

OCF Security Specification

VERSION 2.0.5 | September 2019



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

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

1 **LEGAL DISCLAIMER**

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

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

15 Copyright © 2017-2019 Open Connectivity Foundation, Inc. All rights reserved.

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

17 **CONTENTS**

18 1 Scope 13

19 2 Normative References 13

20 3 Terms, definitions, and abbreviated terms 15

21 3.1 Terms and definitions..... 15

22 3.2 Abbreviated terms..... 18

23 4 Document Conventions and Organization 21

24 4.1 Conventions..... 21

25 4.2 Notation..... 22

26 4.3 Data types 23

27 4.4 Document structure..... 23

28 5 Security Overview..... 24

29 5.1 Preamble 24

30 5.2 Access Control..... 26

31 5.2.1 ACL Architecture 27

32 5.2.2 Access Control Scoping Levels..... 29

33 5.3 Onboarding Overview 31

34 5.3.1 Onboarding General 31

35 5.3.2 Onboarding Steps..... 32

36 5.3.3 Establishing a Device Owner 33

37 5.3.4 Provisioning for Normal Operation 34

38 5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview –
39 moved to OCF Cloud Security document 34

40 5.3.6 OCF Compliance Management System..... 34

41 5.4 Provisioning..... 34

42 5.4.1 Provisioning General 34

43 5.4.2 Provisioning other services..... 35

44 5.4.3 Provisioning Credentials for Normal Operation 35

45 5.4.4 Role Assignment and Provisioning for Normal Operation 35

46 5.4.5 ACL provisioning 36

47 5.5 Secure Resource Manager (SRM)..... 36

48 5.6 Credential Overview..... 37

49 6 Security for the Discovery Process 38

50 6.1 Preamble 38

51 6.2 Security Considerations for Discovery..... 38

52 7 Security Provisioning 41

53 7.1 Device Identity..... 41

54 7.1.1 General Device Identity 41

55 7.1.2 Device Identity for Devices with UAID [Deprecated]..... 41

56 7.2 Device Ownership..... 41

57 7.3 Device Ownership Transfer Methods..... 42

58 7.3.1 OTM implementation requirements 42

59 7.3.2 SharedKey Credential Calculation 44

60	7.3.3	Certificate Credential Generation.....	44
61	7.3.4	Just-Works OTM.....	44
62	7.3.5	Random PIN Based OTM.....	46
63	7.3.6	Manufacturer Certificate Based OTM.....	49
64	7.3.7	Vendor Specific OTMs.....	51
65	7.3.8	Establishing Owner Credentials.....	52
66	7.3.9	Security considerations regarding selecting an Ownership Transfer Method	
67		- Moved to OCF Onboarding Tool document.....	60
68	7.3.10	Security Profile Assignment.....	60
69	7.4	Provisioning.....	60
70	7.4.1	Provisioning Flows.....	60
71	7.5	Device Provisioning for OCF Cloud – moved to OCF Cloud Security document.....	62
72	8	Device Onboarding State Definitions.....	62
73	8.1	Device Onboarding General.....	62
74	8.2	Device Onboarding-Reset State Definition.....	64
75	8.3	Device Ready-for-OTM State Definition.....	64
76	8.4	Device Ready-for-Provisioning State Definition.....	65
77	8.5	Device Ready-for-Normal-Operation State Definition.....	65
78	8.6	Device Soft Reset State Definition.....	65
79	9	Security Credential Management.....	67
80	9.1	Preamble.....	67
81	9.2	Credential Lifecycle.....	67
82	9.2.1	Credential Lifecycle General.....	67
83	9.2.2	Creation.....	67
84	9.2.3	Deletion.....	67
85	9.2.4	Refresh.....	67
86	9.2.5	Revocation.....	67
87	9.3	Credential Types.....	68
88	9.3.1	Preamble.....	68
89	9.3.2	Pair-wise Symmetric Key Credentials.....	68
90	9.3.3	Group Symmetric Key Credentials.....	68
91	9.3.4	Asymmetric Authentication Key Credentials.....	69
92	9.3.5	Asymmetric Key Encryption Key Credentials.....	69
93	9.3.6	Certificate Credentials.....	69
94	9.3.7	Password Credentials.....	70
95	9.4	Certificate Based Key Management.....	70
96	9.4.1	Overview.....	70
97	9.4.2	X.509 Digital Certificate Profiles.....	71
98	9.4.3	Certificate Revocation List (CRL) Profile.....	80
99	9.4.4	Resource Model.....	81
100	9.4.5	Certificate Provisioning.....	81
101	9.4.6	CRL Provisioning.....	82
102	10	Device Authentication.....	84
103	10.1	Device Authentication General.....	84

