each guess in the  
1653 attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an effective way  
1654 to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify the reduction,  
1655 since an X-fold increase in time spent by the honest peers does not directly translate to an X-fold  
1656 increase in time by the attacker. This asymmetry is because the attacker may use specialized  
1657 implementations and hardware not available to honest peers. For this reason, when deciding how  
1658 much entropy to use for a PIN, it is recommended that implementers assume PBKDF2 provides no  
1659 security, and ensure the PIN has sufficient entropy.

1660 The Random PIN device OTM security depends on an assumption that a secure out-of-band  
1661 method for communicating a randomly generated PIN from the new device to the OBT exists. If the  
1662 OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN, and  
1663 the attacks described above apply. The out-of-band mechanism should be chosen such that it  
1664 requires proximity between the OBT and the new device. The attacker is assumed to not have  
1665 compromised the out-of-band-channel. As an example OOB channel, the device may display a PIN  
1666 to be entered into the OBT software. Another example is for the device to encode the PIN as a 2D  
1667 barcode and display it for a camera on the OBT device to capture and decode.

### 1668 7.3.6 Manufacturer Certificate Based OTM

#### 1669 7.3.6.1 Manufacturer Certificate Based OTM General

1670 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the  
1671 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device  
1672 and the OBT.

1673 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust  
1674 anchor.

1675 For some environments, policies or administrators, additional information about device  
1676 characteristics may be sought. This list of additional attestations that OCF may or may not have  
1677 tested (understanding that some attestations are incapable of testing or for which testing may be  
1678 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3  
1679 extension described in 9.4.2.2.6.

1680 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with  
1681 certificate data to authenticate their identities with the OBT in the process of bringing a new device  
1682 into operation on an OCF Security Domain. The onboarding process involves several discrete steps:

1683 1) Pre-on-board conditions

1684 a) The credential element of the Device's credential Resource ("/oic/sec/cred") containing the  
1685 manufacturer certificate shall be identified by the "credusage" Property containing the string  
1686 "oic.sec.cred.mfgcert" to indicate that the credential contains a manufacturer certificate.

1687 b) The manufacturer certificate chain shall be contained in the identified credential element's  
1688 "publicdata" Property.

1689 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1690 d) If the device requires authentication of the OBT as part of ownership transfer, it is presumed  
1691 that the OBT has been registered and has obtained a certificate for its unique and immutable  
1692 ECC asymmetric key pair signed by the predetermined Trust Anchor.

1693 e) User has configured the OBT app with network access info and account info (if any).

1694 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the  
1695 Device may authenticate the OBT to verify the OBT signature.

1696 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready  
1697 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted link  
1698 in accordance with the negotiated cipher suite.

#### 1699 **7.3.6.2 Certificate Profiles**

1700 See 9.4.2 for details.

#### 1701 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

1702 In order for full, mutual authentication to occur between the device and the OBT, both the device  
1703 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This implies  
1704 that OCF may need to obtain services from a Certificate Authority (e.g. Symantec, Verisign, etc.)  
1705 to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors are derived.

1706 The OBT shall authenticate the device during onboarding. However, the device is not required to  
1707 authenticate the OBT due to potential resource constraints on the device.

1708 In the case where the Device does NOT authenticate the OBT software, there is the possibility of  
1709 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,  
1710 which can compromise OCF Security Domain access credentials and/or personal information.

#### 1711 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1712 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.

## Perform Manufacturer Certificate Owner Transfer Method

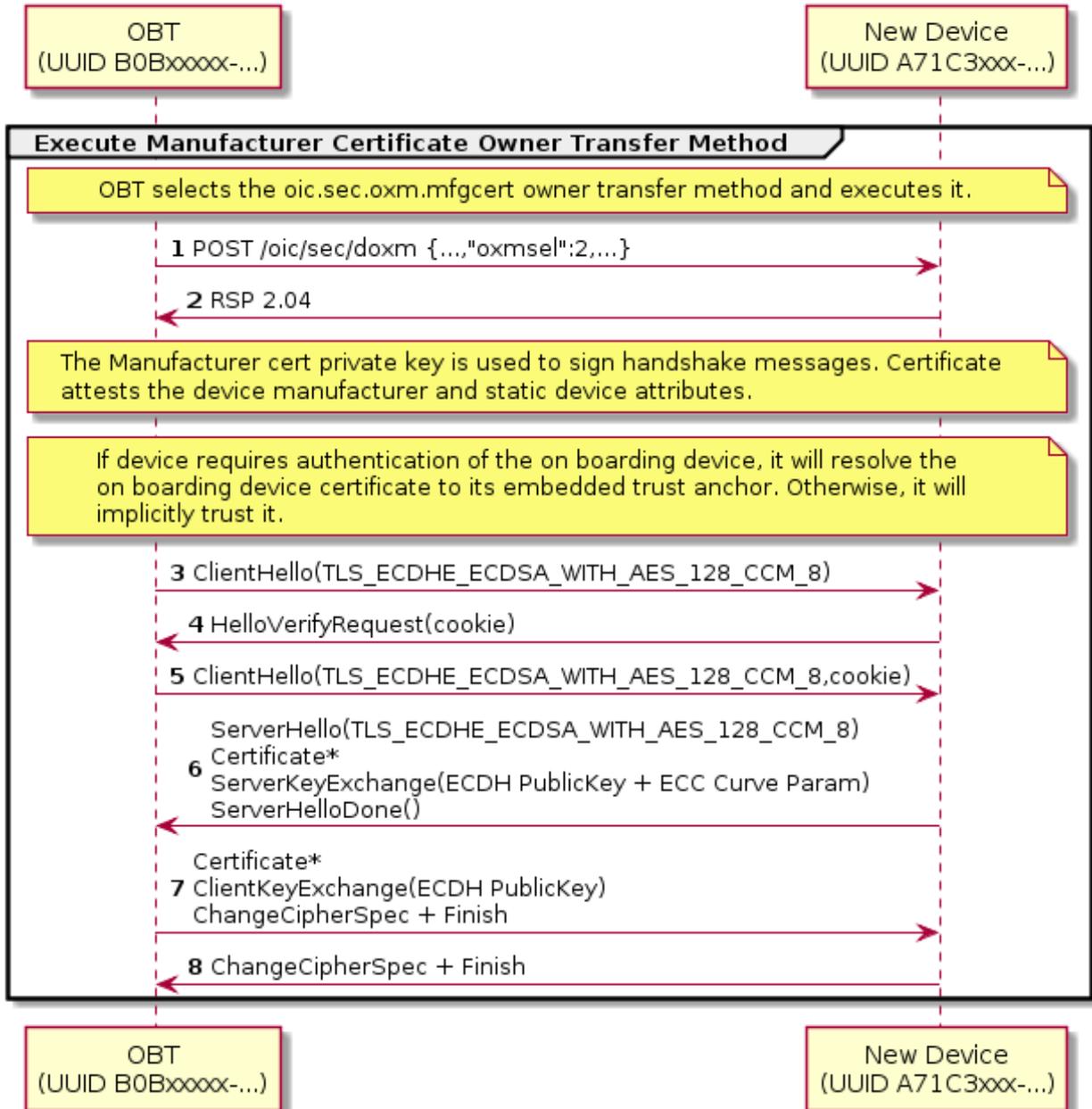


Figure 16 – Manufacturer Certificate Based OTM Sequence

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.
--	--

1717 **7.3.6.5 Security Considerations**

1718 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of  
1719 assurance that the private key cannot be compromised.

1720 The Platform manufacturer issues the manufacturer certificate and attests the private key  
1721 protection mechanism.

1722 **7.3.7 Vendor Specific OTMs**

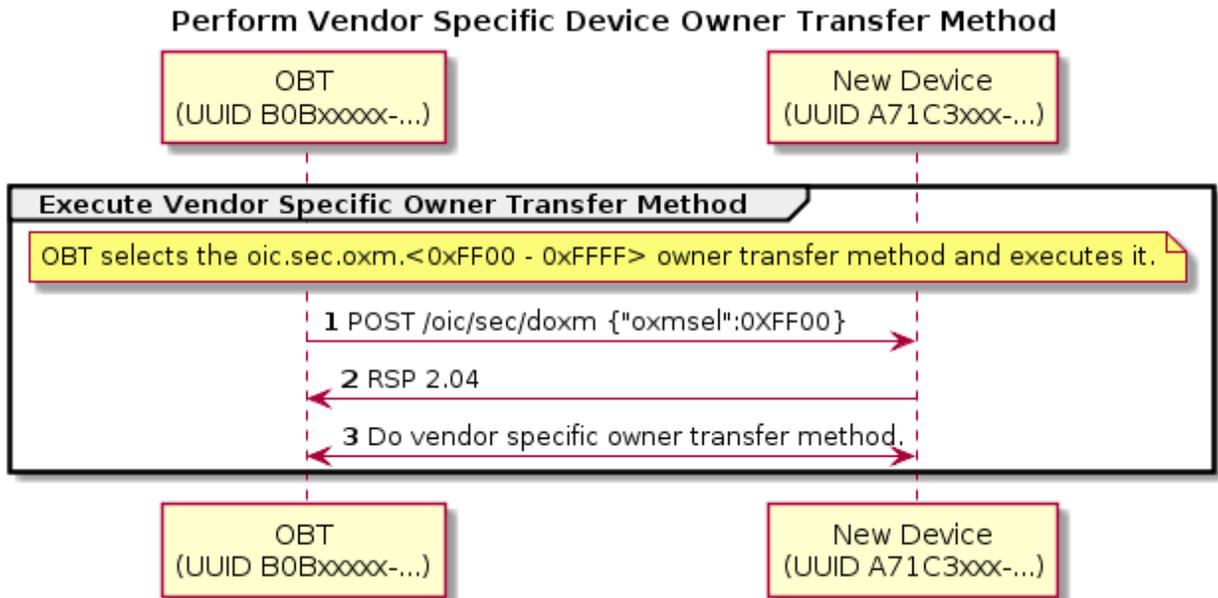
1723 **7.3.7.1 Vendor Specific OTM General**

1724 The OCF anticipates situations where a vendor will need to implement an OTM that accommodates  
1725 manufacturing or Device constraints. The Device OTM resource is extensible for this purpose.  
1726 Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1727 – The OBT must determine which credential types are supported by the Device. This is  
1728 accomplished by querying the Device's "/oic/sec/doxm" Resource to identify supported  
1729 credential types.
- 1730 – The OBT provisions the Device with OC(s).
- 1731 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1732 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security  
1733 Domain subsequent to ownership establishment.
- 1734 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks  
1735 to be performed by a security service.

1736 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1737 Vendor-specific OTM sequence example is shown in Figure 17 and steps described in Table 5.



1738

1739 **Figure 17 – Vendor-specific Owner Transfer Sequence**

1740

1741

**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

1742 **7.3.7.3 Security Considerations**

1743 The vendor is responsible for considering security threats and mitigation strategies.

1744 **7.3.8 Establishing Owner Credentials**

1745 Once the OBT and the new Device have authenticated and established an encrypted connection  
1746 using one of the defined OTM methods.

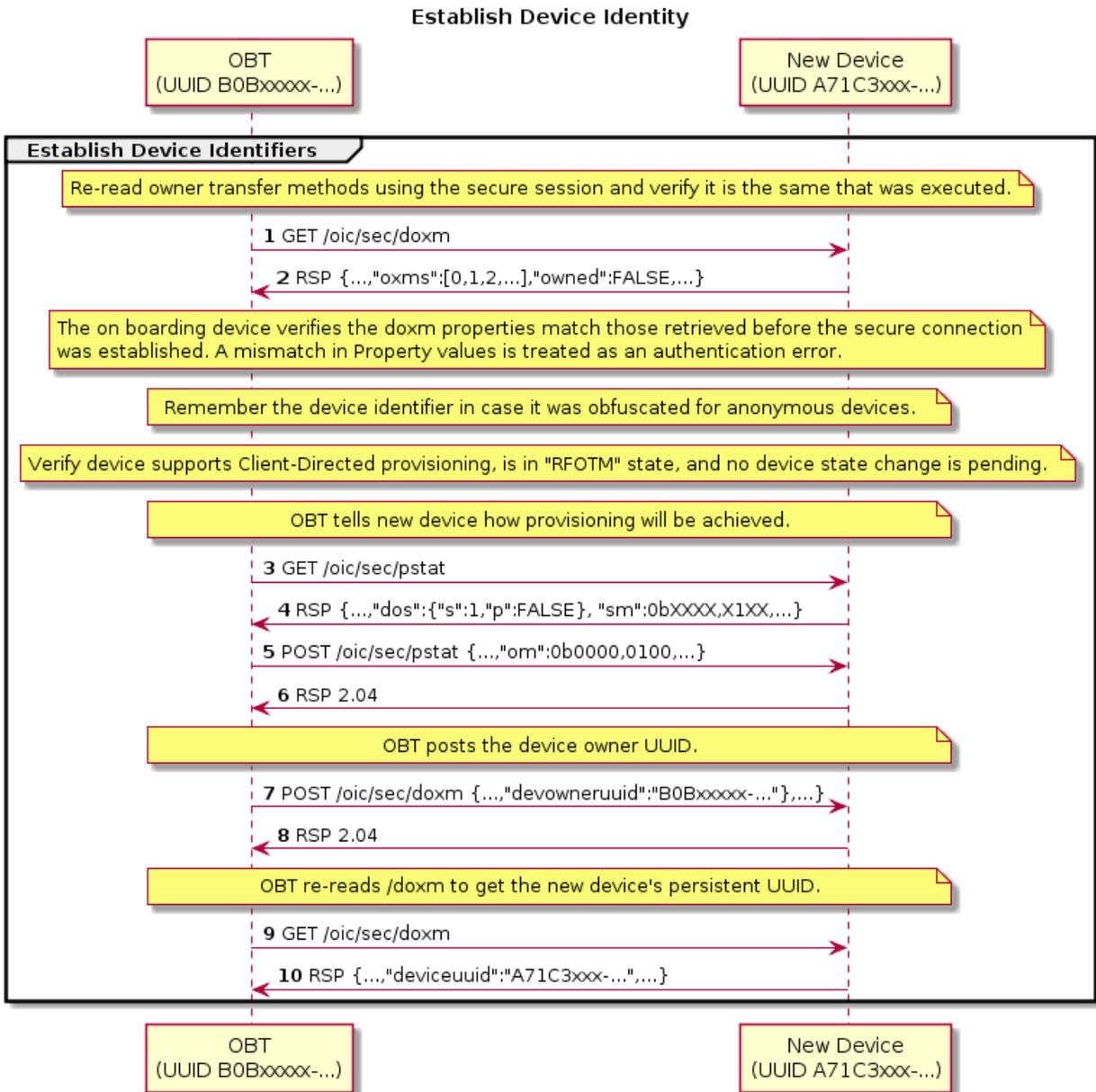
1747 Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security  
1748 Domain access information, provisioning functions, shared keys, or Kerberos tickets.

1749 The OBT might then provision the new Device with additional credentials for Device management  
1750 and Device-to-Device communications. These credentials may consist of certificates with  
1751 signatures, UAID based on the Device public key, PSK, etc.

1752 The steps for establishing Device's owner credentials (OC) are:

- 1753 1) The OBT shall establish the Device ID and Device owner uuid - See Figure 18 and Table 6.
- 1754 2) The OBT then establishes Device's OC - See Figure 19 and Table 7. This can be either:
  - 1755 a) Symmetric credential - See Figure 20 and Table 8.
  - 1756 b) Asymmetric credential - See Figure 21 and Table 9.
- 1757 3) Configure Device services - See Figure 22 and Table 10.
- 1758 4) Configure Device for peer to peer interaction - See Figure 23 and Table 11.

1759



1760  
1761  
1762  
1763

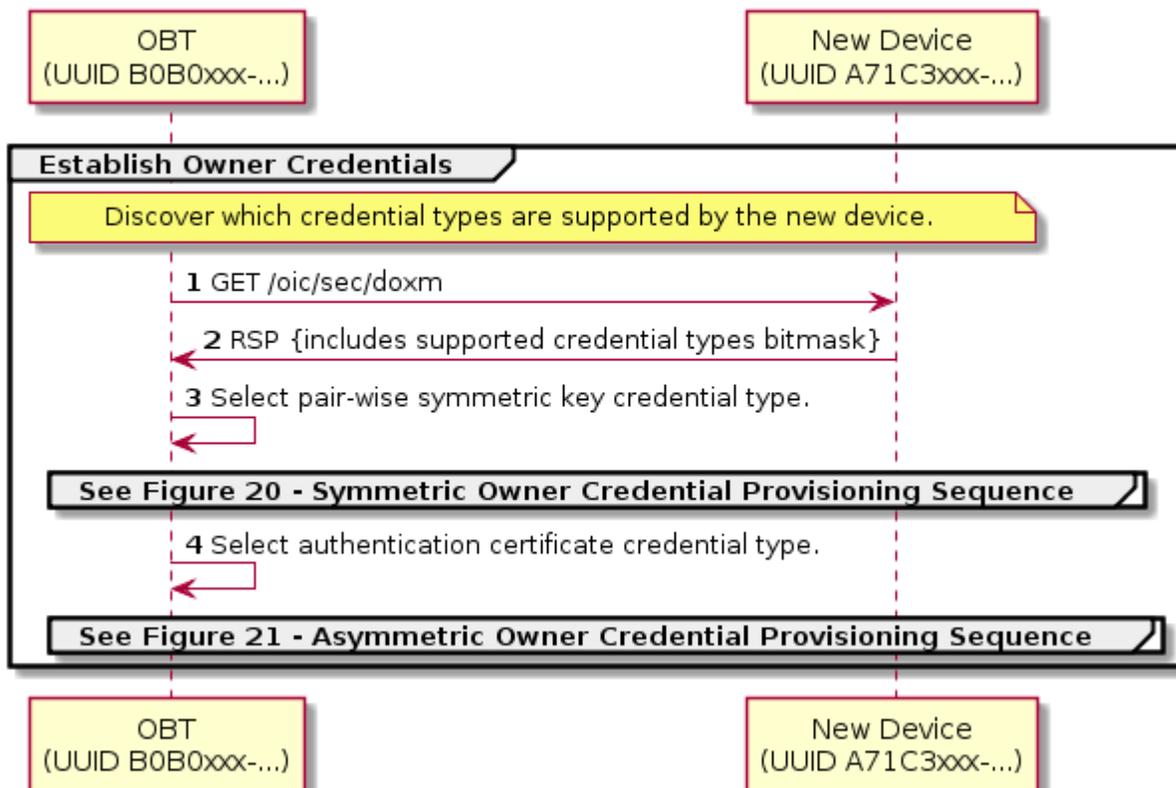
**Figure 18 – Establish Device Identity Flow**

**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



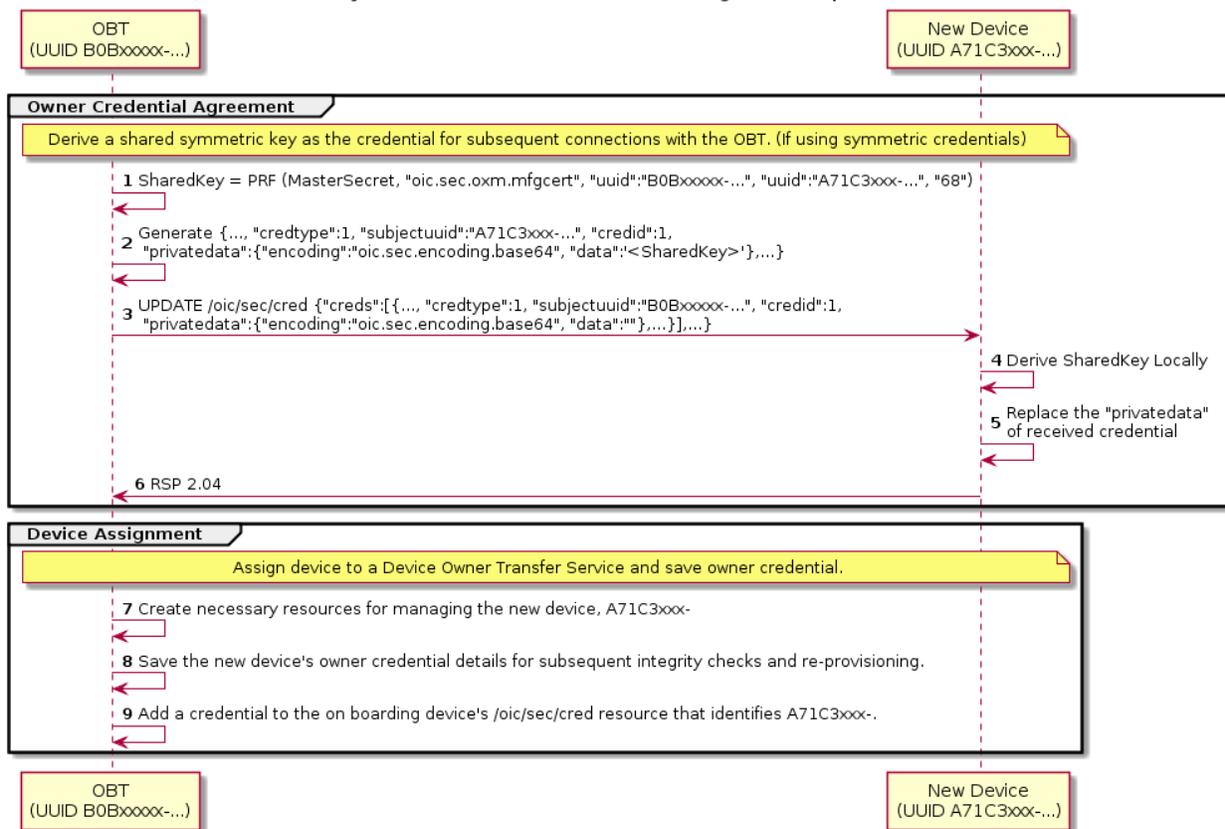
1764  
1765  
1766  
1767

Figure 19 – Owner Credential Selection Provisioning Sequence

Table 7 – Owner Credential Selection Details

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

### Symmetric Owner Credential (OC) Assignment Sequence



1768

1769

1770

1771

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

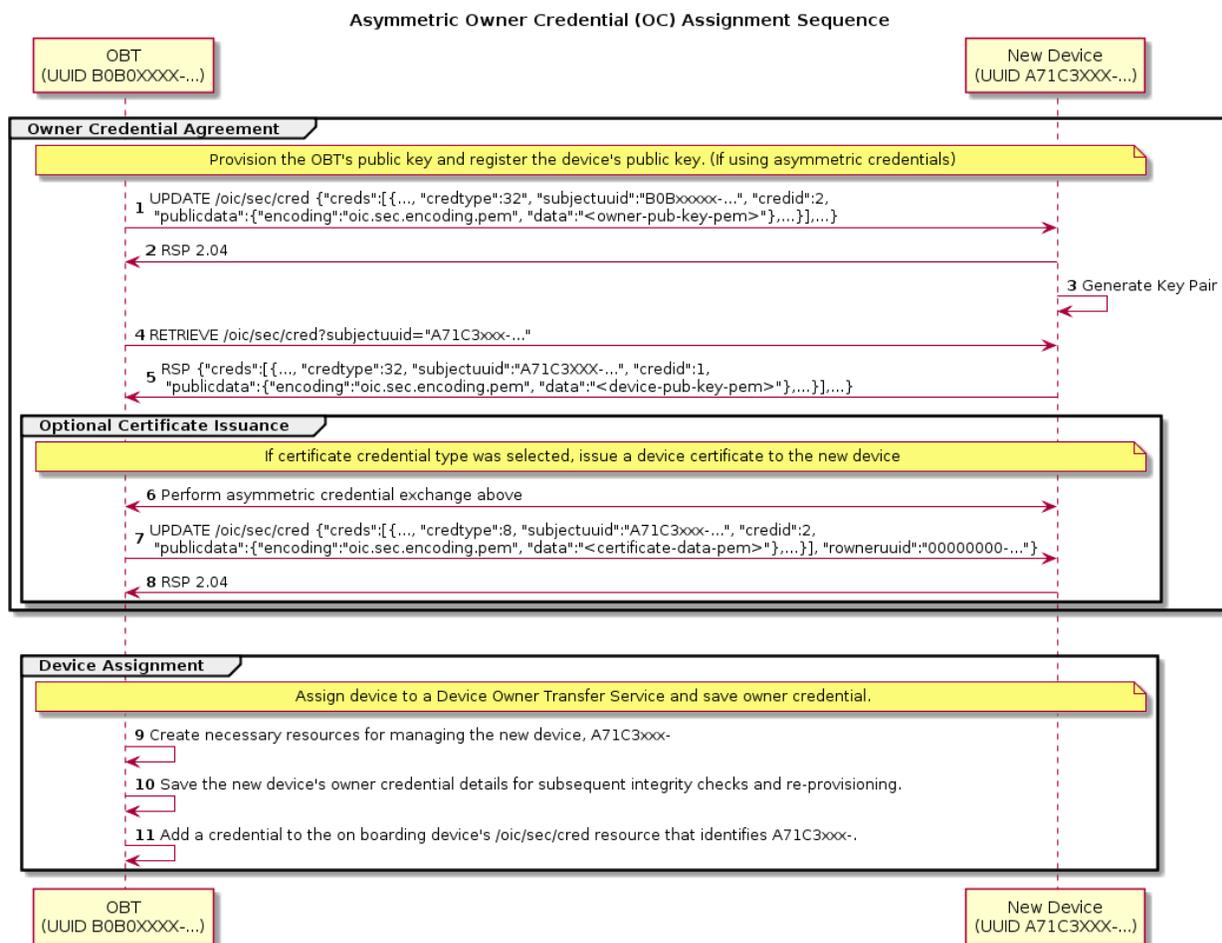
**Table 8 – Symmetric Owner Credential Assignment Details**

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.
---	---

1772 In particular, if the OBT selects symmetric owner credentials:

- 1773 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
- 1774 described in 7.3.2.
- 1775 – The OBT shall send an empty key to the new Device's "/oic/sec/cred" Resource, identified as a
- 1776 symmetric pair-wise key.
- 1777 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
- 1778 generate the Shared Key using the SharedKey Credential Calculation method described in 7.3.2
- 1779 and store it with the owner credential.
- 1780 – The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred"
- 1781 Resource to authenticate the owner during subsequent connections.



1782

1783

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

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

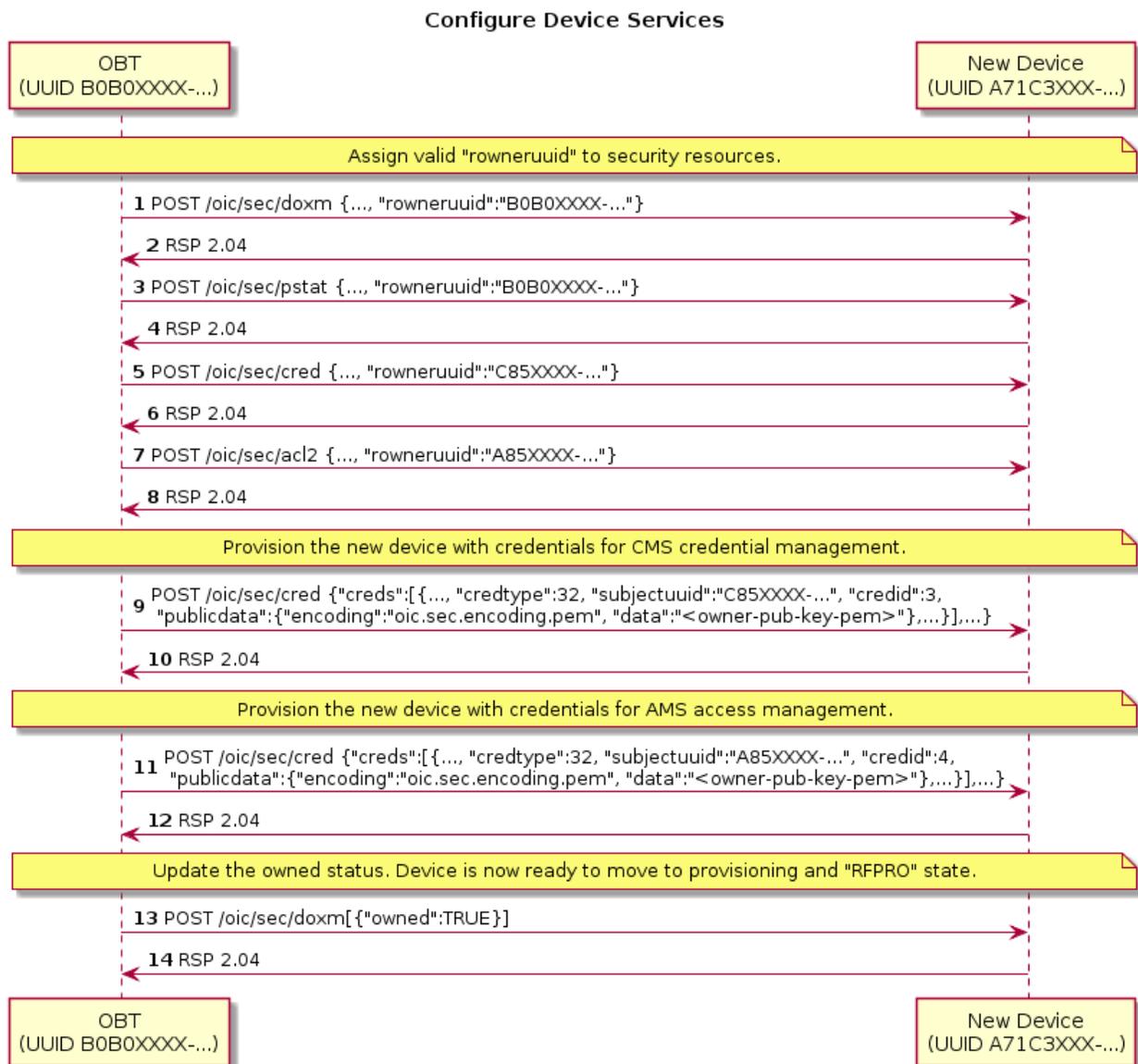
Step	Description
If an asymmetric or certificate owner credential type was selected by the OBT	
1, 2	The OBT creates an asymmetric type credential Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric type credential Resource Property set generated at step 25. It may be used subsequently to authenticate the new Device.
If certificate owner credential type is selected by the OBT	
6-8	The steps for creating an asymmetric credential type are performed. In addition, the OBT instantiates a newly-created certificate (or certificate chain) on the new Device.
9	The onboarding service creates a subjects resource for the new device (e.g./A71C3xxx-...)
10	The onboarding service provisions its "/oic/svc/dots/subjects/A71C3xxx-/cred" resource with the owner credential. Credential type is PUBLIC KEY.
11	(optional) The onboarding service provisions 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.

1786 If the OBT selects asymmetric owner credentials:

- 1787 – The OBT shall add its public key to the new Device's "/oic/sec/cred" Resource, identified as an
- 1788 Asymmetric Encryption Key.
- 1789 – The OBT shall query the "/oic/sec/cred" Resource from the new Device, supplying the new
- 1790 Device's UUID via the SubjectID query parameter. In response, the new Device shall return the
- 1791 public Asymmetric Encryption Key, which the OBT shall retain for future owner authentication
- 1792 of the new Device.

1793 If the OBT selects certificate owner credentials:

- 1794 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the
- 1795 public key returned by the new Device, signed by a mutually-trusted CA, and complying with
- 1796 the Certificate Credential Generation requirements defined in 7.3.3.
- 1797 – The OBT shall add the newly-created certificate chain to the "/oic/sec/cred" Resource, identified
- 1798 as an Asymmetric Signing Key with Certificate.

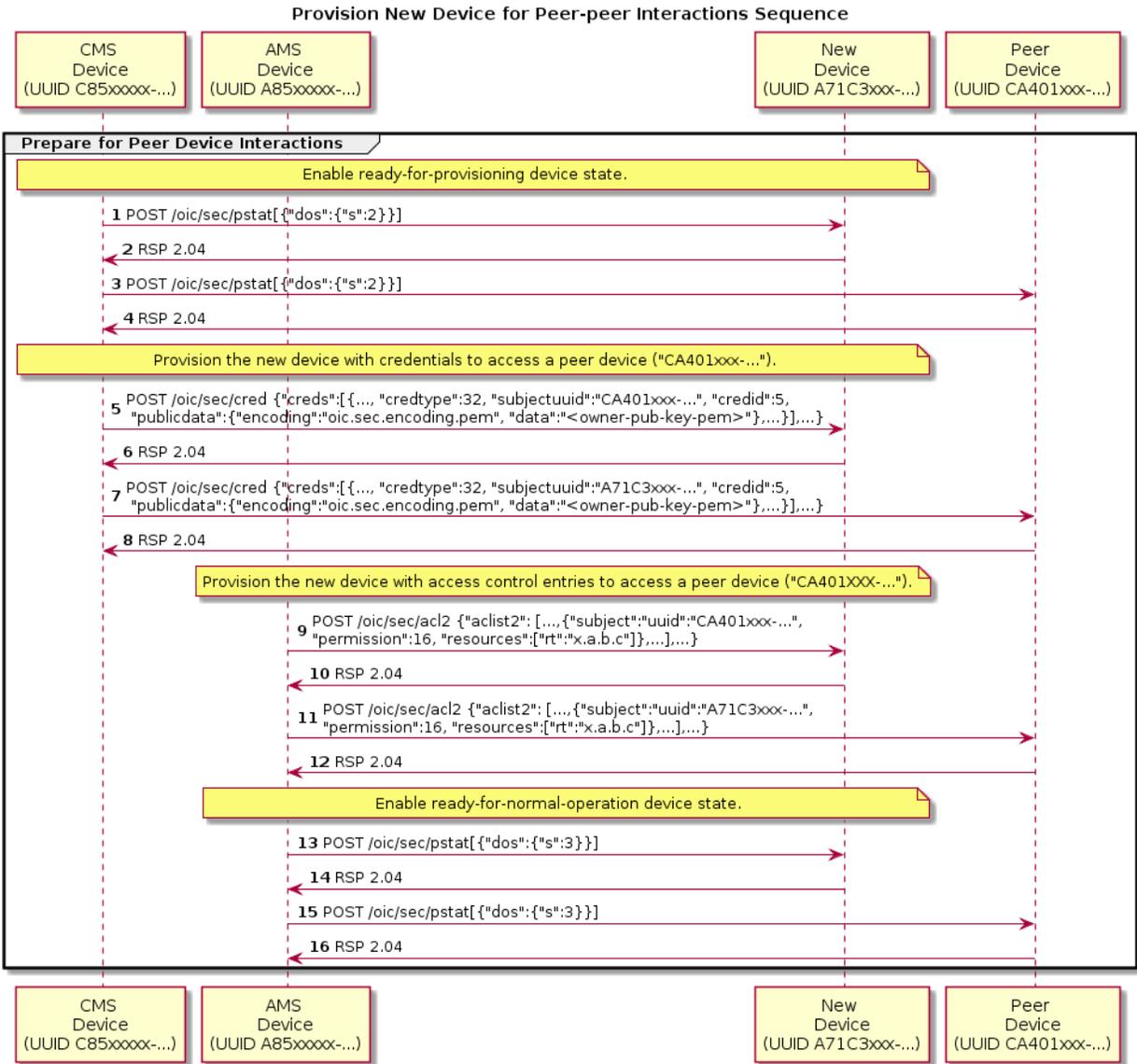


1799  
1800  
1801  
1802

**Figure 22 – Configure Device Services**

**Table 10 – Configure Device Services Detail**

Step	Description
1 - 8	The OBT assigns rowneruid 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.



1803

1804

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

1805

1806

**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.

1807 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1808 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are  
1809 acceptable when transferring ownership of a new Device. Some of the factors to be considered  
1810 when determining those requirements are:

- 1811 – The security considerations described for each of the OTMs
- 1812 – The probability that a man-in-the-middle attacker might be present in the environment used to  
1813 perform the ownership transfer

1814 For example, the operator of an OBT might require that all of the Devices being onboarded support  
1815 either the Random PIN or the Manufacturer Certificate OTM.

1816 When such a local OTM policy exists, the OBT should try to use just the OTMs that are acceptable  
1817 according to that policy, regardless of the doxm contents obtained during step 1 from the sequence  
1818 diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an unauthenticated and/or  
1819 unencrypted connection between the OBT and the Device, the contents of the response to the GET  
1820 request might have been tampered by a man-in-the-middle attacker. For example, the list of OTMs  
1821 supported by the new Device might have been altered by the attacker.

1822 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new  
1823 Device to fail. In such cases, the OBT has no way of determining if the session failed because the  
1824 new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle injected  
1825 such a failure into the communication between the OBT and the new Device.

1826 The current version of this document leaves the design and user experience related to the OTM  
1827 policy as OBT implementation details.

1828 **7.3.10 Security Profile Assignment**

1829 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results  
1830 could be accessed from a manufacturer's certificate, OCF web server or other public repository.  
1831 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device is  
1832 authorized to possess and configures the Device with the subset of evaluated security profiles best  
1833 suited for the OCF Security Domain owner's intended segmentation strategy.

1834 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles" Property  
1835 of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification process.  
1836 The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the values  
1837 contained in the "supportedprofiles". The manufacturer default value shall be re-asserted when the  
1838 Device transitions to RESET Device State.

1839 The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in  
1840 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as  
1841 directed by a Security Profile.

1842 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a subset  
1843 of the OCF Security Profiles values the Device achieved as part of OCF Conformance testing. The  
1844 DOTS may locate conformance results by inspecting manufacturer certificates supplied with the  
1845 OCF Device by selecting the "credusage" Property of the "/oic/sec/cred" Resource having the value  
1846 of "oic.sec.cred.mfgcert". The DOTS may further locate conformance results by visiting a well-  
1847 known OCF web site URI corresponding to the ocfCPLAttributes extension fields (clause 9.4.2.2.7).  
1848 The DOTS may select a subset of Security Profiles (from those evaluated by OCF conformance  
1849 testing) based on a local policy.

1850 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries to  
1851 allow DOTS access subsequent to onboarding.

1852 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the value  
1853 that most correctly depicts the OCF Security Domain owner's intended Device deployment strategy.

1854 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID") to  
1855 indicate the OCF Security Domain owner's intention to segment the OCF Security Domain  
1856 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the  
1857 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security  
1858 Profiles when issuing role credentials.

1859 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control  
1860 entries that include the role designation(s).

## 1861 **7.4 Provisioning**

### 1862 **7.4.1 Provisioning Flows**

#### 1863 **7.4.1.1 Provisioning Flows General**

1864 As part of onboarding a new Device a secure channel is formed between the new Device and the  
1865 OBT. Subsequent to the Device ownership status being changed to "owned", there is an opportunity  
1866 to begin provisioning. The OBT decides how the new Device will be managed going forward and  
1867 provisions the support services that should be subsequently used to complete Device provisioning  
1868 and on-going Device management.

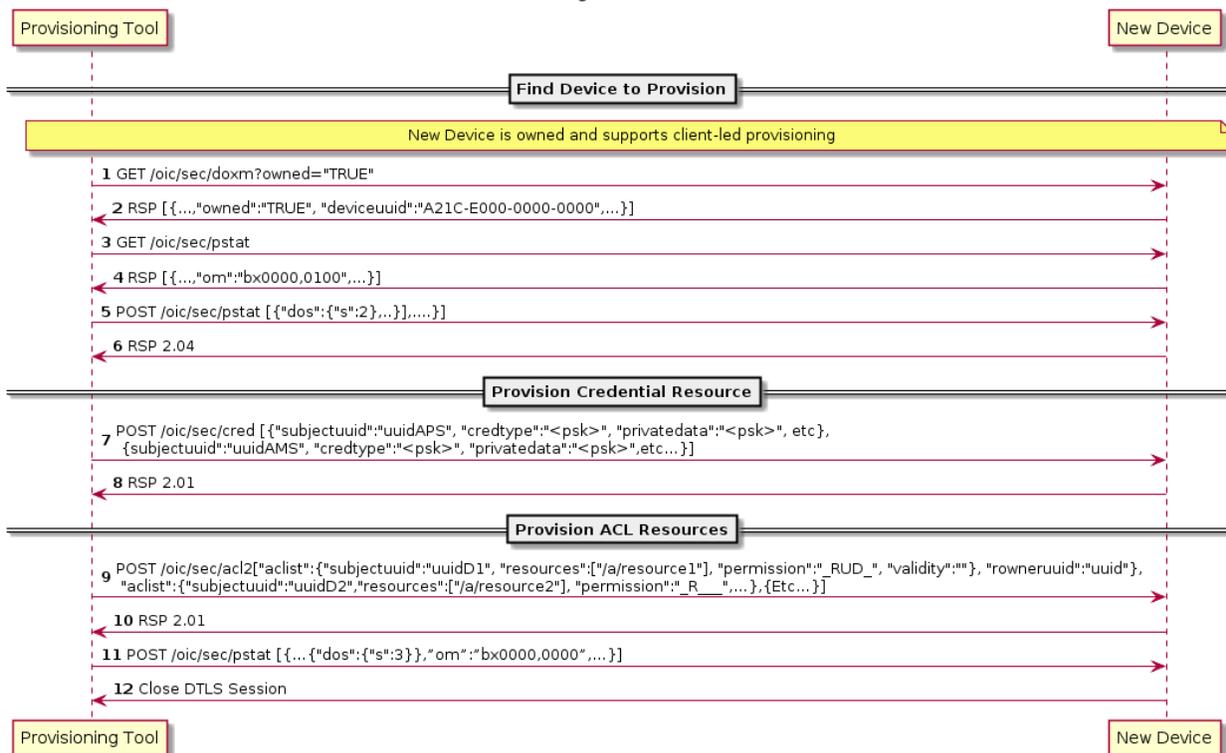
1869 The Device employs a Server-directed or Client-directed provisioning strategy. The "/oic/sec/pstat"  
1870 Resource identifies the provisioning strategy and current provisioning status. The provisioning  
1871 service should determine which provisioning strategy is most appropriate for the OCF Security  
1872 Domain. See 13.8 for additional detail.