104	10.2	Device Authentication with Symmetric Key Credentials	84
105	10.3	Device Authentication with Raw Asymmetric Key Credentials.....	84
106	10.4	Device Authentication with Certificates	84
107	10.4.1	Device Authentication with Certificates General.....	84
108	10.4.2	Role Assertion with Certificates	85
109	10.4.3	OCF PKI Roots	86
110	10.4.4	PKI Trust Store.....	86
111	10.4.5	Path Validation and extension processing.....	87
112	10.5	Device Authentication with OCF Cloud – moved to OCF Cloud Security	
113		document.....	88
114	11	Message Integrity and Confidentiality	89
115	11.1	Preamble	89
116	11.2	Session Protection with DTLS.....	89
117	11.2.1	DTLS Protection General.....	89
118	11.2.2	Unicast Session Semantics.....	89
119	11.2.3	Cloud Session Semantics – moved to OCF Cloud Security document	89
120	11.3	Cipher Suites	89
121	11.3.1	Cipher Suites General	89
122	11.3.2	Cipher Suites for Device Ownership Transfer	89
123	11.3.3	Cipher Suites for Symmetric Keys.....	90
124	11.3.4	Cipher Suites for Asymmetric Credentials.....	91
125	11.3.5	Cipher suites for OCF Cloud Credentials – moved to OCF Cloud Security	
126		document	91
127	12	Access Control	92
128	12.1	ACL Generation and Management	92
129	12.2	ACL Evaluation and Enforcement.....	92
130	12.2.1	ACL Evaluation and Enforcement General.....	92
131	12.2.2	Host Reference Matching	92
132	12.2.3	Resource Wildcard Matching	92
133	12.2.4	Multiple Criteria Matching	93
134	12.2.5	Subject Matching using Wildcards	93
135	12.2.6	Subject Matching using Roles.....	93
136	12.2.7	ACL Evaluation.....	94
137	13	Security Resources	96
138	13.1	Security Resources General	96
139	13.2	Device Owner Transfer Resource	97
140	13.2.1	Device Owner Transfer Resource General.....	97
141	13.2.2	Persistent and Semi-Persistent Device Identifiers.....	100
142	13.2.3	Onboarding Considerations for Device Identifier	100
143	13.2.4	OCF defined OTMs.....	101
144	13.3	Credential Resource	101
145	13.3.1	Credential Resource General.....	101
146	13.3.2	Properties of the Credential Resource	106
147	13.3.3	Key Formatting	108

148	13.3.4	Credential Refresh Method Details [Deprecated]	108
149	13.4	Certificate Revocation List	109
150	13.4.1	CRL Resource Definition	109
151	13.5	ACL Resources	109
152	13.5.1	ACL Resources General	109
153	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures.	109
154	13.5.3	ACL Resource	110
155	13.6	Access Manager ACL Resource [Deprecated]	115
156	13.7	Signed ACL Resource [Deprecated]	115
157	13.8	Provisioning Status Resource	115
158	13.9	Certificate Signing Request Resource	120
159	13.10	Roles Resource	121
160	13.11	Account Resource – moved to OCF Cloud Security document.....	122
161	13.12	Account Session Resource – moved to OCF Cloud Security document	122
162	13.13	Account Token Refresh Resource – moved to OCF Cloud Security document	122
163	13.14	Security Virtual Resources (SVRs) and Access Policy	122
164	13.15	SVRs, Discoverability and OCF Endpoints	122
165	13.16	Additional Privacy Consideration for Core and SVRs Resources	123
166	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ...	123
167	13.16.2	Privacy Protecting the Device Identifiers.....	125
168	13.16.3	Privacy Protecting the Protocol Independent Device Identifier	125
169	13.16.4	Privacy Protecting the Platform Identifier	126
170	13.17	Easy Setup Resource Device State	126
171	14	Security Hardening Guidelines/ Execution Environment Security	129
172	14.1	Preamble	129
173	14.2	Execution Environment Elements	129
174	14.2.1	Execution Environment Elements General	129
175	14.2.2	Secure Storage.....	129
176	14.2.3	Secure execution engine	132
177	14.2.4	Trusted input/output paths	132
178	14.2.5	Secure clock.....	132
179	14.2.6	Approved algorithms.....	132
180	14.2.7	Hardware tamper protection.....	133
181	14.3	Secure Boot.....	133
182	14.3.1	Concept of software module authentication.....	133
183	14.3.2	Secure Boot process	135
184	14.3.3	Robustness Requirements.....	135
185	14.4	Attestation	135
186	14.5	Software Update	135
187	14.5.1	Overview:	135
188	14.5.2	Recognition of Current Differences	136
189	14.5.3	Software Version Validation	137
190	14.5.4	Software Update.....	137
191	14.5.5	Recommended Usage.....	137

192	14.6	Non-OCF Endpoint interoperability.....	138
193	14.7	Security Levels	138
194	14.8	Security Profiles.....	139
195	14.8.1	Security Profiles General.....	139
196	14.8.2	Identification of Security Profiles (Normative)	139
197	14.8.3	Security Profiles	141
198	15	Device Type Specific Requirements.....	146
199	15.1	Bridging Security	146
200	15.1.1	Universal Requirements for Bridging to another Ecosystem	146
201	15.1.2	Additional Security Requirements specific to Bridged Protocols	147
202	Annex A (informative)	Access Control Examples.....	149
203	A.1	Example OCF ACL Resource	149
204	Annex B (Informative)	Execution Environment Security Profiles	150
205	Annex C (normative)	Resource Type definitions.....	151
206	C.1	List of Resource Type definitions	151
207	C.2	Access Control List-2.....	151
208	C.2.1	Introduction	151
209	C.2.2	Well-known URI	151
210	C.2.3	Resource type	151
211	C.2.4	OpenAPI 2.0 definition.....	151
212	C.2.5	Property definition	160
213	C.2.6	CRUDN behaviour	161
214	C.3	Credential.....	161
215	C.3.1	Introduction	161
216	C.3.2	Well-known URI.....	161
217	C.3.3	Resource type	161
218	C.3.4	OpenAPI 2.0 definition.....	162
219	C.3.5	Property definition	172
220	C.3.6	CRUDN behaviour	172
221	C.4	Certificate Revocation.....	172
222	C.4.1	Introduction	172
223	C.4.2	Well-known URI.....	172
224	C.4.3	Resource type	172
225	C.4.4	OpenAPI 2.0 definition.....	173
226	C.4.5	Property definition	175
227	C.4.6	CRUDN behaviour	175
228	C.5	Certificate Signing Request.....	175
229	C.5.1	Introduction	175
230	C.5.2	Well-known URI.....	175
231	C.5.3	Resource type	176
232	C.5.4	OpenAPI 2.0 definition.....	176
233	C.5.5	Property definition	177
234	C.5.6	CRUDN behaviour	178
235	C.6	Device Owner Transfer Method.....	178

236	C.6.1	Introduction	178
237	C.6.2	Well-known URI	178
238	C.6.3	Resource type	178
239	C.6.4	OpenAPI 2.0 definition.....	178
240	C.6.5	Property definition	181
241	C.6.6	CRUDN behaviour	183
242	C.7	Device Provisioning Status	184
243	C.7.1	Introduction	184
244	C.7.2	Well-known URI	184
245	C.7.3	Resource type	184
246	C.7.4	OpenAPI 2.0 definition.....	184
247	C.7.5	Property definition	188
248	C.7.6	CRUDN behaviour	192
249	C.8	Asserted Roles	192
250	C.8.1	Introduction	192
251	C.8.2	Well-known URI	193
252	C.8.3	Resource type	193
253	C.8.4	OpenAPI 2.0 definition.....	193
254	C.8.5	Property definition	202
255	C.8.6	CRUDN behaviour	202
256	C.9	Security Profile	202
257	C.9.1	Introduction	202
258	C.9.2	Well-known URI	202
259	C.9.3	Resource type	202
260	C.9.4	OpenAPI 2.0 definition.....	202
261	C.9.5	Property definition	205
262	C.9.6	CRUDN behaviour	205
263	Annex D (informative)	OID definitions	206
264	Annex E (informative)	Security considerations specific to Bridged Protocols	208
265	E.1	Security Considerations specific to the AllJoyn Protocol	208
266	E.2	Security Considerations specific to the Bluetooth LE Protocol.....	208
267	E.3	Security Considerations specific to the oneM2M Protocol	208
268	E.4	Security Considerations specific to the U+ Protocol	209
269	E.5	Security Considerations specific to the Z-Wave Protocol.....	209
270	E.6	Security Considerations specific to the Zigbee Protocol	210
271			