#### 1873 **7.4.1.2 Client-directed Provisioning**

1874 Client-directed provisioning relies on a provisioning service that identifies Servers in need of  
1875 provisioning then performs all necessary provisioning duties.

1876 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



1877

1878

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

1879

1880

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

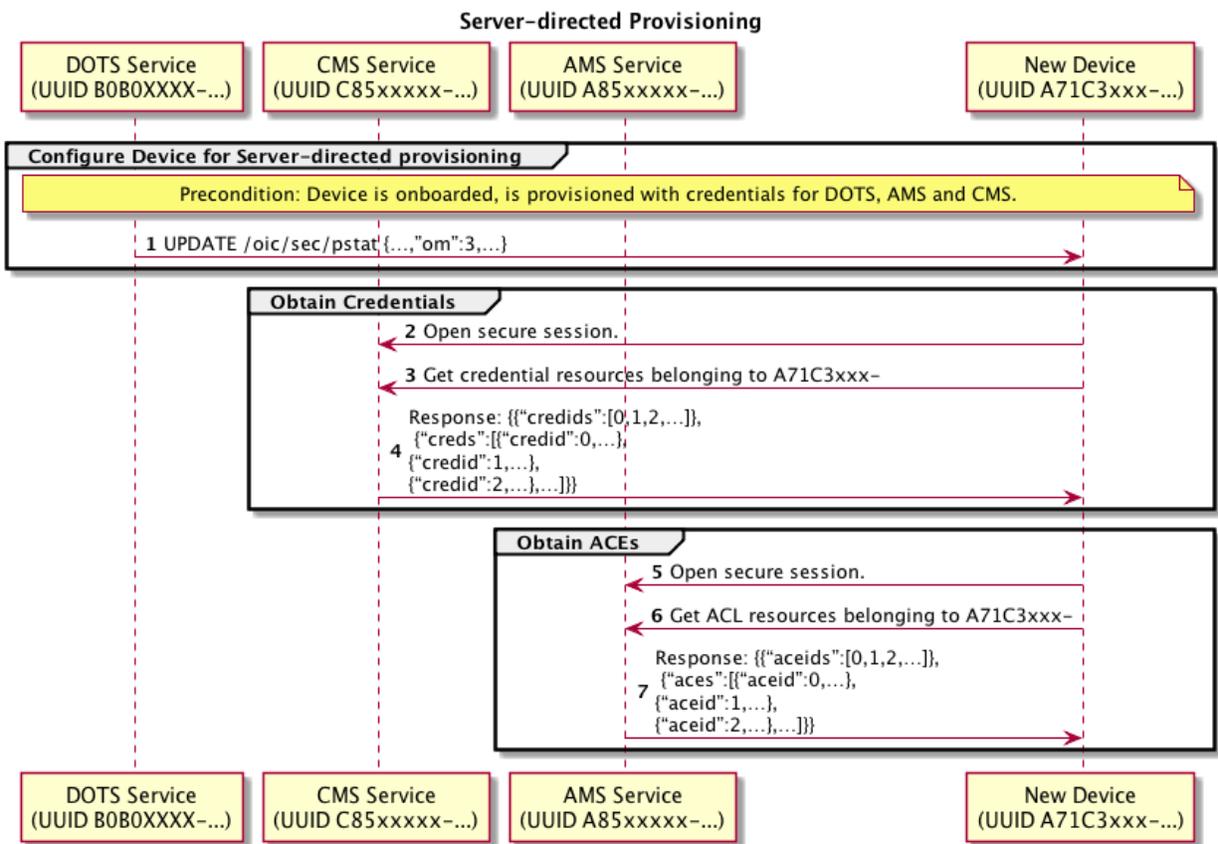
Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The "/oic/sec/doxm" Resource identifies the Device and it's owned status.
3	Provisioning Tool (PT) obtains the new Device's provisioning status found in "/oic/sec/pstat" Resource
4	The "pstat" Resource describes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

1881 **7.4.1.3 Server-directed Provisioning**

1882 Server-directed provisioning relies on the Server (i.e. new Device) for directing much of the  
 1883 provisioning work. As part of the onboarding process the support services used by the Server to  
 1884 seek additional provisioning are provisioned. The new Device uses a self-directed, state-driven  
 1885 approach to analyse current provisioning state, and tries to drive toward target state. This example  
 1886 assumes a single support service is used to provision the new Device.

1887 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1888 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**

1889

1890 **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.

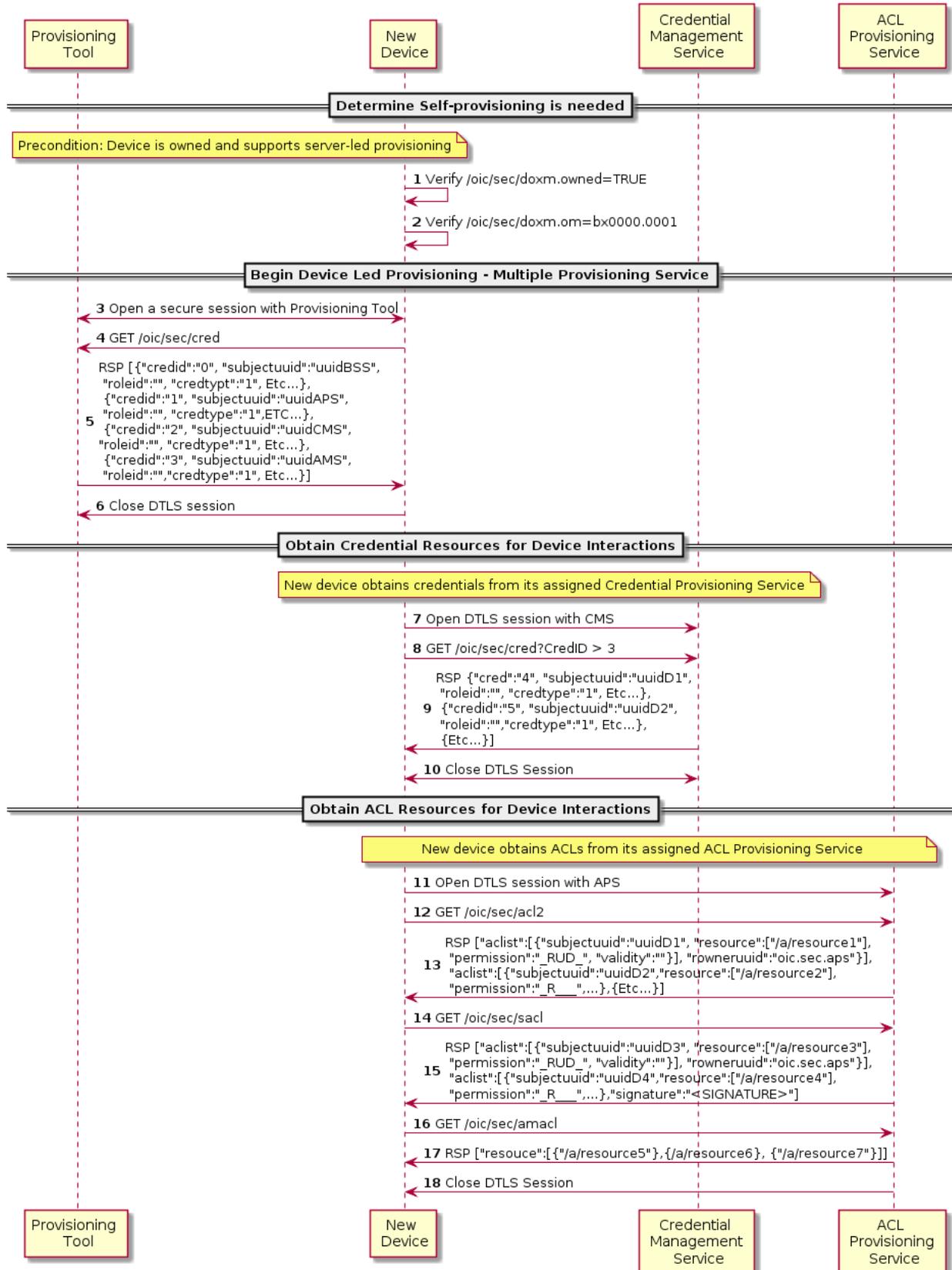
8 – 9	The new Device gets the "/oic/sec/cred" Resources. It contains credentials for the provisioned services and other Devices.
11 – 12	The new Device gets the "/oic/sec/acl2" Resource.
14	The secure session is closed.

1891 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1892 A Server-directed provisioning flow, involving multiple support services distributes the provisioning  
 1893 work across multiple support services. Employing multiple support services is an effective way to  
 1894 distribute provisioning workload or to deploy specialized support. The example in Figure 26  
 1895 demonstrates using a provisioning tool to configure two support services, a CMS and an AMS.  
 1896 Steps for the example are described in Table 14.

DRAFT

### OCF Server Led Provisioning with Multiple Service Providers



1898 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1899 **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.

1900 **7.5 Device Provisioning for OCF Cloud – moved to OCF Cloud Security document**

1901 **8 Device Onboarding State Definitions**

1902 **8.1 Device Onboarding General**

1903 As explained in 5.3, the process of onboarding completes after the ownership of the Device has  
 1904 been transferred and the Device has been provisioned with relevant configuration/services as  
 1905 explained in 5.4. The Figure 27 shows the various states a Device can be in during the Device  
 1906 lifecycle.

1907 The "/pstat.dos.s" Property is RW by the "/oic/sec/pstat" resource owner (e.g. "doxs" service) so  
 1908 that the resource owner can remotely update the Device state. When the Device is in RFNOP or  
 1909 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the  
 1910 Device state is SRESET the Device OC may be the only indication of authorization to access the  
 1911 Device. The Device owner may perform low-level consistency checks and re-provisioning to get  
 1912 the Device suitable for a transition to RFPRO.



1929 **8.2 Device Onboarding-Reset State Definition**

1930 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset  
1931 also defines a state where the Device asset is ready to be transferred to another party.

1932 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces Platform  
1933 reset. All Devices hosted on the same Platform transition their Device states to RESET when the  
1934 Platform reset is asserted.

1935 The following Resources and their specific properties shall have the value as specified:

- 1936 1) The "owned" Property of the "/oic/sec/doxm" Resource shall transition to FALSE.
- 1937 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 1938 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
1939 implemented.
- 1940 4) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
1941 default value.
- 1942 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
1943 default value, if this Property is implemented.
- 1944 6) The "sct" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's default  
1945 value.
- 1946 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
1947 default value.
- 1948 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1949 9) The "dos" Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal  
1950 "RESET" state and dos.p shall equal "FALSE".
- 1951 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the  
1952 manufacturer default value.
- 1953 11) The "sm" (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set  
1954 to the manufacturer default value.
- 1955 12) The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and  
1956 "/oic/sec/cred" Resources shall be nil UUID.
- 1957 13) The "supportedprofiles" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer  
1958 default value.
- 1959 14) The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer  
1960 default value.

1961 **8.3 Device Ready-for-OTM State Definition**

1962 The following Resources and their specific properties shall have the value as specified when the  
1963 Device enters ready for ownership transfer:

- 1964 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to  
1965 TRUE.
- 1966 2) The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
1967 implemented.
- 1968 3) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 1969 4) The "deviceid" Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is  
1970 implemented. The value of the di Property in "/oic/d" is undefined.
- 1971 5) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
1972 default value.

- 1973 6) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1974 7) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFOTM" state  
1975 and dos.p shall equal "FALSE".
- 1976 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

#### 1977 **8.4 Device Ready-for-Provisioning State Definition**

1978 The following Resources and their specific properties shall have the value as specified when the  
1979 Device enters ready for provisioning:

- 1980 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 1981 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 1982 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
1983 set to the value that was determined during RFOTM processing. Also the value of the "di"  
1984 Property in "/oic/d" Resource shall be the same as the "deviceid" Property in the "/oic/sec/doxm"  
1985 Resource.
- 1986 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM  
1987 used during ownership transfer.
- 1988 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1989 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFPRO" state  
1990 and "dos.p" shall equal "FALSE".
- 1991 7) The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner  
1992 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
1993 "rowneruuid" may result in an orphan Resource.
- 1994 8) The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by  
1995 "rowneruuid" and "devowneruuid" Properties.

#### 1996 **8.5 Device Ready-for-Normal-Operation State Definition**

1997 The following Resources and their specific properties shall have the value as specified when the  
1998 Device enters ready for normal operation:

- 1999 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2000 2) The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2001 3) The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
2002 set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d" shall  
2003 be the same as the deviceuuid.
- 2004 4) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM  
2005 used during ownership transfer.
- 2006 5) The "isop" Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once  
2007 transition to RFNOP is otherwise complete.
- 2008 6) The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFNOP" state  
2009 and dos.p shall equal "FALSE".
- 2010 7) The "rowneruuid" Property of every installed Resource shall be set to a valid resource owner  
2011 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2012 "rowneruuid" results in an orphan Resource.
- 2013 8) The "/oic/sec/cred" Resource shall contain credentials for each service referenced by  
2014 "rowneruuid" and "devowneruuid" Properties.

#### 2015 **8.6 Device Soft Reset State Definition**

2016 The soft reset state is defined (e.g. "/pstat.dos.s" = SRESET) where entrance into this state means  
2017 the Device is not operational but remains owned by the current owner. The Device may exit



2029 When in SRESET, the following Resources and their specific Properties shall have the values as  
2030 specified.

- 2031 1) The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2032 2) The "devowneruid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2033 3) The "devowner" Property of the "/oic/sec/doxm" Resource shall be non-null, if this Property is  
2034 implemented.
- 2035 4) The "deviceuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2036 5) The "deviceid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2037 6) The "sct" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 2038 7) The "oxmsel" Property of the "/oic/sec/doxm" Resource shall retains its value.
- 2039 8) The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2040 9) The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- 2041 10) The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-directed  
2042 mode".
- 2043 11) The "sm" (supported operational modes) Property of "/oic/sec/pstat" Resource may be updated  
2044 by the Device owner (aka DOTS).
- 2045 12) The "rowneruid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", "/oic/sec/amacl",  
2046 "/oic/sec/sacl", and "/oic/sec/cred" Resources may be reset by the Device owner (aka DOTS)  
2047 and re-provisioned.

2048

## 2049 **9 Security Credential Management**

### 2050 **9.1 Preamble**

2051 This clause provides an overview of the credential types in OCF, along with details of credential  
2052 use, provisioning and ongoing management.

### 2053 **9.2 Credential Lifecycle**

#### 2054 **9.2.1 Credential Lifecycle General**

2055 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)  
2056 issuance and (5) revocation.

#### 2057 **9.2.2 Creation**

2058 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is  
2059 authorized by matching the rowneruuid Property of the "/oic/sec/cred" resource to the DeviceID of  
2060 the credential the CMS used to establish the secure connection.

2061 Credential Resources created using a CMS may involve specialized credential issuance protocols  
2062 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate  
2063 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of a  
2064 provisioning action by a DOTS, CMS or AMS.

#### 2065 **9.2.3 Deletion**

2066 The CMS should delete known compromised credential Resources. The Device (e.g. the Device  
2067 where the credential Resource is hosted) should delete credential Resources that have expired.

2068 An expired credential Resource may be deleted to manage memory and storage space.

2069 Deletion in OCF key management is equivalent to credential suspension.

#### 2070 **9.2.4 Refresh**

2071 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2072 The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may be  
2073 applied when a credential is about to expire or is about to exceed a maximum threshold for bytes  
2074 encrypted.

2075 A credential refresh method specifies the options available when performing key refresh. The  
2076 Period Property informs when the credential should expire. The Device may proactively obtain a  
2077 new credential using a credential refresh method using current unexpired credentials to refresh the  
2078 existing credential. If the Device does not have an internal time source, the current time should be  
2079 obtained from a CMS at regular intervals.

2080 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the CMS  
2081 credentials to the Device. If the onboarding established credentials are allowed to expire the DOTS  
2082 shall re-onboard the Device to re-apply device owner transfer steps.

2083 All Devices shall support at least one credential refresh method.

#### 2084 **9.2.5 Revocation**

2085 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where the  
2086 revocation method involves provisioning of a revocation object that identifies a credential that is to  
2087 be revoked prior to its normal expiration period, a credential Resource is created containing the  
2088 revocation information that supersedes the originally issued credential. The revocation object

2089 expiration should match that of the revoked credential so that the revocation object is cleaned up  
2090 upon expiry.

2091 It is conceptually reasonable to consider revocation applying to a credential or to a Device. Device  
2092 revocation asserts all credentials associated with the revoked Device should be considered for  
2093 revocation. Device revocation is necessary when a Device is lost, stolen or compromised. Deletion  
2094 of credentials on a revoked Device might not be possible or reliable.

## 2095 **9.3 Credential Types**

### 2096 **9.3.1 Preamble**

2097 The "/oic/sec/cred" Resource maintains a credential type Property that supports several  
2098 cryptographic keys and other information used for authentication and data protection. The  
2099 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric  
2100 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.  
2101 PIN/password).

### 2102 **9.3.2 Pair-wise Symmetric Key Credentials**

2103 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The  
2104 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2105 Pair-wise keys could be established through ad-hoc key agreement protocols.

2106 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2107 The PublicData Property may contain a token encrypted to the peer Device containing the pair-  
2108 wise key.

2109 The OptionalData Property may contain revocation status.

2110 The Device implementer should apply hardened key storage techniques that ensure the  
2111 PrivateData remains private.

2112 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2113 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2114 unauthorized modifications.

### 2115 **9.3.3 Group Symmetric Key Credentials**

2116 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are  
2117 used for efficient sharing of data among group participants.

2118 Group keys do not provide authentication of Devices but only establish membership in a group.

2119 The CMS shall provision group symmetric key credentials to the group members. The CMS  
2120 maintains the group memberships.

2121 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2122 The PublicData Property may contain the group name.

2123 The OptionalData Property may contain revocation status.

2124 The Device implementer should apply hardened key storage techniques that ensure the  
2125 PrivateData remains private.

2126 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2127 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2128 unauthorized modifications.

### 2129 **9.3.4 Asymmetric Authentication Key Credentials**

#### 2130 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2131 Asymmetric authentication key credentials contain either a public and private key pair or only a  
2132 public key. The private key is used to sign Device authentication challenges. The public key is used  
2133 to verify a device authentication challenge-response.

2134 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2135 The PublicData Property contains the public key.

2136 The OptionalData Property may contain revocation status.

2137 The Device implementer should apply hardened key storage techniques that ensure the  
2138 PrivateData remains private.

2139 Devices should generate asymmetric authentication key pairs internally to ensure the private key  
2140 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material  
2141 between Devices.

2142 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2143 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2144 unauthorized modifications.

#### 2145 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2146 Devices should employ industry-standard high-assurance techniques when allowing off-device key  
2147 pair creation and provisioning. Use of such key pairs should be minimized, particularly if the key  
2148 pair is immutable and cannot be changed or replaced after provisioning.

2149 When used as part of onboarding, these key pairs can be used to prove the Device possesses the  
2150 manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept  
2151 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the  
2152 Device, and then provisions new OCF Security Domain credentials for use.

### 2153 **9.3.5 Asymmetric Key Encryption Key Credentials**

2154 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when  
2155 distributing or storing the key.

2156 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2157 The PublicData Property contains the public key.

2158 The OptionalData Property may contain revocation status.

2159 The Device implementer should apply hardened key storage techniques that ensure the  
2160 PrivateData remains private.

2161 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2162 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2163 unauthorized modifications.

### 2164 **9.3.6 Certificate Credentials**

2165 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a CMS  
2166 or an external certificate authority (CA).

2167 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2168 The issued certificate is stored with the asymmetric key credential Resource.

2169 Other objects useful in managing certificate lifecycle such as certificate revocation status are  
2170 associated with the credential Resource.

2171 Either an asymmetric key credential Resource or a self-signed certificate credential is used to  
2172 terminate a path validation.

2173 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2174 The PublicData Property contains the issued certificate.

2175 The OptionalData Property may contain revocation status.

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

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

### 2181 **9.3.7 Password Credentials**

2182 Shared secret credentials are used to maintain a PIN or password that authorizes Device access  
2183 to a foreign system or Device that doesn't support any other OCF credential types.

2184 The PrivateData Property in the "/oic/sec/cred" Resource contains the PIN, password and other  
2185 values useful for changing and verifying the password.

2186 The PublicData Property may contain the user or account name if applicable.

2187 The OptionalData Property may contain revocation status.

2188 The Device implementer should apply hardened key storage techniques that ensure the  
2189 PrivateData remains private.

2190 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2191 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2192 unauthorized modifications.

## 2193 **9.4 Certificate Based Key Management**

### 2194 **9.4.1 Overview**

2195 To achieve authentication and transport security during communications in OCF Security Domain,  
2196 certificates containing public keys of communicating parties and private keys can be used.

2197 The certificate and private key may be issued by a local or remote certificate authority (CA). For  
2198 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of identity.  
2199 In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to validate  
2200 proof of identity and validity.

2201 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,  
2202 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in X.509  
2203 are not supported so that the format intends to meet the constrained Device's requirement.

2204 As for the certificate and CRL management in the Server, the process of storing, retrieving and  
2205 parsing Resources of the certificates and CRL will be performed at the security resource manager  
2206 layer; the relevant interfaces may be exposed to the upper layer.

2207 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of  
2208 certificates and CRL will be stored and managed in SVR database.

2209 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a CMS  
2210 to a Device when it is newly onboarded. The issuing CMS should process certificate revocations  
2211 for certificates it issues. If a certificate private key is compromised, the CMS should revoke the  
2212 certificate. If CRLs are used by a Device, the CMS should regularly (for example; every 3 months)  
2213 update the "/oic/sec/crl" resource for the Devices it manages.

## 2214 **9.4.2 X.509 Digital Certificate Profiles**

### 2215 **9.4.2.1 Digital Certificate Profile General**

2216 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in  
2217 IETF RFC 5280.

2218 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications for  
2219 those communities wishing to make use of X.509 technology. The X.509 v3 certificate format is  
2220 described in detail, with additional information regarding the format and semantics of OCF specific  
2221 extension(s). The supported standard certificate extensions are also listed.

2222 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this  
2223 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are  
2224 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,  
2225 compliant entities shall ignore their contents.

2226 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as  
2227 defined in ISO/IEC 8825-1 to encode certificates.

2228 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public  
2229 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs  
2230 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and  
2231 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2232 The following clauses specify the supported standard and custom extensions for the OCF  
2233 certificates profile.

### 2234 **9.4.2.2 Certificate Profile and Fields**

#### 2235 **9.4.2.2.1 Root CA Certificate Profile**

2236 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2237 **Table 15 – 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)

2238 Table 17 describes X.509 v3 extensions required for Root CA Certificates.

2239 **Table 16 - 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)

2240 **9.4.2.2.2 Intermediate CA Certificate Profile**

2241 Table 18 describes X.509 v1 fields required for Intermediate CA Certificates.

2242 **Table 17 - 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)

2243 Table 19 describes X.509 v3 extensions required for Intermediate CA Certificates.

**Table 18 – 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

#### 2245 9.4.2.2.3 End-Entity Black Certificate Profile

2246 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security  
2247 profile.

2248

**Table 19 – 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)

2249 Table 21 describes X.509 v3 extensions required for End-Entity Certificates.

**Table 20 – X.509 v3 extensions for End-Entity Certificates**

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

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

2251 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2252 The OCF Compliance Extension defines required parameters to correctly identify the type of Device,  
2253 its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2254 The extension carries an "ocfVersion" field which provides the specific base version of the OCF  
2255 documents the device implements. The "ocfVersion" field shall contain a sequence of three integers  
2256 ("major", "minor", and "build"). For example, if an entity is certified to be compliant with OCF

2257 specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be set to  
2258 "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote  
2259 compliance to a specified base version of the OCF documents.

2260 The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more  
2261 supported Security Profiles associated with the certificate in string form (UTF-8). All Security  
2262 Profiles associated with the certificate should be identified by this field.

2263 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".  
2264 The fields carry human-readable descriptions of the Device's name and manufacturer, respectively.

2265 The ASN.1 definition of the OCFCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined as  
2266 follows:

```
2267 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2268                               private(4) enterprise(1) OCF(51414) }
2269
2270 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2271
2272 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2273
2274 ocfVersion ::= SEQUENCE {
2275     major    INTEGER,
2276             --Major version number
2277     minor    INTEGER,
2278             --Minor version number
2279     build    INTEGER,
2280             --Build/Micro version number
2281 }
2282
2283 ocfCompliance ::= SEQUENCE {
2284     version          ocfVersion,
2285                     --Device/OCF version
2286     securityProfile  SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2287                     --Sequence of OCF Security Profile OID strings
2288                     --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2289
2290     deviceName       UTF8String,
2291                     --Name of the device
2292     deviceManufacturer UTF8String,
2293                     --Human-Readable Manufacturer
2294                     --of the device
2295 }
```

#### 2295 **9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension**

2296 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for devices  
2297 to signal to the network the access and network functionality they require to properly function.  
2298 Access controls can be more easily achieved and deployed at scale when the MUD extension is  
2299 used. The current draft of the MUD v3 extension at this time of writing is:

2300 <https://tools.ietf.org/html/rfc8520#section-11>

2301 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2302 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2303                       internet(1) security(5) mechanisms(5) pkix(7)
2304                       id-mod(0) id-mod-mudURLExtn2016(88) }
2305
2306 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2307 -- EXPORTS ALL --
2308 IMPORTS
2309     EXTENSION
```

```

2310 FROM PKIX-CommonTypes-2009
2311     { iso(1) identified-organization(3) dod(6) internet(1)
2312       security(5) mechanisms(5) pkix(7) id-mod(0)
2313       id-mod-pkixCommon-02(57) }
2314 id-pe
2315 FROM PKIX1Explicit-2009
2316     { iso(1) identified-organization(3) dod(6) internet(1)
2317       security(5) mechanisms(5) pkix(7) id-mod(0)
2318       id-mod-pkix1-explicit-02(51) } ;
2319 MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2320 ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2321                               IDENTIFIED BY id-pe-mud-url }
2322
2323 id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2324
2325 MUDURLSyntax ::= IA5String
2326
2327 END

```

#### 2328 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2329 The OCF Security Claims Extension defines a list of OIDs representing security claims that the  
2330 manufacturer/integrator is making as to the security posture of the device above those required by  
2331 the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2332 The purpose of this extension is to allow for programmatic evaluation of assertions made about  
2333 security to enable some platforms/policies/administrators to better understand what is being  
2334 onboarded or challenged.

2335 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is defined  
2336 as follows:

```

2337 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2338                               private(4) enterprise(1) OCF(51414) }
2339
2340 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2341
2342 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2343
2344 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2345 --Device claims that the boot process follows a procedure trusted
2346 --by the firmware and the BIOS
2347
2348 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2349 --Device claims that credentials are stored in a specialized hardware
2350 --protection environment such as a Trusted Platform Module (TPM) or
2351 --similar mechanism.
2352
2353 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2354
2355 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

#### 2356 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2357 The OCF Certified Product List Extension defines required parameters to utilize the OCF  
2358 Compliance Management System Certified Product List (OCMS-CPL). This clause is only  
2359 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to verify  
2360 the compliance level of a device.

2361 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model and  
2362 Version.

2363 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique  
2364 PEN established in the IANA PEN list located at: [https://www.iana.org/enterprise-](https://www.iana.org/assignments/enterprise-)  
2365 numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as  
2366 reported during OCF Certification.

2367 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field  
2368 found in end-products shall be the same information as reported during OCF Certification.

2369 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found  
2370 in end-products shall be the same information as reported during OCF Certification.

2371 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is defined  
2372 as follows:

```
2373 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2374                               private(4) enterprise(1) OCF(51414) }
2375
2376 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2377
2378   id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2379
2380     cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2381     cpl-at-model   ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2382     cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2383
2384
2385   ocfCPLAttributes ::= SEQUENCE {
2386     cpl-at-IANAPen      UTF8String,
2387                       --Manufacturer's registered IANA Private Enterprise Number
2388     cpl-at-model       UTF8String,
2389                       --Device OCF Security Profile
2390     cpl-at-version     UTF8String
2391                       --Name of the device
2392   }
```

### 2393 9.4.2.3 Supported Certificate Extensions

2394 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the  
2395 clause number from that RFC to include it by reference. Each extension is summarized here, and  
2396 any modifications to the RFC definition are listed. Devices MUST implement and understand the  
2397 extensions listed here; other extensions from the RFC are not included in this document and  
2398 therefore are not required. 10.4 describes what Devices must implement when validating certificate  
2399 chains, including processing of extensions, and actions to take when certain extensions are absent.

#### 2400 – Authority Key Identifier (4.2.1.1)

2401 The Authority Key Identifier (AKI) extension provides a means of identifying the public key  
2402 corresponding to the private key used to sign a certificate. This document makes the following  
2403 modifications to the referenced definition of this extension:

2404 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier  
2405 sequence are not permitted; only keyIdentifier is allowed. This results in the following grammar  
2406 definition:

```
2407 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2408
2409 AuthorityKeyIdentifier ::= SEQUENCE {
2410   keyIdentifier          [0] KeyIdentifier
2411 }
2412
2413 KeyIdentifier ::= OCTET STRING
```

#### 2413 – Subject Key Identifier (4.2.1.2)

2414 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that  
2415 contain a particular public key.

2416 This document makes the following modification to the referenced definition of this extension:

2417 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's  
2418 SubjectPublicKeyInfo field or a method that generates unique values. This document  
2419 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey  
2420 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains must  
2421 not assume any particular method of computing key identifiers, however, and must only base  
2422 matching AKI's and SKI's in certification path constructions on key identifiers seen in certificates.

2423 – Subject Alternative Name

2424 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role  
2425 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and  
2426 interpreted as described below. When no EKU is present, or has another value, the  
2427 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to  
2428 encode one or more Role ID values in role certificates, binding the roles to the subject public  
2429 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2430 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2431
2432 SubjectAltName ::= GeneralNames
2433
2434 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2435
2436 GeneralName ::= CHOICE {
2437     otherName                [0]     OtherName,
2438     rfc5322Name              [1]     IA5String,
2439     dNSName                  [2]     IA5String,
2440     x400Address              [3]     OAddress,
2441     directoryName            [4]     Name,
2442     ediPartyName             [5]     EDIPartyName,
2443     uniformResourceIdentifier [6]     IA5String,
2444     iPAddress                [7]     OCTET STRING,
2445     registeredID             [8]     OBJECT IDENTIFIER }
2446
2447     EDIPartyName ::= SEQUENCE {
2448         nameAssigner [0]     DirectoryString OPTIONAL,
2449         partyName    [1]     DirectoryString }
2450
```

2451 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a  
2452 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name  
2453 shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational  
2454 Unit) components. The OU component, if present, shall specify the authority that defined the  
2455 semantics of the role. If the OU component is absent, the certificate issuer has defined the role.  
2456 The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may  
2457 be present, but shall not be interpreted as roles. Therefore, if the certificate issuer includes  
2458 non-role names in the subjectAltName extension, the extension should not be marked critical.

2459 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted  
2460 character set [0-9a-z-A-z '()+,./:=?].

2461 – Key Usage (4.2.1.3)

2462 The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing)  
2463 of the key contained in the certificate. The usage restriction might be employed when a key that  
2464 could be used for more than one operation is to be restricted.

2465 This document does not modify the referenced definition of this extension.

2466 – Basic Constraints (4.2.1.9)

2467 The basic constraints extension identifies whether the subject of the certificate is a CA and the  
2468 maximum depth of valid certification paths that include this certificate. Without this extension,  
2469 a certificate cannot be an issuer of other certificates.

2470 This document does not modify the referenced definition of this extension.

2471 – Extended Key Usage (4.2.1.12)

2472  
2473 Extended Key Usage describes allowed purposes for which the certified public key may can be  
2474 used. When a Device receives a certificate, it determines the purpose based on the context of  
2475 the interaction in which the certificate is presented, and verifies the certificate can be used for  
2476 that purpose.

2477 This document makes the following modifications to the referenced definition of this extension:  
2478 CAs SHOULD mark this extension as critical.

2479 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).

2480

2481 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2482 – Identity certificate 1.3.6.1.4.1.44924.1.6

2483 – Role certificate 1.3.6.1.4.1.44924.1.7

#### 2484 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2485 See 9.4.3.5 for details.

#### 2486 **9.4.2.5 Encoding of Certificate**

2487 See 9.4.2 for details.

### 2488 **9.4.3 Certificate Revocation List (CRL) Profile**

#### 2489 **9.4.3.1 CRL General**

2490 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use within  
2491 OCF applications for those communities wishing to support revocation features in their PKIs.

2492 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in  
2493 IETF RFC 5280 – Clause 5.1

#### 2494 **9.4.3.2 CRL Profile and Fields**

2495 This clause intentionally left empty.

#### 2496 **9.4.3.3 Encoding of CRL**

2497 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]  
2498 should be used to encode CRL.

#### 2499 **9.4.3.4 CRLs Supported Standard Extensions**

2500 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55] provide  
2501 methods for associating additional attributes with CRLs. The following list of X.509 extensions  
2502 should be supported in this certificate profile:

2503 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides  
2504 a means of identifying the public key corresponding to the private key used to sign a CRL.  
2505 Conforming CRL issuers should use the key identifier method, and shall include this extension  
2506 in all CRLs issued

2507 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that  
2508 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer

2509 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for  
2510 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]  
2511 [X9.55]. Although this document does not provide any recommendation about the use of specific  
2512 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not marked  
2513 critical.

#### 2514 **9.4.3.5 Encryption Ciphers and TLS support**

2515 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support  
2516 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 cipher suite as defined in IETF RFC 7251 and may  
2517 support additional ciphers as defined in the TLS v1.2 specifications.

#### 2518 **9.4.4 Resource Model**

2519 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated  
2520 with a separate crl Resource that is defined for maintaining the revocation list.

2521 The cred Resource contains the certificate information pertaining to the Device. The PublicData  
2522 Property holds the device certificate and CA certificate chain. PrivateData Property holds the  
2523 Device private key paired to the certificate. (See 13.3 for additional detail regarding the  
2524 "/oic/sec/cred" Resource).

2525 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained  
2526 through the CMS. The Device must consider revoked certificates as part of certificate path  
2527 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain a  
2528 full list, the Device must query the CMS for current revocation status. (See 13.4 for additional detail  
2529 regarding the "/oic/sec/crl" Resource).

#### 2530 **9.4.5 Certificate Provisioning**

2531 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have  
2532 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)  
2533 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate shall  
2534 have the format described in 9.4.2.

2535 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,  
2536 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer them  
2537 to the Device including its CA certificate chain. Optionally, the CMS may also transfer one or more  
2538 role certificates, which shall have the format described in clause 9.4.2. . The subjectPublicKey of  
2539 each role certificate shall match the subjectPublicKey in the Device certificate.

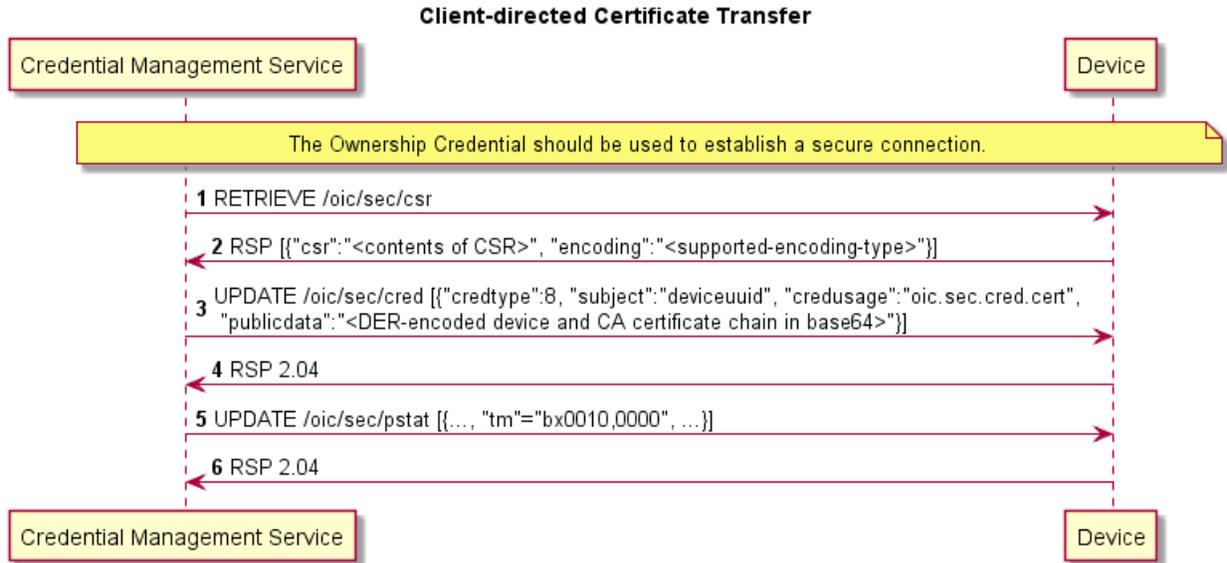
2540 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in  
2541 IETF RFC 2986, and is included here by reference.

2542 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2543 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device  
2544 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.  
2545 The Device determines the public key to present; this may be an already-provisioned key it has  
2546 selected for use with authentication, or if none is present, it may generate a new key pair  
2547 internally and provide the public part. The key pair shall be compatible with the allowed  
2548 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF  
2549 authentication.

2550 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on  
2551 its own, then it is not capable of using certificates. The Device shall advertise this fact both by  
2552 setting the 0x8 bit position in the sct Property of "/oic/sec/doxm" to 0, and return an error that  
2553 the "/oic/sec/csr" resource does not exist.

2554 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using the  
 2555 same credid, to maintain the association with the private key. The credential type  
 2556 ("oic.sec.cred") used to transfer certificates in Figure 29 is also used to transfer role certificates,  
 2557 by including multiple credentials in the POST from CMS to Device. Identity certificates shall be  
 2558 stored with the credusage Property set to "oic.sec.cred.cert" and role certificates shall be stored  
 2559 with the credusage Property set to "oic.sec.cred.rolecert".



2560

2561 **Figure 29 – Client-directed Certificate Transfer**

2562 **9.4.6 CRL Provisioning**

2563 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the function  
 2564 to register revocation certificates, to sign CRL and to transfer it to Devices.

2565 The CMS sends the CRL to the Device.

2566 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2567 – change of issuer name
- 2568 – change of association between Devices and CA
- 2569 – certificate compromise
- 2570 – suspected compromise of the corresponding private key

2571 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In some  
 2572 special cases, Devices may request CRL to a given CMS.

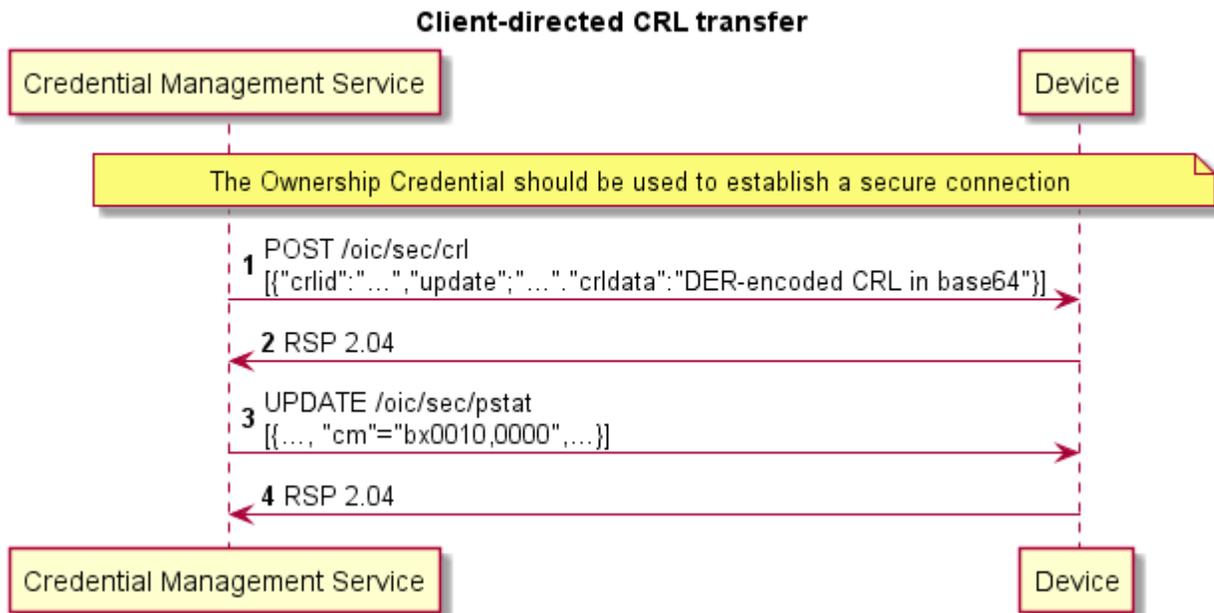
2573 There are two options to update and deliver CRL;

- 2574 – CMS pushes CRL to each Device
- 2575 – each Device periodically requests to update CRL

2576 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

- 2577 1) The CMS may retrieve the CRL Resource Property.
- 2578 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.

2579

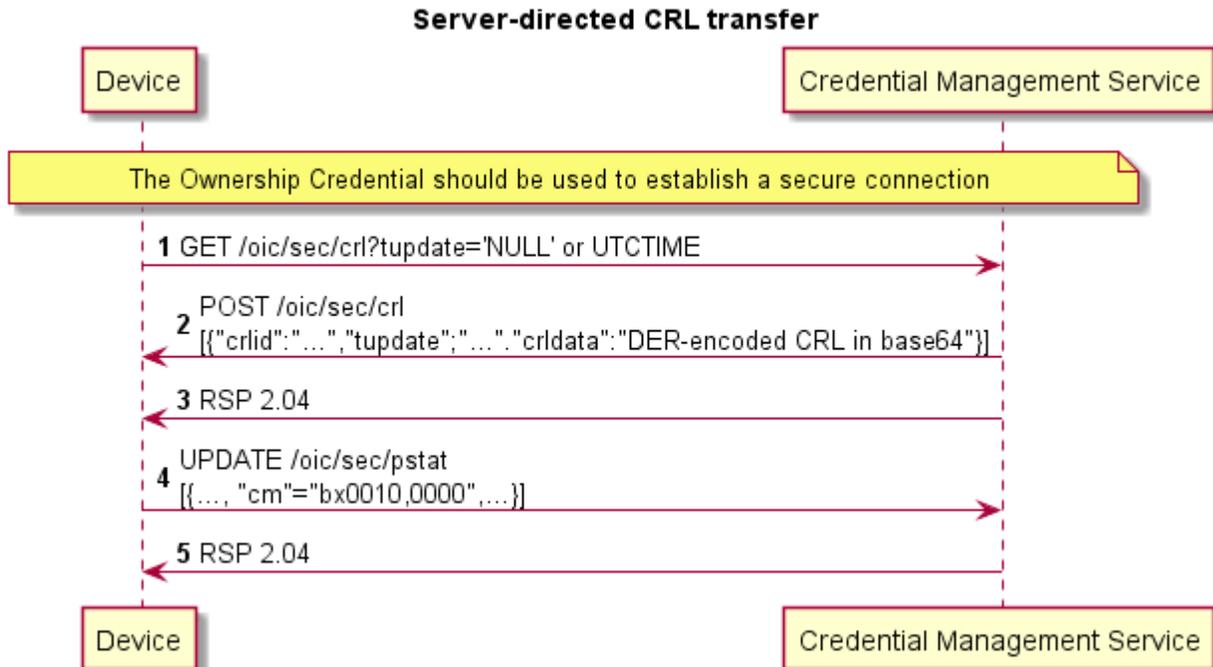


**Figure 30 – Client-directed CRL Transfer**

2581

2582 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.

- 2583 1) The Device retrieves the CRL Resource Property "update" to the CMS.  
 2584 2) If the CMS recognizes the updated CRL information after the designated "update" time, it may  
 2585 transfer its CRL to the Device.



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

2586  
 2587  
 2588

2589 **10 Device Authentication**

2590 **10.1 Device Authentication General**

2591 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the  
2592 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or  
2593 more roles that the server can use in access control decisions. Roles may be asserted when the  
2594 Device authentication is done with certificates.

2595 **10.2 Device Authentication with Symmetric Key Credentials**

2596 When using symmetric keys to authenticate, the Server Device shall include the  
2597 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client shall  
2598 validate that it has a credential with the Subject ID set to the Server's Device ID, and a credential  
2599 type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error or other  
2600 suitable error.

2601 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that  
2602 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a  
2603 credential with the matching Subject ID and type. If it does not, the Server shall respond with an  
2604 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS  
2605 protocol, and both Client and Server shall compute the resulting premaster secret.

2606 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2607 When using raw asymmetric keys to authenticate, the Client and the Server shall include a suitable  
2608 public key from a credential that is bound to their Device. Each Device shall verify that the provided  
2609 public key matches the `PublicData` field of a credential they have, and use the corresponding  
2610 Subject ID of the credential to identify the peer Device.

2611 **10.4 Device Authentication with Certificates**

2612 **10.4.1 Device Authentication with Certificates General**

2613 When using certificates to authenticate, the Client and Server shall each include their certificate  
2614 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.  
2615 Each Device shall validate the certificate chain presented by the peer Device. Each certificate  
2616 signature shall be verified until a public key is found within the `"/oic/sec/cred"` Resource with the  
2617 `"oic.sec.cred.trustca"` credusage. Credential Resource found in `"/oic/sec/cred"` is used to terminate  
2618 certificate path validation. Also, the validity period and revocation status should be checked for all  
2619 above certificates, but at this time a failure to obtain a certificate's revocation status (CRL or OCSP  
2620 response) MAY continue to allow the use of the certificate if all other verification checks succeed.

2621 If available, revocation information should be used to verify the revocation status of the certificate.  
2622 The URL referencing the revocation information should be retrieved from the certificate (via the  
2623 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used  
2624 to gather relevant revocation information like CRLs or OCSP responses.

2625 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2626 Devices must follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2627 particular:

- 2628 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is  
2629 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate chain  
2630 MUST be rejected. If the `pathLenConstraint` field is present, Devices will confirm the number of  
2631 certificates between this certificate and the End-Entity certificate is less than or equal to  
2632 `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity certificate can  
2633 be issued by this certificate. If the `pathLenConstraint` field is absent, there is no limit to the  
2634 chain length.

- 2635 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is present,  
2636 and that the keyCertSign bit is asserted.
- 2637 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.  
2638 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt  
2639 validation with the public keys of possible issuer certificates whose subject name equals the  
2640 issuer name of this certificate.
- 2641 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage (EKU)  
2642 suitable to the purpose for which it is being presented. An End-Entity certificate which contains  
2643 no ECU extension is not valid for any purpose and must be rejected. Any certificate which  
2644 contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other valid  
2645 EKUs are also present.
- 2646 – Devices MUST verify "transitive ECU" for certificate chains. Issuer certificates (any certificate  
2647 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the certificate  
2648 chain is being presented. An issuer certificate is valid for a purpose if it contains an ECU  
2649 extension and the ECU OID for that purpose is listed in the extension, OR it does not have an  
2650 ECU extension. An issuer certificate SHOULD contain an ECU extension and a complete list of  
2651 EKUs for the purposes for which it is authorized to issue certificates. An issuer certificate  
2652 without an ECU extension is valid for all purposes; this differs from End-Entity certificates  
2653 without an ECU extension.
- 2654 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2655 If the Device does not recognize an extension, it must examine the `critical` field. If the field is  
2656 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the  
2657 certificate as if the extension were absent and proceed accordingly. This applies to all certificates  
2658 in a chain.

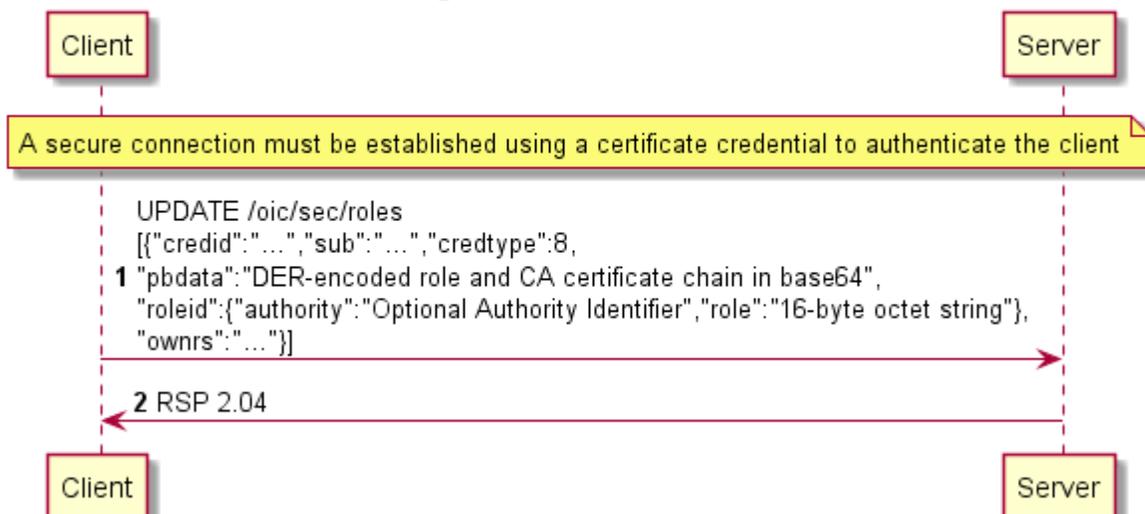
2659 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

#### 2660 **10.4.2 Role Assertion with Certificates**

2661 This clause describes role assertion by a client to a server using a certificate role credential. If a  
2662 server does not support the certificate credential type, clients should not attempt to assert roles  
2663 with certificates.

2664 Following authentication with a certificate, a client may assert one or more roles by updating the  
2665 server's roles resource with the role certificates it wants to use. The role credentials must be  
2666 certificate credentials and shall include a certificate chain. The server shall validate each certificate  
2667 chain as specified in clause 10.3. Additionally, the public key in the End-Entity certificate used for  
2668 Device authentication must be identical to the public key in all role (End-Entity) certificates. Also,  
2669 the subject distinguished name in the End-Entity authentication and role certificates must match.  
2670 The roles asserted are encoded in the subjectAltName extension in the certificate. The  
2671 subjectAltName field can have multiple values, allowing a single certificate to encode multiple roles  
2672 that apply to the client. The server shall also check that the ECU extension of the role certificate(s)  
2673 contains the value 1.3.6.1.4.1.44924.1.7 (see clause 9.4.2.2) indicating the certificate may be used  
2674 to assert roles. Figure 32 describes how a client Device asserts roles to a server.

## Asserting Certificate Role Credentials



2675

2676

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

2677 Additional comments for Figure 32

- 2678 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If  
2679 the server does not support certificate credentials, it should return "501 Not Implemented"
- 2680 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration shall  
2681 not exceed the validity period of the role certificate. When fresh CRL information is obtained,  
2682 the certificates in "/oic/sec/roles" should be checked, and the role removed if the certificate is  
2683 revoked or expired.
- 2684 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role  
2685 by a client. It is recommended that servers use the validity period of the certificate as a duration,  
2686 effectively allowing the CMS to decide the duration.
- 2687 4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see  
2688 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not  
2689 be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the  
2690 value in the certificate shall be used for validation. For example, if the Period field is set in the  
2691 credential, the server shall treat the validity period in the certificate as authoritative. Similar for  
2692 the roleid data (authority, role).
- 2693 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)
- 2694 6) Clients may GET the "/oic/sec/roles" resource to determine the roles that have been previously  
2695 asserted. An array of credential objects shall be returned. If there are no valid certificates  
2696 corresponding to the currently connected and authenticated Client's identity, then an empty  
2697 array (i.e. []) shall be returned.

### 2698 10.4.3 OCF PKI Roots

2699 This clause intentionally left empty.

### 2700 10.4.4 PKI Trust Store

2701 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store the  
2702 OCF Root CA certificates in the "oic/sec/cred" resource and SHOULD physically store this resource  
2703 in a hardened memory location where the certificates cannot be tampered with.

2704 **10.4.5 Path Validation and extension processing**

2705 Devices SHALL follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2706 addition, the following are best practices and SHALL be adhered to by any OCF-compliant  
2707 application handling digital certificates

2708 – Validity Period checking

2709 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and  
2710 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,  
2711 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2712 – Revocation checking

2713 Relying applications SHOULD check the revocation status for all certificates, but at this time,  
2714 an application MAY continue to allow the use of the certificate upon a failure to obtain a  
2715 certificate's revocation status (CRL or OCSP response), if all other verification checks succeed.

2716 – basicConstraints

2717 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify that  
2718 the basicConstraints extension is present, flagged critical, and that the cA boolean value in the  
2719 extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2720 If the pathLenConstraint field is present, Devices will confirm the number of certificates between  
2721 this certificate and the End-Entity certificate is less than or equal to pathLenConstraint. In  
2722 particular, if pathLenConstraint is zero, only an End-Entity certificate can be issued by this  
2723 certificate. If the pathLenConstraint field is absent, there is no limit to the chain length.

2724 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged  
2725 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a  
2726 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a  
2727 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a  
2728 null value.

2729 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-compliant  
2730 Root CA certificates SHALL NOT contain a pathLenConstraint. This allows additional tiers of  
2731 Intermediate CAs to be implemented in the future without changing the Root CA trust anchors,  
2732 should such a requirement emerge.

2733 – keyUsage

2734 For all certificates, Devices shall verify that the key usage extension is present and flagged  
2735 critical.

2736 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits SHALL  
2737 be asserted.

2738 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be  
2739 asserted.

2740 – extendedKeyUsage:

2741 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be  
2742 rejected.

2743 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)  
2744 are required for compatibility with various TLS implementations.

2745 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)  
2746 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be  
2747 present and End-Entity certificates with ECU extensions containing both OIDs SHALL be  
2748 rejected.

2749 – certificatePolicies

2750 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one  
2751 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)  
2752 corresponding to the version of the OCF Certificate Policy under which it was issued. Additional  
2753 manufacturer-specific CP OIDs may also be populated.

2754 **10.5 Device Authentication with OCF Cloud – moved to OCF Cloud Security document**

2755 .

2756

DRAFT

2757 **11 Message Integrity and Confidentiality**

2758 **11.1 Preamble**

2759 Secured communications between Clients and Servers are protected against eavesdropping,  
2760 tampering, or message replay, using security mechanisms that provide message confidentiality and  
2761 integrity.

2762 **11.2 Session Protection with DTLS**

2763 **11.2.1 DTLS Protection General**

2764 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices  
2765 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See  
2766 11.3 for a list of required and optional cipher suites for message communication.

2767 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1  
2768 or lower.

2769 Multicast session semantics are not yet defined in this version of the security document.

2770 **11.2.2 Unicast Session Semantics**

2771 For unicast messages between a Client and a Server, both Devices shall authenticate each other.  
2772 See clause 10 for details on Device Authentication.

2773 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.  
2774 The sending Device shall encrypt and authenticate messages as defined by the selected cipher  
2775 suite and the receiving Device shall verify and decrypt the messages before processing them.

2776 **11.2.3 Cloud Session Semantics – moved to OCF Cloud Security document**

2777 **11.3 Cipher Suites**

2778 **11.3.1 Cipher Suites General**

2779 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher  
2780 suites allowed during ownership transfer and normal operation. The following RFCs provide  
2781 additional information about the cipher suites used in OCF.

2782 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2783 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2784 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and  
2785 PSKs

2786 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2787 **11.3.2 Cipher Suites for Device Ownership Transfer**

2788 **11.3.2.1 Just Works Method Cipher Suites**

2789 The Just Works OTM may use the following (D)TLS cipher suites.

2790 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,

2791 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

2792 All Devices supporting Just Works OTM shall implement:

2793 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256 (with the value 0xFF00)

2794 All Devices supporting Just Works OTM should implement:

2795 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256 (with the value 0xFF01)

2796 **11.3.2.2 Random PIN Method Cipher Suites**

2797 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2798 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2799 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2800 All Devices supporting Random Pin Based OTM shall implement:

2801 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256

2802 **11.3.2.3 Certificate Method Cipher Suites**

2803 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2804 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

2805 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2806 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2807 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2808 Using the following curve:

2809 secp256r1 (See IETF RFC 4492)

2810 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2811 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2812 Devices supporting Manufacturer Certificate Based OTM should implement:

2813 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2814 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2815 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2816 **11.3.3 Cipher Suites for Symmetric Keys**

2817 The following cipher suites are defined for (D)TLS communication using PSKs:

2818 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2819 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2820 TLS\_PSK\_WITH\_AES\_128\_CCM\_8, (\* 8 OCTET Authentication tag \*)

2821 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

2822 TLS\_PSK\_WITH\_AES\_128\_CCM, (\* 16 OCTET Authentication tag \*)

2823 TLS\_PSK\_WITH\_AES\_256\_CCM,

2824 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2825 All Devices shall implement the following:

2826 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2827

2828 Devices should implement the following:

2829 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2830 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2831 TLS\_PSK\_WITH\_AES\_128\_CCM\_8,  
2832 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,  
2833 TLS\_PSK\_WITH\_AES\_128\_CCM,  
2834 TLS\_PSK\_WITH\_AES\_256\_CCM

#### 2835 **11.3.4 Cipher Suites for Asymmetric Credentials**

2836 The following cipher suites are defined for (D)TLS communication with asymmetric keys or  
2837 certificates:

2838 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,  
2839 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,  
2840 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,  
2841 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2842 Using the following curve:

2843 secp256r1 (See IETF RFC 4492)

2844 All Devices supporting Asymmetric Credentials shall implement:

2845 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2846 All Devices supporting Asymmetric Credentials should implement:

2847 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,  
2848 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,  
2849 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

#### 2850 **11.3.5 Cipher suites for OCF Cloud Credentials – moved to OCF Cloud Security document**

2851

2852 **12 Access Control**

2853 **12.1 ACL Generation and Management**

2854 This clause will be expanded in a future version of the document.

2855 **12.2 ACL Evaluation and Enforcement**

2856 **12.2.1 ACL Evaluation and Enforcement General**

2857 The Server enforces access control over application Resources before exposing them to the  
2858 requestor. The Security Layer in the Server authenticates the requestor when access is received  
2859 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL  
2860 entries that specify the requestor's identity, role or may match authenticated requestors using a  
2861 subject wildcard.

2862 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a  
2863 subject wildcard match of anonymous requestors.

2864 Access is denied if a requested Resource is not matched by an ACL entry.

2865 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources  
2866 may be granted prior to provisioning of ACL Resources.

2867 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries (ACE2)  
2868 that employ a Resource matching algorithm that uses an array of Resource references to match  
2869 Resources to which the ACE2 access policy applies. Matching consists of comparing the values of  
2870 the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources are matched  
2871 in two ways:

- 2872 1) host reference ("href")
- 2873 2) resource wildcard ("wc").

2874 **12.2.2 Host Reference Matching**

2875 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for  
2876 Resource matching.

2877 – The href Property shall be used to find an exact match of the Resource name if present.

2878 **12.2.3 Resource Wildcard Matching**

2879 When present, a wildcard (wc) expression shall be used to match multiple Resources using a  
2880 wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

2881 A wildcard expression may be used to match multiple Resources using a wildcard Property  
2882 contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined  
2883 in Table 23.

2884 **Table 21 – 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.

2885 NOTE Discoverable resources appear in the "/oic/res" Resource, while non-discoverable resources may appear in other  
2886 collection resources but do not appear in the /res collection.

#### 2887 **12.2.4 Multiple Criteria Matching**

2888 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for  
2889 each array element. For example, if a first array element of the "resources" Property contains  
2890 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",  
2891 then Resources that match either of the two "href" criteria shall be included in the set of matched  
2892 Resources.

2893 Example 1 JSON for Resource matching

```
2894 {  
2895 //Matches Resources named "/x/door1" or "/x/door2"  
2896 "resources": [  
2897   {  
2898     "href": "/x/door1"  
2899   },  
2900   {  
2901     "href": "/x/door2"  
2902   },  
2903 ]  
2904 }
```

2905 Example 2 JSON for Resource matching

```
2906 {  
2907 // Matches all Resources  
2908 "resources": [  
2909   {  
2910     "wc": "*"   
2911   }  
2912 ]  
2913 }
```

#### 2914 **12.2.5 Subject Matching using Wildcards**

2915 When the ACE subject is specified as the wildcard string "\*" any requestor is matched. The OCF  
2916 server may authenticate the OCF client, but is not required to.

2917 Examples: JSON for subject wildcard matching

```
2918 //matches all subjects that have authenticated and confidentiality protections in place.  
2919 "subject" : {  
2920   "conntype" : "auth-crypt"  
2921 }  
2922 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
2923 "subject" : {  
2924   "conntype" : "anon-clear"  
2925 }
```

#### 2926 **12.2.6 Subject Matching using Roles**

2927 When the ACE subject is specified as a role, a requestor shall be matched if either:

2928 1) The requestor authenticated with a symmetric key credential, and the role is present in the  
2929 roleid Property of the credential's entry in the credential resource, or

2930 2) The requestor authenticated with a certificate, and a valid role certificate is present in the roles  
2931 resource with the requestor's certificate's public key at the time of evaluation. Validating role  
2932 certificates is defined in 10.3.1.

## 2933 **12.2.7 ACL Evaluation**

### 2934 **12.2.7.1 ACE2 matching algorithm**

2935 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 2936 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these ACE2  
2937 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the signature,  
2938 at least once, following update of the "/oic/sec/sacl" Resource.
- 2939 2) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.
- 2940 3) Access shall be granted when all these criteria are met:
  - 2941 a) The requestor is matched by the ACE2 "subject" Property.
  - 2942 b) The requested Resource is matched by the ACE2 resources Property and the requested  
2943 Resource shall exist on the local Server.
  - 2944 c) The "period" Property constraint shall be satisfied.
  - 2945 d) The "permission" Property constraint shall be applied.

2946 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching  
2947 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then UNION  
2948 (Perm1, Perm2)=CRUDN.

2949 The Server shall enforce access based on the effective permissions granted.

2950 Batch requests to Resource containing Links require additional considerations when accessing the  
2951 linked Resources. ACL considerations for batch request to the Atomic Measurement Resource  
2952 Type are provided in clause 12.2.7.2. ACL considerations for batch request to the Collection  
2953 Resource Type are provided in 12.2.7.3.

### 2954 **12.2.7.2 (Currently blank)**

2955 This clause intentionally left empty.

### 2956 **12.2.7.3 ACL considerations for a batch OCF Interface request to a Collection**

2957 This clause addresses the additional authorization processes which take place when a Server  
2958 receives a batch OCF Interface request from a Client to a Collection hosted on that Server,  
2959 assuming there is an ACE matching the Collection which permits the original Client request. For  
2960 the purposes of this clause, the Server hosting this Collection is called the "Collection host". The  
2961 additional authorization process is dependent on whether the linked Resource is hosted on the  
2962 Collection host or the linked Resource is hosted on another Server:

- 2963 – For each generated request to a linked Resource hosted on the Collection host, the Collection  
2964 host shall apply the ACE2 matching algorithm in clause 12.2.7.1 to determine whether the linked  
2965 Resource is permitted to process the generated request, with the following clarifications:
  - 2966 – The requestor in clause 12.2.7.1 shall be the Client which sent the original Client request.
  - 2967 – The requested Resource in clause 12.2.7.1 shall be the linked Resource, which shall be  
2968 matched using at least one of:
    - 2969 – a Resource Wildcard matching the linked Resource, or
    - 2970 – an exact match of the local path of the linked Resource with a "href" Property in the  
2971 "resources" array in the ACE2.

2972           – an exact match of the full URI of the linked Resource with a "href" Property in the  
2973           "resources" array in the ACE2.

2974 NOTE The full URI of a linked Resource is obtained by concatenating the "anchor" Property of the Link, if present, and  
2975 the "href" Property of the Link. The local path can then be determined from the full URI.

2976 If the linked Resource is not permitted to process the generated request, then the Collection host  
2977 shall treat such cases as a linked Resource which cannot process the request when composing the  
2978 aggregated response to the original Client Request, as specified for the batch OCF Interface in the  
2979 ISO/IEC 30118-1:2018.

DRAFT

2980 **13 Security Resources**

2981 **13.1 Security Resources General**

2982 OCF Security Resources are shown in Figure 34.

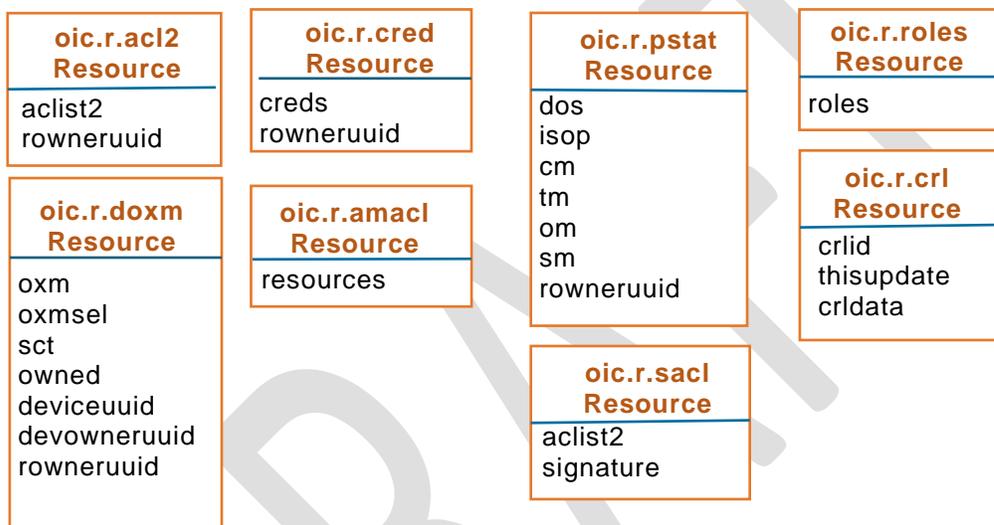
2983 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

2984 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

2985 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

2986 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

2987



2988

**Figure 33 – OCF Security Resources**

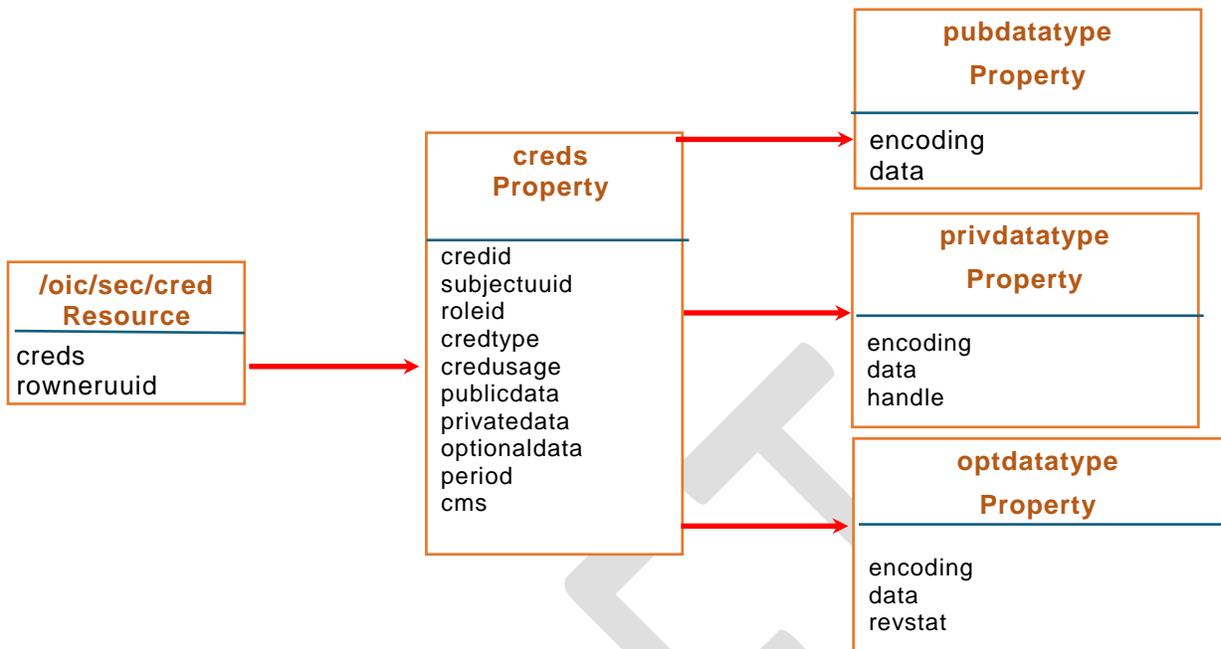


Figure 34 – "/oic/sec/cred" Resource and Properties

2989

2990

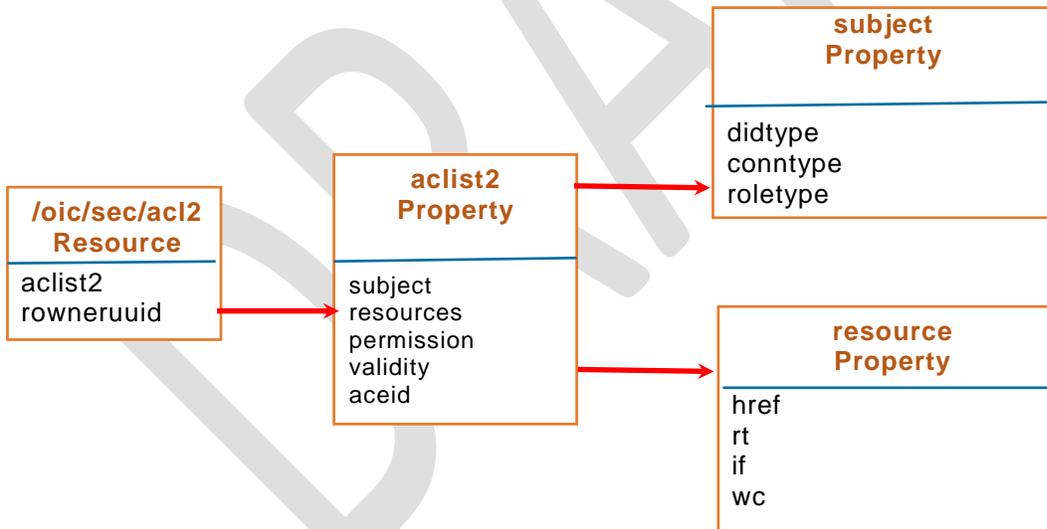
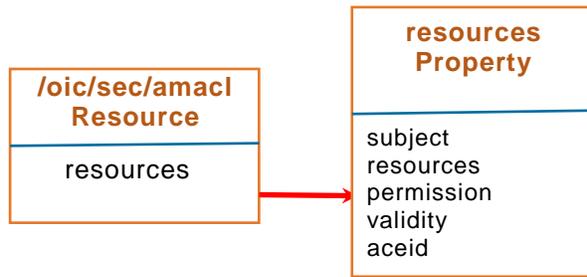
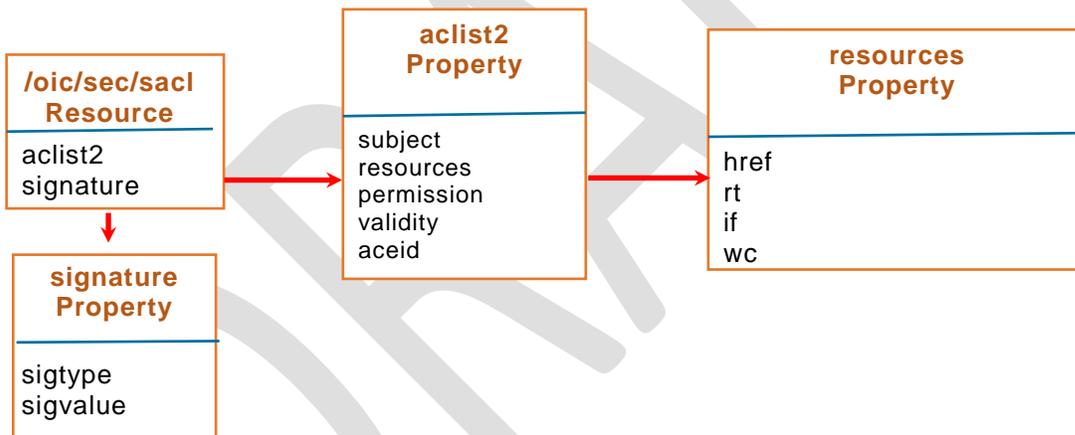


Figure 35 – "/oic/sec/acl2" Resource and Properties

2991



2992 **Figure 36 – "/oic/sec/amacl" Resource and Properties**



2993 **Figure 37 – "/oic/sec/sacl" Resource and Properties**

2994 **13.2 Device Owner Transfer Resource**

2995 **13.2.1 Device Owner Transfer Resource General**

2996 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

2997 Resource discovery processing respects the CRUDN constraints supplied as part of the security  
2998 Resource definitions contained in this document.

2999 "/oic/sec/doxm" Resource is defined in Table 24.

3000

**Table 22 – 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

3001 Table 25 defines the Properties of the "/oic/sec/doxm" Resource.

3002

**Table 23 – 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.

3003 Table 26 defines the Properties of the "oic.sec.didtype".

3004 **Table 24 – 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

3005 The oxms Property contains a list of OTM where the entries appear in the order of preference. This  
 3006 Property contains the higher priority methods appearing before the lower priority methods. The  
 3007 DOTS queries this list at the time of onboarding and selects the most appropriate method.

3008 The DOTS shall update the oxmsel Property of the "/oic/sec/doxm" Resource with the OTM that  
 3009 was used to onboard the Device.

3010 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3011 <DoxmType> ::= <NSS>
3012 <NSS> ::= <Identifier> | { {<NID> "."} <NameSpaceQualifier> "."} <Method>
3013 <NID> ::= <Vendor-or-Organization>
3014 <Identifier> ::= INTEGER
3015 <NameSpaceQualifier> ::= String
3016 <Method> ::= String
3017 <Vendor-Organization> ::= String

```

3018 When an OTM successfully completes, the "owned" Property is set to "1" (TRUE). Consequently,  
 3019 subsequent attempts to take ownership of the Device will fail.

3020 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in the  
 3021 "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource is  
 3022 UPDATED to non-nil UUID value.

3023 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource  
3024 after it has updated the devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-nil-  
3025 UUID value.

3026 The Device vendor shall determine that the Device identifier ("deviceuuid") is persistent (not  
3027 updatable) or that it is non-persistent (updatable by the owner transfer service – aka. DOTS).

3028 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall  
3029 fail with the error PROPERTY\_NOT\_FOUND.

3030 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is non-persistent, the request to  
3031 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is  
3032 RESET. If the UPDATE to deviceuuid Property of the "/oic/sec/doxm" Resource fails while in the  
3033 RFOTM Device state the device state shall transition to RESET where the Server shall set the  
3034 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g. "00000000-  
3035 0000-0000-0000-000000000000").

3036 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the  
3037 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the "deviceuuid"  
3038 Property of the "/oic/sec/doxm" Resource each time the device enters RESET Device state. The  
3039 temporary deviceuuid value is used while the device state is in the RESET state and while in the  
3040 RFOTM device state until the DOTS establishes a secure OTM connection. The DOTS should  
3041 RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm" Resource after it has  
3042 updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-nil-UUID value.

3043 The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall expose a persistent value (i.e. is  
3044 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTS via  
3045 an OCF Interface to the deviceuuid Property of the "/oic/sec/doxm" Resource during RFOTM Device  
3046 state.).

3047 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a  
3048 secure OTM connection and UPDATES the "devowneruuid" Property to a non-nil UUID value.  
3049 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the "deviceuuid" Property of  
3050 the "/oic/sec/doxm" Resource shall reveal the persistent or semi-persistent value to authenticated  
3051 requestors and shall reveal the temporary non-repeated value to unauthenticated requestors.

3052 See 13.16 for additional details related to privacy sensitive considerations.

### 3053 **13.2.2 Persistent and Semi-Persistent Device Identifiers**

3054 The Device vendor determines whether a device identifier can be set by a configuration tool or  
3055 whether it is immutable. If it is an immutable value this document refers to it as a persistent device  
3056 identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four device  
3057 identifiers that could be considered persistent or semi-persistent:

- 3058 1) "deviceuuid" Property of "/oic/sec/doxm"
- 3059 2) "di" Property of "/oic/d"
- 3060 3) "piid" Property of "/oic/d"
- 3061 4) "pi" Property of "/oic/p"

### 3062 **13.2.3 Onboarding Considerations for Device Identifier**

3063 The "deviceuuid" is used to onboard the Device. The other identifiers ("di", "piid" and "pi") are not  
3064 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the  
3065 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security Domain  
3066 owner may have a preference for persistent or semi-persistent device identifiers. Detecting whether  
3067 the Device is using persistent or semi-persistent deviceuuid can be achieved by attempting to  
3068 update it.

3069 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is persistent, then an UPDATE request,  
 3070 at the appropriate time during onboarding shall fail with an appropriate error response.

3071 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device  
 3072 state is RFOTM and when devowneruid Property value of the "/oic/sec/doxm" Resource has a  
 3073 non-nil UUID value.

3074 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is semi-persistent, subsequent to a  
 3075 successful UPDATE request to change it; the Device shall remember the semi-persistent value  
 3076 until the next successful UPDATE request or until the Device state transitions to RESET.

3077 See 13.16 for addition behaviour regarding "deviceuuid".

3078

3079 **13.2.4 OCF defined OTMs**

3080 Table 27 defines the Properties of the "oic.sec.doxmtype".

3081

**Table 25 – Properties of the "oic.sec.doxmtype" type**

Value Type Name	Value Type URN (optional)	Enumeration Value (mandatory)	Description
OCFJustWorks	oic.sec.doxm.jw	0	The just-works method relies on anonymous Diffie-Hellman key agreement protocol to allow an DOTS to assert ownership of the new Device. The first DOTS to make the assertion is accepted as the Device owner. The just-works method results in a shared secret that is used to authenticate the Device to the DOTS and likewise authenticates the DOTS to the Device. The Device allows the DOTS to take ownership of the Device, after which a second attempt to take ownership by a different DOTS will fail <sup>a</sup> .
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a DOTS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTS signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec.doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.doxm.self	4	The manufacturer shall set the "/doxm.doxmsel" value to (4). The Server shall reset this value to (4) upon entering RESET Device state.
OCF Reserved	<Reserved>	5~0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	Reserved for vendor-specific OTM use

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

3082 **13.3 Credential Resource**

3083 **13.3.1 Credential Resource General**

3084 The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and  
3085 support services as well as credentials used to verify Clients and support services.

3086 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared  
3087 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to  
3088 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3089 In order to provide an interface which allows management of the "creds" Array Property, the  
3090 RETRIEVE, UPDATE and DELETE operations on the "oic.r.cred" Resource shall behave as follows:

3091 1) A RETRIEVE shall return the full Resource representation, except that any write-only Properties  
3092 shall be omitted (e.g. private key data).

3093 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3094 UPDATE request, as follows:

3095 a) If an UPDATE representation includes the "creds" array Property, then:

3096 i) Supplied "creds" with a "credid" that matches an existing "credid" shall replace  
3097 completely the corresponding "cred" in the existing "creds" array.

3098 ii) Supplied "creds" without a "credid" shall be appended to the existing "creds" array, and  
3099 a unique (to the cred Resource) "credid" shall be created and assigned to the new "cred"  
3100 by the Server. The "credid" of a deleted "cred" should not be reused, to improve the  
3101 determinism of the interface and reduce opportunity for race conditions.

3102 iii) Supplied "creds" with a "credid" that does not match an existing "credid" shall be  
3103 appended to the existing "creds" array, using the supplied "credid".

3104 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device  
3105 States in which an UPDATE of the "creds" array Property is always rejected. If OCF  
3106 Device is in a Device State where the Access Mode in this row contains "R", then the  
3107 OCF Device shall reject all UPDATES of the "creds" array Property.

3108 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not remove  
3109 the "oic.r.cred" Resource.

3110 4) A DELETE with one or more "credid" query parameters shall remove the "cred"(s) with the  
3111 corresponding "credid"(s) from the "creds" array.

3112 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in  
3113 which a DELETE is always rejected. If OCF Device is in a Device State where the Access Mode  
3114 in this row contains "R", then the OCF Device shall reject all DELETES.

3115 NOTE The "oic.r.cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3116 in ISO/IEC 30118-1:2018.

3117 "oic.r.cred" Resource is defined in Table 28.

3118 **Table 26 – 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

3119 Table 29 defines the Properties of the "/oic/sec/cred" Resource.

Table 27 – 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.

3121 All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end  
3122 interaction.

3123 The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid Property.

3124 ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access  
3125 modes.

3126 Table 30 defines the Properties of "oic.sec.cred".

Table 28 – 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".

3128 Table 31 defines the Properties of "oic.sec.credusagetype".

3129 **Table 29: 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

3130 Table 32 defines the Properties of "oic.sec.pubdatatype".

3131 **Table 30 – 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

3132 Table 33 defines the Properties of "oic.sec.privdatatype".

3133 **Table 31 – 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

3134 Table 34 defines the Properties of "oic.sec.optdatatype".

3135 **Table 32 – 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

3136 Table 35 defines the Properties of "oic.sec.roletype".

3137 **Table 33 – 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.

3138 **13.3.2 Properties of the Credential Resource**

3139 **13.3.2.1 Credential ID**

3140 Credential ID ("credid") is a local reference to an entry in a "creds" Property array of the  
3141 "/oic/sec/cred" Resource. The SRM generates it. The "credid" Property shall be used to  
3142 disambiguate array elements of the "creds" Property.

3143 **13.3.2.2 Subject UUID**

3144 The "subjectuuid" Property identifies the Device to which an entry in a "creds" Property array of the  
3145 "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication  
3146 challenge-response or to authenticate an authentication challenge.

3147 A "subjectuuid" Property that matches the Server's own "deviceuuid" Property, distinguishes the  
3148 array entries in the "creds" Property that pertain to this Device.

3149 The "subjectuuid" Property shall be used to identify a group to which a group key is used to protect  
3150 shared data.

3151 When certificate chain is used during secure connection establishment, the "subjectuud" Property  
3152 shall also be used to verify the identity of the responder. The presented certificate chain shall be  
3153 accepted, if there is a matching Credential entry on the Device that satisfies all of the following:

- 3154 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3155 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity certificate  
3156 in the presented chain. If Subject UUID of the entry is set as a wildcard "\*", this condition is  
3157 automatically satisfied.
- 3158 – Credential Usage of the entry is "oic.sec.cred.trustca".

#### 3159 **13.3.2.3 Role ID**

3160 The roleid Property identifies a role that has been granted to the credential.

#### 3161 **13.3.2.4 Credential Type**

3162 The "credtype" Property is used to interpret several of the other Property values whose contents  
3163 can differ depending on credential type. These Properties include "publicdata", "privatedata" and  
3164 "optionaldata". The "credtype" Property value of "0" ("no security mode") is reserved for testing and  
3165 debugging circumstances. Production deployments shall not allow provisioning of credentials of  
3166 type "0". The SRM should introduce checking code that prevents its use in production deployments.

#### 3167 **13.3.2.5 Public Data**

3168 The "publicdata" Property contains information that provides additional context surrounding the  
3169 issuance of the credential. For example, it might contain information included in a certificate or  
3170 response data from a CMS. It might contain wrapped data.

#### 3171 **13.3.2.6 Private Data**

3172 The "privatedata" Property contains secret information that is used to authenticate a Device, protect  
3173 data or verify an authentication challenge-response.

3174 The "privatedata" Property shall not be disclosed outside of the SRM's trusted computing perimeter.  
3175 A secure element (SE) or trusted execution environment (TEE) should be used to implement the  
3176 SRM's trusted computing perimeter. The privatedata contents may be referenced using a handle;  
3177 for example, if used with a secure storage sub-system.

#### 3178 **13.3.2.7 Optional Data**

3179 The "optionaldata" Property contains information that is optionally supplied, but facilitates key  
3180 management, scalability or performance optimization.