FIGURES

272	
273	Figure 1 – OCF Interaction..... 22
274	Figure 2 – OCF Layers 24
275	Figure 3 – OCF Security Enforcement Points 26
276	Figure 4 – Use case-1 showing simple ACL enforcement 28
277	Figure 5 – Use case 2: A policy for the requested Resource is missing 29
278	Figure 6 – Example Resource definition with opaque Properties 30
279	Figure 7 – Property Level Access Control 30
280	Figure 8 – Onboarding Overview..... 32
281	Figure 9 – OCF Onboarding Process 33
282	Figure 10 – OCF's SRM Architecture 36
283	Figure 11 – Discover New Device Sequence 43
284	Figure 12 – A Just Works OTM 45
285	Figure 13 – Random PIN-based OTM 47
286	Figure 14 – Manufacturer Certificate Based OTM Sequence 50
287	Figure 15 – Vendor-specific Owner Transfer Sequence..... 52
288	Figure 16 – Establish Device Identity Flow..... 53
289	Figure 17 – Owner Credential Selection Provisioning Sequence 54
290	Figure 18 – Symmetric Owner Credential Provisioning Sequence 55
291	Figure 19 – Asymmetric Owner Credential Provisioning Sequence..... 56
292	Figure 20 – Configure Device Services 58
293	Figure 21 – Provision New Device for Peer to Peer Interaction Sequence..... 59
294	Figure 22 – Example of Client-directed provisioning..... 61
295	Figure 23 – Device state model..... 63
296	Figure 24 – Client-directed Certificate Transfer 82
297	Figure 25 – Client-directed CRL Transfer 83
298	Figure 26 – Asserting a role with a certificate role credential. 86
299	Figure 27 – OCF Security Resources 96
300	Figure 28 – "/oic/sec/cred" Resource and Properties..... 97
301	Figure 29 – "/oic/sec/acl2" Resource and Properties 97
302	Figure 30 – Example of Soft AP and Easy Setup Resource in different Device states 127
303	Figure 31 – Software Module Authentication 134
304	Figure 32 – Verification Software Module 134
305	Figure 33 – Software Module Authenticity 135
306	Figure 34 – State transitioning diagram for software download 136
307	Figure A-1 – Example "/oic/sec/acl2" Resource..... 149
308	Figure E-1 Security Considerations for BLE Bridge 208
309	Figure E-2 Security Considerations for Z-Wave Bridge..... 210
310	Figure E-3 Security Considerations for Zigbee Bridge 211

312	Tables	
313	Table 1 – Discover New Device Details.....	43
314	Table 2 – A Just Works OTM Details.....	45
315	Table 3 – Random PIN-based OTM Details.....	47
316	Table 4 – Manufacturer Certificate Based OTM Details.....	51
317	Table 5 – Vendor-specific Owner Transfer Details.....	52
318	Table 6 – Establish Device Identity Details.....	53
319	Table 7 – Owner Credential Selection Details.....	54
320	Table 8 – Symmetric Owner Credential Assignment Details.....	55
321	Table 9 – Asymmetric Owner Credential Assignment Details.....	57
322	Table 10 – Configure Device Services Detail.....	58
323	Table 11 – Provision New Device for Peer to Peer Details.....	59
324	Table 12 – Steps describing Client -directed provisioning.....	61
325	Table 13 – X.509 v1 fields for Root CA Certificates.....	71
326	Table 14 - X.509 v3 extensions for Root CA Certificates.....	72
327	Table 15 - X.509 v1 fields for Intermediate CA Certificates.....	72
328	Table 16 – X.509 v3 extensions for Intermediate CA Certificates.....	72
329	Table 17 – X.509 v1 fields for End-Entity Certificates.....	73
330	Table 18 – X.509 v3 extensions for End-Entity Certificates.....	73
331	Table 19 – ACE2 Wildcard Matching Strings Description.....	92
332	Table 20 – Definition of the "/oic/sec/doxm" Resource.....	98
333	Table 21 – Properties of the "/oic/sec/doxm" Resource.....	98
334	Table 22 – Properties of the "oic.sec.didtype" type.....	99
335	Table 23 – Properties of the "oic.sec.doxmtype" type.....	101
336	Table 24 – Definition of the "/oic/sec/cred" Resource.....	102
337	Table 25 – Properties of the "/oic/sec/cred" Resource.....	103
338	Table 26 – Properties of the "oic.sec.creds" Property.....	104
339	Table 27: Properties of the "oic.sec.credusagetype" Property.....	105
340	Table 28 – Properties of the "oic.sec.pubdatatype" Property.....	105
341	Table 29 – Properties of the "oic.sec.privdatatype" Property.....	105
342	Table 30 – Properties of the "oic.sec.optdatatype" Property.....	106
343	Table 31 – Definition of the "oic.sec.roletype" type.....	106
344	Table 32 – 128-bit symmetric key.....	108
345	Table 33 – 256-bit symmetric key.....	108
346	Table 34 – Definition of the "oic/sec/crl" Resource.....	109
347	Table 35 – Properties of the "oic/sec/crl" Resource.....	109
348	Table 36 – BNF Definition of OCF ACL.....	109