#### 3181 **13.3.2.8 Period**

3182 The "period" Property identifies the validity period for the credential. If no validity period is specified,  
3183 the credential lifetime is undetermined. Constrained devices that do not implement a date-time  
3184 capability shall obtain current date-time information from its CMS.

#### 3185 **13.3.2.9 Credential Refresh Method Type Definition**

3186 The CMS shall implement the credential refresh methods specified in the "crms" Property of the  
3187 "oic.sec.creds" array in the "/oic/sec/cred" Resource.

3188 Table 36 defines the values of "oic.sec.crmttype".

**Table 34 – Value Definition of the "oic.sec.crmtype" Property**

Value Type Name	Value Type URN	Applicable Credential Type	Description
Provisioning Service	oic.sec.crm.pro	All	A CMS initiates re-issuance of credentials nearing expiration. The Server should delete expired credentials to manage storage resources. The Resource Owner Property references the provisioning service. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify additional key management service that supports this credential refresh method.
Pre-shared Key	oic.sec.crm.psk	[1]	The Server performs ad-hoc key refresh by initiating a DTLS connection with the Device prior to credential expiration using a Diffie-Hellman based ciphersuite and the current PSK. The new DTLS MasterSecret value becomes the new PSK. The Server selects the new validity period. The new validity period value is sent to the Device who updates the validity period for the current credential. The Device acknowledges this update by returning a successful response or denies the update by returning a failure response. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify a key management service that supports this credential refresh method.
Random PIN	oic.sec.crm.rdp	[16]	The Server performs ad-hoc key refresh following the "oic.sec.crm.psk" approach, but in addition generates a random PIN value that is communicated out-of-band to the remote Device. The current PSK + PIN are hashed to form a new PSK' that is used with the DTLS ciphersuite. I.e. PSK' = SHA256(PSK, PIN). The Server uses its "/oic/sec/cred.rownruuid" Resource to identify a key management service that supports this credential refresh method.
SKDC	oic.sec.crm.skdc	[1, 2, 4, 32]	The Server issues a request to obtain a ticket for the Device. The Server updates the credential using the information contained in the response to the ticket request. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify the key management service that supports this credential refresh method.
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.

### 3190 13.3.2.10 Credential Usage

3191 Credential Usage indicates to the Device the circumstances in which a credential should be used.  
3192 Five values are defined:

- 3193 – "oic.sec.cred.trustca": This certificate is a trust anchor for the purposes of certificate chain  
3194 validation, as defined in 10.4. OCF Server SHALL remove any "/oic/sec/cred" entries with an  
3195 "oic.sec.cred.trustca" credusage upon transitioning to RFOTM. OCF Servers SHALL use  
3196 "/oic/sec/cred" entries that have an "oic.sec.cred.trustca" Value of "credusage" Property only  
3197 as trust anchors for post-onboarding (D)TLS session establishment in RFNOP state; these  
3198 entries are not to be used for onboarding (D)TLS sessions.
- 3199 – "oic.sec.cred.cert": This "credusage" is used for certificates for which the Device possesses the  
3200 private key and uses it for identity authentication in a secure session, as defined in clause 10.4.
- 3201 – "oic.sec.cred.rolecert": This "credusage" is used for certificates for which the Device possesses  
3202 the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3203 – "oic.sec.cred.mfgtrustca": This certificate is a trust anchor for the purposes of the Manufacturer  
3204 Certificate Based OTM as defined in clause 7.3.6. OCF Servers SHALL use "/oic/sec/cred"  
3205 entries that have an "oic.sec.cred.mfgtrustca" Value of "credusage" Property only as trust  
3206 anchors for onboarding (D)TLS session establishment; these entries are not to be used for post-  
3207 onboarding (D)TLS sessions.

3208 – "oic.sec.cred.mfgcert": This certificate is used for certificates for which the Device possesses  
3209 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as  
3210 defined in clause 7.3.6.

### 3211 13.3.3 Key Formatting

#### 3212 13.3.3.1 Symmetric Key Formatting

3213 Symmetric keys shall have the format described in Table 37 and Table 38.

3214 **Table 35 – 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.

3215

3216 **Table 36 – 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.

#### 3217 13.3.3.2 Asymmetric Keys

3218 Asymmetric key formatting is not available in this revision of the document.

#### 3219 13.3.3.3 Asymmetric Keys with Certificate

3220 Key formatting is defined by certificate definition.

#### 3221 13.3.3.4 Passwords

3222 Password formatting is not available in this revision of the document.

### 3223 13.3.4 Credential Refresh Method Details

#### 3224 13.3.4.1 Provisioning Service

3225 The resource owner identifies the provisioning service. If the Server determines a credential  
3226 requires refresh and the other methods do not apply or fail, the Server will request re-provisioning  
3227 of the credential before expiration. If the credential is allowed to expire, the Server should delete  
3228 the Resource.

#### 3229 13.3.4.2 Pre-Shared Key

##### 3230 13.3.4.2.1 Pre-Shared Key General

3231 Using this mode, the current PSK is used to establish a Diffie-Hellman session key in DTLS. The  
3232 TLS\_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3233  $PSK = TLS\_PRF(\text{MasterSecret}, \text{Message}, \text{length});$

3234 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of the  
3235 above ciphersuites.

3236 – Message is the concatenation of the following values:

3237 – RM - Refresh method – I.e. "oic.sec.crm.psk"

3238 – Device ID\_A is the string representation of the Device ID that supplied the DTLS ClientHello.

3239 – Device ID\_B is the Device responding to the DTLS ClientHello message  
3240 – Length of Message in bytes.  
3241 Both Server and Client use the PSK to update the "/oic/sec/cred" Resource's "privatedata" Property.  
3242 If Server initiated the credential refresh, it selects the new validity period. The Server sends the  
3243 chosen validity period to the Client over the newly established DTLS session so it can update the  
3244 corresponding credential Resource for the Server.

#### 3245 **13.3.4.2.2 Random PIN**

3246 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.  
3247 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session  
3248 key should be used to switch from PIN to PSK mode.

3249 The PIN is randomly generated by the Server and communicated to the Client through an out-of-  
3250 band method. The OOB method used is out-of-scope.

3251 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used to  
3252 generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a DTLS  
3253 ciphersuite that accepts a PSK.

3254  $PPSK = PBKDF2(PRF, PIN, RM, Device\ ID, c, dkLen)$

3255 The PBKDF2 function has the following parameters:

- 3256 – PRF – Uses the DTLS PRF.
- 3257 – PIN – Shared between Devices.
- 3258 – RM - Refresh method – I.e. "oic.sec.crm.rdp"
- 3259 – Device ID – UUID of the new Device.
- 3260 – c – Iteration count initialized to 1000, incremented upon each use.
- 3261 – dkLen – Desired length of the derived PSK in octets.

3262 Both Server and Client use the PPSK to update the "/oic/sec/cred" Resource's PrivateData Property.  
3263 If Server initiated the credential refresh, it selects the new validity period. The Server sends the  
3264 chosen validity period to the Client over the newly established DTLS session so it can update its  
3265 corresponding credential Resource for the Server.

#### 3266 **13.3.4.2.3 SKDC**

3267 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid  
3268 Property value that matches a CMS that implements SKDC functionality and where the Client  
3269 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request message  
3270 is delivered to the CMS and in response returns the ticket request. The Server updates or  
3271 instantiates a "/oic/sec/cred" Resource guided by the ticket response contents.

#### 3272 **13.3.4.2.4 PKCS10**

3273 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruuid  
3274 Property value that matches a CMS that supports the "oic.sec.crm.pk10" credential refresh method.  
3275 A PKCS10 formatted message is delivered to the service. After the refreshed certificate is issued,  
3276 the CMS pushes the certificate to the Server. The Server updates or instantiates an "/oic/sec/cred"  
3277 Resource guided by the certificate contents.

#### 3278 **13.3.4.3 Resource Owner**

3279 The Resource Owner Property allows credential provisioning to occur soon after Device onboarding  
3280 before access to support services has been established. It identifies the entity authorized to  
3281 manage the "/oic/sec/cred" Resource in response to Device recovery situations.

3282 **13.4 Certificate Revocation List**

3283 **13.4.1 CRL Resource Definition**

3284 Device certificates and private keys are kept in "cred" Resource. CRL is maintained and updated  
 3285 with a separate "crl" Resource that is newly defined for maintaining the revocation list.

3286 "oic.r.crl" Resource is defined in Table 39.

3287 **Table 37 – 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

3288 Table 40 defines the Properties of "oic.r.crl".

3289 **Table 38 – 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

3290 **13.5 ACL Resources**

3291 **13.5.1 ACL Resources General**

3292 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be  
 3293 expressed using three ACL Resource Types: "/oic/sec/acl2", "/oic/sec/amacl" and "/oic/sec/sacl".  
 3294 The subject (e.g. "deviceuuid" of the Client) requesting access to a Resource shall be authenticated  
 3295 prior to applying the ACL check. Resources that are available to multiple Clients can be matched  
 3296 using a wildcard subject. All Resources accessible via the unsecured communication endpoint shall  
 3297 be matched using a wildcard subject.

3298 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3299 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3300 **Table 39 – 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>

3301 The <DeviceId> token means the requestor must possess a credential that uses <UUID> as its  
3302 identity in order to match the requestor to the <ACE> policy.

3303 The <RoleId> token means the requestor must possess a role credential with <Character> as its  
3304 role in order to match the requestor to the <ACE> policy.

3305 The <Wildcard> token "\*" means any requestor is matched to the <ACE> policy, with or without  
3306 authentication.

3307 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the <ACE>  
3308 policy to Resources.

3309 The <OIC\_LINK> token contains values used to query existence of hosted Resources.

3310 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>  
3311 and <ResourceRef> matching does not produce the empty set match.

3312 Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE ("D"),  
3313 NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the  
3314 respective permission is not granted.

3315 The empty set match result defaults to a condition where no access rights are granted.

3316 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.  
3317 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively  
3318 be granted and rescinded according to the pattern.

### 3319 13.5.3 ACL Resource

3320 An "acl2" is a list of type "ace2".

3321 In order to provide an interface which allows management of array elements of the "aclist2"  
3322 Property associated with a "/oic/sec/acl2" Resource. The RETRIEVE, UPDATE and DELETE  
3323 operations on the "/oic/sec/acl2" Resource SHALL behave as follows:

- 3324 1) A RETRIEVE shall return the full Resource representation.
- 3325 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3326 UPDATE request, as follows:
  - 3327 a) If an UPDATE representation includes the array Property, then:
    - 3328 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace completely  
3329 the corresponding ACE in the existing "aces2" array.
    - 3330 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array, and  
3331 a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new ACE  
3332 by the Server. The "aceid" of a deleted ACE should not be reused, to improve the  
3333 determinism of the interface and reduce opportunity for race conditions.

3334           iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be  
3335           appended to the existing "aces2" array, using the supplied "aceid".

3336   The rows in Table 44 defines the Properties of "oic.sec.acl2".

3337           iv) Table 44 corresponding to the "aclist2" array Property dictate the Device States in which  
3338           an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is in a  
3339           Device State where the Access Mode in this row contains "R", then the OCF Device  
3340           shall reject all UPDATEs of the "aclist2" array Property.

3341   3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not remove  
3342   the "oic.r.ace2" Resource.

3343   4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the  
3344   corresponding "aceid"(s) from the "aces2" array.

3345   The rows in Table 44 defines the Properties of "oic.sec.acl2".

3346   5) Table 44 corresponding to the "aclist2" array Property dictate the Device States in which a  
3347   DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this  
3348   row contains "R", then the OCF Device shall reject all DELETES.

3349   NOTE   The "oic.r.acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3350   in ISO/IEC 30118-1:2018.

3351   Evaluation of local ACL Resource completes when all ACL Resource have been queried and no  
3352   entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl2",  
3353   "/oic/sec/sacl" and "/oic/sec/amacl" do not match the subject and the requested Resource.

3354   It is possible the AMS has an ACL policy that satisfies a resource access request, but the necessary  
3355   ACE has not been provisioned to Server. The Server may open a secure connection to the AMS to  
3356   request ACL provisioning. The Server may use filter criteria that returns a subset of the AMS ACL  
3357   policy. The AMS shall obtain the Server Device ID using the secure connection context.

3358   The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS to  
3359   process an "/oic/sec/amacl" Resource. The AMS shall match the AMACL policy and return the  
3360   Permission Property or an error if no match is found.

3361   If the requested Resource is still not matched, the Server returns an error. The requester should  
3362   query the Server to discover the configured AMS services. The Client should contact the AMS to  
3363   request a sacl ("/oic/sec/sacl") Resource. Performing the following operations implement this type  
3364   of request:

3365   1) Client: Open secure connection to AMS.

3366   2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3367   3) AMS: constructs a "/oic/sec/sacl" Resource that is signed by the AMS and returns it in response  
3368   to the RETRIEVE command.

3369   4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3370   5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3371   6) Client: retries original Resource access request. This time the new ACL is included in the local  
3372   ACL evaluation.

3373   The ACL contained in the "/oic/sec/sacl" Resource should grant longer term access that satisfies  
3374   repeated Resource requests.

3375   Table 42 defines the values of "oic.sec.crudntype".

3376

**Table 40 – Value Definition of the "oic.sec.crudntype" Property**

Value	Access Policy	Description	RemarksNotes
bx0000,0000 (0)	No permissions	No permissions	N/A
bx0000,0001 (1)	C	CREATE	N/A
bx0000,0010 (2)	R	RETREIVE, OBSERVE, DISCOVER	The "R" permission bit covers both the Read permission and the Observe permission.
bx0000,0100 (4)	U	WRITE, UPDATE	N/A
bx0000,1000 (8)	D	DELETE	N/A
bx0001,0000 (16)	N	NOTIFY	The "N" permission bit is ignored in OCF 1.0, since "R" covers the Observe permission. It is documented for future versions

3377 "oic.sec.acl2" Resource is defined in Table 28.

3378

**Table 41 – 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

3379 Table 44 defines the Properties of "oic.sec.acl2".

**Table 42 – Properties of the "oic.sec.acl2" Resource**

Property Name	Value Type	Mandatory	Device State	Access Mode	Description
aclist2	array of oic.sec.ace2	Yes	N/A		The aclist2 Property is an array of ACE records of type "oic.sec.ace2". The Server uses this list to apply access control to its local resources.
N/A	N/A	N/A	RESET	R	Server shall set to manufacturer defaults.
			RFOTM	RW	Set by DOTS after successful OTM
			RFPRO	RW	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
			RFNOP	R	Access to NCRs is permitted after a matching ACE2 is found.
			SRESET	RW	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm Resource") should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
rowneruuid	uuid	Yes	N/A		The resource owner Property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
			RFOTM	RW	The DOTS should configure the rowneruuid Property of "/oic/sec/acl2" Resource when a successful owner transfer session is established.
			RFPRO	R	n/a
			RFNOP	R	n/a
			SRESET	RW	The DOTS (referenced via devowneruuid Property or rowneruuid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3381

3382 Table 45 defines the Properties of "oic.sec.ace2".

3383

**Table 43 – "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.

3384 Table 46 defines the Properties of "oic.sec.ace2.resource-ref".

3385

**Table 44 – "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.

3386 Table 47 defines the values of "oic.sec.ace2.resource-ref".

3387

**Table 45 – 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

3388 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack  
 3389 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's  
 3390 request using policies contained in ACL resources.

3391 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified  
 3392 Resource references include the device identifier in the href Property that identifies the remote  
 3393 Resource Server that hosts the Resource. Partially qualified references mean that the local  
 3394 Resource Server hosts the Resource. If a fully qualified resource reference is given, the  
 3395 Intermediary enforcing access shall have a secure channel to the Resource Server and the  
 3396 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource  
 3397 access enforcement point.

3398 Resource Servers should include references to Device and ACL Resources where access  
3399 enforcement is to be applied. However, access enforcement logic shall not depend on these  
3400 references for access control processing as access to Server Resources will have already been  
3401 granted.

3402 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and modify  
3403 this Resource. This prevents non-terminating dependency on some other ACL Resource.  
3404 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL  
3405 Resource.

3406 An ACE2 entry is considered "currently valid" if the validity period of the ACE2 entry includes the  
3407 time of the request. The validity period in the ACE2 may be a recurring time period (e.g., daily from  
3408 1:00-2:00). Matching the resource(s) specified in a request to the resource Property of the ACE2  
3409 is defined in clause 12.2. For example, one way they can match is if the Resource URI in the  
3410 request exactly matches one of the resource references in the ACE2 entries.

3411 A request will match an ACE2 if any of the following are true:

3412 1) The ACE2 "subject" Property is of type "oic.sec.didtype" has a UUID value that matches the  
3413 "deviceuuid" Property associated with the secure session;

3414 AND the Resource of the request matches one of the resources Property of the ACE2  
3415 "oic.sec.ace2.resource-ref";

3416 AND the ACE2 is currently valid.

3417 2) The ACE2 "subject" Property is of type "oic.sec.conntype" and has the wildcard value that  
3418 matches the currently established connection type;

3419 AND the resource of the request matches one of the resources Property of the ACE2  
3420 "oic.sec.ace2.resource-ref";

3421 AND the ACE2 is currently valid.

3422 3) When Client authentication uses a certificate credential;

3423 AND one of the "roleid" values contained in the role certificate matches the "roleid" Property of  
3424 the ACE2 "oic.sec.roletype";

3425 AND the role certificate public key matches the public key of the certificate used to establish  
3426 the current secure session;

3427 AND the resource of the request matches one of the array elements of the "resources" Property  
3428 of the ACE2 "oic.sec.ace2.resource-ref";

3429 AND the ACE2 is currently valid.

3430 4) When Client authentication uses a certificate credential;

3431 AND the CoAP payload query string of the request specifies a role, which is member of the set  
3432 of roles contained in the role certificate;

3433 AND the roleid values contained in the role certificate matches the "roleid" Property of the ACE2  
3434 "oic.sec.roletype";

3435 AND the role certificate public key matches the public key of the certificate used to establish  
3436 the current secure session;

3437 AND the resource of the request matches one of the resources Property of the ACE2  
3438 "oic.sec.ace2.resource-ref";

3439 AND the ACE2 is currently valid.

3440 5) When Client authentication uses a symmetric key credential;

3441 AND one of the "roleid" values associated with the symmetric key credential used in the secure  
3442 session, matches the "roleid" Property of the ACE2 "oic.sec.roletype";

3443 AND the resource of the request matches one of the array elements of the "resources" Property  
3444 of the ACE2 "oic.sec.ace2.resource-ref";

3445 AND the ACE2 is currently valid.

3446 6) When Client authentication uses a symmetric key credential;

3447 AND the CoAP payload query string of the request specifies a role, which is contained in the  
3448 "oic.r.cred.creds.roleid" Property of the current secure session;

3449 AND CoAP payload query string of the request specifies a role that matches the "roleid"  
3450 Property of the ACE2 "oic.sec.roletype";

3451 AND the resource of the request matches one of the array elements of the "resources" Property  
3452 of the ACE2 "oic.sec.ace2.resource-ref";

3453 AND the ACE2 is currently valid.

3454 A request is granted if ANY of the 'matching' ACE2 entries contain the permission to allow the  
3455 request. Otherwise, the request is denied.

3456 There is no way for an ACE2 entry to explicitly deny permission to a resource. Therefore, if one  
3457 Device with a given role should have slightly different permissions than another Device with the  
3458 same role, they must be provisioned with different roles.

3459 The Server is required to verify that any hosted Resource has authorized access by the Client  
3460 requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the  
3461 Resource request processing should be applied securely and efficiently. See Annex A for example.

### 3462 13.6 Access Manager ACL Resource

3463 "oic.r.amacl" Resource is defined in Table 48.

3464 **Table 46 – 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

3465 Table 49 defines the Properties of "oic.r.amacl".

3466 **Table 47 – Properties of the "oic.r.amacl" Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandator y	Description
Resources	resources	oic.sec.ace2.resource-ref	array	RW	Yes	Multiple links to this host's Resources

3467 The AMS should be used to centralize management of access policy, but requires Servers to open  
3468 a connection to the AMS whenever the named Resources are accessed. See A.2 for example.

### 3469 13.7 Signed ACL Resource

3470 "oic.r.sacl" Resource is defined in Table 50.

3471

**Table 48 – 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

3472 Table 51 defines the Properties of "oic.r.sacl".

3473

**Table 49 – Properties of the "oic.r.sacl" Resource**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	State	Description
ACE List	aclist2	oic.sec.ace2	array	Yes	N/A	N/A	Access Control Entries in the ACL Resource
					N/A	RESET	Server shall set to manufacturer defaults.
					N/A	RFOTM	Set by DOTS after successful OTM
					N/A	RFPRO	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
					N/A	RFNOP	Access to NCRs is permitted after a matching ACE is found.
					N/A	SRESET	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm" Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
Signature	signature	oic.sec.sigtype	N/A	Yes	N/A	N/A	The signature over the ACL Resource

3474 Table 52 defines the Properties of "oic.sec.sigtype".

3475

**Table 50 – 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

3476 **13.8 Provisioning Status Resource**

3477 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning should  
3478 be Client-directed or Server-directed. Client-directed provisioning relies on a Client device to  
3479 determine what, how and when Server Resources should be instantiated and updated. Server-  
3480 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-  
3481 directed provisioning depends on configuration of the rowneruuid Property of the "/oic/sec/doxm",  
3482 "/oic/sec/cred" and "/oic/sec/acl2" Resources to identify the device ID of the trusted DOTS, CMS  
3483 and AMS services respectively. Furthermore, the "/oic/sec/cred" Resource should be provisioned  
3484 at ownership transfer with credentials necessary to open a secure connection with appropriate  
3485 support service.

3486 "oic.r.pstat" Resource is defined in Table 53.

3487 **Table 51 – 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

3488 Table 54 defines the Properties of "oic.r.pstat".

Table 52 – Properties of the "oic.r.pstat" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	N/A	Yes	RW		Device Onboarding State
Is Device Operational	isop	Boolean	T F	Yes	R	RESET	Server shall set to FALSE
					R	RFOTM	Server shall set to FALSE
					R	RFPRO	Server shall set to FALSE
					R	RFNOP	Server shall set to TRUE
					R	SRESET	Server shall set to FALSE
Current Mode	cm	oic.sec.dpmttype	bitmask	Yes	R		Current Mode
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes	RW		Target Mode
Operational Mode	om	oic.sec.pomttype	bitmask	Yes	R	RESET	Server shall set to manufacturer default.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by DOTS.
Supported Mode	sm	oic.sec.pomttype	bitmask	Yes	R	All states	Supported provisioning services operation modes
Device UUID	deviceuuid	String	uuid	Yes	RW	All states	[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruuid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
					RW	RFOTM	The DOTS should configure the rowneruuid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid does not refer to a valid DOTS the Server shall transition to RESET Device state.

3490 The provisioning status Resource "/oic/sec/pstat" is used to enable Devices to perform self-directed  
3491 provisioning. Devices are aware of their current configuration status and a target configuration  
3492 objective. When there is a difference between current and target status, the Device should consult

3493 the rowneruid Property of "/oic/sec/cred" Resource to discover whether any suitable provisioning  
 3494 services exist. The Device should request provisioning if configured to do so. The om Property of  
 3495 "/oic/sec/pstat" Resource will specify expected Device behaviour under these circumstances.

3496 Self-directed provisioning enables Devices to function with greater autonomy to minimize  
 3497 dependence on a central provisioning authority that should be a single point of failure in the OCF  
 3498 Security Domain.

3499 Table 55 defines the Properties of "/oic/sec/dostype".

3500 **Table 53 – Properties of the "/oic/sec/dostype" Property**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	s	UINT16	enum (0=RESET, 1=RFOTM, 2=RFPRO, 3=RFNOP, 4=SRESET	Y	R	RESET	The Device is in a hard reset state.
					RW	RFOTM	Set by DOTS after successful OTM to RFPRO.
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by CMS, AMS, DOTS after successful authentication
Pending state	p	Boolean	T   F	Y	R	All States	TRUE (1) – "s" state is pending until all necessary changes to Device resources are complete FALSE (0) – "s" state changes are complete

3501 In all Device states:

- 3502 – An authenticated and authorised Client may change the Device state of a Device by updating  
 3503 pstat.dos.s to the desired value. The allowed Device state transitions are defined in Figure 27.
- 3504 – Prior to updating "pstat.dos.s", the Client configures the Device to meet entry conditions for the  
 3505 new Device state. The SVR definitions define the entity (Client or Server) expected to perform  
 3506 the specific SVR configuration change to meet the entry conditions. Once the Client has  
 3507 configured the aspects for which the Client is responsible, it may update "pstat.dos.s". The  
 3508 Server then makes any changes for which the Server is responsible, including updating required  
 3509 SVR values, and set pstat.dos.s to the new value.
- 3510 – The "pstat.dos.p" Property is read-only by all Clients.
- 3511 – The Server sets "pstat.dos.p" to TRUE before beginning the process of updating "pstat.dos.s",  
 3512 and sets it back to FALSE when the "pstat.dos.s" change is completed.
- 3513 Any requests to update "pstat.dos.s" while "pstat.dos.p" is TRUE are denied.

3514 When Device state is RESET:

- 3515 – All SVR content is removed and reset to manufacturer default values.
- 3516 – The default manufacturer Device state is RESET.
- 3517 – NCRs are reset to manufacturer default values.
- 3518 – NCRs are inaccessible.
- 3519 – After successfully processing RESET the SRM transitions to RFOTM by setting "s" Property of  
 3520 "/oic/sec/dostype" Resource to RFOTM.

3521 When Device state is RFOTM:

- 3522 – NCRs are inaccessible.
- 3523 – Before OTM is successful, the deviceuuid Property of "/oic/sec/doxm" Resource shall be set to  
3524 a temporary non-repeated value as defined in clauses 13.2 and 13.16.
- 3525 – Before OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-only by  
3526 unauthenticated requestors
- 3527 – After the OTM is successful, the "s" Property of "/oic/sec/dostype" Resource is read-write by  
3528 authorized requestors.
- 3529 – The negotiated Device OC is used to create an authenticated session over which the DOTS  
3530 directs the Device state to transition to RFPRO.
- 3531 – If an authenticated session cannot be established the ownership transfer session should be  
3532 disconnected and SRM sets back the Device state to RESET state.
- 3533 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the  
3534 SRM asserts the OTM failed, should be disconnected, and transitions to RESET  
3535 ("/pstat.dos.s"=RESET).
- 3536 – The DOTS UPDATES the "devowneruuid" Property in the "/doxm" Resource to a non-nil UUID  
3537 value. The DOTS (or other authorized client) may update it multiple times while in RFOTM. It is  
3538 not updatable while in other device states except when the Device state returns to RFOTM  
3539 through RESET.
- 3540 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done, the  
3541 DOTS UPDATES the "owned" Property in the "/doxm" Resource to "true".

3542 When Device state is RFPRO:

- 3543 – The s Property of "/oic/sec/dostype" Resource is read-only by unauthorized requestors and  
3544 read-write by authorized requestors.
- 3545 – NCRs are inaccessible, except for Easy Setup Resources, if supported.
- 3546 – The OCF Server may re-create NCRs.
- 3547 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
- 3548 – An authorized Client may perform consistency checks on SVRs to determine which shall be re-  
3549 provisioned.
- 3550 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if  
3551 the Device has already transitioned from SRESET but consistency checks continue to fail.
- 3552 – The authorized Client sets the "/pstat.dos.s"=RFNOP.

3553 When Device state is RFNOP:

- 3554 – The "/pstat.dos.s" Property is read-only by unauthorized requestors and read-write by  
3555 authorized requestors.
- 3556 – NCRs, SVRs and core Resources are accessible following normal access processing.
- 3557 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or  
3558 RESET.

3559 When Device state is SRESET:

- 3560 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to  
3561 access or reference them.
- 3562 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These  
3563 include devowneruuid Property of the "/oic/sec/doxm" Resource,

3564 "creds":[...,{"subjectuid":<devowneruid>},...] Property of the "/oic/sec/cred" Resource and  
 3565 s Property of the "/oic/sec/dostype" Resource of "/oic/sec/pstat" Resource.

3566 – The certificates that identify and authorize the Device owner are sufficient to re-create  
 3567 minimalist "/cred" and "/doxm" resources enabling Device owner control of SRESET. If the SRM  
 3568 can't establish these Resources, then it will transition to RESET state.

3569 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as  
 3570 needed to ensure they are available for continued provisioning in RFPRO or for normal  
 3571 functioning in RFNOP.

3572 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING  
 3573 "dos.s" Property of the "/pstat" Resource with RFPRO or RFNOP values

3574 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to Device  
 3575 owner privileges.

3576 – The SRM asserts a Client-directed operational mode (e.g. "/pstat.om"=CLIENT\_DIRECTED).

3577 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.  
 3578 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning  
 3579 mode without selecting any particular value.

3580 "oic.sec.dpmttype" is defined in Table 56.

3581 **Table 54 – 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

3582 Table 57 and Table 58 define the values of "oic.sec.dpmttype".

3583 **Table 55 – 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)

3584 **Table 56 – 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

3585 The *provisioning operation mode* type is an 8-bit mask enumerating the various provisioning  
 3586 operation modes.

3587 "oic.sec.pomtype" is defined in Table 59.

3588 **Table 57 – 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

3589 Table 60 defines the values of "oic.sec.pomtype".

3590 **Table 58 – 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

3591 **13.9 Certificate Signing Request Resource**

3592 The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be  
 3593 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC  
 3594 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of  
 3595 "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr"  
 3596 Resource.

3597 "oic.r.csr" Resource is defined in Table 61.

3598 **Table 59 – 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

3599 Table 62 defines the Properties of "oic.r.csr".

**Table 60 – 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

3601 The Device chooses which public key to use, and may optionally generate a new key pair for this  
3602 purpose.

3603 In the CSR, the Common Name component of the Subject Name shall contain a string of the format  
3604 "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122. The  
3605 Common Name, and other components of the Subject Name, may contain other data. If the Device  
3606 chooses to include additional information in the Common Name component, it shall delimit it from  
3607 the UUID field by white space, a comma, or a semicolon.

3608 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to generate  
3609 a new key pair, the Device may begin generation of a key pair as a result of a RETRIEVE of this  
3610 resource. If the Device cannot immediately respond to the RETRIEVE request due to time required  
3611 to generate a key pair, the Device shall return an "operation pending" error. This indicates to the  
3612 Client that the Device is not yet ready to respond, but will be able at a later time. The Client should  
3613 retry the request after a short delay.

### 3614 **13.10 Roles Resource**

3615 The roles Resource maintains roles that have been asserted with role certificates, as described in  
3616 clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role  
3617 certificate. Servers shall only grant access to the roles information associated with the public key  
3618 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.  
3619 See 10.4.2 for how role certificates are validated.

3620 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS session  
3621 with a Client, if is not already created. The roles Resource shall only expose a secured OCF  
3622 Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long as the  
3623 (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least until the  
3624 certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of clause  
3625 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A Server  
3626 should regularly inspect the contents of the roles resource and purge contents based on a policy it  
3627 determines based on its resource constraints. For example, expired certificates, and certificates  
3628 from Clients that have not been heard from for some arbitrary period of time could be candidates  
3629 for purging.

3630 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In  
3631 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall behave  
3632 as follows. Unlisted operations are implementation specific and not reliable.

3633 1) A RETRIEVE request shall return all previously asserted roles associated with the currently  
3634 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query  
3635 parameter is not supported; all previously asserted roles associated with the currently  
3636 connected and authenticated Client's identity are returned.

- 3637 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties  
3638 included in the array as follows:
- 3639 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the  
3640 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid" value.
- 3641 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,  
3642 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed  
3643 response and a duplicate entry shall not be added to the array.
- 3644 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored  
3645 by the Server. The Server shall assign a unique "credid" value for every entry of the "roles"  
3646 array.
- 3647 3) A DELETE request without a "credid" query parameter shall remove all entries from the  
3648 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
3649 Client's identity.
- 3650 4) A DELETE request with a "credid" query parameter shall remove only the entries of the  
3651 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
3652 Client's identity and where the corresponding "credid" matches the entry.
- 3653 NOTE The "oic.r.roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3654 in ISO/IEC 30118-1:2018.
- 3655 "oic.r.roles" Resource is defined in Table 63.

3656 **Table 61 – 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

3657 Table 64 defines the Properties of "oic.r.roles".

3658 **Table 62 – 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

3659 Because "oic.r.roles" shares the "oic.sec.cred" schema with "oic.r.cred", "subjectuud" is a required Property. However,  
3660 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property  
3661 is contained in an UPDATE request to the "/oic/sec/roles" Resource.

3662 **13.11 Account Resource – moved to OCF Cloud Security document**

3663 **13.12 Account Session Resource – moved to OCF Cloud Security document**

3664 **13.13 Account Token Refresh Resource – moved to OCF Cloud Security document**

3665 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3666 The SVRs expose the security-related Properties of the Device.

3667 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to unauthenticated  
3668 (anonymous) Clients could create privacy or security concerns.

3669 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for  
3670 the "oic.r.doxxm" Resource to anonymous requesters, so that the Device can be discovered and  
3671 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the "oic.r.doxxm"  
3672 Resource to anonymous requesters, to preserve privacy.

3673 **13.15 SVRs, Discoverability and OCF Endpoints**

3674 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy Parameter  
3675 clause 7.8.2.1.2).

3676 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS) (reference  
3677 ISO/IEC 30118-1:2018, clause 10).

3678 The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM  
3679 (reference ISO/IEC 30118-1:2018, clause 10).

3680 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

3681 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

3682 Unique identifiers are a privacy consideration due to their potential for being used as a tracking  
3683 mechanism. These include the following Resources and Properties:

- 3684 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3685 – "/oic/p" Resource containing the "pi" Property.
- 3686 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3687 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous  
3688 requestors. This implies any Client Device, including those with malicious intent, are able to reliably  
3689 obtain identifiers useful for building a log of activity correlated with a specific Platform and Device.

3690 There are two strategies for privacy protection of Devices:

- 3691 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers
- 3692 2) Limit identifier persistence to make it impractical for tracking use.

3693 Both techniques can be used effectively together to limit exposure to privacy attacks.

3694 1) A Platform / Device manufacturer should specify a default ACL policy that restricts anonymous  
3695 requestors from accessing unique identifiers. An OCF Security Domain owner should modify  
3696 the ACL policy to grant access to authenticated Devices who, presumably, do not present a  
3697 privacy threat.

3698 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the  
3699 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and  
3700 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated  
3701 identifiers shall be:

- 3702 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 3703 b) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
3704 resistant

3705 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier used  
3706 to begin the onboarding process. However, attackers could obtain the value too and use it for  
3707 Device tracking throughout the Device's lifetime.

3708 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the  
3709 deviceuuid Property of the "/oic/sec/doxm" Resource to unauthenticated "/oic/res" and  
3710 "/oic/sec/doxm" Resource RETRIEVE requests when the devowneruuid Property of "/oic/sec/doxm"  
3711 Resource is the nil-UUID. The Server shall expose a new temporary non-repeated deviceuuid  
3712 Property of the "/oic/sec/doxm" Resource when the device state transitions to RESET. This ensures  
3713 the deviceuuid Property of the "/oic/sec/doxm" cannot be used to track across multiple owners.

3714 The devowneruuid Property of "/oic/sec/doxm" Resource is initialized to the nil-UUID upon entering  
3715 RESET; which is retained until being set to a non-nil-UUID value during RFOTM device state. The

3716 device shall supply a temporary, non-repeated deviceuuid Property of "/oic/sec/doxm" Resource to  
 3717 RETRIEVE requests on "/oic/sec/doxm" and "/oic/res" Resources while devowneruuid Property of  
 3718 "/oic/sec/doxm" Resource is the nil-UUID. During the OTM process the DOTS shall UPDATE  
 3719 devowneruuid Property of the "/oic/sec/doxm" Resource to a non-nil UUID value which is the trigger  
 3720 for the Device to expose its persistent or semi-persistent device identifier. Therefore, the Device  
 3721 shall supply deviceuuid Property of "/oic/sec/doxm" Resource in response to RETRIEVE requests  
 3722 while the devowneruuid Property of the "/oic/sec/doxm" Resource is a non-nil-UUID value.

3723 The DOTS or AMS may also provision an ACL policy that restricts access to the "/oic/sec/doxm"  
 3724 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent  
 3725 device identifier via the deviceuuid Property value of the "/oic/sec/doxm" Resource.

3726 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent  
 3727 or semi-persistent identifier using the "/oic/sec/cred" Resource to first establish an authenticated  
 3728 connection. This is achieved by first provisioning a "/oic/sec/cred" Resource entry that contains the  
 3729 Server's deviceuuid Property value of the "/oic/sec/doxm" Resource.

3730 The "di" Property in the "/oic/d" Resource shall mirror that of the deviceuuid Property of the  
 3731 "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the  
 3732 "/oic/d" resource such that only authenticated Clients are able to obtain the "di" Property of "/oic/d"  
 3733 Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

3734 Servers should expose a temporary, non-repeated, piid Property of "/oic/p" Resource Value upon  
 3735 entering RESET Device state. Servers shall expose a persistent value via the "piid" Property of  
 3736 "/oic/p" Property when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. An ACL  
 3737 policy on the "/oic/d" Resource should protect the "piid" Property of "/oic/p" Resource from being  
 3738 disclosed to unauthenticated requestors.

3739 Servers shall expose a temporary, non-repeated, "pi" Property value upon entering RESET Device  
 3740 state. Servers shall expose a persistent or semi-persistent platform identifier value via the "pi"  
 3741 Property of the "/oic/p" Resource when onboarding sets "devowneruuid" Property to a non-nil-UUID  
 3742 value. An ACL policy on the "/oic/p" Resource should protect the "pi" Property from being disclosed  
 3743 to unauthenticated requestors.

3744 Table 71 depicts Core Resource Properties Access Modes given various Device States.

3745 **Table 63 – 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.

3746 Four identifiers are thought to be privacy sensitive:

3747 – "/oic/d" Resource containing the "di" and "piid" Properties.  
3748 – "/oic/p" Resource containing the "pi" Property.  
3749 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.  
3750 There are three strategies for privacy protection of Devices:

- 3751 1) Apply access control to restrict read access to Resources containing unique identifiers. This  
3752 ensures privacy sensitive identifiers do not leave the Device.
- 3753 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy sensitive  
3754 identifiers are less effective for tracking and correlation.
- 3755 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can  
3756 do so.

3757 These techniques can be used to limit exposure to privacy attacks. For example:

- 3758 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent  
3759 identifiers can be created.
- 3760 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to  
3761 facilitate onboarding.
- 3762 – Persistent and semi-persistent identifiers can be encrypted before sending them to another  
3763 Device.

3764 A temporary, non-repeated identifier shall be:

- 3765 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 3766 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
3767 resistant

3768 NOTE This requirement is met through a vendor attestation certification mechanism.

3769 **13.16.2 Privacy Protecting the Device Identifiers**

3770 The "di" Property Value of the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of  
3771 the "/oic/sec/doxm" Resource. The Device should use a new, temporary non-repeated identifier in  
3772 place of the "deviceuuid" Property Value of "/oic/sec/doxm" Resource upon entering the RESET  
3773 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID  
3774 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value of  
3775 the "/oic/sec/doxm" Resource after the DOTS sets the "devowneruuid" Property to a non-nil-UUID  
3776 value. The temporary identifier should not change more frequently than once per Device state  
3777 transition to RESET.

3778 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3779 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"  
3780 Property values:
  - 3781 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di") Property  
3782 value only if responding to an authenticated requestor and the "deviceuuid" (or "di") value  
3783 is confidentiality protected .
  - 3784 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if  
3785 responding to an unauthenticated requestor.
- 3786 – The AMS should provision an ACL policy on the "/oic/sec/doxm" and "/oic/d" resources to  
3787 further protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

3788 See 13.2 for deviceuuid Property lifecycle requirements.

3789 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary  
3790 discovery requests. This is achieved by provisioning a "/oic/sec/cred" Resource entry that contains the Server's  
3791 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

### 3792 13.16.3 Privacy Protecting the Protocol Independent Device Identifier

3793 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property  
3794 Value of "/oic/d" Resource upon entering the RESET Device state. If a temporary, non-repeated  
3795 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID  
3796 value. The Device should use its persistent "piid" Property value after the DOTS sets the  
3797 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change more  
3798 frequently than once per Device state transition to RESET.

3799 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3800 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
  - 3801 – The Device should include its persistent "piid" Property value only if responding to an
  - 3802 authenticated requestor and the "piid" value is confidentiality protected.
  - 3803 – The Device should include a temporary non-repeated "piid" Property value if responding to
  - 3804 an unauthenticated requestor.
- 3805 – The AMS should provision an ACL policy on the "/oic/d" Resource to further protect the piid
- 3806 Property of "/oic/p" Resource from being disclosed unnecessarily.

### 3807 13.16.4 Privacy Protecting the Platform Identifier

3808 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property Value  
3809 of the "/oic/p" Resource upon entering the RESET Device state. This value should be exposed  
3810 while the "devowneruuid" Property has a nil UUID value. The Device should use its persistent (or  
3811 semi-persistent) "pi" Property value after the DOTS sets the "devowneruuid" Property to a non-nil-  
3812 UUID value. The temporary identifier should not change more frequently than once per Device state  
3813 transition to RESET.

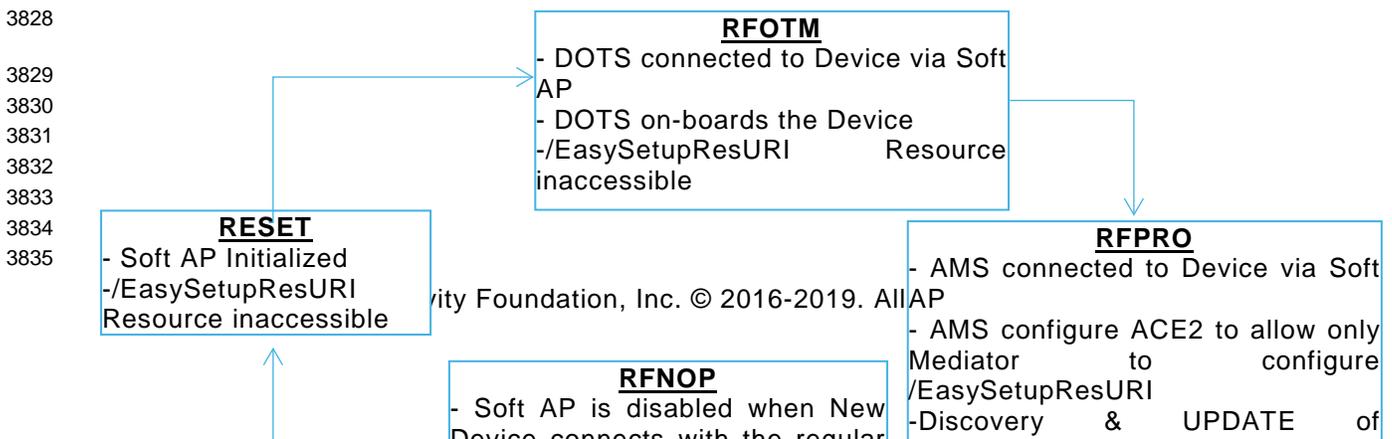
3814 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3815 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
  - 3816 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
  - 3817 responding to an authenticated requestor and the "pi" value is confidentiality protected.
  - 3818 – The Device should include a temporary non-repeated "pi" Property value if responding to
  - 3819 an unauthenticated requestor.
- 3820 – The AMS should provision an ACL policy on the "/oic/p" Resource to protect the pi Property
- 3821 from being disclosed unnecessarily.

### 3822 13.17 Easy Setup Resource Device State

3823 This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined  
3824 in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of  
3825 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup  
3826 Devices.

3827 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



3836  
3837  
3838  
3839  
3840  
3841  
3842  
3843  
3844

**Figure 38 – Example of Soft AP and Easy Setup Resource in different Device states**

3846 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's  
3847 state.

3848 While it is reasonable for a user to expect that power cycling a new Device will turn on the Soft AP  
3849 for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it is a  
3850 security risk to make this the default behaviour of a device that remains unenrolled beyond a  
3851 reasonable period after first boot.

3852 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

- 3853 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes  
3854 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for Easy  
3855 Setup.
- 3856 – If a new Device tried and failed to complete Easy Setup Enrolment immediately following the  
3857 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically for  
3858 another 30 minutes upon being power cycled, provided that the power cycle occurs within 3  
3859 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup Soft  
3860 AP directly without a factory reset, it is not necessary to turn it back on if it was on immediately  
3861 prior to power cycle, because the user obviously knows how to initiate the process manually.
- 3862 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft  
3863 AP should not turn back on for Easy Setup until another factory reset occurs, or the user initiates  
3864 the Easy Setup Soft AP directly.
- 3865 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new  
3866 Device to connect to the Enroller.
- 3867 – The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the  
3868 Enroller.
- 3869 – Once a new Device has successfully connected to the Enroller, it shall not turn the Easy Setup  
3870 Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or the user  
3871 initiates the Easy Setup Soft AP directly.
- 3872 – Just Works OTM shall not be enabled on Devices which support Easy Setup.
- 3873 – The Soft AP shall be secured (e.g. shall not expose an open AP).
- 3874 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase  
3875 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on  
3876 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.
- 3877 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the  
3878 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by an  
3879 attacker with knowledge of the Device type, model, manufacturer, or any other information  
3880 discoverable through Device's exposed interfaces.

3881 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the  
3882 "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the

3883 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller, for  
3884 use in connecting the Enrollee to the Enroller.

3885 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the  
3886 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.

3887 The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET  
3888 state. After ownership transfer process is completed with the DOTS, and the Device enters in  
3889 RFPRO Device state, the "/example/EasySetupResURI" may be Discoverable. The DOTS may be  
3890 hosted on the Mediator Device.

3891 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership  
3892 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used by  
3893 AMS for "/oic/sec/acl2" Resource provisioning in RFPRO state. The CoAPS session authentication  
3894 and encryption is already defined in the Security spec.

3895 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following  
3896 Resources to be only configurable by the Mediator Device with permission to UPDATE or  
3897 RETRIEVE access:

- 3898 – "/example/EasySetupResURI"
- 3899 – "/example/WifiConfResURI"
- 3900 – "/example/DevConfResURI"

3901 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
3902 {  
3903     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
3904     "resources": [  
3905         { "href": "/example/EasySetupResURI" },  
3906         { "href": "/example/WiFiConfResURI" },  
3907         { "href": "/example/DevConfResURI" },  
3908     ],  
3909     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
3910 }
```

3911 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to  
3912 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

3913 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE  
3914 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

## 3915 **14 Security Hardening Guidelines/ Execution Environment Security**

### 3916 **14.1 Preamble**

3917 This is an informative clause. Many TGs in OCF have security considerations for their protocols  
3918 and environments. These security considerations are addressed through security mechanisms  
3919 specified in the security documents for OCF. However, effectiveness of these mechanisms depends  
3920 on security robustness of the underlying hardware and software Platform. This clause defines the  
3921 components required for execution environment security.

### 3922 **14.2 Execution Environment Elements**

#### 3923 **14.2.1 Execution Environment Elements General**

3924 Execution environment within a computing Device has many components. To perform security  
3925 functions in a robustness manner, each of these components has to be secured as a separate  
3926 dimension. For instance, an execution environment performing AES cannot be considered secure  
3927 if the input path entering keys into the execution engine is not secured, even though the partitions  
3928 of the CPU, performing the AES encryption, operate in isolation from other processes. Different  
3929 dimensions referred to as elements of the execution environment are listed below. To qualify as a  
3930 secure execution environment (SEE), the corresponding SEE element must qualify as secure.

- 3931 – (Secure) Storage
- 3932 – (Secure) Execution engine
- 3933 – (Trusted) Input/output paths
- 3934 – (Secure) Time Source/clock
- 3935 – (Random) number generator
- 3936 – (Approved) cryptographic algorithms
- 3937 – Hardware Tamper (protection)

3938 **NOTE** Software security practices (such as those covered by OWASP) are outside scope of this document, as  
3939 development of secure code is a practice to be followed by the open source development community. This document will  
3940 however address the underlying Platform assistance required for executing software. Examples are secure boot and  
3941 secure software upgrade.

3942 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,  
3943 14.2.7.

#### 3944 **14.2.2 Secure Storage**

##### 3945 **14.2.2.1 Secure Storage General**

3946 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive  
3947 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,  
3948 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive  
3949 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both its  
3950 integrity and confidentiality be maintained.

3951 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive  
3952 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either  
3953 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication and  
3954 encryption, it must maintain its integrity against intentional or accidental alteration.

3955 A partial list of Sensitive Data is outlined in Table 72:

**Table 64 – Examples of Sensitive Data**

<b>Data</b>	<b>Integrity protection</b>	<b>Confidentiality protection</b>
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
Access Token	Yes	Yes

3957 Exact method of protection for secure storage is implementation specific, but typically combinations  
3958 of hardware and software methods are used.

#### 3959 **14.2.2.2 Hardware Secure Storage**

3960 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric  
3961 and asymmetric private keys, access credentials, and personal private data. Hardware secure  
3962 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes  
3963 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

3964 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides  
3965 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or electronic  
3966 attacks. It is not necessary to prevent the attacks themselves, but an attempted attack should not  
3967 result in an unauthorized entity successfully retrieving Sensitive Data.

3968 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data  
3969 from attacks that include but are not limited to:

- 3970 1) Physical decapping of chip packages to optically read NVRAM contents
- 3971 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 3972 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit patterns  
3973 of Critical Sensitive Data
- 3974 4) Use of malicious software or firmware to read memory contents at rest or in transit within a  
3975 microcontroller
- 3976 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

#### 3977 **14.2.2.3 Software Storage**

3978 It is generally NOT recommended to rely solely on software and unsecured memory to store  
3979 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and encryption  
3980 keys should be housed in hardware secure storage whenever possible.

3981 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable  
3982 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

#### 3983 **14.2.2.4 Additional Security Guidelines and Best Practices**

3984 Some general practices that can help ensure that Sensitive Data is not compromised by various  
3985 forms of security attacks:

- 3986 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG  
3987 used for authentication challenges can substantially degrade security strength. For this reason,  
3988 it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source be used  
3989 for all authentication challenges.
- 3990 2) Secure download and boot – To prevent the loading and execution of malicious software, where  
3991 it is practical, it is recommended that Secure Download and Secure Boot methods that  
3992 authenticate a binary's source as well as its contents be used.
- 3993 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered  
3994 unsecure and shall not be used for any security-related function:
  - 3995 a) SHA-1
  - 3996 b) MD5
  - 3997 c) RC4
  - 3998 d) RSA 1024
- 3999 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is  
4000 stored in Secure Storage, any use of that data that requires its transmission out of that Secure  
4001 Storage should be encrypted to prevent eavesdropping by malicious software within an  
4002 MCU/MPU.
- 4003 5) It is recommended to avoid using wildcard in Subject Id ("\*"), when setting up "oic.r.cred"  
4004 Resource entries, since this opens up an identity spoofing opportunity.
- 4005 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device  
4006 meets security requirements for its intended uses. As an example, IoTivity is a reference  
4007 implementation intended to be used as a basis for a product, but IoTivity has not undergone  
4008 3rd party security review, penetration testing, etc. Any Device based on IoTivity should undergo  
4009 appropriate penetration testing and security review prior to sale or deployment.
- 4010 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to  
4011 consumers. Changes should be made to a public and accessible website. Expectations should  
4012 be clear as to what will be supported and for how long the Device vendor expects to support  
4013 security updates to the software, operating system, drivers, networking, firmware and hardware  
4014 of the device.
- 4015 8) Device vendor has not implemented test or debug interfaces on the Device which are operable  
4016 or which can be enabled which might present an attack vector on the Device which circumvents  
4017 the interface-level security or access policies of the Device.
- 4018 9) Device vendor understands that if an application running on the Device has access to  
4019 cryptographic elements such as the private keys or Ownership Credential, then those elements  
4020 have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or a Device  
4021 with access to the Internet beyond the local network, the execution of critical functions should  
4022 take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4023 10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management or  
4024 access, or other security-critical function, should be sufficiently unique (do not duplicate  
4025 passphrases. The creation of these passphrases or PINS should not be algorithmically  
4026 deterministic nor should they use insufficient entropy in their creation.
- 4027 11) Ensure that there are no remaining "VENDOR\_TODO" items in the source code.

4028 12) If the implementation of this document uses the "Just Works" onboarding method, understand  
4029 that there is a man-in-the-middle vulnerability during the onboarding process where a malicious  
4030 party could intercept messages between the device being onboarded and the OBT and could  
4031 persist, acting as an intermediary with access to message traffic, during the lifetime of that  
4032 onboarded device. The recommended best practice would be to use an alternate ownership  
4033 transfer method (OTM) instead of "Just Works".

4034 13) It is recommended that at least one static and dynamic analysis tool<sup>1</sup> be applied to any  
4035 proposed major production release of the software before its release, and any vulnerabilities  
4036 resolved.

4037 14) To avoid a malicious device being able to covertly join an OCF Security Domain, implementers  
4038 of any OBT may eliminate completely autonomous sequences where a device is brought into  
4039 the OCF Security Domain without any authorization by the owner. Consider either including a  
4040 confirmation with the OCF Security Domain owner/operator (e.g. "Do you want to add  
4041 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'? Yes/No/Cancel?") or a confirmation  
4042 with a security policy (e.g. an enterprise policy where the OCF Security Domain admin can  
4043 bulk-onboard devices).

#### 4044 **14.2.3 Secure execution engine**

4045 Execution engine is the part of computing Platform that processes security functions, such as  
4046 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine requires  
4047 the following

4048 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This  
4049 includes isolation of CPU caches, and all of execution elements that needed to be considered  
4050 as part of trusted (crypto) boundary.

4051 – Isolation of data paths into and out of execution engine. For instance, both unencrypted but  
4052 sensitive data prior to encryption or after decryption, or cryptographic keys used for  
4053 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

#### 4054 **14.2.4 Trusted input/output paths**

4055 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be protected.  
4056 This includes paths into and out secure execution engine and secure memory.

4057 Path protection can be both hardware based (e.g. use of a privileged bus) or software based (using  
4058 encryption over an untrusted bus).

#### 4059 **14.2.5 Secure clock**

4060 Many security functions depend on time-sensitive credentials. Examples are time stamped  
4061 Kerberos tickets, OAuth tokens, X.509 certificates, OSCP response, software upgrades, etc. Lack  
4062 of secure source of clock can mean an attacker can modify the system clock and fool the validation  
4063 mechanism. Thus an SEE needs to provide a secure source of time that is protected from tampering.  
4064 Trustworthiness from security robustness standpoint is not the same as accuracy. Protocols such  
4065 as NTP can provide rather accurate time sources from the network, but are not immune to attacks.  
4066 A secure time source on the other hand can be off by seconds or minutes depending on the time-  
4067 sensitivity of the corresponding security mechanism. Secure time source can be external as long  
4068 as it is signed by a trusted source and the signature validation in the local Device is a trusted  
4069 process (e.g. backed by secure boot).

#### 4070 **14.2.6 Approved algorithms**

4071 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and  
4072 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by  
4073 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only  
4074 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic

---

<sup>1</sup> A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4075 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are  
4076 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms (even  
4077 if they deemed stronger by some parties) must be considered non-approved.

4078 The set of algorithms to be considered for approval are algorithms for

- 4079 – Hash functions
- 4080 – Signature algorithms
- 4081 – Encryption algorithms
- 4082 – Key exchange algorithms
- 4083 – Pseudo Random functions (PRF) used for key derivation

4084 This list will be included in this or a separate security robustness rules document and must be  
4085 followed for all security specifications within OCF.

#### 4086 **14.2.7 Hardware tamper protection**

4087 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not  
4088 requirements) regarding tamper protection for cryptographic module

- 4089 – Production-grade (lowest level): this means components that include conformal sealing coating  
4090 applied over the module's circuitry to protect against environmental or other physical damage.  
4091 This does not however require zeroization of secret material during physical maintenance. This  
4092 definition is borrowed from FIPS 140-2 security level 1.
- 4093 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,  
4094 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from FIPS  
4095 140-2 security level 2.
- 4096 – Tamper resistance (highest level), this means there is a response to physical tempering that  
4097 typically includes zeroization of sensitive material on the module. This definition is borrowed  
4098 from FIPS 140-2 security level 3.

4099 It is difficult of specify quantitative certification test cases for accreditation of these levels. Content  
4100 protection regimes usually talk about different tools (widely available, specialized and professional  
4101 tools) used to circumvent the hardware protections put in place by manufacturing. If needed, OCF  
4102 can follow that model, if and when OCF engage in distributing sensitive key material (e.g. PKI) to  
4103 its members.

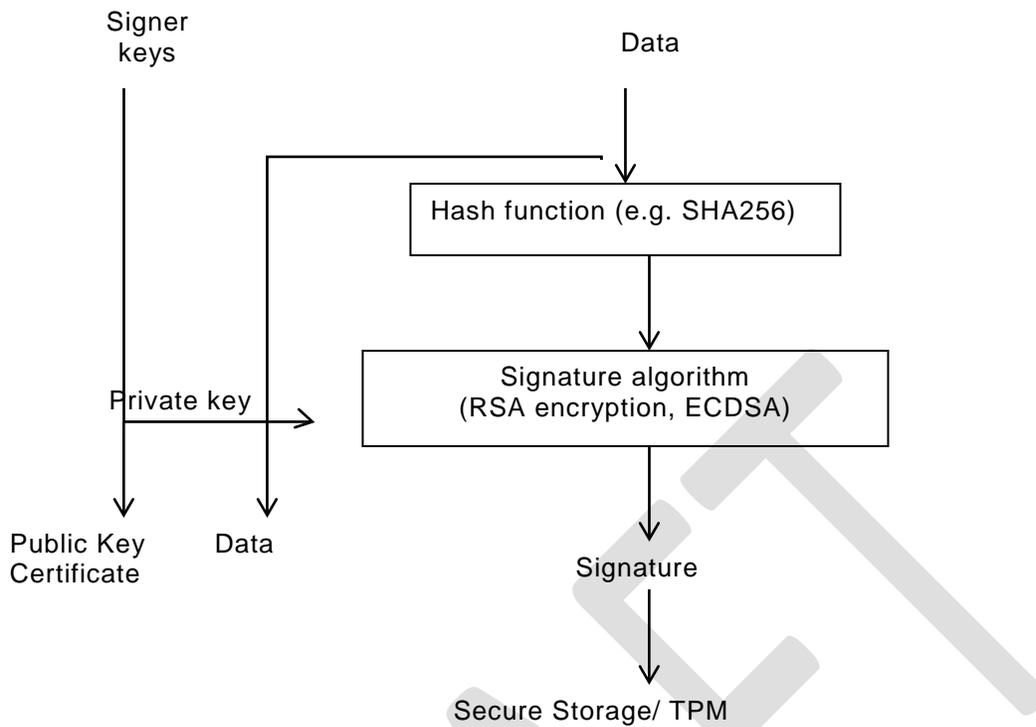
### 4104 **14.3 Secure Boot**

#### 4105 **14.3.1 Concept of software module authentication**

4106 In order to ensure that all components of a Device are operating properly and have not been  
4107 tampered with, it is best to ensure that the Device is booted properly. There may be multiple stages  
4108 of boot. The end result is an application running on top an operating system that takes advantage  
4109 of memory, CPU and peripherals through drivers.

4110 The general concept is that each software module is invoked only after cryptographic integrity  
4111 verification is complete. The integrity verification relies on the software module having been hashed  
4112 (e.g. SHA\_1, SHA\_256) and then signed with a cryptographic signature algorithm with (e.g. RSA),  
4113 with a key that only a signing authority has access to.

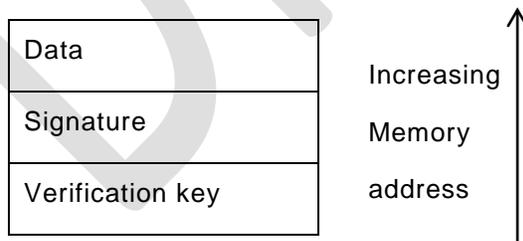
4114 Figure 40 depicts software module authentication.



**Figure 39 – Software Module Authentication**

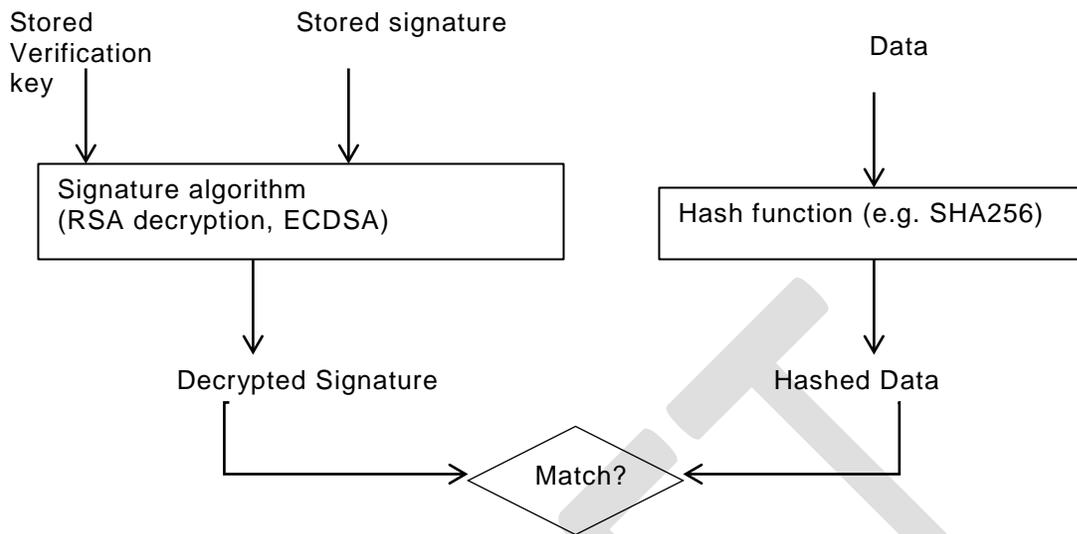
4115  
 4116 After the data is signed with the signer’s signing key (a private key), the verification key (the public  
 4117 key corresponding to the private signing key) is provided for later verification. For lower level  
 4118 software modules, such as bootloaders, the signatures and verification keys are inserted inside  
 4119 tamper proof memory, such as one-time programmable memory or TPM. For higher level software  
 4120 modules, such as application software, the signing is typically performed according to the PKCS#7  
 4121 format IETF RFC 2315, where the signedData format includes both indications for signature  
 4122 algorithm, hash algorithm as well as the signature verification key (or certificate). Secure boot does  
 4123 not require use of PKCS#7 format.

4124 Figure 41 depicts verification software module.



**Figure 40 – Verification Software Module**

4125  
 4126 As shown in Figure 42. the verification module first decrypts the signature with the verification key  
 4127 (public key of the signer). The verification module also calculates a hash of the data and then  
 4128 compares the decrypted signature (the original) with the hash of data (actual) and if the two values  
 4129 match, the software module is authentic.



4130 **Figure 41 – Software Module Authenticity**

4131 **14.3.2 Secure Boot process**

4132 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/  
 4133 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to  
 4134 find out where the boot code is and then run the boot code (second-stage boot loader). The second  
 4135 stage bootloader is typically the process that loads the operating system (Kernel) and transfers the  
 4136 execution to the where the Kernel code is. Once the Kernel starts, it may load external Kernel  
 4137 modules and drivers.

4138 When performing a secure boot, it is required that the integrity of each boot loader is verified before  
 4139 executing the boot loader stage. As mentioned, while the signature and verification key for the  
 4140 lowest level bootloader is typically stored in tamper-proof memory, the signature and verification  
 4141 key for higher levels should be embedded (but attached in an easily accessible manner) in the data  
 4142 structures software.

4143 **14.3.3 Robustness Requirements**

4144 **14.3.3.1 Robustness General**

4145 To qualify as high robustness secure boot process, the signature and hash algorithms shall be one  
 4146 of the approved algorithms, the signature values and the keys used for verification shall be stored  
 4147 in secure storage and the algorithms shall run inside a secure execution environment and the keys  
 4148 shall be provided the SEE over trusted path.

4149 **14.3.3.2 Next steps**

4150 Develop a list of approved algorithms and data formats

4151 **14.4 Attestation**

4152 **14.5 Software Update**

4153 **14.5.1 Overview:**

4154 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;  
 4155 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and  
 4156 end-of-life stages for the Device remain outstanding. It is possible for the Device to require update

4157 during any of these stages, although the most likely times are during onboarding, regular operation  
 4158 and maintenance. The aspects of the software include, but are not limited to, firmware, operating  
 4159 system, networking stack, application code, drivers, etc.

4160 **14.5.2 Recognition of Current Differences**

4161 Different manufacturers approach software update utilizing a collection of tools and strategies:  
 4162 over-the-air or wired USB connections, full or partial replacement of existing software, signed and  
 4163 verified code, attestation of the delivery package, verification of the source of the code, package  
 4164 structures for the software, etc.

4165 It is recommended that manufacturers review their processes and technologies for compliance with  
 4166 industry best-practices that a thorough security review of these takes place and that periodic review  
 4167 continue after the initial architecture has been established.

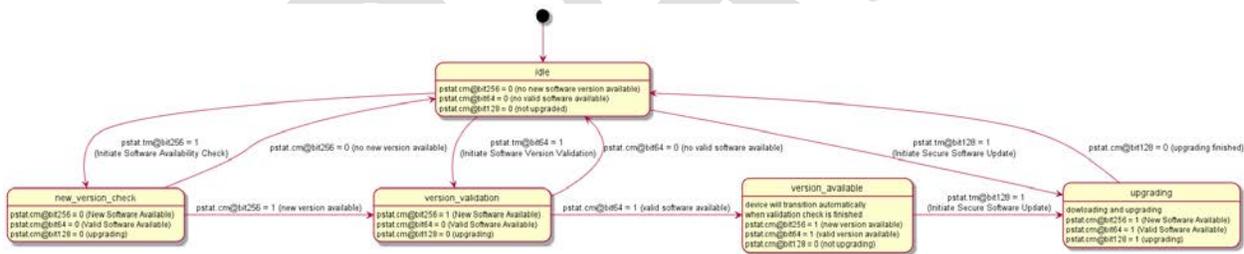
4168 This document applies to software updates as recommended to be implemented by OCF Devices;  
 4169 it does not have any bearing on the above-mentioned alternative proprietary software update  
 4170 mechanisms. The described steps are being triggered by an OCF Client, the actual implementation  
 4171 of the steps and how the software package is downloaded and upgraded is vendor specific.

4172 The triggers that can be invoked from OCF clients can perform:

- 4173 1) Check if new software is available
- 4174 2) Download and verify the integrity of the software package
- 4175 3) Install the verified software package

4176 The triggers are not sequenced, each trigger can be invoked individually.

4177 The state of the transitions of software update is in Figure 43.



4178  
 4179 **Figure 42 – State transitioning diagram for software download**

4180  
 4181 **Table 65 – 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

4182  
 4183 **14.5.2.1 Checking availability of new software**

4184 Setting the Initiate Software Availability Check bit in the "/oic/sec/pstat.tm" Property (see Table 54  
 4185 of clause 13.8) indicates a request to initiate the process to check if new software is available, e.g.  
 4186 the process whereby the Device checks if a newer software version is available on the external

4187 endpoint. Once the Device has determined if an newer software version is available, it sets the  
4188 Initiate Software Availability Check bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE), indicating  
4189 that new software is available or to 0 (FALSE) if no newer software version is available, See also  
4190 Table 73 where the bits in property TM indicates that the action is initiated and the CM bits are  
4191 indicating the result of the action. The Device receiving this trigger is not downloading and not  
4192 validating the software to determine if new software is available. The version check is determined  
4193 by the current software version and the software version on the external endpoint. The  
4194 determination if a software package is newer is vendor defined.

### 4195 **14.5.3 Software Version Validation**

4196 Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see Table 54  
4197 defines the Properties of "oic.r.pstat".

4198 Table 54 of 13.8) indicates a request to initiate the software version validation process, the process  
4199 whereby the Device validates the software (including firmware, operating system, Device drivers,  
4200 networking stack, etc.) against a trusted source to see if, at the conclusion of the check, the  
4201 software update process will need to be triggered (see clause 14.5.4). When the Initiate Software  
4202 Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the  
4203 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a  
4204 software version check. Once the Device has determined if a valid software is available, it sets the  
4205 Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if an update  
4206 is available or 0 (FALSE) if no update is available. To signal completion of the Software Version  
4207 Validation process, the Device sets the Initiate Software Version Validation bit in the  
4208 "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation bit of  
4209 "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation process. The  
4210 Software Version Validation process can download the software from the external endpoint to verify  
4211 the integrity of the software package.

### 4212 **14.5.4 Software Update**

4213 Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 54 of  
4214 clause 13.8) indicates a request to initiate the software update process. When the Initiate Secure  
4215 Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the  
4216 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a  
4217 software update process. Once the Device has completed the software update process, it sets the  
4218 Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if/when the  
4219 software was successfully updated or 0 (FALSE) if no update was performed. To signal completion  
4220 of the Secure Software Update process, the Device sets the Initiate Secure Software Update bit in  
4221 the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure Software Update bit of  
4222 "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the update process.

#### 4223 **14.5.4.1 State of Device after software update**

4224 The state of all resources implemented in the Device should be the same as after boot, meaning  
4225 that the software update is not resetting user data and retaining a correct state.

4226 User data of a Device is defined as:

- 4227 – Retain the SVR states, e.g. the on boarded state, registered clients.
- 4228 – Retain all created resources
- 4229 – Retain all stored data of a resource
- 4230 – For example the preferences stored for the brewing resource ("oic.r.brewing").

### 4231 **14.5.5 Recommended Usage**

4232 The Initiate Secure Software Update bit of "/oic/sec/pstat.tm" should only be set by a Client after  
4233 the Initiate Software Version Validation check is complete.

4234 The process of updating Device software may involve state changes that affect the Device  
4235 Operational State ("/oic/sec/pstat.dos"). Devices with an interest in the Device(s) being updated  
4236 should monitor "/oic/sec/pstat.dos" and be prepared for pending software update(s) to affect Device  
4237 state(s) prior to completion of the update.

4238 The Device itself may indicate that it is autonomously initiating a software version check/update or  
4239 that a check/update is complete by setting the "pstat.tm" and "pstat.cm" Initiate Software Version  
4240 Validation and Secure Software Update bits when starting or completing the version check or  
4241 update process. As is the case with a Client-initiated update, Clients can be notified that an  
4242 autonomous version check or software update is pending and/or complete by observing pstat  
4243 resource changes.

4244 The "oic.r.softwareupdate" Resource Type specifies additional features to control the software  
4245 update process see core specification.

#### 4246 **14.6 Non-OCF Endpoint interoperability**

#### 4247 **14.7 Security Levels**

4248 Security Levels are a way to differentiate Devices based on their security criteria. This need for  
4249 differentiation is based on the requirements from different verticals such as industrial and health  
4250 care and may extend into smart home. This differentiation is distinct from Device classification  
4251 (e.g. IETF RFC 7228)

4252 These categories of security differentiation may include, but is not limited to:

- 4253 1) Security Hardening
- 4254 2) Identity Attestation
- 4255 3) Certificate/Trust
- 4256 4) Onboarding Technique
- 4257 5) Regulatory Compliance
  - 4258 a) Data at rest
  - 4259 b) Data in transit
- 4260 6) Cipher Suites – Crypto Algorithms & Curves
- 4261 7) Key Length
- 4262 8) Secure Boot/Update

4263 In the future security levels can be used to define interoperability.

4264 The following applies to the OCF Security Specification 1.1:

4265 The current document does not define any other level beyond Security Level 0. All Devices will be  
4266 designated as Level 0. Future versions may define additional levels.

4267 Additional comments:

- 4268 – The definition of a given security level will remain unchanged between versions of the document.
- 4269 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 4270 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of the  
4271 higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and a later  
4272 document version defines a security level 1, the Device could be evaluated and classified as  
4273 level 1 if it meets level 1 requirements).
- 4274 – The security levels may need to be visible to the end user.

4275 **14.8 Security Profiles**

4276 **14.8.1 Security Profiles General**

4277 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This need  
4278 for differentiation is based on the requirements from different verticals such as industrial and health  
4279 care and may extend into smart home. This differentiation is distinct from device classification (e.g.  
4280 IETF RFC 7228)

4281 These categories of security differentiation may include, but is not limited to:

4282 1) Security Hardening and assurances criteria

4283 2) Identity Attestation

4284 3) Certificate/Trust

4285 4) Onboarding Technique

4286 5) Regulatory Compliance

4287 a) Data at rest

4288 b) Data in transit

4289 6) Cipher Suites – Crypto Algorithms & Curves

4290 7) Key Length

4291 8) Secure Boot/Update

4292 Each Security Profile definition must specify the version or versions of the OCF Security  
4293 Specification(s) that form a baseline set of normative requirements. The profile definition may  
4294 include security requirements that supersede baseline requirements (not to relax security  
4295 requirements).

4296 Security Profiles have the following properties:

4297 – A given profile definition is not specific to the version of the document that defines it. For  
4298 example, the profile may remain constant for subsequent OCF Security Specification versions.

4299 – A specific OCF Device and platform combination may be used to satisfy the security profile.

4300 – Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles  
4301 simultaneously.

4302 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found to  
4303 satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the document,  
4304 and a later document version defines a security profile Black, the device could be evaluated  
4305 and classified as profile Black if it meets profile Black requirements).

4306 – A machine-readable representation of compliance results specifically describing profiles  
4307 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or  
4308 manifest may contain security profiles attributes).

4309 **14.8.2 Identification of Security Profiles (Normative)**

4310 **14.8.2.1 Security Profiles in Prior Documents**

4311 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles  
4312 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in  
4313 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use  
4314 the OCF Security Specification version to characterize expected security behaviour.

4315 **14.8.2.2 Security Profile Resource Definition**

4316 The "oic.sec.sp" Resource is used by the OCF Device to show which OCF Security Profiles the  
4317 OCF Device is capable of supporting and which are authorized for use by the OCF Security Domain

4318 owner. Properties of the Resource identify which OCF Security Profile is currently operational. The  
 4319 ocfSecurityProfileOID value type shall represent OID values and may reference an entry in the form  
 4320 of strings (UTF-8).

4321 "oic.sec.sp" Resource is defined in Table 74.

4322 **Table 66 – 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

4323 Table 75 defines the Properties of "oic.sec.sp".

4324 **Table 67 – 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")

4325 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or  
 4326 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4327 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)  
 4328 private(4) enterprise(1) OCF(51414) }

4329  
 4330 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4331  
 4332 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4333  
 4334 sp-undefined ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }

4335 --The Security Profile is not specified

4336 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }

4337 --This specifies the OCF Baseline Security Profile(s)

4338 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }

4339 --This specifies the OCF Black Security Profile(s)

4340 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }

4341 --This specified the OCF Blue Security Profile(s)

4342 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }

4343 --This specifies the OCF Purple Security Profile(s)

4344  
 4345 --versioned Security Profiles

4346 sp-undefined-v0 ::= ocfSecurityProfileOID (id-sp-undefined 0)

4347 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4348 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}

4349 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4350 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}

4351 --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"

4352 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}

4353 --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"

4354 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}

4355 --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"

4356  
 4357 ocfSecurityProfileOID ::= UTF8String

4358

### 4359 **14.8.3 Security Profiles**

#### 4360 **14.8.3.1 Security Profiles General**

4361 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to  
4362 the Security Profile clauses for additional details).

4363 The OCF Conformance criteria may require vendor attestation that establishes the expected  
4364 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific  
4365 requirements).

#### 4366 **14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)**

4367 The Security Profile "sp-unspecified-v0" is reserved for future use.

#### 4368 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4369 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where  
4370 the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the  
4371 "supportedprofiles" Property of the "/oic/sec/sp" Resource.

4372 It indicates the OCF Device satisfies the normative security requirements for this document.

4373 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-  
4374 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other  
4375 profiles.

4376 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to  
4377 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

#### 4378 **14.8.3.4 Security Profile Black (sp-black-v0)**

##### 4379 **14.8.3.4.1 Black Profile General**

4380 The need for Security Profile Black v0 is to support devices and manufacturers who wish to certify  
4381 their devices meeting this specific set of security criteria. A Device may satisfy the Black  
4382 requirements as well as requirements of other profiles, the Black Security Profile is not necessarily  
4383 mutually exclusive with other Security Profiles unless those requirements conflict with the explicit  
4384 requirements of the Black Security Profile.

##### 4385 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4386 Security Profile Black devices could include any device a manufacturer wishes to certify at this  
4387 profile, but healthcare devices and industrial devices with additional security requirements are the  
4388 initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or devices  
4389 with exceptional profiles of trust bestowed upon them, may wish to certify at this profile; these types  
4390 of devices may include, but are not limited to the following:

- 4391 – Bridges (Mapping devices between ecosystems handling virtual devices from different  
4392 ecosystems)
- 4393 – Resource Directories (Devices trusted to manage OCF Security Domain resources)
- 4394 – Remote Access (Devices which have external access but can also act within the OCF Security  
4395 Domain)
- 4396 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)
- 4397 – Industrial Devices (Devices with advanced management, security and attestation requirements)

4398 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4399 Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security  
4400 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4401 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4402 – Support for AES 128 encryption for data at rest and in transit.
- 4403 – Hardening minimums: manufacturer assertion of secure credential storage
- 4404 – In 13) in enumerated item #10 "The "/oic/sec/cred" Resource should contain credential(s) if  
4405 required by the selected OTM" is changed to require the credential be stored: "The  
4406 "/oic/sec/cred" Resource shall contain credential(s)."
- 4407 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4408 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by  
4409 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4410 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-  
4411 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4412 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to  
4413 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4414 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework  
4415 described in the supporting documents:

- 4416 – Certificate Profile (See 9.4.2)
- 4417 – Certificate Policy (see Certificate Policy document:  
4418 <https://openconnectivity.org/specs/OCF%20Certificate%20Policy.pdf>)

4419 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4420 **14.8.3.5.1 Blue Profile General**

4421 The Security Profile Blue is used when manufacturers issue platform certificates for platforms  
4422 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is  
4423 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF  
4424 Security Domain owners evaluate manufacturer supplied certificates and attributed data to  
4425 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.  
4426 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may  
4427 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4428 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting  
4429 Criteria defined by OCF.

4430 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4431 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from  
4432 OCF Device vendor and where platform vendors may implement trusted platforms that may conform  
4433 to industry standards defining trusted platforms. The OCF Security Profile Blue specifies  
4434 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance criteria  
4435 produced by OCF conformance operations. The OCF Security Domain owner evaluates these data  
4436 when an OCF Device is onboarded into the OCF Security Domain. Based on this evaluation the  
4437 OCF Security Domain owner determines which Security Profile may be applied during OCF Device  
4438 operation. All OCF Device types may be considered for evaluation using the OCF Security Profile  
4439 Blue.

4440 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4441 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for this  
4442 document version are satisfied and the following additional criteria are satisfied.

4443 OCF Blue profile defines the following OCF Device quality assurances:

- 4444 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF Device  
4445 was hosted on one or more platforms that satisfies OCF Blue platform security assurances and  
4446 platform security and privacy functionality requirements.
- 4447 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF and  
4448 published by OCF in a machine readable format.
- 4449 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned  
4450 signing key.
- 4451 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4452 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by the  
4453 ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4454 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in  
4455 its certificate.
- 4456 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF  
4457 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's  
4458 "securityprofiles" field.

4459 OCF Blue profile defines the following OCF Device security functionality:

- 4460 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage  
4461 functions are hardened by the platform.
- 4462 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials using  
4463 the "/oic/sec/cred" Resource where the "credusage" Property contains the value  
4464 "oic.sec.cred.mfgcert".
- 4465 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an IEEE802.1AR  
4466 IDevID and should verify the "TCG Endorsement Key Credential". All TCG-defined  
4467 manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the  
4468 "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to  
4469 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4470 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See  
4471 NIST SP 800-57).
- 4472 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST SP  
4473 800-57).
- 4474 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST  
4475 SP 800-57).
- 4476 – OCF Device(s) should protect the "/oic/sec/cred" resource using the platform provided secure  
4477 storage.
- 4478 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned  
4479 certificates) using platform provided secure storage.
- 4480 – OCF Device(s) should check certificate revocation status for locally issued certificates.
- 4481 – OCF OBTs (aka DOTS) shall check certificate revocation status for all certificates in  
4482 manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation fails  
4483 and the connection is refused. The DOTS may disregard revocation status results if unavailable.

4484 OCF Blue profile defines the following platform security assurances:

- 4485 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage  
4486 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL  
4487 Level 2.

4488 – Platforms implementing trusted platform functionality should be evaluated with a minimum  
4489 Common Criteria EAL Level 1.

4490 OCF Blue profile defines the following platform security and privacy functionality:

4491 – The Platform shall implement cryptographic service provider (CSP) functionality.

4492 – Platform CSP functionality shall include cryptographic algorithms, random number generation,  
4493 secure time.

4494 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST SP  
4495 800-57).

4496 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See  
4497 NIST SP 800-57).

4498 – Platforms hosting OCF Device(s) should implement a platform identifier following IEEE802.1AR  
4499 or Trusted Computing Group(TCG) specifications.

4500 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF Device(s)  
4501 should supply TCG-defined manufacture certificates; also known as "TCG Endorsement Key  
4502 Credential" (which complies with IETF RFC 5280) and "TCG Platform Credential" (which  
4503 complies with IETF RFC 5755).

4504 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,  
4505 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

4506 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to  
4507 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

4508 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"  
4509 Property to one of the other values found in the "supportedprofiles" Property.

#### 4510 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**

4511 Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause  
4512 14.8.2 must support following minimum requirements

4513 – Hardening minimums: secure credential storage, software integrity validation, secure update.

4514 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension  
4515 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-  
4516 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"

4517 – The OCF Device shall include a X.509v3 OCFPLAttributes Extension (clause 9.4.2.2.7) in its  
4518 End-Entity Certificate when manufacturer certificate is used.

4519 Security Profile Purple has following optional security hardening requirements that the device can  
4520 additionally support.

4521 – Hardening additions: secure boot, hardware backed secure storage

4522 – The OCF Device shall include a X.509v3 OCFSecurityClaims Extension (clause 9.4.2.2.6) in its  
4523 End-Entity Certificate and it shall include corresponding OIDs to the hardening additions  
4524 implemented and attested by the vendor. If there is no additional support for hardening  
4525 requirements, X.509v3 OCFSecurityClaims Extension shall be omitted.

4526 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism  
4527 for security critical executables such as cryptographic modules or secure service applications, and  
4528 they should be validated before the execution. The key used for validating the integrity must be  
4529 pinned at the least to the validating software module.

4530 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4531 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM) to  
4532 be executed by the processor on power-on, and secure boot parameters to be provisioned by  
4533 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the  
4534 security critical executables and stop the boot process if any integrity of them is compromised.

4535 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile  
4536 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic  
4537 attacks.

4538 More details on security hardening guidelines for software integrity validation, secure boot, secure  
4539 update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4540 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA Vetting  
4541 Criteria defined by OCF.

4542 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-purple-  
4543 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other profiles.

4544 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to  
4545 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

DRAFT

4546 **15 Device Type Specific Requirements**

4547 **15.1 Bridging Security**

4548 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4549 The Bridge shall go through OCF ownership transfer as any other onboarder would.

4550 The software of an Bridge shall be field updatable. (This requirement need not be tested but can  
4551 be certified via a vendor declaration.)

4552 Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be provisioned  
4553 as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged Devices are  
4554 treated the same way as physical Devices. They are entities that have to be provisioned in their  
4555 network.

4556 Each VOD shall implement the behaviour required by ISO/IEC 30118-1:2018 and this document.  
4557 Each VOD shall perform authentication, access control, and encryption according to the security  
4558 settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security  
4559 requirements of the Bridged Protocol.

4560 In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall  
4561 use appropriate ecosystem-specific security options for communication between the Virtual Bridged  
4562 Devices instantiated by the Bridge and Bridged Devices. This security shall include mutual  
4563 authentication, and encryption and integrity protection of messages in the bridged ecosystem.

4564 A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see  
4565 clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which  
4566 instantiated that VOD.

4567 A VOD may authenticate itself to the OCF Cloud (see clause 10.5.2) using the Manufacturer  
4568 Certificate and corresponding private key of the Bridge which instantiated that VOD.

4569 A Bridge and the VODs created by that Bridge shall operate as independent Devices, with the  
4570 following exceptions:

- 4571 – If a Bridge creates a VOD while the Bridge is in an Unowned State, then the VOD shall be  
4572 created in an Unowned State.
- 4573 – An Unowned VOD shall not accept DTLS connection attempts nor TLS connection attempts nor  
4574 any other requests, including discovery requests, while the Bridge (that created that VOD) is  
4575 Unowned.
- 4576 – At any time when a Bridge is transitioning from Owned to Unowned State, all Unowned VODs  
4577 (created by that Bridge prior to the transition) shall drop any existing TLS and/or DTLS  
4578 connections.
- 4579 – At any time when a Bridge is transitioning from Unowned to Owned State, the Bridge shall  
4580 trigger all Unowned VODs (created by that Bridge prior to the transition) to become accessible  
4581 in RFOTM state, with internal state as if the VOD has just transitioned from RESET to RFOTM.
- 4582 – If a Bridge creates a VOD while the Bridge is in an Owned State, then the VOD shall become  
4583 accessible in RFOTM state, with internal state as if the VOD has just transitioned from RESET  
4584 to RFOTM.

4585 Table 76 intends to clarify this behaviour.

4586  
4587

**Table 68 – 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

4588 The "vods" Property of the "oic.r.vodlist" Resource on a Bridge reflects the details of all currently  
4589 Owned VODs which have been created by that Bridge since the most recent hardware reset (if any)  
4590 of the Bridge Platform (which removes all the created VODs), regardless of whether the VODs have  
4591 the same owner as the Bridge or not. The entries in the "vods" Property are added and removed  
4592 according to the following criteria:

- 4593 – Whenever a VOD created by a Bridge transitions from being Unowned to being Owned, then  
4594 an entry for that VOD shall be added to the "vods" Property of the "oic.r.vodlist" Resource of  
4595 that Bridge.
- 4596 – Whenever a VOD created by a Bridge transitions from being Owned to being Unowned, then  
4597 entry for that VOD shall be removed from the "vods" Property of the "oic.r.vodlist" Resource of  
4598 that Bridge. If that Bridge is currently in Unowned state, then the "oic.r.vodlist" Resource is not  
4599 accessible, and the entry for that VOD shall be removed from the "vods" Property before or  
4600 during the transition of that Bridge to the Owned state.
- 4601 – All other modifications of the list are not allowed.

4602 A Bridge shall only expose a secure OCF Endpoint for the "oic.r.vodlist" Resource.

4603 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4604 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4605 For AllJoyn translator, an OCF OBT shall be able to block the communication of all OCF Devices  
4606 with all Bridged Devices that don't communicate securely with the Bridge, by using the Bridge  
4607 Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

4608 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4609 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4610 communicate securely with the Bridge.

4611 **15.1.2.3 Additional Security Requirements specific to the oneM2M Protocols**

4612 The Bridge shall implement oneM2M application access control as defined in the oneM2M Release  
4613 3 Specifications.

4614 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4615 communicate securely with the Bridge.

4616 **15.1.2.4 Additional Security Requirements specific to the U+ Protocol**

4617 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4618 communicate securely with the Bridge.

4619 **15.1.2.5 Additional Security Requirements specific to the Z-Wave Protocol**

4620 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4621 communicate securely with the Bridge.

4622 **15.1.2.6 Additional Security Requirements specific to the Zigbee Protocol**

4623 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4624 communicate securely with the Bridge.

4625

4626

4627

4628

4629

4630

4631

4632

4633

4634

4635

4636

4637

4638

4639

4640

4641

4642

4643

4644

4645

4646 .



4647  
4648  
4649

## Annex A (informative) Access Control Examples

4650

### Example OCF ACL Resource

4651 Figure A-1 shows how a "/oic/sec/acl2" Resource could be configured to enforce an example  
4652 access policy on the Server.