349	Table 37 – Value Definition of the "oic.sec.crudntype" Property	111
350	Table 38 – Definition of the "oic/sec/acl2" Resource	111
351	Table 39 – Properties of the "/oic/sec/acl2" Resource	112
352	Table 40 – "oic.sec.ace2" data type definition.	113
353	Table 41 – "oic.sec.ace2.resource-ref" data type definition.	113
354	Table 42 – Value definition "oic.sec.conntype" Property.....	113
355	Table 43 – Definition of the "/oic/sec/pstat" Resource	115
356	Table 44 – Properties of the "/oic/sec/pstat" Resource.....	116
357	Table 45 – Properties of the ".oic.sec.dostype" Property.....	117
358	Table 46 – Definition of the "oic.sec.dpmttype" Property	119
359	Table 47 – Value Definition of the "oic.sec.dpmttype" Property (Low-Byte)	119
360	Table 48 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte).....	119
361	Table 49 – Definition of the "oic.sec.pomtype" Property	119
362	Table 50 – Value Definition of the "oic.sec.pomtype" Property	120
363	Table 51 – Definition of the "/oic/sec/csr" Resource	120
364	Table 52 – Properties of the "oic.r.csr" Resource	120
365	Table 53 – Definition of the "/oic/sec/roles" Resource	122
366	Table 54 – Properties of the "/oic/sec/roles" Resource.....	122
367	Table 55 – Core Resource Properties Access Modes given various Device States	124
368	Table 56 – Examples of Sensitive Data.....	130
369	Table 57 – Description of the software update bits.....	136
370	Table 58 – Definition of the "/oic/sec/sp" Resource	140
371	Table 59 – Properties of the "/oic/sec/sp" Resource.....	140
372	Table 60 – Dependencies of VOD Behaviour on Bridge state, as clarification of	
373	accompanying text.....	147
374	Table B.1 – OCF Security Profile	150
375	Table C.1 – Alphabetized list of security resources	151
376	Table C-1 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".	160
377	Table C-2 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".	161
378	Table C-3 – The Property definitions of the Resource with type "rt" = "oic.r.cred".	172
379	Table C-4 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".	172
380	Table C-5 – The Property definitions of the Resource with type "rt" = "oic.r.crl".	175
381	Table C-6 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".	175
382	Table C-7 – The Property definitions of the Resource with type "rt" = "oic.r.csr".....	177
383	Table C-8 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".	178
384	Table C-9 – The Property definitions of the Resource with type "rt" = "oic.r.doxm".	181
385	Table C-10 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".	183
386	Table C-11 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".	188
387	Table C-12 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".	192

388	Table C-13 – The Property definitions of the Resource with type "rt" = "oic.r.roles".....	202
389	Table C-14 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".	202
390	Table C-15 – The Property definitions of the Resource with type "rt" = "oic.r.sp".....	205
391	Table C-16 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".	205
392	Table E.1 GAP security mode	208
393	Table E.2 TLS 1.2 Cipher Suites used by U+	209
394	Table E.3 Z-Wave Security Class.....	210
395	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers	211
396		
397		