```
4653 {  
4654   "aclist2": [  
4655     {  
4656       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve, Update,  
4657       Delete and Notify)  
4658       "subject": {"uuid": "XXXX-...-XX01"},  
4659       "resources": [  
4660         {"href": "/oic/sh/light/1"},  
4661         {"href": "/oic/sh/temp/0"}  
4662       ],  
4663       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC  
4664       "validity": [  
4665         // The period starting at 18:00:00 UTC, on January 1, 2015 and  
4666         // ending at 07:00:00 UTC on January 2, 2015  
4667         "period": ["20150101T180000Z/20150102T070000Z"],  
4668         // Repeats the {period} every week until the last day of Jan. 2015.  
4669         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]  
4670       ],  
4671       "aceid": 1  
4672     }  
4673   ],  
4674   // An ACL provisioning and management service should be identified as  
4675   // the resource owner  
4676   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"  
4677 }  
4678
```

4678 **Figure A-1 – Example "/oic/sec/acl2" Resource**

4679

### Example AMS

4680 Figure A-2 demonstrates how the "/oic/sec/amacl" Resource should be configured to achieve this  
4681 objective.

```
4682 {  
4683   "resources": [  
4684     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was  
4685     // supplied then use the sacl validation credential to enforce access.  
4686     {"href": "/oic/sh/light/1"},  
4687     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was  
4688     // supplied then use the sacl validation credential to enforce access.  
4689     {"href": "/oma/3"},  
4690   ]  
4691 }
```

```
4690 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4691 // the sacl validation credential to enforce access.
4692 {"wc": ""}]
4693 }
4694
```

**Figure A-2 Example "/oic/sec/amacl" Resource**

DRAFT

4695  
4696  
4697

## Annex B (Informative) Execution Environment Security Profiles

4698 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security  
4699 robustness requirements meeting all IOT applications and services will not serve the needs of OCF,  
4700 and security profiles of varying degree of robustness (trustworthiness), cost and complexity have  
4701 to be defined. To address a large ecosystem of vendors, the profiles can only be defined as  
4702 requirements and the exact solutions meeting those requirements are specific to the vendors' open  
4703 or proprietary implementations, and thus in most part outside scope of this document.

4704 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228  
4705 (Terminology for constrained node networks) methodology, we limit the number of security profiles  
4706 to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities criteria for  
4707 each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4708 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are  
4709 either capable of no security functionality or easily breakable security that depend on environmental  
4710 (e.g. availability of human) factors to perform security functions. This means the class 0 will not be  
4711 equipped with an SEE.

4712

**Table B.1 – OCF Security Profile**

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4713 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of  
4714 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,  
4715 memory, storage).

4716  
4717  
4718

## Annex C (normative) Resource Type definitions

### 4719 C.1 List of Resource Type definitions

4720 Table C.1 contains the list of defined security resources in this document.

4721 **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

4722

### 4723 C.2 Account Token – moved to OCF Cloud Security document

### 4724 C.3 Access Control List [DEPRECATED]

4725 This clause intentionally left empty.

### 4726 C.4 Access Control List-2

#### 4727 C.4.1 Introduction

4728 This Resource specifies the local access control list.  
4729 When used without query parameters, all the ACE entries are returned.  
4730 When used with a query parameter, only the ACEs matching the specified  
4731 parameter are returned.  
4732

#### 4733 C.4.2 Well-known URI

4734 /oic/sec/acl2

#### 4735 C.4.3 Resource type

4736 The Resource Type is defined as: "oic.r.acl2".

#### 4737 C.4.4 OpenAPI 2.0 definition

```
4738 {
4739   "swagger": "2.0",
4740   "info": {
4741     "title": "Access Control List-2",
4742     "version": "20190111",
4743     "license": {
4744       "name": "OCF Data Model License",
4745       "url":
4746         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
4747         CENSE.md",
4748       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
4749         reserved."
4750     },
4751     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4752   },
4753   "schemes": ["http"],
4754   "consumes": ["application/json"],
4755   "produces": ["application/json"],
4756   "paths": {
4757     "/oic/sec/acl2" : {
4758       "get": {
4759         "description": "This Resource specifies the local access control list.\nWhen used without
4760         query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
4761         matching the specified\nparameter are returned.\n",
4762         "parameters": [
4763           {"$ref": "#/parameters/interface"},
4764           {"$ref": "#/parameters/ace-filtered"}
4765         ],
4766         "responses": {
4767           "200": {
4768             "description": "",
4769             "x-example":
4770               {
4771                 "rt" : ["oic.r.acl2"],
4772                 "aclist2": [
4773                   {
4774                     "aceid": 1,
4775                     "subject": {
4776                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4777                       "role": "SOME_STRING"
4778                     },
4779                     "resources": [
4780                       {
4781                         "href": "/light",
4782                         "rt": ["oic.r.light"],
4783                         "if": ["oic.if.baseline", "oic.if.a"]
4784                       },
4785                       {
4786                         "href": "/door",
4787                         "rt": ["oic.r.door"],
4788                         "if": ["oic.if.baseline", "oic.if.a"]
4789                       }
4790                     ],
4791                     "permission": 24
4792                   }
4793                 ],
4794                 "aceid": 2,
4795                 "subject": {
4796                   "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4797                 },
4798                 "resources": [
4799                   {
4800                     "href": "/light",
4801                     "rt": ["oic.r.light"],
4802                     "if": ["oic.if.baseline", "oic.if.a"]
4803                   },
4804                   {
4805                     "href": "/door",
4806                     "rt": ["oic.r.door"],
```

```

4807         "if": ["oic.if.baseline", "oic.if.a"]
4808     }
4809 ],
4810 "permission": 24
4811 },
4812 {
4813     "aceid": 3,
4814     "subject": {"conntype": "anon-clear"},
4815     "resources": [
4816         {
4817             "href": "/light",
4818             "rt": ["oic.r.light"],
4819             "if": ["oic.if.baseline", "oic.if.a"]
4820         },
4821         {
4822             "href": "/door",
4823             "rt": ["oic.r.door"],
4824             "if": ["oic.if.baseline", "oic.if.a"]
4825         }
4826     ],
4827     "permission": 16,
4828     "validity": [
4829         {
4830             "period": "20160101T180000Z/20170102T070000Z",
4831             "recurrence": [ "DSTART:XXXXX",
4832 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4833         },
4834         {
4835             "period": "20160101T180000Z/PT5H30M",
4836             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4837         }
4838     ]
4839     }
4840 ],
4841     "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
4842 },
4843     "schema": { "$ref": "#/definitions/Acl2" }
4844 },
4845     "400": {
4846         "description": "The request is invalid."
4847     }
4848 },
4849 },
4850 "post": {
4851     "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
4852 update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
4853 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
4854 the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
4855     "parameters": [
4856         { "$ref": "#/parameters/interface" },
4857         { "$ref": "#/parameters/ace-filtered" },
4858     ],
4859     "name": "body",
4860     "in": "body",
4861     "required": true,
4862     "schema": { "$ref": "#/definitions/Acl2-Update" },
4863     "x-example":
4864     {
4865         "aclist2": [
4866             {
4867                 "aceid": 1,
4868                 "subject": {
4869                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4870                     "role": "SOME_STRING"
4871                 },
4872                 "resources": [
4873                     {
4874                         "href": "/light",
4875                         "rt": ["oic.r.light"],
4876                         "if": ["oic.if.baseline", "oic.if.a"]
4877                     }

```

```

4878         {
4879             "href": "/door",
4880             "rt": ["oic.r.door"],
4881             "if": ["oic.if.baseline", "oic.if.a"]
4882         }
4883     ],
4884     "permission": 24
4885 },
4886 {
4887     "aceid": 3,
4888     "subject": {
4889         "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4890     },
4891     "resources": [
4892         {
4893             "href": "/light",
4894             "rt": ["oic.r.light"],
4895             "if": ["oic.if.baseline", "oic.if.a"]
4896         },
4897         {
4898             "href": "/door",
4899             "rt": ["oic.r.door"],
4900             "if": ["oic.if.baseline", "oic.if.a"]
4901         }
4902     ],
4903     "permission": 24
4904 }
4905 ],
4906 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4907 }
4908 },
4909 ],
4910 "responses": {
4911     "400": {
4912         "description": "The request is invalid."
4913     },
4914     "201": {
4915         "description": "The ACL entry is created."
4916     },
4917     "204": {
4918         "description": "The ACL entry is updated."
4919     }
4920 },
4921 },
4922 "delete": {
4923     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
4924 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
4925 the\nspecified parameter are deleted.\n",
4926     "parameters": [
4927         {"$ref": "#/parameters/interface"},
4928         {"$ref": "#/parameters/ace-filtered"}
4929     ],
4930     "responses": {
4931         "200": {
4932             "description": "The matching ACEs or the entire ACL Resource has been successfully
4933 deleted."
4934         },
4935         "400": {
4936             "description": "The request is invalid."
4937         }
4938     }
4939 }
4940 },
4941 },
4942 "parameters": {
4943     "interface": {
4944         "in": "query",
4945         "name": "if",
4946         "type": "string",
4947         "enum": ["oic.if.baseline"]
4948     },

```

```

4949     "ace-filtered" : {
4950         "in" : "query",
4951         "name" : "aceid",
4952         "required" : false,
4953         "type" : "integer",
4954         "description" : "Only applies to the ACE with the specified aceid.",
4955         "x-example" : 2112
4956     }
4957 },
4958 "definitions": {
4959     "Acl2" : {
4960         "properties": {
4961             "rowneruuid" : {
4962                 "description": "The value identifies the unique Resource owner\nFormat pattern according
4963 to IETF RFC 4122.",
4964                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4965 9]{12}$",
4966                 "type": "string"
4967             },
4968             "rt" : {
4969                 "description": "Resource Type of the Resource.",
4970                 "items": {
4971                     "maxLength": 64,
4972                     "type": "string",
4973                     "enum": ["oic.r.acl2"]
4974                 },
4975                 "minItems": 1,
4976                 "maxItems": 1,
4977                 "readOnly": true,
4978                 "type": "array"
4979             },
4980             "aclist2" : {
4981                 "description": "Access Control Entries in the ACL Resource.",
4982                 "items": {
4983                     "properties": {
4984                         "aceid": {
4985                             "description": "An identifier for the ACE that is unique within the ACL. In cases
4986 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
4987                             "minimum": 1,
4988                             "type": "integer"
4989                         },
4990                         "permission": {
4991                             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
4992 permissions.",
4993                             "x-detail-desc": [
4994                                 "0 - No permissions",
4995                                 "1 - Create permission is granted",
4996                                 "2 - Read, observe, discover permission is granted",
4997                                 "4 - Write, update permission is granted",
4998                                 "8 - Delete permission is granted",
4999                                 "16 - Notify permission is granted"
5000                             ],
5001                             "maximum": 31,
5002                             "minimum": 0,
5003                             "type": "integer"
5004                         },
5005                         "resources": {
5006                             "description": "References the application's Resources to which a security policy
5007 applies.",
5008                             "items": {
5009                                 "description": "Each Resource must have at least one of these properties set.",
5010                                 "properties": {
5011                                     "href": {
5012                                         "description": "When present, the ACE only applies when the href matches\nThis
5013 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5014                                         "format": "uri",
5015                                         "maxLength": 256,
5016                                         "type": "string"
5017                                     },
5018                                     "if": {
5019                                         "description": "When present, the ACE only applies when the if (interface)

```

```

5020 matches\nThe interface set supported by this Resource.",
5021     "items": {
5022         "enum": [
5023             "oic.if.baseline",
5024             "oic.if.ll",
5025             "oic.if.b",
5026             "oic.if.rw",
5027             "oic.if.r",
5028             "oic.if.a",
5029             "oic.if.s"
5030         ],
5031         "type": "string"
5032     },
5033     "minItems": 1,
5034     "type": "array"
5035 },
5036 "rt": {
5037     "description": "When present, the ACE only applies when the rt (Resource type)
5038 matches\nResource Type of the Resource.",
5039     "items": {
5040         "maxLength": 64,
5041         "type": "string"
5042     },
5043     "minItems": 1,
5044     "type": "array"
5045 },
5046 "wc": {
5047     "description": "A wildcard matching policy.",
5048     "pattern": "^[+*]$",
5049     "type": "string"
5050 },
5051 },
5052 "type": "object"
5053 },
5054 "type": "array"
5055 },
5056 "subject": {
5057     "anyOf": [
5058         {
5059             "description": "This is the Device identifier.",
5060             "properties": {
5061                 "uuid": {
5062                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5063 4122.",
5064                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5065 fA-F0-9]{12}$",
5066                     "type": "string"
5067                 }
5068             },
5069             "required": [
5070                 "uuid"
5071             ],
5072             "type": "object"
5073         },
5074         {
5075             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5076 <Authority> refers to the local entity or Device.",
5077             "properties": {
5078                 "authority": {
5079                     "description": "The Authority component of the entity being identified. A
5080 NULL <Authority> refers to the local entity or Device.",
5081                     "type": "string"
5082                 },
5083                 "role": {
5084                     "description": "The ID of the role being identified.",
5085                     "type": "string"
5086                 }
5087             },
5088             "required": [
5089                 "role"
5090             ],

```

```

5091         "type": "object"
5092     },
5093     {
5094         "properties": {
5095             "conntype": {
5096                 "description": "This property allows an ACE to be matched based on the
5097 connection or message type.",
5098                 "x-detail-desc": [
5099                     "auth-crypt - ACE applies if the Client is authenticated and the data
5100 channel or message is encrypted and integrity protected",
5101                     "anon-clear - ACE applies if the Client is not authenticated and the data
5102 channel or message is not encrypted but may be integrity protected"
5103                 ],
5104                 "enum": [
5105                     "auth-crypt",
5106                     "anon-clear"
5107                 ],
5108                 "type": "string"
5109             }
5110         },
5111         "required": [
5112             "conntype"
5113         ],
5114         "type": "object"
5115     }
5116 ]
5117 },
5118 "validity": {
5119     "description": "validity is an array of time-pattern objects.",
5120     "items": {
5121         "description": "The time-pattern contains a period and recurrence expressed in
5122 RFC5545 syntax.",
5123         "properties": {
5124             "period": {
5125                 "description": "String represents a period using the RFC5545 Period.",
5126                 "type": "string"
5127             },
5128             "recurrence": {
5129                 "description": "String array represents a recurrence rule using the RFC5545
5130 Recurrence.",
5131                 "items": {
5132                     "type": "string"
5133                 },
5134                 "type": "array"
5135             }
5136         },
5137         "required": [
5138             "period"
5139         ],
5140         "type": "object"
5141     },
5142     "type": "array"
5143 }
5144 },
5145 "required": [
5146     "aceid",
5147     "resources",
5148     "permission",
5149     "subject"
5150 ],
5151 "type": "object"
5152 },
5153 "type": "array"
5154 },
5155 "n": {
5156     "$ref":
5157 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5158 schema.json#/definitions/n"
5159 },
5160 "id": {
5161     "$ref":

```

```

5162 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5163 schema.json#/definitions/id"
5164 },
5165 "if" : {
5166 "description": "The interface set supported by this Resource.",
5167 "items": {
5168 "enum": [
5169 "oic.if.baseline"
5170 ],
5171 "type": "string"
5172 },
5173 "minItems": 1,
5174 "maxItems": 1,
5175 "readOnly": true,
5176 "type": "array"
5177 }
5178 },
5179 "type" : "object",
5180 "required": ["acllist2", "rowneruuid"]
5181 },
5182 "Acl2-Update" : {
5183 "properties": {
5184 "rowneruuid" : {
5185 "description": "The value identifies the unique Resource owner\nFormat pattern according
5186 to IETF RFC 4122.",
5187 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5188 9]{12}$",
5189 "type": "string"
5190 },
5191 "acllist2" : {
5192 "description": "Access Control Entries in the ACL Resource.",
5193 "items": {
5194 "properties": {
5195 "aceid": {
5196 "description": "An identifier for the ACE that is unique within the ACL. In cases
5197 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5198 "minimum": 1,
5199 "type": "integer"
5200 },
5201 "permission": {
5202 "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5203 permissions.",
5204 "x-detail-desc": [
5205 "0 - No permissions",
5206 "1 - Create permission is granted",
5207 "2 - Read, observe, discover permission is granted",
5208 "4 - Write, update permission is granted",
5209 "8 - Delete permission is granted",
5210 "16 - Notify permission is granted"
5211 ],
5212 "maximum": 31,
5213 "minimum": 0,
5214 "type": "integer"
5215 },
5216 "resources": {
5217 "description": "References the application's Resources to which a security policy
5218 applies.",
5219 "items": {
5220 "description": "Each Resource must have at least one of these properties set.",
5221 "properties": {
5222 "href": {
5223 "description": "When present, the ACE only applies when the href matches\nThis
5224 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5225 "format": "uri",
5226 "maxLength": 256,
5227 "type": "string"
5228 },
5229 "if": {
5230 "description": "When present, the ACE only applies when the if (interface)
5231 matches\nThe interface set supported by this Resource.",
5232 "items": {

```

```

5233         "enum": [
5234             "oic.if.baseline",
5235             "oic.if.ll",
5236             "oic.if.b",
5237             "oic.if.rw",
5238             "oic.if.r",
5239             "oic.if.a",
5240             "oic.if.s"
5241         ],
5242         "type": "string"
5243     },
5244     "minItems": 1,
5245     "type": "array"
5246 },
5247 "rt": {
5248     "description": "When present, the ACE only applies when the rt (Resource type)
5249 matches\nResource Type of the Resource.",
5250     "items": {
5251         "maxLength": 64,
5252         "type": "string"
5253     },
5254     "minItems": 1,
5255     "type": "array"
5256 },
5257 "wc": {
5258     "description": "A wildcard matching policy.",
5259     "x-detail-desc": [
5260         "+ - Matches all discoverable Resources",
5261         "- - Matches all non-discoverable Resources",
5262         "* - Matches all Resources"
5263     ],
5264     "enum": [
5265         "+",
5266         "-",
5267         "*"
5268     ],
5269     "type": "string"
5270 },
5271 },
5272 "type": "object"
5273 },
5274 "type": "array"
5275 },
5276 "subject": {
5277     "anyOf": [
5278         {
5279             "description": "This is the Device identifier.",
5280             "properties": {
5281                 "uuid": {
5282                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5283 4122.",
5284                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5285 fA-F0-9]{12}$",
5286                     "type": "string"
5287                 }
5288             },
5289             "required": [
5290                 "uuid"
5291             ],
5292             "type": "object"
5293         },
5294         {
5295             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5296 <Authority> refers to the local entity or Device.",
5297             "properties": {
5298                 "authority": {
5299                     "description": "The Authority component of the entity being identified. A
5300 NULL <Authority> refers to the local entity or Device.",
5301                     "type": "string"
5302                 },
5303                 "role": {

```

```

5304         "description": "The ID of the role being identified.",
5305         "type": "string"
5306     },
5307 },
5308     "required": [
5309         "role"
5310     ],
5311     "type": "object"
5312 },
5313 {
5314     "properties": {
5315         "conntype": {
5316             "description": "This property allows an ACE to be matched based on the
5317 connection or message type.",
5318             "x-detail-desc": [
5319                 "auth-crypt - ACE applies if the Client is authenticated and the data
5320 channel or message is encrypted and integrity protected",
5321                 "anon-clear - ACE applies if the Client is not authenticated and the data
5322 channel or message is not encrypted but may be integrity protected"
5323             ],
5324             "enum": [
5325                 "auth-crypt",
5326                 "anon-clear"
5327             ],
5328             "type": "string"
5329         }
5330     },
5331     "required": [
5332         "conntype"
5333     ],
5334     "type": "object"
5335 }
5336 ]
5337 },
5338 "validity": {
5339     "description": "validity is an array of time-pattern objects.",
5340     "items": {
5341         "description": "The time-pattern contains a period and recurrence expressed in
5342 RFC5545 syntax.",
5343         "properties": {
5344             "period": {
5345                 "description": "String represents a period using the RFC5545 Period.",
5346                 "type": "string"
5347             },
5348             "recurrence": {
5349                 "description": "String array represents a recurrence rule using the RFC5545
5350 Recurrence.",
5351                 "items": {
5352                     "type": "string"
5353                 },
5354                 "type": "array"
5355             }
5356         },
5357         "required": [
5358             "period"
5359         ],
5360         "type": "object"
5361     },
5362     "type": "array"
5363 }
5364 },
5365 "required": [
5366     "resources",
5367     "permission",
5368     "subject"
5369 ],
5370 "type": "object"
5371 },
5372 "type": "array"
5373 }
5374 },

```

```

5375     "type" : "object"
5376     }
5377   }
5378 }
5379

```

5380 **C.4.5 Property definition**

5381 Table C.4 defines the Properties that are part of the "oic.r.acl2" Resource Type.

5382 **Table C.2 – 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.

5383 **C.4.6 CRUDN behaviour**

5384 Table C.5 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

5385 **Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5386 **C.5 Managed Access Control**

5387 **C.5.1 Introduction**

5388 This Resource specifies the host Resources with access permission that is managed by an AMS.

5389 **C.5.2 Well-known URI**

5390 /oic/sec/amacl

### 5391 C.5.3 Resource type

5392 The Resource Type is defined as: "oic.r.amacl".

### 5393 C.5.4 OpenAPI 2.0 definition

```
5394 {
5395   "swagger": "2.0",
5396   "info": {
5397     "title": "Managed Access Control",
5398     "version": "20190111",
5399     "license": {
5400       "name": "OCF Data Model License",
5401       "url":
5402         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbc8ebdc4ba/LI
5403         CENSE.md",
5404       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5405         reserved."
5406     },
5407     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5408   },
5409   "schemes": ["http"],
5410   "consumes": ["application/json"],
5411   "produces": ["application/json"],
5412   "paths": {
5413     "/oic/sec/amacl" : {
5414       "get": {
5415         "description": "This Resource specifies the host Resources with access permission that is
5416         managed by an AMS.\n",
5417         "parameters": [
5418           { "$ref": "#/parameters/interface" }
5419         ],
5420         "responses": {
5421           "200": {
5422             "description": "",
5423             "x-example":
5424               {
5425                 "rt": ["oic.r.amacl"],
5426                 "resources": [
5427                   {
5428                     "href": "/temp",
5429                     "rt": ["oic.r.temperature"],
5430                     "if": ["oic.if.baseline", "oic.if.a"]
5431                   },
5432                   {
5433                     "href": "/temp",
5434                     "rt": ["oic.r.temperature"],
5435                     "if": ["oic.if.baseline", "oic.if.s"]
5436                   }
5437                 ]
5438               },
5439           "schema": { "$ref": "#/definitions/Amacl" }
5440         }
5441       }
5442     },
5443     "post": {
5444       "description": "Sets the new amacl data.\n",
5445       "parameters": [
5446         { "$ref": "#/parameters/interface" },
5447         {
5448           "name": "body",
5449           "in": "body",
5450           "required": true,
5451           "schema": { "$ref": "#/definitions/Amacl" },
5452           "x-example":
5453             {
5454               "resources": [
5455                 {
5456                   "href": "/temp",
5457                   "rt": ["oic.r.temperature"],
5458                   "if": ["oic.if.baseline", "oic.if.a"]
5459                 }
5460               ]
5461             }
5462         }
5463       ]
5464     }
5465   }
5466 }
```

```

5460         {
5461             "href": "/temp",
5462             "rt": ["oic.r.temperature"],
5463             "if": ["oic.if.baseline", "oic.if.s"]
5464         }
5465     ]
5466 }
5467 }
5468 ],
5469 "responses": {
5470     "400": {
5471         "description": "The request is invalid."
5472     },
5473     "201": {
5474         "description": "The AMACL entry is created."
5475     },
5476     "204": {
5477         "description": "The AMACL entry is updated."
5478     }
5479 }
5480 },
5481 "put": {
5482     "description": "Creates the new acl data.\n",
5483     "parameters": [
5484         {"$ref": "#/parameters/interface"},
5485         {
5486             "name": "body",
5487             "in": "body",
5488             "required": true,
5489             "schema": { "$ref": "#/definitions/Amacl" },
5490             "x-example":
5491             {
5492                 "resources": [
5493                     {
5494                         "href": "/temp",
5495                         "rt": ["oic.r.temperature"],
5496                         "if": ["oic.if.baseline", "oic.if.a"]
5497                     },
5498                     {
5499                         "href": "/temp",
5500                         "rt": ["oic.r.temperature"],
5501                         "if": ["oic.if.baseline", "oic.if.s"]
5502                     }
5503                 ]
5504             }
5505         }
5506     ],
5507     "responses": {
5508         "400": {
5509             "description": "The request is invalid."
5510         },
5511         "201": {
5512             "description": "The AMACL entry is created."
5513         }
5514     }
5515 },
5516 "delete": {
5517     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
5518 entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
5519 matched entry is deleted.\n",
5520     "parameters": [
5521         {"$ref": "#/parameters/interface"},
5522         {
5523             "in": "query",
5524             "description": "Delete the ACE identified by the string matching the subject value.\n",
5525             "type": "string",
5526             "name": "subject"
5527         }
5528     ],
5529     "responses": {
5530         "200": {

```

```

5531         "description" : "The ACE instance or the the entire AMACL Resource has been
5532 successfully deleted."
5533     },
5534     "400": {
5535         "description" : "The request is invalid."
5536     }
5537 }
5538 }
5539 },
5540 },
5541 "parameters": {
5542     "interface" : {
5543         "in" : "query",
5544         "name" : "if",
5545         "type" : "string",
5546         "enum" : ["oic.if.baseline"]
5547     }
5548 },
5549 "definitions": {
5550     "Amacl" : {
5551         "properties": {
5552             "rt" : {
5553                 "description": "Resource Type of the Resource.",
5554                 "items": {
5555                     "maxLength": 64,
5556                     "type": "string",
5557                     "enum": ["oic.r.amacl"]
5558                 },
5559                 "minItems": 1,
5560                 "maxItems": 1,
5561                 "readOnly": true,
5562                 "type": "array"
5563             },
5564             "n": {
5565                 "$ref":
5566 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5567 schema.json#/definitions/n"
5568             },
5569             "id": {
5570                 "$ref":
5571 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5572 schema.json#/definitions/id"
5573             },
5574             "resources" : {
5575                 "description": "Multiple links to this host's Resources.",
5576                 "items": {
5577                     "description": "Each Resource must have at least one of these properties set.",
5578                     "properties": {
5579                         "href": {
5580                             "description": "When present, the ACE only applies when the href matches\nThis is
5581 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5582                             "format": "uri",
5583                             "maxLength": 256,
5584                             "type": "string"
5585                         },
5586                         "if": {
5587                             "description": "When present, the ACE only applies when the if (interface)
5588 matches\nThe interface set supported by this Resource.",
5589                             "items": {
5590                                 "enum": [
5591                                     "oic.if.baseline",
5592                                     "oic.if.ll",
5593                                     "oic.if.b",
5594                                     "oic.if.rw",
5595                                     "oic.if.r",
5596                                     "oic.if.a",
5597                                     "oic.if.s"
5598                                 ],
5599                                 "type": "string"
5600                             },
5601                             "minItems": 1,

```

```

5602         "type": "array"
5603     },
5604     "rt": {
5605         "description": "When present, the ACE only applies when the rt (Resource type)
5606 matches\nResource Type of the Resource.",
5607         "items": {
5608             "maxLength": 64,
5609             "type": "string"
5610         },
5611         "minItems": 1,
5612         "type": "array"
5613     },
5614     "wc": {
5615         "description": "A wildcard matching policy.",
5616         "pattern": "^[+*]*$",
5617         "type": "string"
5618     }
5619 },
5620 "type": "object"
5621 },
5622 "type": "array"
5623 },
5624 "if" : {
5625     "description": "The interface set supported by this Resource.",
5626     "items": {
5627         "enum": [
5628             "oic.if.baseline"
5629         ],
5630         "type": "string"
5631     },
5632     "minItems": 1,
5633     "maxItems": 1,
5634     "readOnly": true,
5635     "type": "array"
5636 },
5637 },
5638 "type" : "object",
5639 "required": ["resources"]
5640 }
5641 }
5642 }
5643

```

### 5644 C.5.5 Property definition

5645 Table C.6 defines the Properties that are part of the "oic.r.amacl" Resource Type.

5646 **Table C.4 – 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	

### 5647 C.5.6 CRUDN behaviour

5648 Table C.7 defines the CRUDN operations that are supported on the "oic.r.amacl" Resource Type.

Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl".

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

## 5650 C.6 Credential

### 5651 C.6.1 Introduction

5652 This Resource specifies credentials a Device may use to establish secure communication.

5653 Retrieves the credential data.

5654 When used without query parameters, all the credential entries are returned.

5655 When used with a query parameter, only the credentials matching the specified  
5656 parameter are returned.

5657

5658 Note that write-only credential data will not be returned.

5659

### 5660 C.6.2 Well-known URI

5661 /oic/sec/cred

### 5662 C.6.3 Resource type

5663 The Resource Type is defined as: "oic.r.cred".

### 5664 C.6.4 OpenAPI 2.0 definition

```
5665 {
5666   "swagger": "2.0",
5667   "info": {
5668     "title": "Credential",
5669     "version": "v1.0-20181031",
5670     "license": {
5671       "name": "OCF Data Model License",
5672       "url":
5673         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5674         CENSE.md",
5675       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5676       reserved.",
5677     },
5678     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5679   },
5680   "schemes": ["http"],
5681   "consumes": ["application/json"],
5682   "produces": ["application/json"],
5683   "paths": {
5684     "/oic/sec/cred" : {
5685       "get": {
5686         "description": "This Resource specifies credentials a Device may use to establish secure
5687         communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
5688         credential entries are returned.\nWhen used with a query parameter, only the credentials matching
5689         the specified\nparameter are returned.\n\nNote that write-only credential data will not be
5690         returned.\n",
5691         "parameters": [
5692           {"$ref": "#/parameters/interface"}
5693           ,{"$ref": "#/parameters/cred-filtered-credid"}
5694           ,{"$ref": "#/parameters/cred-filtered-subjectuid"}
5695         ],
5696         "responses": {
5697           "200": {
5698             "description": "",
5699             "x-example":
5700               {
5701                 "rt": ["oic.r.cred"],
5702                 "creds": [
5703                   {
5704                     "credid": 55,
5705                     "subjectuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
```

```

5706         "roleid": {
5707             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5708             "role": "SOME_STRING"
5709         },
5710         "credtype": 32,
5711         "publicdata": {
5712             "encoding": "oic.sec.encoding.base64",
5713             "data": "BASE-64-ENCODED-VALUE"
5714         },
5715         "privatedata": {
5716             "encoding": "oic.sec.encoding.base64",
5717             "data": "BASE-64-ENCODED-VALUE",
5718             "handle": 4
5719         },
5720         "optionaldata": {
5721             "revstat": false,
5722             "encoding": "oic.sec.encoding.base64",
5723             "data": "BASE-64-ENCODED-VALUE"
5724         },
5725         "period": "20160101T180000Z/20170102T070000Z",
5726         "crms": [ "oic.sec.crm.pk10" ]
5727     },
5728     {
5729         "credid": 56,
5730         "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5731         "roleid": {
5732             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5733             "role": "SOME_STRING"
5734         },
5735         "credtype": 1,
5736         "publicdata": {
5737             "encoding": "oic.sec.encoding.base64",
5738             "data": "BASE-64-ENCODED-VALUE"
5739         },
5740         "privatedata": {
5741             "encoding": "oic.sec.encoding.base64",
5742             "data": "BASE-64-ENCODED-VALUE",
5743             "handle": 4
5744         },
5745         "optionaldata": {
5746             "revstat": false,
5747             "encoding": "oic.sec.encoding.base64",
5748             "data": "BASE-64-ENCODED-VALUE"
5749         },
5750         "period": "20160101T180000Z/20170102T070000Z",
5751         "crms": [ "oic.sec.crm.pk10" ]
5752     }
5753 ],
5754     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5755 }
5756 ,
5757 "schema": { "$ref": "#/definitions/Cred" }
5758 },
5759 "400": {
5760     "description": "The request is invalid."
5761 }
5762 }
5763 },
5764 "post": {
5765     "description": "Updates the credential Resource with the provided
5766 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
5767 Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
5768 Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
5769 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
5770 Resource.\n",
5771     "parameters": [
5772         { "$ref": "#/parameters/interface" },
5773         {
5774             "name": "body",
5775             "in": "body",
5776             "required": true,

```

```

5777     "schema": { "$ref": "#/definitions/Cred-Update" },
5778     "x-example":
5779     {
5780         "creds": [
5781             {
5782                 "credid": 55,
5783                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5784                 "roleid": {
5785                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5786                     "role": "SOME_STRING"
5787                 },
5788                 "credtype": 32,
5789                 "publicdata": {
5790                     "encoding": "oic.sec.encoding.base64",
5791                     "data": "BASE-64-ENCODED-VALUE"
5792                 },
5793                 "privatedata": {
5794                     "encoding": "oic.sec.encoding.base64",
5795                     "data": "BASE-64-ENCODED-VALUE",
5796                     "handle": 4
5797                 },
5798                 "optionaldata": {
5799                     "revstat": false,
5800                     "encoding": "oic.sec.encoding.base64",
5801                     "data": "BASE-64-ENCODED-VALUE"
5802                 },
5803                 "period": "20160101T180000Z/20170102T070000Z",
5804                 "crms": [ "oic.sec.crm.pk10" ]
5805             },
5806             {
5807                 "credid": 56,
5808                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5809                 "roleid": {
5810                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5811                     "role": "SOME_STRING"
5812                 },
5813                 "credtype": 1,
5814                 "publicdata": {
5815                     "encoding": "oic.sec.encoding.base64",
5816                     "data": "BASE-64-ENCODED-VALUE"
5817                 },
5818                 "privatedata": {
5819                     "encoding": "oic.sec.encoding.base64",
5820                     "data": "BASE-64-ENCODED-VALUE",
5821                     "handle": 4
5822                 },
5823                 "optionaldata": {
5824                     "revstat": false,
5825                     "encoding": "oic.sec.encoding.base64",
5826                     "data": "BASE-64-ENCODED-VALUE"
5827                 },
5828                 "period": "20160101T180000Z/20170102T070000Z",
5829                 "crms": [ "oic.sec.crm.pk10" ]
5830             }
5831         ],
5832         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5833     }
5834 },
5835 ],
5836 "responses": {
5837     "400": {
5838         "description": "The request is invalid."
5839     },
5840     "201": {
5841         "description": "The credential entry is created."
5842     },
5843     "204": {
5844         "description": "The credential entry is updated."
5845     }
5846 },
5847 },

```

```

5848     "delete": {
5849         "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
5850 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
5851 matching\nthe query parameter are deleted.\n",
5852         "parameters": [
5853             {"$ref": "#/parameters/interface"},
5854             {"$ref": "#/parameters/cred-filtered-credid"},
5855             {"$ref": "#/parameters/cred-filtered-subjectuuid"}
5856         ],
5857         "responses": {
5858             "400": {
5859                 "description": "The request is invalid."
5860             },
5861             "204": {
5862                 "description": "The specific credential(s) or the the entire credential Resource has
5863 been successfully deleted."
5864             }
5865         }
5866     }
5867 },
5868 },
5869 "parameters": {
5870     "interface": {
5871         "in": "query",
5872         "name": "if",
5873         "type": "string",
5874         "enum": ["oic.if.baseline"]
5875     },
5876     "cred-filtered-credid": {
5877         "in": "query",
5878         "name": "credid",
5879         "required": false,
5880         "type": "integer",
5881         "description": "Only applies to the credential with the specified credid.",
5882         "x-example": 2112
5883     },
5884     "cred-filtered-subjectuuid": {
5885         "in": "query",
5886         "name": "subjectuuid",
5887         "required": false,
5888         "type": "string",
5889         "description": "Only applies to credentials with the specified subject UUID.",
5890         "x-example": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5891     }
5892 },
5893 "definitions": {
5894     "Cred": {
5895         "properties": {
5896             "rowneruuid": {
5897                 "description": "Format pattern according to IETF RFC 4122.",
5898                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5899 9]{12}$",
5900                 "type": "string"
5901             },
5902             "rt": {
5903                 "description": "Resource Type of the Resource.",
5904                 "items": {
5905                     "maxLength": 64,
5906                     "type": "string",
5907                     "enum": ["oic.r.cred"]
5908                 },
5909                 "minItems": 1,
5910                 "readOnly": true,
5911                 "type": "array",
5912                 "uniqueItems": true
5913             },
5914             "n": {
5915                 "$ref":
5916 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5917 schema.json#/definitions/n"
5918             }
5919         }
5920     }

```

```

5919     "id": {
5920         "$ref":
5921         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5922         schema.json#/definitions/id"
5923     },
5924     "creds" : {
5925         "description": "List of credentials available at this Resource.",
5926         "items": {
5927             "properties": {
5928                 "credid": {
5929                     "description": "Local reference to a credential Resource.",
5930                     "type": "integer"
5931                 },
5932                 "credtype": {
5933                     "description": "Representation of this credential's type\nCredential Types - Cred
5934 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
5935 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
5936 password32 - Asymmetric encryption key.",
5937                     "maximum": 63,
5938                     "minimum": 0,
5939                     "type": "integer"
5940                 },
5941                 "credusage": {
5942                     "description": "A string that provides hints about how/where the cred is used\nThe
5943 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
5944 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
5945 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
5946                     "enum": [
5947                         "oic.sec.cred.trustca",
5948                         "oic.sec.cred.cert",
5949                         "oic.sec.cred.rolecert",
5950                         "oic.sec.cred.mfgtrustca",
5951                         "oic.sec.cred.mfgcert"
5952                     ],
5953                     "type": "string"
5954                 },
5955                 "crms": {
5956                     "description": "The refresh methods that may be used to update this credential.",
5957                     "items": {
5958                         "description": "Each enum represents a method by which the credentials are
5959 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
5960 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
5961 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
5962 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
5963                         "enum": [
5964                             "oic.sec.crm.pro",
5965                             "oic.sec.crm.psk",
5966                             "oic.sec.crm.rdp",
5967                             "oic.sec.crm.skdc",
5968                             "oic.sec.crm.pk10"
5969                         ],
5970                         "type": "string"
5971                     },
5972                     "type": "array",
5973                     "uniqueItems" : true
5974                 },
5975                 "optionaldata": {
5976                     "description": "Credential revocation status information\nOptional credential
5977 contents describes revocation status for this credential.",
5978                     "properties": {
5979                         "data": {
5980                             "description": "The encoded structure.",
5981                             "type": "string"
5982                         },
5983                         "encoding": {
5984                             "description": "A string specifying the encoding format of the data contained in
5985 the optdata.",
5986                             "x-detail-desc": [
5987                                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
5988                                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
5989                                 "oic.sec.encoding.base64 - Base64 encoded object.",

```

```

5990         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
5991         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
5992         "oic.sec.encoding.raw - Raw hex encoded data."
5993     ],
5994     "enum": [
5995         "oic.sec.encoding.jwt",
5996         "oic.sec.encoding.cwt",
5997         "oic.sec.encoding.base64",
5998         "oic.sec.encoding.pem",
5999         "oic.sec.encoding.der",
6000         "oic.sec.encoding.raw"
6001     ],
6002     "type": "string"
6003 },
6004     "revstat": {
6005         "description": "Revocation status flag - true = revoked.",
6006         "type": "boolean"
6007     }
6008 },
6009     "required": [
6010         "revstat"
6011     ],
6012     "type": "object"
6013 },
6014     "period": {
6015         "description": "String with RFC5545 Period.",
6016         "type": "string"
6017     },
6018     "privatedata": {
6019         "description": "Private credential information\nCredential Resource non-public
6020 contents.",
6021         "properties": {
6022             "data": {
6023                 "description": "The encoded value.",
6024                 "maxLength": 3072,
6025                 "type": "string"
6026             },
6027             "encoding": {
6028                 "description": "A string specifying the encoding format of the data contained in
6029 the privdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6030 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6031 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6032 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data.",
6033                 "enum": [
6034                     "oic.sec.encoding.jwt",
6035                     "oic.sec.encoding.cwt",
6036                     "oic.sec.encoding.base64",
6037                     "oic.sec.encoding.uri",
6038                     "oic.sec.encoding.handle",
6039                     "oic.sec.encoding.raw"
6040                 ],
6041                 "type": "string"
6042             },
6043             "handle": {
6044                 "description": "Handle to a key storage Resource.",
6045                 "type": "integer"
6046             }
6047         },
6048         "required": [
6049             "encoding"
6050         ],
6051         "type": "object"
6052     },
6053     "publicdata": {
6054         "description": "Public credential information.",
6055         "properties": {
6056             "data": {
6057                 "description": "The encoded value.",
6058                 "maxLength": 3072,
6059                 "type": "string"
6060             },

```

```

6061         "encoding": {
6062             "description": "A string specifying the encoding format of the data contained in
6063 the pubdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6064 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6065 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.pem - Encoding for PEM encoded
6066 certificate or chain\noic.sec.encoding.der - Encoding for DER encoded
6067 certificate\noic.sec.encoding.raw - Raw hex encoded data.",
6068             "enum": [
6069                 "oic.sec.encoding.jwt",
6070                 "oic.sec.encoding.cwt",
6071                 "oic.sec.encoding.base64",
6072                 "oic.sec.encoding.uri",
6073                 "oic.sec.encoding.pem",
6074                 "oic.sec.encoding.der",
6075                 "oic.sec.encoding.raw"
6076             ],
6077             "type": "string"
6078         }
6079     },
6080     "type": "object"
6081 },
6082     "roleid": {
6083         "description": "The role this credential possesses\nSecurity role specified as an
6084 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6085         "properties": {
6086             "authority": {
6087                 "description": "The Authority component of the entity being identified. A NULL
6088 <Authority> refers to the local entity or Device.",
6089                 "type": "string"
6090             },
6091             "role": {
6092                 "description": "The ID of the role being identified.",
6093                 "type": "string"
6094             }
6095         },
6096         "required": [
6097             "role"
6098         ],
6099         "type": "object"
6100     },
6101     "subjectuuid": {
6102         "anyOf": [
6103             {
6104                 "description": "The id of the Device, which the cred entry applies to or \"*\n
6105 for wildcard identity.",
6106                 "pattern": "^\\*$",
6107                 "type": "string"
6108             },
6109             {
6110                 "description": "Format pattern according to IETF RFC 4122.",
6111                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6112 F0-9]{12}$",
6113                 "type": "string"
6114             }
6115         ]
6116     }
6117 },
6118     "type": "object"
6119 },
6120     "type": "array"
6121 },
6122     "if" : {
6123         "description": "The interface set supported by this Resource.",
6124         "items": {
6125             "enum": [
6126                 "oic.if.baseline"
6127             ],
6128             "type": "string"
6129         },
6130         "minItems": 1,
6131         "readOnly": true,

```

```

6132         "type": "array"
6133     },
6134 },
6135 "type": "object",
6136 "required": ["creds", "rowneruuid"]
6137 },
6138 "Cred-Update" : {
6139     "properties": {
6140         "rowneruuid" : {
6141             "description": "Format pattern according to IETF RFC 4122.",
6142             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6143 9]{12}$",
6144             "type": "string"
6145         },
6146         "creds" : {
6147             "description": "List of credentials available at this Resource.",
6148             "items": {
6149                 "properties": {
6150                     "credid": {
6151                         "description": "Local reference to a credential Resource.",
6152                         "type": "integer"
6153                     },
6154                     "credtype": {
6155                         "description": "Representation of this credential's type\nCredential Types - Cred
6156 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6157 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6158 password32 - Asymmetric encryption key.",
6159                         "maximum": 63,
6160                         "minimum": 0,
6161                         "type": "integer"
6162                     },
6163                     "credusage": {
6164                         "description": "A string that provides hints about how/where the cred is used\nThe
6165 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6166 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6167 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6168                         "enum": [
6169                             "oic.sec.cred.trustca",
6170                             "oic.sec.cred.cert",
6171                             "oic.sec.cred.rolecert",
6172                             "oic.sec.cred.mfgtrustca",
6173                             "oic.sec.cred.mfgcert"
6174                         ],
6175                         "type": "string"
6176                     },
6177                     "crms": {
6178                         "description": "The refresh methods that may be used to update this credential.",
6179                         "items": {
6180                             "description": "Each enum represents a method by which the credentials are
6181 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6182 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6183 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6184 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6185                             "enum": [
6186                                 "oic.sec.crm.pro",
6187                                 "oic.sec.crm.psk",
6188                                 "oic.sec.crm.rdp",
6189                                 "oic.sec.crm.skdc",
6190                                 "oic.sec.crm.pk10"
6191                             ],
6192                             "type": "string"
6193                         },
6194                     },
6195                 },
6196                 "type": "array"
6197             },
6198             "optionaldata": {
6199                 "description": "Credential revocation status information\nOptional credential
6200 contents describes revocation status for this credential.",
6201                 "properties": {
6202                     "data": {
6203                         "description": "The encoded structure.",
6204                         "type": "string"

```

```

6203     },
6204     "encoding": {
6205         "description": "A string specifying the encoding format of the data contained in
6206 the optdata.",
6207         "x-detail-desc": [
6208             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6209             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6210             "oic.sec.encoding.base64 - Base64 encoded object.",
6211             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6212             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6213             "oic.sec.encoding.raw - Raw hex encoded data."
6214         ],
6215         "enum": [
6216             "oic.sec.encoding.jwt",
6217             "oic.sec.encoding.cwt",
6218             "oic.sec.encoding.base64",
6219             "oic.sec.encoding.pem",
6220             "oic.sec.encoding.der",
6221             "oic.sec.encoding.raw"
6222         ],
6223         "type": "string"
6224     },
6225     "revstat": {
6226         "description": "Revocation status flag - true = revoked.",
6227         "type": "boolean"
6228     }
6229 },
6230 "required": [
6231     "revstat"
6232 ],
6233 "type": "object"
6234 },
6235 "period": {
6236     "description": "String with RFC5545 Period.",
6237     "type": "string"
6238 },
6239 "privatedata": {
6240     "description": "Private credential information\nCredential Resource non-public
6241 contents.",
6242     "properties": {
6243         "data": {
6244             "description": "The encoded value.",
6245             "maxLength": 3072,
6246             "type": "string"
6247         },
6248         "encoding": {
6249             "description": "A string specifying the encoding format of the data contained in
6250 the privdata.",
6251             "x-detail-desc": [
6252                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6253                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6254                 "oic.sec.encoding.base64 - Base64 encoded object.",
6255                 "oic.sec.encoding.uri - URI reference.",
6256                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
6257 referenced using a handle.",
6258                 "oic.sec.encoding.raw - Raw hex encoded data."
6259             ],
6260             "enum": [
6261                 "oic.sec.encoding.jwt",
6262                 "oic.sec.encoding.cwt",
6263                 "oic.sec.encoding.base64",
6264                 "oic.sec.encoding.uri",
6265                 "oic.sec.encoding.handle",
6266                 "oic.sec.encoding.raw"
6267             ],
6268             "type": "string"
6269         },
6270         "handle": {
6271             "description": "Handle to a key storage Resource.",
6272             "type": "integer"
6273         }
6274     }

```

```

6274     },
6275     "required": [
6276         "encoding"
6277     ],
6278     "type": "object"
6279 },
6280 "publicdata": {
6281     "properties": {
6282         "data": {
6283             "description": "The encoded value.",
6284             "maxLength": 3072,
6285             "type": "string"
6286         },
6287         "encoding": {
6288             "description": "Public credential information\nA string specifying the encoding
6289 format of the data contained in the pubdata.",
6290             "x-detail-desc": [
6291                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6292                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6293                 "oic.sec.encoding.base64 - Base64 encoded object.",
6294                 "oic.sec.encoding.uri - URI reference.",
6295                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6296                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6297                 "oic.sec.encoding.raw - Raw hex encoded data."
6298             ],
6299             "enum": [
6300                 "oic.sec.encoding.jwt",
6301                 "oic.sec.encoding.cwt",
6302                 "oic.sec.encoding.base64",
6303                 "oic.sec.encoding.uri",
6304                 "oic.sec.encoding.pem",
6305                 "oic.sec.encoding.der",
6306                 "oic.sec.encoding.raw"
6307             ],
6308             "type": "string"
6309         }
6310     },
6311     "type": "object"
6312 },
6313 "roleid": {
6314     "description": "The role this credential possesses\nSecurity role specified as an
6315 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6316     "properties": {
6317         "authority": {
6318             "description": "The Authority component of the entity being identified. A NULL
6319 <Authority> refers to the local entity or Device.",
6320             "type": "string"
6321         },
6322         "role": {
6323             "description": "The ID of the role being identified.",
6324             "type": "string"
6325         }
6326     },
6327     "required": [
6328         "role"
6329     ],
6330     "type": "object"
6331 },
6332 "subjectuuid": {
6333     "anyOf": [
6334         {
6335             "description": "The id of the Device, which the cred entry applies to or \"*\n
6336 for wildcard identity.",
6337             "pattern": "^\\*$",
6338             "type": "string"
6339         },
6340         {
6341             "description": "Format pattern according to IETF RFC 4122.",
6342             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6343 F0-9]{12}$",
6344             "type": "string"

```

```

6345     }
6346   ]
6347 }
6348 },
6349   "type": "object"
6350 },
6351   "type": "array"
6352 },
6353   "if" :
6354   {
6355     "description": "The interface set supported by this Resource.",
6356     "items": {
6357       "enum": [
6358         "oic.if.baseline"
6359       ],
6360       "type": "string"
6361     },
6362     "minItems": 1,
6363     "readOnly": true,
6364     "type": "array"
6365   }
6366 },
6367   "type" : "object"
6368 }
6369 }
6370 }
6371

```

6372 **C.6.5 Property definition**

6373 Table C.8 defines the Properties that are part of the "oic.r.cred" Resource Type.

6374 **Table C.6 – 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.

6375 **C.6.6 CRUDN behaviour**

6376 Table C.9 defines the CRUDN operations that are supported on the "oic.r.cred" Resource Type.

6377 **Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

6378 **C.7 Certificate Revocation**

6379 **C.7.1 Introduction**

6380 This Resource specifies certificate revocation lists as X.509 objects.

6381 **C.7.2 Well-known URI**

6382 /oic/sec/crl

6383 **C.7.3 Resource type**

6384 The Resource Type is defined as: "oic.r.crl".

6385 **C.7.4 OpenAPI 2.0 definition**

```
6386 {
6387   "swagger": "2.0",
6388   "info": {
6389     "title": "Certificate Revocation",
6390     "version": "v1.0-20150819",
6391     "license": {
6392       "name": "OCF Data Model License",
6393       "url":
6394         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6395         CENSE.md",
6396       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6397         reserved."
6398     },
6399     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6400   },
6401   "schemes": ["http"],
6402   "consumes": ["application/json"],
6403   "produces": ["application/json"],
6404   "paths": {
6405     "/oic/sec/crl" : {
6406       "get": {
6407         "description": "This Resource specifies certificate revocation lists as X.509 objects.\n",
6408         "parameters": [
6409           {"$ref": "#/parameters/interface"}
6410         ],
6411         "responses": {
6412           "200": {
6413             "description": "",
6414             "x-example":
6415               {
6416                 "rt": ["oic.r.crl"],
6417                 "crlid": 1,
6418                 "thisupdate": "2016-04-12T23:20:50.52Z",
6419                 "crldata": "Base64ENCODEDCRL"
6420               },
6421             "schema": { "$ref": "#/definitions/Crl" }
6422           }
6423         }
6424       },
6425       "post": {
6426         "description": "Updates the CRL data.\n",
6427         "parameters": [
6428           {"$ref": "#/parameters/interface"},
6429           {
6430             "name": "body",
6431             "in": "body",
```

```

6432         "required": true,
6433         "schema": { "$ref": "#/definitions/Crl-Update" },
6434         "x-example":
6435             {
6436                 "crlid": 1,
6437                 "thisupdate": "2016-04-12T23:20:50.52Z",
6438                 "crldata": "Base64ENCODEDCRL"
6439             }
6440     },
6441 ],
6442     "responses": {
6443         "400": {
6444             "description": "The request is invalid."
6445         },
6446         "204": {
6447             "description": "The CRL entry is updated."
6448         }
6449     }
6450 }
6451 }
6452 },
6453 "parameters": {
6454     "interface": {
6455         "in": "query",
6456         "name": "if",
6457         "type": "string",
6458         "enum": ["oic.if.baseline"]
6459     }
6460 },
6461 "definitions": {
6462     "Crl": {
6463         "properties": {
6464             "rt": {
6465                 "description": "Resource Type of the Resource.",
6466                 "items": {
6467                     "maxLength": 64,
6468                     "type": "string",
6469                     "enum": ["oic.r.crl"]
6470                 },
6471                 "minItems": 1,
6472                 "readOnly": true,
6473                 "type": "array"
6474             },
6475             "n": {
6476                 "$ref":
6477 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6478 schema.json#/definitions/n"
6479             },
6480             "id": {
6481                 "$ref":
6482 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6483 schema.json#/definitions/id"
6484             },
6485             "crldata": {
6486                 "description": "Base64 BER encoded CRL data.",
6487                 "type": "string"
6488             },
6489             "crlid": {
6490                 "description": "Local reference to a CRL Resource.",
6491                 "type": "integer"
6492             },
6493             "thisupdate": {
6494                 "description": "UTC time of last CRL update.",
6495                 "type": "string"
6496             },
6497             "if": {
6498                 "description": "The interface set supported by this Resource.",
6499                 "items": {
6500                     "enum": [
6501                         "oic.if.baseline"
6502                     ],

```

```

6503         "type": "string"
6504     },
6505     "minItems": 1,
6506     "readOnly": true,
6507     "type": "array"
6508 }
6509 },
6510     "type": "object",
6511     "required": ["crlid", "thisupdate", "crldata"]
6512 }
6513 },
6514 "Crl-Update": {
6515     "properties": {
6516         "crldata": {
6517             "description": "Base64 BER encoded CRL data.",
6518             "type": "string"
6519         },
6520         "crlid": {
6521             "description": "Local reference to a CRL Resource.",
6522             "type": "integer"
6523         },
6524         "thisupdate": {
6525             "description": "UTC time of last CRL update.",
6526             "type": "string"
6527         }
6528     },
6529     "type": "object"
6530 }
6531 }
6532 }
6533

```

6534 **C.7.5 Property definition**

6535 Table C.10 defines the Properties that are part of the "oic.r.crl" Resource Type.

6536 **Table C.8 – 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.

6537 **C.7.6 CRUDN behaviour**

6538 Table C.11 defines the CRUDN operations that are supported on the "oic.r.crl" Resource Type.

6539 **Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".**

Create	Read	Update	Delete	Notify
	get	post		observe

6540 **C.8 Certificate Signing Request**

6541 **C.8.1 Introduction**

6542 This Resource specifies a Certificate Signing Request.

6543 **C.8.2 Well-known URI**

6544 /oic/sec/csr

6545 **C.8.3 Resource type**

6546 The Resource Type is defined as: "oic.r.csr".

6547 **C.8.4 OpenAPI 2.0 definition**

```

6548 {
6549   "swagger": "2.0",
6550   "info": {
6551     "title": "Certificate Signing Request",
6552     "version": "v1.0-20150819",
6553     "license": {
6554       "name": "OCF Data Model License",
6555       "url":
6556         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6557         CENSE.md",
6558       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6559         reserved."
6560     },
6561     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6562   },
6563   "schemes": ["http"],
6564   "consumes": ["application/json"],
6565   "produces": ["application/json"],
6566   "paths": {
6567     "/oic/sec/csr" : {
6568       "get": {
6569         "description": "This Resource specifies a Certificate Signing Request.\n",
6570         "parameters": [
6571           {"$ref": "#/parameters/interface"}
6572         ],
6573         "responses": {
6574           "200": {
6575             "description": "",
6576             "x-example":
6577               {
6578                 "rt": ["oic.r.csr"],
6579                 "encoding" : "oic.sec.encoding.pem",
6580                 "csr": "PEMENCODEDCSR"
6581               },
6582             "schema": { "$ref": "#/definitions/Csr" }
6583           },
6584           "404": {
6585             "description": "The Device does not support certificates and generating CSRs."
6586           },
6587           "503": {
6588             "description": "The Device is not yet ready to return a response. Try again later."
6589           }
6590         }
6591       }
6592     }
6593   },

```

```

6594     "parameters": {
6595         "interface" : {
6596             "in" : "query",
6597             "name" : "if",
6598             "type" : "string",
6599             "enum" : ["oic.if.baseline"]
6600         }
6601     },
6602     "definitions": {
6603         "Csr" : {
6604             "properties": {
6605                 "rt" : {
6606                     "description": "Resource Type of the Resource.",
6607                     "items": {
6608                         "maxLength": 64,
6609                         "type": "string",
6610                         "enum": ["oic.r.csr"]
6611                     },
6612                     "minItems": 1,
6613                     "readOnly": true,
6614                     "type": "array"
6615                 },
6616                 "encoding": {
6617                     "description": "A string specifying the encoding format of the data contained in CSR.",
6618                     "x-detail-desc": [
6619                         "oic.sec.encoding.pem - Encoding for PEM encoded CSR.",
6620                         "oic.sec.encoding.der - Encoding for DER encoded CSR."
6621                     ],
6622                     "enum": [
6623                         "oic.sec.encoding.pem",
6624                         "oic.sec.encoding.der"
6625                     ],
6626                     "readOnly": true,
6627                     "type": "string"
6628                 },
6629                 "n": {
6630                     "$ref":
6631 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6632 schema.json#/definitions/n"
6633                 },
6634                 "id": {
6635                     "$ref":
6636 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6637 schema.json#/definitions/id"
6638                 },
6639                 "csr": {
6640                     "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
6641                     "maxLength": 3072,
6642                     "readOnly": true,
6643                     "type": "string"
6644                 },
6645                 "if": {
6646                     "description": "The interface set supported by this Resource.",
6647                     "items": {
6648                         "enum": [
6649                             "oic.if.baseline"
6650                         ],
6651                         "type": "string"
6652                     },
6653                     "minItems": 1,
6654                     "readOnly": true,
6655                     "type": "array"
6656                 }
6657             },
6658             "type" : "object",
6659             "required": ["csr", "encoding"]
6660         }
6661     }
6662 }
6663

```

6664 **C.8.5 Property definition**

6665 Table C.12 defines the Properties that are part of the "oic.r.csr" Resource Type.

6666 **Table C.10 – 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.

6667 **C.8.6 CRUDN behaviour**

6668 Table C.13 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

6669 **Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".**

Create	Read	Update	Delete	Notify
	get			observe

6670 **C.9 Device Owner Transfer Method**

6671 **C.9.1 Introduction**

6672 This Resource specifies properties needed to establish a Device owner.

6673

6674 **C.9.2 Well-known URI**

6675 /oic/sec/doxm

6676 **C.9.3 Resource type**

6677 The Resource Type is defined as: "oic.r.doxm".

6678 **C.9.4 OpenAPI 2.0 definition**

6679 {  
 6680 "swagger": "2.0",  
 6681 "info": {  
 6682 "title": "Device Owner Transfer Method",  
 6683 "version": "v1.0-20181001",  
 6684 "license": {  
 6685 "name": "OCF Data Model License",  
 6686 "url":  
 6687 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
 6688 CENSE.md",

```

6689         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6690 reserved."
6691     },
6692     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6693 },
6694 "schemes": ["http"],
6695 "consumes": ["application/json"],
6696 "produces": ["application/json"],
6697 "paths": {
6698     "/oic/sec/doxm" : {
6699         "get": {
6700             "description": "This Resource specifies properties needed to establish a Device owner.\n",
6701             "parameters": [
6702                 {"$ref": "#/parameters/interface"}
6703             ],
6704             "responses": {
6705                 "200": {
6706                     "description": "",
6707                     "x-example":
6708                         {
6709                             "rt": ["oic.r.doxm"],
6710                             "oxms": [ 0, 2, 3 ],
6711                             "oxmsel": 0,
6712                             "sct": 16,
6713                             "owned": true,
6714                             "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
6715                             "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6716                             "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6717                         }
6718                     ,
6719                     "schema": { "$ref": "#/definitions/Doxm" }
6720                 },
6721                 "400": {
6722                     "description": "The request is invalid."
6723                 }
6724             }
6725         },
6726         "post": {
6727             "description": "Updates the DOXM Resource data.\n",
6728             "parameters": [
6729                 {"$ref": "#/parameters/interface"},
6730                 {
6731                     "name": "body",
6732                     "in": "body",
6733                     "required": true,
6734                     "schema": { "$ref": "#/definitions/Doxm-Update" },
6735                     "x-example":
6736                         {
6737                             "oxmsel": 0,
6738                             "owned": true,
6739                             "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
6740                             "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6741                             "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6742                         }
6743                 }
6744             ],
6745             "responses": {
6746                 "400": {
6747                     "description": "The request is invalid."
6748                 },
6749                 "204": {
6750                     "description": "The DOXM entry is updated."
6751                 }
6752             }
6753         }
6754     }
6755 },
6756 "parameters": {
6757     "interface" : {
6758         "in" : "query",
6759         "name" : "if",

```

```

6760         "type" : "string",
6761         "enum" : ["oic.if.baseline"]
6762     },
6763 },
6764 "definitions": {
6765     "Doxm" : {
6766         "properties": {
6767             "rowneruuid": {
6768                 "description": "Format pattern according to IETF RFC 4122.",
6769                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6770 9]{12}$",
6771                 "type": "string"
6772             },
6773             "oxms": {
6774                 "description": "List of supported owner transfer methods.",
6775                 "items": {
6776                     "description": "The Device owner transfer methods that may be selected at Device on-
6777 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
6778 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
6779 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
6780 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
6781 (deprecated).",
6782                     "type": "integer"
6783                 },
6784                 "readOnly": true,
6785                 "type": "array"
6786             },
6787             "devowneruuid": {
6788                 "description": "Format pattern according to IETF RFC 4122.",
6789                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6790 9]{12}$",
6791                 "type": "string"
6792             },
6793             "deviceuuid": {
6794                 "description": "The uuid formatted identity of the Device\nFormat pattern according to
6795 IETF RFC 4122.",
6796                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6797 9]{12}$",
6798                 "type": "string"
6799             },
6800             "owned": {
6801                 "description": "Ownership status flag.",
6802                 "type": "boolean"
6803             },
6804             "n": {
6805                 "$ref":
6806 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6807 schema.json#/definitions/n"
6808             },
6809             "id": {
6810                 "$ref":
6811 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6812 schema.json#/definitions/id"
6813             },
6814             "oxmsel": {
6815                 "description": "The selected owner transfer method used during on-boarding\nThe Device
6816 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
6817 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
6818 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
6819 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
6820 method (oic.sec.doxm.dcap) (deprecated).",
6821                 "type": "integer"
6822             },
6823             "sct": {
6824                 "description": "Bitmask encoding of supported credential types\nCredential Types -
6825 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6826 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6827 password32 - Asymmetric encryption key.",
6828                 "maximum": 63,
6829                 "minimum": 0,
6830                 "type": "integer",

```

```

6831         "readOnly": true
6832     },
6833     "rt" : {
6834         "description": "Resource Type of the Resource.",
6835         "items": {
6836             "maxLength": 64,
6837             "type": "string",
6838             "enum": ["oic.r.doxm"]
6839         },
6840         "minItems": 1,
6841         "readOnly": true,
6842         "type": "array"
6843     },
6844     "if": {
6845         "description": "The interface set supported by this Resource.",
6846         "items": {
6847             "enum": [
6848                 "oic.if.baseline"
6849             ],
6850             "type": "string"
6851         },
6852         "minItems": 1,
6853         "readOnly": true,
6854         "type": "array"
6855     }
6856 },
6857 "type" : "object",
6858 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
6859 },
6860 "Doxm-Update" : {
6861     "properties": {
6862         "rowneruuid": {
6863             "description": "Format pattern according to IETF RFC 4122.",
6864             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6865 9]{12}$",
6866             "type": "string"
6867         },
6868         "devowneruuid": {
6869             "description": "Format pattern according to IETF RFC 4122.",
6870             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6871 9]{12}$",
6872             "type": "string"
6873         },
6874         "deviceuuid": {
6875             "description": "The uuid formatted identity of the Device\nFormat pattern according to
6876 IETF RFC 4122.",
6877             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6878 9]{12}$",
6879             "type": "string"
6880         },
6881         "owned": {
6882             "description": "Ownership status flag.",
6883             "type": "boolean"
6884         },
6885         "oxmsel": {
6886             "description": "The selected owner transfer method used during on-boarding\nThe Device
6887 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
6888 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
6889 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
6890 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
6891 method (oic.sec.doxm.dcap) (deprecated).",
6892             "type": "integer"
6893         }
6894     },
6895     "type" : "object"
6896 }
6897 }
6898 }
6899

```

6900 **C.9.5 Property definition**

6901 Table C.14 defines the Properties that are part of the "oic.r.doxm" Resource Type.

6902 **Table C.12 – 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 method method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw) method1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp) method2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert) method3 - 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.

6903 **C.9.6 CRUDN behaviour**

6904 Table C.15 defines the CRUDN operations that are supported on the "oic.r.doxm" Resource Type.

6905 **Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".**

Create	Read	Update	Delete	Notify
	get	post		observe

6906 **C.10 Device Provisioning Status**

6907 **C.10.1 Introduction**

6908 This Resource specifies Device provisioning status.

6909

6910 **C.10.2 Well-known URI**

6911 /oic/sec/pstat

6912 **C.10.3 Resource type**

6913 The Resource Type is defined as: "oic.r.pstat".

6914 **C.10.4 OpenAPI 2.0 definition**

```
6915 {
6916   "swagger": "2.0",
6917   "info": {
6918     "title": "Device Provisioning Status",
6919     "version": "v1.0-20191001",
6920     "license": {
6921       "name": "OCF Data Model License",
6922       "url":
6923         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6924         CENSE.md",
6925       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6926         reserved."
6927     },
6928     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6929   },
6930   "schemes": ["http"],
6931   "consumes": ["application/json"],
6932   "produces": ["application/json"],
6933   "paths": {
6934     "/oic/sec/pstat" : {
6935       "get": {
6936         "description": "This Resource specifies Device provisioning status.\n",
6937         "parameters": [
6938           {"$ref": "#/parameters/interface"}
6939         ],
6940         "responses": {
6941           "200": {
6942             "description" : "",
6943             "x-example":
6944               {
6945                 "rt": ["oic.r.pstat"],
6946                 "dos": {"s": 3, "p": true},
6947                 "isop": true,
6948                 "cm": 8,
6949                 "tm": 60,
6950                 "om": 2,
6951                 "sm": 7,
6952                 "owneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
6953               },
6954             "schema": { "$ref": "#/definitions/Pstat" }
6955           },
6956           "400": {
6957             "description": "The request is invalid."
6958           }
6959         }
6960       }
6961     }
6962   }
6963 }
```

```

6960     },
6961     "post": {
6962         "description": "Sets or updates Device provisioning status data.\n",
6963         "parameters": [
6964             { "$ref": "#/parameters/interface" },
6965             {
6966                 "name": "body",
6967                 "in": "body",
6968                 "required": true,
6969                 "schema": { "$ref": "#/definitions/Pstat-Update" },
6970                 "x-example":
6971                 {
6972                     "dos": { "s": 3 },
6973                     "tm": 60,
6974                     "om": 2,
6975                     "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
6976                 }
6977             }
6978         ],
6979         "responses": {
6980             "400": {
6981                 "description": "The request is invalid."
6982             },
6983             "204": {
6984                 "description": "The PSTAT entry is updated."
6985             }
6986         }
6987     }
6988 },
6989 },
6990 "parameters": {
6991     "interface": {
6992         "in": "query",
6993         "name": "if",
6994         "type": "string",
6995         "enum": ["oic.if.baseline"]
6996     }
6997 },
6998 "definitions": {
6999     "Pstat": {
7000         "properties": {
7001             "rowneruuid": {
7002                 "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7003 according to IETF RFC 4122.",
7004                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7005 9]{12}$",
7006                 "type": "string"
7007             },
7008             "rt": {
7009                 "description": "Resource Type of the Resource.",
7010                 "items": {
7011                     "maxLength": 64,
7012                     "type": "string",
7013                     "enum": ["oic.r.pstat"]
7014                 },
7015                 "minItems": 1,
7016                 "readOnly": true,
7017                 "type": "array"
7018             },
7019             "om": {
7020                 "description": "Current operational mode\nDevice provisioning operation may be server
7021 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7022 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7023 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7024 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7025                 "maximum": 7,
7026                 "minimum": 1,
7027                 "type": "integer"
7028             },
7029             "cm": {
7030                 "description": "Current Device provisioning mode\nDevice provisioning mode maintains a

```

```

7031 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7032 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7033 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7034 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7035 Software Version Validation128 - Initiate Secure Software Update.",
7036     "maximum": 255,
7037     "minimum": 0,
7038     "type": "integer",
7039     "readOnly": true
7040   },
7041   "n": {
7042     "$ref":
7043     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7044     schema.json#/definitions/n"
7045   },
7046   "id": {
7047     "$ref":
7048     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7049     schema.json#/definitions/id"
7050   },
7051   "isop": {
7052     "description": "true indicates Device is operational.",
7053     "readOnly": true,
7054     "type": "boolean"
7055   },
7056   "tm": {
7057     "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7058     bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7059     in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7060     - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7061     services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7062     Software Version Validation128 - Initiate Secure Software Update.",
7063     "maximum": 255,
7064     "minimum": 0,
7065     "type": "integer"
7066   },
7067   "sm": {
7068     "description": "Supported operational modes\nDevice provisioning operation may be server
7069     directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7070     and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7071     services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7072     - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7073     "maximum": 7,
7074     "minimum": 1,
7075     "type": "integer",
7076     "readOnly": true
7077   },
7078   "dos": {
7079     "description": "Device on-boarding state\nDevice operation state machine.",
7080     "properties": {
7081       "p": {
7082         "default": true,
7083         "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7084         to Device Resources are complete.",
7085         "readOnly": true,
7086         "type": "boolean"
7087       },
7088       "s": {
7089         "description": "The current or pending operational state.",
7090         "x-detail-desc": [
7091           "0 - RESET - Device reset state.",
7092           "1 - RFOFM - Ready for Device owner transfer method state.",
7093           "2 - RFPPO - Ready for Device provisioning state.",
7094           "3 - RFNOP - Ready for Device normal operation state.",
7095           "4 - SRESET - The Device is in a soft reset state."
7096         ],
7097         "maximum": 4,
7098         "minimum": 0,
7099         "type": "integer"
7100       }
7101     }

```

```

7102         "required": [
7103             "s"
7104         ],
7105         "type": "object"
7106     },
7107     "if" : {
7108         "description": "The interface set supported by this Resource.",
7109         "items": {
7110             "enum": [
7111                 "oic.if.baseline"
7112             ],
7113             "type": "string"
7114         },
7115         "minItems": 1,
7116         "readOnly": true,
7117         "type": "array"
7118     }
7119 },
7120 "type": "object",
7121 "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
7122 },
7123 "Pstat-Update" : {
7124     "properties": {
7125         "rowneruuid": {
7126             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7127 according to IETF RFC 4122.",
7128             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7129 9]{12}$",
7130             "type": "string"
7131         },
7132         "om": {
7133             "description": "Current operational mode\nDevice provisioning operation may be server
7134 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7135 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7136 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7137 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7138             "maximum": 7,
7139             "minimum": 1,
7140             "type": "integer"
7141         },
7142         "tm": {
7143             "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7144 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7145 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7146 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7147 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7148 Software Version Validation128 - Initiate Secure Software Update.",
7149             "maximum": 255,
7150             "minimum": 0,
7151             "type": "integer"
7152         },
7153         "dos": {
7154             "description": "Device on-boarding state\nDevice operation state machine.",
7155             "properties": {
7156                 "p": {
7157                     "default": true,
7158                     "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7159 to Device Resources are complete.",
7160                     "readOnly": true,
7161                     "type": "boolean"
7162                 },
7163                 "s": {
7164                     "description": "The current or pending operational state.",
7165                     "x-detail-desc": [
7166                         "0 - RESET - Device reset state.",
7167                         "1 - RFOFM - Ready for Device owner transfer method state.",
7168                         "2 - RFPRO - Ready for Device provisioning state.",
7169                         "3 - RFNOP - Ready for Device normal operation state.",
7170                         "4 - SRESET - The Device is in a soft reset state."
7171                     ],
7172                     "maximum": 4,

```

```

7173         "minimum": 0,
7174         "type": "integer"
7175     },
7176 },
7177 "required": [
7178     "s"
7179 ],
7180 "type": "object"
7181 },
7182 },
7183 "type" : "object"
7184 }
7185 }
7186 }
7187 }

```

7188 **C.10.5 Property definition**

7189 Table C.16 defines the Properties that are part of the "oic.r.pstat" Resource Type.

7190 **Table C.14 – 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

				<p>Device. The value can be either 8 or 16 character in length. If it's only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.</p>
sm	integer	Yes	Read Only	<p>Supported operational modes  Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 -</p>

				Unused16 - Unused32 - Unused64 - Unused128 - Unused.
om	integer	Yes	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
tm	integer	Yes	Read Write	Target Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If it's only 8 characters it represents the lower byte value1 - Manufacturer reset state2 -

				Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.
dos	object: see schema	Yes	Read Write	Device on-boarding state Device operation state machine.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.

7191 **C.10.6 CRUDN behaviour**

7192 Table C.17 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

7193 **Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".**

Create	Read	Update	Delete	Notify
	get	post		observe

7194 **C.11 Asserted Roles**

7195 **C.11.1 Introduction**

7196 This Resource specifies roles that have been asserted.

7197

7198 **C.11.2 Well-known URI**

7199 /oic/sec/roles

### 7200 C.11.3 Resource type

7201 The Resource Type is defined as: "oic.r.roles".

### 7202 C.11.4 OpenAPI 2.0 definition

```
7203 {
7204   "swagger": "2.0",
7205   "info": {
7206     "title": "Asserted Roles",
7207     "version": "v1.0-20170323",
7208     "license": {
7209       "name": "OCF Data Model License",
7210       "url":
7211         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7212         CENSE.md",
7213       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7214         reserved."
7215     },
7216     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7217   },
7218   "schemes": ["http"],
7219   "consumes": ["application/json"],
7220   "produces": ["application/json"],
7221   "paths": {
7222     "/oic/sec/roles" : {
7223       "get": {
7224         "description": "This Resource specifies roles that have been asserted.\n",
7225         "parameters": [
7226           {"$ref": "#/parameters/interface"}
7227         ],
7228         "responses": {
7229           "200": {
7230             "description": "",
7231             "x-example":
7232               {
7233                 "roles" :[
7234                   {
7235                     "credid":1,
7236                     "credtype":8,
7237                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7238                     "publicdata":
7239                       {
7240                         "encoding":"oic.sec.encoding.pem",
7241                         "data":"PEMENCODEDROLECERT"
7242                       },
7243                     "optionaldata":
7244                       {
7245                         "revstat": false,
7246                         "encoding":"oic.sec.encoding.pem",
7247                         "data":"PEMENCODEDISSUERCERT"
7248                       }
7249                   },
7250                   {
7251                     "credid":2,
7252                     "credtype":8,
7253                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
7254                     "publicdata":
7255                       {
7256                         "encoding":"oic.sec.encoding.pem",
7257                         "data":"PEMENCODEDROLECERT"
7258                       },
7259                     "optionaldata":
7260                       {
7261                         "revstat": false,
7262                         "encoding":"oic.sec.encoding.pem",
7263                         "data":"PEMENCODEDISSUERCERT"
7264                       }
7265                   }
7266                 ],
7267                 "rt":["oic.r.roles"],
7268                 "if":["oic.if.baseline"]
7269             }
7270           }
7271         }
7272       }
7273     }
7274   }
7275 }
```

```

7269         }
7270         ,
7271         "schema": { "$ref": "#/definitions/Roles" }
7272     },
7273     "400": {
7274         "description": "The request is invalid."
7275     }
7276 }
7277 },
7278 "post": {
7279     "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
7280 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
7281 same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
7282 ignored, the Resource assigns its own.\n",
7283     "parameters": [
7284         { "$ref": "#/parameters/interface" },
7285     ],
7286     "name": "body",
7287     "in": "body",
7288     "required": true,
7289     "schema": { "$ref": "#/definitions/Roles-update" },
7290     "x-example":
7291     {
7292         "roles" :[
7293             {
7294                 "credid":1,
7295                 "credtype":8,
7296                 "subjectuuid":"00000000-0000-0000-0000-000000000000",
7297                 "publicdata":
7298                 {
7299                     "encoding":"oic.sec.encoding.pem",
7300                     "data":"PEMENCODEDROLECERT"
7301                 },
7302                 "optionaldata":
7303                 {
7304                     "revstat": false,
7305                     "encoding":"oic.sec.encoding.pem",
7306                     "data":"PEMENCODEDISSUERCERT"
7307                 }
7308             },
7309             {
7310                 "credid":2,
7311                 "credtype":8,
7312                 "subjectuuid":"00000000-0000-0000-0000-000000000000",
7313                 "publicdata":
7314                 {
7315                     "encoding":"oic.sec.encoding.pem",
7316                     "data":"PEMENCODEDROLECERT"
7317                 },
7318                 "optionaldata":
7319                 {
7320                     "revstat": false,
7321                     "encoding":"oic.sec.encoding.pem",
7322                     "data":"PEMENCODEDISSUERCERT"
7323                 }
7324             }
7325         ]
7326     }
7327 },
7328 ],
7329 "responses": {
7330     "400": {
7331         "description": "The request is invalid."
7332     },
7333     "204": {
7334         "description": "The roles entry is updated."
7335     }
7336 },
7337 },
7338 "delete": {
7339     "description": "Deletes roles Resource entries.\n\nWhen DELETE is used without query

```

```

7340 parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
7341 entries matching\nthe query parameter are deleted.\n",
7342     "parameters": [
7343         {"$ref": "#/parameters/interface"},
7344         {"$ref": "#/parameters/roles-filtered"}
7345     ],
7346     "responses": {
7347         "200": {
7348             "description": "The specified or all roles Resource entries have been successfully
7349 deleted."
7350         },
7351         "400": {
7352             "description": "The request is invalid."
7353         }
7354     }
7355 }
7356 },
7357 },
7358 "parameters": {
7359     "interface": {
7360         "in": "query",
7361         "name": "if",
7362         "type": "string",
7363         "enum": ["oic.if.baseline"]
7364     },
7365     "roles-filtered": {
7366         "in": "query",
7367         "name": "credid",
7368         "required": false,
7369         "type": "integer",
7370         "description": "Only applies to the credential with the specified credid.",
7371         "x-example": 2112
7372     }
7373 },
7374 "definitions": {
7375     "Roles": {
7376         "properties": {
7377             "rt": {
7378                 "description": "Resource Type of the Resource.",
7379                 "items": {
7380                     "maxLength": 64,
7381                     "type": "string",
7382                     "enum": ["oic.r.roles"]
7383                 },
7384                 "minItems": 1,
7385                 "readOnly": true,
7386                 "type": "array"
7387             },
7388             "n": {
7389                 "$ref":
7390 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7391 schema.json#/definitions/n"
7392             },
7393             "id": {
7394                 "$ref":
7395 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7396 schema.json#/definitions/id"
7397             },
7398             "roles": {
7399                 "description": "List of role certificates.",
7400                 "items": {
7401                     "properties": {
7402                         "credid": {
7403                             "description": "Local reference to a credential Resource.",
7404                             "type": "integer"
7405                         },
7406                         "credtype": {
7407                             "description": "Representation of this credential's type\nCredential Types - Cred
7408 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7409 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7410 password32 - Asymmetric encryption key."

```