398 **1 Scope**

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

403 **2 Normative References**

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

407 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)
408 Specification -- Part 1: Core specification
409 <https://www.iso.org/standard/53238.html>
410 Latest version available at:
411 https://openconnectivity.org/specs/OCF_Core_Specification.pdf

412 ISO/IEC 30118-3:2018 Information technology -- Open Connectivity Foundation (OCF)
413 Specification -- Part 3: Bridging specification
414 <https://www.iso.org/standard/74240.html>
415 Latest version available at:
416 https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

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

421 OCF Device to Cloud Services Specification
422 Latest version available at:
423 https://openconnectivity.org/specs/OCF_Device_To_Cloud_Services_Specification.pdf

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

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

427 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September
428 2000, <https://tools.ietf.org/html/rfc2898>

429 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November
430 2000, <https://tools.ietf.org/html/rfc2986>

431 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December
432 2005, <https://tools.ietf.org/html/rfc4279>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

467

468 **3 Terms, definitions, and abbreviated terms**

469 **3.1 Terms and definitions**

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

472 ISO and IEC maintain terminological databases for use in standardization at the following
473 addresses:

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

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

476 **3.1.1**

477 **Access Management Service (AMS)**

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

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

481 **3.1.2**

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

483 **3.1.3**

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

485 **3.1.4**

486 **Client**

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

488 **3.1.5**

489 **Credential Management Service (CMS)**

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

492 **3.1.6**

493 **Device**

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

495 **3.1.7**

496 **Device Class**

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

499 **3.1.8**

500 **Device ID**

501 a stack instance identifier.

502 **3.1.9**

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

504 a logical entity that establishes device ownership

505 **3.1.10**

506 **3.1.11 Device Registration – moved to OCF Cloud Security document**

507 **End-Entity**

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

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

510 **3.1.12**

511 **Entity**

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

513 **3.1.13**
514 **OCF Interface**
515 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

516 **3.1.14**
517 **Intermediary**
518 a Device that implements both Client and Server roles and may perform protocol translation, virtual
519 device to physical device mapping or Resource translation

520 **3.1.15**
521 **OCF Cipher Suite**
522 a set of algorithms and parameters that define the cryptographic functionality of a Device. The OCF
523 Cipher Suite includes the definition of the public key group operations, signatures, and specific
524 hashing and encoding used to support the public key.

525 **3.1.16**
526 **OCF Cloud User – moved to OCF Cloud Security spec**

527 **3.1.17**
528 **OCF Rooted Certificate Chain**
529 a collection of X.509 v3 certificates in which each certificate chains to a trust anchor certificate
530 which has been issued by a certificate authority under the direction, authority, and approval of the
531 Open Connectivity Foundation Board of Directors as a trusted root for the OCF ecosystem.

532 **3.1.18**
533 **Onboarding Tool (OBT)**
534 a tool that implements DOTS(3.1.9), AMS(3.1.1) and CMS(3.1.5) functionality

535 **3.1.19**
536 **Out of Band Communication Channel**
537 any mechanism for delivery of a secret from one party to another, not specified by OCF

538 **3.1.20**
539 **Owner Credential (OC)**
540 credential, provisioned by an OBT(3.1.18) to a Device during onboarding, for the purposes of
541 mutual authentication of the Device and OBT(3.1.18) during subsequent interactions

542 **3.1.21**
543 **Platform ID**
544 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

545 **3.1.22**
546 **Property**
547 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

548 **3.1.23**
549 **Resource**
550 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

551 **3.1.24**
552 **Role (Network context)**
553 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

554 **3.1.25**
555 **Role Identifier**
556 a Property of an OCF credentials Resource or element in a role certificate that identifies a privileged
557 role that a Server Device associates with a Client Device for the purposes of making authorization
558 decisions when the Client Device requests access to Device Resources.

559 **3.1.26**
560 **Secure Resource Manager (SRM)**
561 a module in the OCF Core that implements security functionality that includes management of
562 security Resources such as ACLs, credentials and Device owner transfer state.

563 **3.1.27**
564 **Security Virtual Resource (SVR)**
565 a resource supporting security features.