```

7411         "maximum": 63,
7412         "minimum": 0,
7413         "type": "integer"
7414     },
7415     "credusage": {
7416         "description": "A string that provides hints about how/where the cred is used\nThe
7417 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7418 Certificateoic.sec.cred.rolcert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7419 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7420         "enum": [
7421             "oic.sec.cred.trustca",
7422             "oic.sec.cred.cert",
7423             "oic.sec.cred.rolcert",
7424             "oic.sec.cred.mfgtrustca",
7425             "oic.sec.cred.mfgcert"
7426         ],
7427         "type": "string"
7428     },
7429     "crms": {
7430         "description": "The refresh methods that may be used to update this credential.",
7431         "items": {
7432             "description": "Each enum represents a method by which the credentials are
7433 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7434 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7435 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7436 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7437             "enum": [
7438                 "oic.sec.crm.pro",
7439                 "oic.sec.crm.psk",
7440                 "oic.sec.crm.rdp",
7441                 "oic.sec.crm.skdc",
7442                 "oic.sec.crm.pk10"
7443             ],
7444             "type": "string"
7445         },
7446         "type": "array"
7447     },
7448     "optionaldata": {
7449         "description": "Credential revocation status information\nOptional credential
7450 contents describes revocation status for this credential.",
7451         "properties": {
7452             "data": {
7453                 "description": "This is the encoded structure.",
7454                 "type": "string"
7455             },
7456             "encoding": {
7457                 "description": "A string specifying the encoding format of the data contained in
7458 the optdata.",
7459                 "x-detail-desc": [
7460                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7461                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7462                     "oic.sec.encoding.base64 - Base64 encoded object.",
7463                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7464                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7465                     "oic.sec.encoding.raw - Raw hex encoded data."
7466                 ],
7467                 "enum": [
7468                     "oic.sec.encoding.jwt",
7469                     "oic.sec.encoding.cwt",
7470                     "oic.sec.encoding.base64",
7471                     "oic.sec.encoding.pem",
7472                     "oic.sec.encoding.der",
7473                     "oic.sec.encoding.raw"
7474                 ],
7475                 "type": "string"
7476             },
7477             "revstat": {
7478                 "description": "Revocation status flag - true = revoked.",
7479                 "type": "boolean"
7480             }
7481         }
7482     },

```

```

7482         "required": [
7483             "revstat"
7484         ],
7485         "type": "object"
7486     },
7487     "period": {
7488         "description": "String with RFC5545 Period.",
7489         "type": "string"
7490     },
7491     "privatedata": {
7492         "description": "Private credential information\nCredential Resource non-public
7493 contents.",
7494         "properties": {
7495             "data": {
7496                 "description": "The encoded value.",
7497                 "maxLength": 3072,
7498                 "type": "string"
7499             },
7500             "encoding": {
7501                 "description": "A string specifying the encoding format of the data contained in
7502 the privdata.",
7503                 "x-detail-desc": [
7504                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7505                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7506                     "oic.sec.encoding.base64 - Base64 encoded object.",
7507                     "oic.sec.encoding.uri - URI reference.",
7508                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
7509 referenced using a handle.",
7510                     "oic.sec.encoding.raw - Raw hex encoded data."
7511                 ],
7512                 "enum": [
7513                     "oic.sec.encoding.jwt",
7514                     "oic.sec.encoding.cwt",
7515                     "oic.sec.encoding.base64",
7516                     "oic.sec.encoding.uri",
7517                     "oic.sec.encoding.handle",
7518                     "oic.sec.encoding.raw"
7519                 ],
7520                 "type": "string"
7521             },
7522             "handle": {
7523                 "description": "Handle to a key storage Resource.",
7524                 "type": "integer"
7525             }
7526         },
7527         "required": [
7528             "encoding"
7529         ],
7530         "type": "object"
7531     },
7532     "publicdata": {
7533         "description": "Public credential information.",
7534         "properties": {
7535             "data": {
7536                 "description": "This is the encoded value.",
7537                 "maxLength": 3072,
7538                 "type": "string"
7539             },
7540             "encoding": {
7541                 "description": "A string specifying the encoding format of the data contained in
7542 the pubdata.",
7543                 "x-detail-desc": [
7544                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7545                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7546                     "oic.sec.encoding.base64 - Base64 encoded object.",
7547                     "oic.sec.encoding.uri - URI reference.",
7548                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7549                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7550                     "oic.sec.encoding.raw - Raw hex encoded data."
7551                 ],
7552                 "enum": [

```

```

7553         "oic.sec.encoding.jwt",
7554         "oic.sec.encoding.cwt",
7555         "oic.sec.encoding.base64",
7556         "oic.sec.encoding.uri",
7557         "oic.sec.encoding.pem",
7558         "oic.sec.encoding.der",
7559         "oic.sec.encoding.raw"
7560     ],
7561     "type": "string"
7562 }
7563 },
7564 "type": "object"
7565 },
7566 "roleid": {
7567     "description": "The role this credential possesses\nSecurity role specified as an
7568 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
7569     "properties": {
7570         "authority": {
7571             "description": "The Authority component of the entity being identified. A NULL
7572 <Authority> refers to the local entity or Device.",
7573             "type": "string"
7574         },
7575         "role": {
7576             "description": "The ID of the role being identified.",
7577             "type": "string"
7578         }
7579     },
7580     "required": [
7581         "role"
7582     ],
7583     "type": "object"
7584 },
7585 "subjectuuid": {
7586     "anyOf": [
7587         {
7588             "description": "The id of the Device, which the cred entry applies to or \"*\n
7589 for wildcard identity.",
7590             "pattern": "^[\\*]*$",
7591             "type": "string"
7592         },
7593         {
7594             "description": "Format pattern according to IETF RFC 4122.",
7595             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7596 F0-9]{12}$",
7597             "type": "string"
7598         }
7599     ]
7600 },
7601 },
7602 "type": "object"
7603 },
7604 "type": "array"
7605 },
7606 "if": {
7607     "description": "The interface set supported by this Resource.",
7608     "items": {
7609         "enum": [
7610             "oic.if.baseline"
7611         ],
7612         "type": "string"
7613     },
7614     "minItems": 1,
7615     "readOnly": true,
7616     "type": "array"
7617 }
7618 },
7619 "type": "object",
7620 "required": ["roles"]
7621 },
7622 "Roles-update" : {
7623     "properties": {

```

```

7624     "roles": {
7625         "description": "List of role certificates.",
7626         "items": {
7627             "properties": {
7628                 "credid": {
7629                     "description": "Local reference to a credential Resource.",
7630                     "type": "integer"
7631                 },
7632                 "credtype": {
7633                     "description": "Representation of this credential's type\nCredential Types - Cred
7634 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7635 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
7636 password32 - Asymmetric encryption key.",
7637                     "maximum": 63,
7638                     "minimum": 0,
7639                     "type": "integer"
7640                 },
7641                 "credusage": {
7642                     "description": "A string that provides hints about how/where the cred is used\nThe
7643 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7644 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7645 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7646                     "enum": [
7647                         "oic.sec.cred.trustca",
7648                         "oic.sec.cred.cert",
7649                         "oic.sec.cred.rolecert",
7650                         "oic.sec.cred.mfgtrustca",
7651                         "oic.sec.cred.mfgcert"
7652                     ],
7653                     "type": "string"
7654                 },
7655                 "crms": {
7656                     "description": "The refresh methods that may be used to update this credential.",
7657                     "items": {
7658                         "description": "Each enum represents a method by which the credentials are
7659 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7660 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7661 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7662 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7663                         "enum": [
7664                             "oic.sec.crm.pro",
7665                             "oic.sec.crm.psk",
7666                             "oic.sec.crm.rdp",
7667                             "oic.sec.crm.skdc",
7668                             "oic.sec.crm.pk10"
7669                         ],
7670                         "type": "string"
7671                     },
7672                     "type": "array"
7673                 },
7674                 "optionaldata": {
7675                     "description": "Credential revocation status information\nOptional credential
7676 contents describes revocation status for this credential.",
7677                     "properties": {
7678                         "data": {
7679                             "description": "This is the encoded structure.",
7680                             "type": "string"
7681                         },
7682                         "encoding": {
7683                             "description": "A string specifying the encoding format of the data contained in
7684 the optdata.",
7685                             "x-detail-desc": [
7686                                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7687                                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7688                                 "oic.sec.encoding.base64 - Base64 encoded object.",
7689                                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7690                                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7691                                 "oic.sec.encoding.raw - Raw hex encoded data."
7692                             ],
7693                             "enum": [
7694                                 "oic.sec.encoding.jwt",

```

```

7695         "oic.sec.encoding.cwt",
7696         "oic.sec.encoding.base64",
7697         "oic.sec.encoding.pem",
7698         "oic.sec.encoding.der",
7699         "oic.sec.encoding.raw"
7700     ],
7701     "type": "string"
7702 },
7703 "revstat": {
7704     "description": "Revocation status flag - true = revoked.",
7705     "type": "boolean"
7706 }
7707 },
7708 "required": [
7709     "revstat"
7710 ],
7711 "type": "object"
7712 },
7713 "period": {
7714     "description": "String with RFC5545 Period.",
7715     "type": "string"
7716 },
7717 "privatedata": {
7718     "description": "Private credential information\nCredential Resource non-public
contents.",
7719     "properties": {
7720         "data": {
7721             "description": "The encoded value.",
7722             "maxLength": 3072,
7723             "type": "string"
7724         },
7725         "encoding": {
7726             "description": "A string specifying the encoding format of the data contained in
the privdata.",
7727             "x-detail-desc": [
7728                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7729                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7730                 "oic.sec.encoding.base64 - Base64 encoded object.",
7731                 "oic.sec.encoding.uri - URI reference.",
7732                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle.",
7733                 "oic.sec.encoding.raw - Raw hex encoded data."
7734             ],
7735             "enum": [
7736                 "oic.sec.encoding.jwt",
7737                 "oic.sec.encoding.cwt",
7738                 "oic.sec.encoding.base64",
7739                 "oic.sec.encoding.uri",
7740                 "oic.sec.encoding.handle",
7741                 "oic.sec.encoding.raw"
7742             ]
7743         },
7744         "handle": {
7745             "description": "Handle to a key storage Resource.",
7746             "type": "integer"
7747         }
7748     }
7749 },
7750 "required": [
7751     "encoding"
7752 ],
7753 "type": "object"
7754 },
7755 "publicdata": {
7756     "description": "Public credential information.",
7757     "properties": {
7758         "data": {
7759             "description": "The encoded value.",
7760             "maxLength": 3072,
7761             "type": "string"
7762         }
7763     }
7764 },
7765

```

```

7766         "encoding": {
7767             "description": "A string specifying the encoding format of the data contained in
the pubdata.",
7768             "x-detail-desc": [
7769                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7770                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7771                 "oic.sec.encoding.base64 - Base64 encoded object.",
7772                 "oic.sec.encoding.uri - URI reference.",
7773                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7774                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7775                 "oic.sec.encoding.raw - Raw hex encoded data."
7776             ],
7777             "enum": [
7778                 "oic.sec.encoding.jwt",
7779                 "oic.sec.encoding.cwt",
7780                 "oic.sec.encoding.base64",
7781                 "oic.sec.encoding.uri",
7782                 "oic.sec.encoding.pem",
7783                 "oic.sec.encoding.der",
7784                 "oic.sec.encoding.raw"
7785             ],
7786             "type": "string"
7787         },
7788     },
7789     "type": "object"
7790 },
7791 "roleid": {
7792     "description": "The role this credential possesses\nSecurity role specified as an
7793 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
7794     "properties": {
7795         "authority": {
7796             "description": "The Authority component of the entity being identified. A NULL
7797 <Authority> refers to the local entity or Device.",
7798             "type": "string"
7799         },
7800         "role": {
7801             "description": "The ID of the role being identified.",
7802             "type": "string"
7803         }
7804     },
7805     "required": [
7806         "role"
7807     ],
7808     "type": "object"
7809 },
7810 "subjectuuid": {
7811     "anyOf": [
7812         {
7813             "description": "The id of the Device, which the cred entry applies to or \"*\
7814 for wildcard identity.",
7815             "pattern": "^\\*$",
7816             "type": "string"
7817         },
7818         {
7819             "description": "Format pattern according to IETF RFC 4122.",
7820             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7821 F0-9]{12}$",
7822             "type": "string"
7823         }
7824     ]
7825 },
7826 },
7827 },
7828 "type": "object"
7829 },
7830 "type": "array"
7831 }
7832 },
7833 "type": "object",
7834 "required": ["roles"]
7835 }
7836 }

```

7837 }  
7838

### 7839 C.11.5 Property definition

7840 Table C.18 defines the Properties that are part of the "oic.r.roles" Resource Type.

7841 **Table C.16 – 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.

### 7842 C.11.6 CRUDN behaviour

7843 Table C.19 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

7844 **Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 7845 C.12 Signed Access Control List

### 7846 C.12.1 Introduction

7847 This Resource specifies a signed ACL object.

7848

### 7849 C.12.2 Well-known URI

7850 /oic/sec/sacl

### 7851 C.12.3 Resource type

7852 The Resource Type is defined as: "oic.r.sacl".

### 7853 C.12.4 OpenAPI 2.0 definition

```
7854 {  
7855   "swagger": "2.0",  
7856   "info": {  
7857     "title": "Signed Access Control List",  
7858     "version": "v1.0-20150819",  
7859     "license": {  
7860       "name": "OCF Data Model License",  
7861       "url":  
7862         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
7863         CENSE.md",  
7864       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
7865       reserved."  
7866     },  
7867     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
7868   },  
7869   "schemes": ["http"],  
7870   "consumes": ["application/json"],
```

```

7871 "produces": ["application/json"],
7872 "paths": {
7873   "/oic/sec/sacl" : {
7874     "get": {
7875       "description": "This Resource specifies a signed ACL object.\n",
7876       "parameters": [
7877         {"$ref": "#/parameters/interface"}
7878       ],
7879       "responses": {
7880         "200": {
7881           "description": "",
7882           "x-example":
7883             {
7884               "rt": ["oic.r.sacl"],
7885               "aclist2": [
7886                 {
7887                   "aceid": 1,
7888                   "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
7889                   "resources": [
7890                     {
7891                       "href": "/temp",
7892                       "rt": ["oic.r.temperature"],
7893                       "if": ["oic.if.baseline", "oic.if.a"]
7894                     },
7895                     {
7896                       "href": "/temp",
7897                       "rt": ["oic.r.temperature"],
7898                       "if": ["oic.if.baseline", "oic.if.s"]
7899                     }
7900                   ],
7901                   "permission": 31,
7902                   "validity": [
7903                     {
7904                       "period": "20160101T180000Z/20170102T070000Z",
7905                       "recurrence": [ "DSTART:XXXXX",
7906 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
7907                     },
7908                     {
7909                       "period": "20160101T180000Z/PT5H30M",
7910                       "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
7911                     }
7912                   ]
7913                 },
7914                 {
7915                   "aceid": 2,
7916                   "subject": {
7917                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
7918                     "role": "SOME_STRING"
7919                   },
7920                   "resources": [
7921                     {
7922                       "href": "/light",
7923                       "rt": ["oic.r.light"],
7924                       "if": ["oic.if.baseline", "oic.if.a"]
7925                     },
7926                     {
7927                       "href": "/door",
7928                       "rt": ["oic.r.door"],
7929                       "if": ["oic.if.baseline", "oic.if.a"]
7930                     }
7931                   ],
7932                   "permission": 15
7933                 }
7934             ],
7935             "signature": {
7936               "sigtype": "oic.sec.sigtype.pk7",
7937               "sigvalue": "ENCODED-SIGNATURE-VALUE"
7938             }
7939           },
7940           "schema": { "$ref": "#/definitions/Sacl" }
7941         }

```

```

7942     }
7943   },
7944   "post": {
7945     "description": "Sets the sacl Resource data.\n",
7946     "parameters": [
7947       { "$ref": "#/parameters/interface" },
7948       {
7949         "name": "body",
7950         "in": "body",
7951         "required": true,
7952         "schema": { "$ref": "#/definitions/Sacl" },
7953         "x-example":
7954           {
7955             "aclist2": [
7956               {
7957                 "aceid": 1,
7958                 "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
7959                 "resources": [
7960                   {
7961                     "href": "/temp",
7962                     "rt": [ "oic.r.temperature" ],
7963                     "if": [ "oic.if.baseline", "oic.if.a" ]
7964                   },
7965                   {
7966                     "href": "/temp",
7967                     "rt": [ "oic.r.temperature" ],
7968                     "if": [ "oic.if.baseline", "oic.if.s" ]
7969                   }
7970                 ],
7971                 "permission": 31,
7972                 "validity": [
7973                   {
7974                     "period": "20160101T180000Z/20170102T070000Z",
7975                     "recurrence": [ "DSTART:XXXXX",
7976 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
7977                   },
7978                   {
7979                     "period": "20160101T180000Z/PT5H30M",
7980                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
7981                   }
7982                 ]
7983               },
7984               {
7985                 "aceid": 2,
7986                 "subject": {
7987                   "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
7988                   "role": "SOME_STRING"
7989                 },
7990                 "resources": [
7991                   {
7992                     "href": "/light",
7993                     "rt": [ "oic.r.light" ],
7994                     "if": [ "oic.if.baseline", "oic.if.a" ]
7995                   },
7996                   {
7997                     "href": "/door",
7998                     "rt": [ "oic.r.door" ],
7999                     "if": [ "oic.if.baseline", "oic.if.a" ]
8000                   }
8001                 ],
8002                 "permission": 15
8003               }
8004             ],
8005             "signature": {
8006               "sigtype": "oic.sec.sigtype.pk7",
8007               "sigvalue": "ENCODED-SIGNATURE-VALUE"
8008             }
8009           }
8010         }
8011       ],
8012       "responses": {

```

```

8013         "400": {
8014             "description": "The request is invalid."
8015         },
8016         "201": {
8017             "description": "The ACL entry is created."
8018         },
8019         "204": {
8020             "description": "The ACL entry is updated."
8021         }
8022     },
8023 },
8024 "put": {
8025     "description": "Sets the sacl Resource data\n",
8026     "parameters": [
8027         { "$ref": "#/parameters/interface" },
8028         {
8029             "name": "body",
8030             "in": "body",
8031             "required": true,
8032             "schema": { "$ref": "#/definitions/Sacl" },
8033             "x-example":
8034                 {
8035                     "aclist2": [
8036                         {
8037                             "aceid": 1,
8038                             "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8039                             "resources": [
8040                                 {
8041                                     "href": "/temp",
8042                                     "rt": [ "oic.r.temperature" ],
8043                                     "if": [ "oic.if.baseline", "oic.if.a" ]
8044                                 },
8045                                 {
8046                                     "href": "/temp",
8047                                     "rt": [ "oic.r.temperature" ],
8048                                     "if": [ "oic.if.baseline", "oic.if.s" ]
8049                                 }
8050                             ],
8051                             "permission": 31,
8052                             "validity": [
8053                                 {
8054                                     "period": "20160101T180000Z/20170102T070000Z",
8055                                     "recurrence": [ "DSTART:XXXXX",
8056 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8057                                 },
8058                                 {
8059                                     "period": "20160101T180000Z/PT5H30M",
8060                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8061                                 }
8062                             ]
8063                         },
8064                         {
8065                             "aceid": 2,
8066                             "subject": {
8067                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8068                                 "role": "SOME_STRING"
8069                             },
8070                             "resources": [
8071                                 {
8072                                     "href": "/light",
8073                                     "rt": [ "oic.r.light" ],
8074                                     "if": [ "oic.if.baseline", "oic.if.a" ]
8075                                 },
8076                                 {
8077                                     "href": "/door",
8078                                     "rt": [ "oic.r.door" ],
8079                                     "if": [ "oic.if.baseline", "oic.if.a" ]
8080                                 }
8081                             ],
8082                             "permission": 15
8083                         }

```

```

8084         ],
8085         "signature": {
8086             "sigtype": "oic.sec.sigtype.pk7",
8087             "sigvalue": "ENCODED-SIGNATURE-VALUE"
8088         }
8089     }
8090 },
8091 ],
8092 "responses": {
8093     "400": {
8094         "description": "The request is invalid."
8095     },
8096     "201": {
8097         "description": "The signed ACL entry is created."
8098     }
8099 },
8100 },
8101 "delete": {
8102     "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
8103 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
8104 specified, only the matched entry is deleted.\n",
8105     "parameters": [
8106         { "$ref": "#/parameters/interface" },
8107         {
8108             "in": "query",
8109             "description": "Delete the signed ACL identified by the string containing subject
8110 UUID.\n",
8111             "type": "string",
8112             "name": "subject"
8113         }
8114     ],
8115     "responses": {
8116         "200": {
8117             "description": "The signed ACL instance or the the entire signed ACL Resource has
8118 been successfully deleted."
8119         },
8120         "400": {
8121             "description": "The request is invalid."
8122         }
8123     }
8124 },
8125 },
8126 },
8127 "parameters": {
8128     "interface": {
8129         "in": "query",
8130         "name": "if",
8131         "type": "string",
8132         "enum": ["oic.if.baseline"]
8133     }
8134 },
8135 "definitions": {
8136     "Sacl": {
8137         "properties": {
8138             "rt": {
8139                 "description": "Resource Type of the Resource.",
8140                 "items": {
8141                     "maxLength": 64,
8142                     "type": "string",
8143                     "enum": ["oic.r.sacl"]
8144                 },
8145                 "minItems": 1,
8146                 "readOnly": true,
8147                 "type": "array"
8148             },
8149             "aclist2": {
8150                 "description": "Access Control Entries in the ACL Resource.",
8151                 "items": {
8152                     "properties": {
8153                         "aceid": {
8154                             "description": "An identifier for the ACE that is unique within the ACL. In cases

```

```

8155 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
8156     "minimum": 1,
8157     "type": "integer"
8158 },
8159 "permission": {
8160     "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
permissions.",
8161     "x-detail-desc": [
8162         "0 - No permissions.",
8163         "1 - Create permission is granted.",
8164         "2 - Read, observe, discover permission is granted.",
8165         "4 - Write, update permission is granted.",
8166         "8 - Delete permission is granted.",
8167         "16 - Notify permission is granted."
8168     ],
8169     "maximum": 31,
8170     "minimum": 0,
8171     "type": "integer"
8172 },
8173 "resources": {
8174     "description": "References the application's Resources to which a security policy
applies.",
8175     "items": {
8176         "description": "Each Resource must have at least one of these properties set.",
8177         "properties": {
8178             "href": {
8179                 "allOf": [
8180                     {
8181                         "description": "When present, the ACE only applies when the href matches."
8182                     },
8183                     {
8184                         "description": "This is the target URI, it can be specified as a Relative
Reference or fully-qualified URI.",
8185                         "format": "uri",
8186                         "maxLength": 256,
8187                         "type": "string"
8188                     }
8189                 ]
8190             }
8191         }
8192     },
8193     "if": {
8194         "description": "When present, the ACE only applies when the if (interface)
matches\nThe interface set supported by this Resource.",
8195         "items": {
8196             "enum": [
8197                 "oic.if.baseline",
8198                 "oic.if.ll",
8199                 "oic.if.b",
8200                 "oic.if.rw",
8201                 "oic.if.r",
8202                 "oic.if.a",
8203                 "oic.if.s"
8204             ],
8205             "type": "string"
8206         },
8207         "minItems": 1,
8208         "type": "array"
8209     },
8210     "rt": {
8211         "description": "When present, the ACE only applies when the rt (resource type)
matches\nResource Type of the Resource.",
8212         "items": {
8213             "maxLength": 64,
8214             "type": "string"
8215         },
8216         "minItems": 1,
8217         "type": "array"
8218     },
8219     "wc": {
8220         "description": "A wildcard matching policy.",
8221         "pattern": "^[+*]$",
8222         "type": "string"
8223     }
8224 }
8225

```

```

8226     }
8227     },
8228     "type": "object"
8229 },
8230 "type": "array"
8231 },
8232 "subject": {
8233   "anyOf": [
8234     {
8235       "description": "Device identifier.",
8236       "properties": {
8237         "uuid": {
8238           "description": "A UUID Device ID\nFormat pattern according to IETF RFC
8239 4122.",
8240           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
8241 fA-F0-9]{12}$",
8242           "type": "string"
8243         }
8244       },
8245       "required": [
8246         "uuid"
8247       ],
8248       "type": "object"
8249     },
8250     {
8251       "description": "Security role specified as an <Authority> & <Rolename>. A NULL
8252 <Authority> refers to the local entity or Device.",
8253       "properties": {
8254         "authority": {
8255           "description": "The Authority component of the entity being identified. A
8256 NULL <Authority> refers to the local entity or Device.",
8257           "type": "string"
8258         },
8259         "role": {
8260           "description": "The ID of the role being identified.",
8261           "type": "string"
8262         }
8263       },
8264       "required": [
8265         "role"
8266       ],
8267       "type": "object"
8268     },
8269     {
8270       "properties": {
8271         "conntype": {
8272           "description": "This property allows an ACE to be matched based on the
8273 connection or message type.",
8274           "x-detail-desc": [
8275             "auth-crypt - ACE applies if the Client is authenticated and the data
8276 channel or message is encrypted and integrity protected.",
8277             "anon-clear - ACE applies if the Client is not authenticated and the data
8278 channel or message is not encrypted but may be integrity protected."
8279           ],
8280           "enum": [
8281             "auth-crypt",
8282             "anon-clear"
8283           ],
8284           "type": "string"
8285         }
8286       },
8287       "required": [
8288         "conntype"
8289       ],
8290       "type": "object"
8291     }
8292   ]
8293 },
8294 "validity": {
8295   "description": "validity is an array of time-pattern objects.",
8296   "items": {

```

```

8297         "description": "The time-pattern contains a period and recurrence expressed in
8298 RFC5545 syntax.",
8299         "properties": {
8300             "period": {
8301                 "description": "String represents a period using the RFC5545 Period.",
8302                 "type": "string"
8303             },
8304             "recurrence": {
8305                 "description": "String array represents a recurrence rule using the RFC5545
8306 Recurrence.",
8307                 "items": {
8308                     "type": "string"
8309                 },
8310                 "type": "array"
8311             }
8312         },
8313         "required": [
8314             "period"
8315         ],
8316         "type": "object"
8317     },
8318     "type": "array"
8319 }
8320 },
8321 "required": [
8322     "aceid",
8323     "resources",
8324     "permission",
8325     "subject"
8326 ],
8327 "type": "object"
8328 },
8329 "type": "array"
8330 },
8331 "n": {
8332     "$ref":
8333     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8334 schema.json#/definitions/n"
8335 },
8336 "id": {
8337     "$ref":
8338     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8339 schema.json#/definitions/id"
8340 },
8341 "signature": {
8342     "description": "The signature over the ACL Resource\nEncoded signature data.",
8343     "properties": {
8344         "sigtype": {
8345             "description": "The string specifies the predefined signature format.",
8346             "x-detail-desc": [
8347                 "RFC7515 JSON web signature (JWS) object.",
8348                 "RFC2315 base64 encoded object.",
8349                 "CBOR encoded JWS object."
8350             ],
8351             "enum": [
8352                 "oic.sec.sigtype.jws",
8353                 "oic.sec.sigtype.pk7",
8354                 "oic.sec.sigtype.cws"
8355             ],
8356             "type": "string"
8357         },
8358         "sigvalue": {
8359             "description": "The encoded signature.",
8360             "type": "string"
8361         }
8362     },
8363     "required": [
8364         "sigtype",
8365         "sigvalue"
8366     ],
8367     "type": "object"

```

```

8368     },
8369     "if": {
8370         "description": "The interface set supported by this Resource.",
8371         "items": {
8372             "enum": [
8373                 "oic.if.baseline"
8374             ],
8375             "type": "string"
8376         },
8377         "minItems": 1,
8378         "readOnly": true,
8379         "type": "array"
8380     }
8381 },
8382 "type": "object",
8383 "required": ["aclist2", "signature"]
8384 }
8385 }
8386 }
8387

```

### 8388 C.12.5 Property definition

8389 Table C.20 defines the Properties that are part of the "oic.r.sacl" Resource Type.

8390 **Table C.18 – 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	

### 8391 C.12.6 CRUDN behaviour

8392 Table C.21 defines the CRUDN operations that are supported on the "oic.r.sacl" Resource Type.

8393 **Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl".**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

## 8394 C.13 Session – moved to OCF Cloud Security document

## 8395 C.14 Security Profile

### 8396 C.14.1 Introduction

8397 Resource specifying supported and active security profile(s).

8398

8399 **C.14.2 Well-known URI**

8400 /oic/sec/sp

8401 **C.14.3 Resource type**

8402 The Resource Type is defined as: "oic.r.sp".

8403 **C.14.4 OpenAPI 2.0 definition**

```
8404 {
8405   "swagger": "2.0",
8406   "info": {
8407     "title": "Security Profile",
8408     "version": "v1.0-20190208",
8409     "license": {
8410       "name": "OCF Data Model License",
8411       "url":
8412         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8413         CENSE.md",
8414       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8415         reserved."
8416     },
8417     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8418   },
8419   "schemes": ["http"],
8420   "consumes": ["application/json"],
8421   "produces": ["application/json"],
8422   "paths": {
8423     "/oic/sec/sp" : {
8424       "get": {
8425         "description": "Resource specifying supported and active security profile(s).\n",
8426         "parameters": [
8427           { "$ref": "#/parameters/interface" }
8428         ],
8429         "responses": {
8430           "200": {
8431             "description": "",
8432             "x-example":
8433               {
8434                 "rt": ["oic.r.sp"],
8435                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
8436                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
8437               },
8438             "schema": { "$ref": "#/definitions/SP" }
8439           },
8440           "400": {
8441             "description": "The request is invalid."
8442           }
8443         }
8444       },
8445       "post": {
8446         "description": "Sets or updates Device provisioning status data.\n",
8447         "parameters": [
8448           { "$ref": "#/parameters/interface" },
8449           {
8450             "name": "body",
8451             "in": "body",
8452             "required": true,
8453             "schema": { "$ref": "#/definitions/SP-Update" },
8454             "x-example":
8455               {
8456                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
8457                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
8458               }
8459           }
8460         ],
8461         "responses": {
8462           "200": {
8463             "description": "",
8464             "x-example":
8465               {
```

```

8466         "rt": ["oic.r.sp"],
8467         "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
8468         "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
8469     },
8470     "schema": { "$ref": "#/definitions/SP" }
8471 },
8472     "400": {
8473         "description" : "The request is invalid."
8474     }
8475 }
8476 }
8477 }
8478 },
8479 "parameters": {
8480     "interface" : {
8481         "in" : "query",
8482         "name" : "if",
8483         "type" : "string",
8484         "enum" : ["oic.if.baseline"]
8485     }
8486 },
8487 "definitions": {
8488     "SP" : {
8489         "properties": {
8490             "rt": {
8491                 "description": "Resource Type of the Resource.",
8492                 "items": {
8493                     "maxLength": 64,
8494                     "type": "string",
8495                     "enum": ["oic.r.sp"]
8496                 },
8497                 "minItems": 1,
8498                 "readOnly": true,
8499                 "type": "array"
8500             },
8501             "n": {
8502                 "$ref":
8503 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8504 schema.json#/definitions/n"
8505             },
8506             "id": {
8507                 "$ref":
8508 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8509 schema.json#/definitions/id"
8510             },
8511             "currentprofile": {
8512                 "description": "Security Profile currently active.",
8513                 "type": "string"
8514             },
8515             "supportedprofiles": {
8516                 "description": "Array of supported Security Profiles.",
8517                 "items": {
8518                     "type": "string"
8519                 },
8520                 "type": "array"
8521             },
8522             "if": {
8523                 "description": "The interface set supported by this Resource.",
8524                 "items": {
8525                     "enum": [
8526                         "oic.if.baseline"
8527                     ],
8528                     "type": "string"
8529                 },
8530                 "minItems": 1,
8531                 "readOnly": true,
8532                 "type": "array"
8533             }
8534         },
8535         "type" : "object",
8536         "required": ["supportedprofiles", "currentprofile"]

```

```

8537 },
8538 "SP-Update" : {
8539   "properties": {
8540     "currentprofile": {
8541       "description": "Security Profile currently active.",
8542       "type": "string"
8543     },
8544     "supportedprofiles": {
8545       "description": "Array of supported Security Profiles.",
8546       "items": {
8547         "type": "string"
8548       },
8549       "type": "array"
8550     }
8551   },
8552   "type" : "object"
8553 }
8554 }
8555 }
8556

```

#### 8557 C.14.5 Property definition

8558 Table C.24 defines the Properties that are part of the "oic.r.sp" Resource Type.

8559 **Table C.20 – 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.

#### 8560 C.14.6 CRUDN behaviour

8561 Table C.25 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

8562 **Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".**

Create	Read	Update	Delete	Notify
	get	post		observe

#### 8563 C.15 Token Refresh – moved to OCF Cloud Security document

8564  
8565  
8566  
8567

## Annex D (informative)

### OID definitions

8568 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to  
8569 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of  
8570 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
8571 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
8572     private(4) enterprise(1) OCF(51414) }
8573
8574 -- OCF Security specific OIDs
8575
8576 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
8577 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
8578
8579 -- OCF Security Categories
8580
8581 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
8582 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
8583
8584 -- OCF Security Profiles
8585
8586 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
8587 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
8588 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
8589 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
8590 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
8591
8592 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)
8593 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
8594 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
8595 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
8596 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
8597
8598 ocfSecurityProfileOID ::= UTF8String
8599
8600 -- OCF Security Certificate Policies
8601
8602 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
8603
8604 -- OCF X.509v3 Extensions
8605
8606 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
8607 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
8608 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
8609 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
8610
8611 ocfVersion ::= SEQUENCE {
8612     major    INTEGER,
8613     minor    INTEGER,
8614     build    INTEGER}
8615
8616 ocfCompliance ::= SEQUENCE {
8617     version        ocfVersion,
8618     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
8619     deviceName     UTF8String,
8620     deviceManufacturer UTF8String}
8621
8622 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
8623 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
8624
8625 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
8626
8627 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
8628
8629 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
8630 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
8631 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
8632
8633 ocfCPLAttributes ::= SEQUENCE {
8634     cpl-at-IANAPen UTF8String,
8635     cpl-at-model UTF8String,
8636     cpl-at-version UTF8String}
```

DRAFT

**Annex E  
(informative)**

**Security considerations specific to Bridged Protocols**

8637  
8638  
8639  
8640

8641 The text in this Annex is provided for information only. This Annex has no normative impact. This  
8642 information is applicable at the time of initial publication and may become out of date.

**E.1 Security Considerations specific to the AllJoyn Protocol**

8643  
8644

This clause intentionally left empty.

**E.2 Security Considerations specific to the Bluetooth LE Protocol**

8645

8646 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security mode  
8647 has several security levels (see Table E.1)

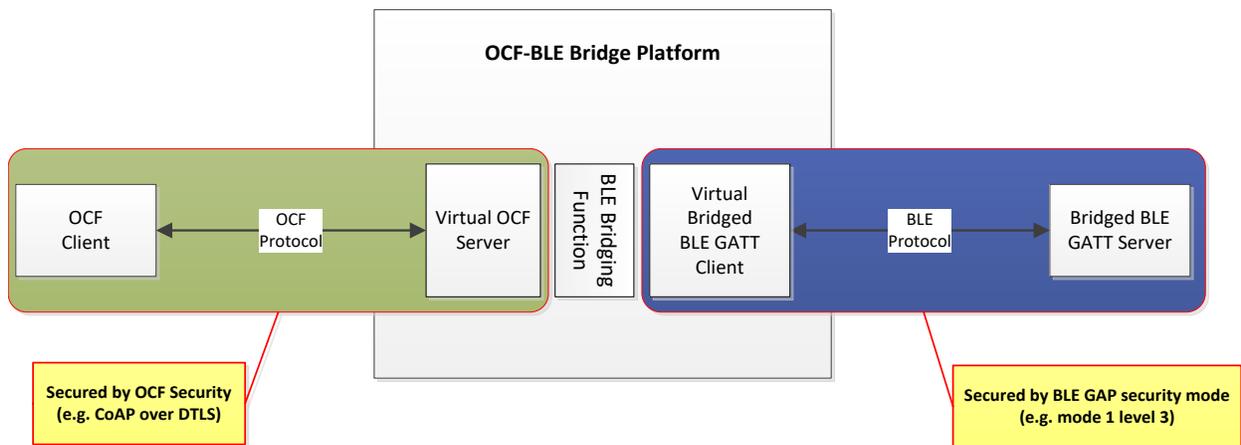
8648 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF  
8649 perspective. The appropriate selection of security mode and level is left to the vendor.

**Table E.1 GAP security mode**

8650

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)

8651 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are  
8652 secured by their own security.



8653

**Figure E-1 Security Considerations for BLE Bridge**

8654

**E.3 Security Considerations specific to the oneM2M Protocol**

8655

8656 This clause intentionally left empty.

8657 **E.4 Security Considerations specific to the U+ Protocol**

8658 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

8659 **Table E.2 TLS 1.2 Cipher Suites used by U+**

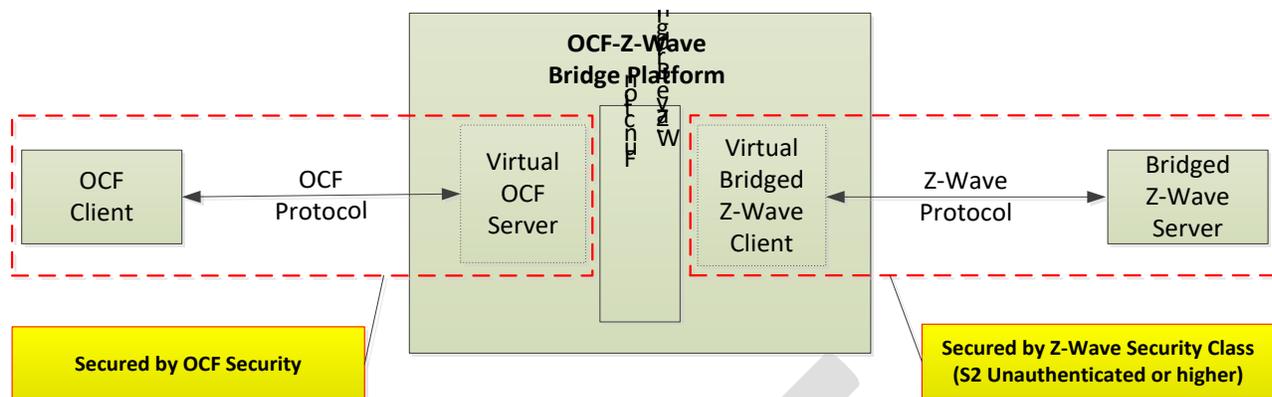
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

8660 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

8661 **E.5 Security Considerations specific to the Z-Wave Protocol**

8662 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2 Security  
 8663 Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for communication  
 8664 with a Virtual Bridged Client would typically be considered secure from an OCF perspective. The  
 8665 appropriate selection for S2 Security Class and Class Name is left to the vendor.

8666 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their  
 8667 own security.



8668

8669

**Figure E-2 Security Considerations for Z-Wave Bridge**

8670 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2  
8671 Unauthenticated provides the following advantages from the security perspective;

- 8672 – The unique device specific key for every secure device enables validation of device identity and  
8673 prevents man-in-the-middle compromises to security
- 8674 – The Secure cryptographic key exchange methods during inclusion achieves high level of  
8675 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 8676 – Out of band key exchange for product authentication which is combined with device specific  
8677 key prevents eavesdropping and man-in-the-middle attack vectors.

8678 See Table E.3 for a summary of Z-Wave Security Classes.

8679

**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

8680 On the other hand, S0 Security Class has the vulnerability of security during inclusion by  
8681 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the  
8682 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices  
8683 might be no longer secure in that case.

## 8684 E.6 Security Considerations specific to the Zigbee Protocol

8685 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the  
8686 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0  
8687 stack, "nwkSecurityLevel", represents the security level of a device.

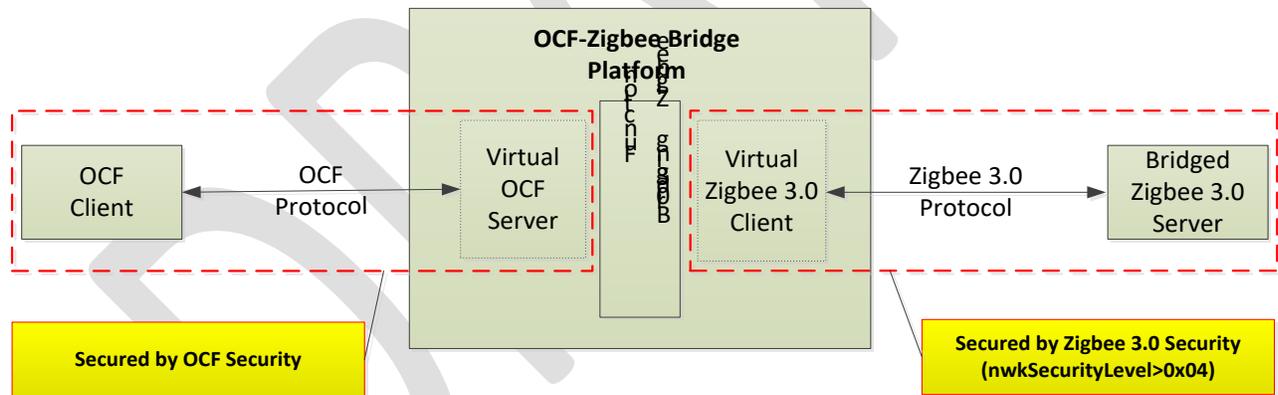
8688 The security level nwkSecurityLevel > 0x04 provides message integrity code (MIC) and/or AES128-  
 8689 CCM encryption (ENC). Zigbee Servers using nwkSecurityLevel > 0x04 would typically be  
 8690 considered secure from an OCF perspective. The appropriate selection for nwkSecurityLevel is left  
 8691 to the vendor.

8692 See Table E.4 for a summary of the Zigbee Security Levels.

8693 **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)

8694 Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are  
 8695 secured by their own security.



8696

8697 **Figure E-3 Security Considerations for Zigbee Bridge**