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

567 **3.1.28**
568 **Server**

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

570 **3.1.29**
571 **Trust Anchor**
572 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.
573 a Device and an OBT(3.1.18)) can assume trust

574 **3.1.30**
575 **Unique Authenticable Identifier**
576 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is
577 used to create the Device ID.

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

579 **3.1.31**
580 **Device Configuration Resource (DCR)**
581 a Resource that is any of the following:

- 582 a) a Discovery Core Resource, or
- 583 b) a Security Virtual Resource, or
- 584 c) a Wi-Fi Easy Setup Resource ("oic.r.easyssetup", "oic.r.wificonf", "oic.r.devconf"), or
- 585 d) a CoAP Cloud Configuration Resource ("oic.r.coapcloudconf"), or
- 586 e) a Software Update Resource ("oic.r.softwareupdate"), or
- 587 f) a Maintenance Resource ("oic.wk.mnt").

588 **3.1.32**
589 **Non-Configuration Resource (NCR)**
590 a Resource that is not a Device Configuration Resource (3.1.31).

591 **3.1.33**
592 **Bridged Device**

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

594 **3.1.34**
595 **Bridged Protocol**

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

597 **3.1.35**
598 **Bridge**

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

600 **3.1.36**
601 **Bridging Platform**

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

603 **3.1.37**
604 **Virtual Bridged Device**
605 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

606 **3.1.38**
607 **Virtual OCF Device**
608 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

609 **3.1.39**
610 **OCF Security Domain**
611 set of onboarded OCF Devices that are provisioned with credentialing information for confidential
612 communication with one another

613 **3.1.40**
614 **Owned (or "in Owned State")**
615 having the "owned" Property of the "/oic/sec/doxm" resource equal to "TRUE"

616 **3.1.41**
617 **Unowned (or "in Unowned State")**
618 having the "owned" Property of the "/oic/sec/doxm" resource equal to "FALSE"

619 **3.1.42 OCF Onboarding**
620 initial establishment of ownership over a Device, and initial provisioning of the Device for normal
621 operation

622 **3.2 Abbreviated terms**

623 **3.2.1**
624 **AC**
625 Access Control

626 **3.2.2**
627 **ACE**
628 Access Control Entry

629 **3.2.3**
630 **ACL**
631 Access Control List

632 **3.2.4**
633 **AES**
634 Advanced Encryption Standard
635 Note 1 to entry: See NIST FIPS 197, "Advanced Encryption Standard (AES)"

636 **3.2.5**
637 **AMS**
638 Access Management Service

639 **3.2.6**
640 **CMS**
641 Credential Management Service

642 **3.2.7**
643 **CRUDN**
644 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

645 **3.2.8**
646 **CSR**
647 Certificate Signing Request

648 **3.2.9**
649 **CVC**
650 Code Verification Certificate

651 **3.2.10**
652 **ECC**
653 Elliptic Curve Cryptography

654 **3.2.11**
655 **ECDSA**
656 Elliptic Curve Digital Signature Algorithm

657 **3.2.12**
658 **EKU**
659 Extended Key Usage

660 **3.2.13**
661 **EPC**
662 Embedded Platform Credential

663 **3.2.14**
664 **EPK**
665 Embedded Public Key

666 **3.2.15**
667 **DOTS**
668 Device Ownership Transfer Service

669 **3.2.16**
670 **DPKP**
671 Dynamic Public Key Pair

672 **3.2.17**
673 **ID**
674 Identity/Identifier

675 **3.2.18**
676 **JSON**
677 JavaScript Object Notation.

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

679 **3.2.19**
680 **JWS**
681 JSON Web Signature.

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

683 **3.2.20**
684 **KDF**
685 Key Derivation Function

686 **3.2.21**
687 **MAC**
688 Message Authentication Code

689 **3.2.22**
690 **MITM**
691 Man-in-the-Middle

692 **3.2.23**
693 **NVRAM**
694 Non-Volatile Random-Access Memory

695 **3.2.24**
696 **OC**
697 Owner Credential

698 **3.2.25**
699 **OCSP**
700 Online Certificate Status Protocol

701 **3.2.26**
702 **OBT**
703 Onboarding Tool

704 **3.2.27**
705 **OID**
706 Object Identifier

707 **3.2.28**
708 **OTM**
709 Owner Transfer Method

710 **3.2.29**
711 **OWASP**
712 Open Web Application Security Project.

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

714 **3.2.30**
715 **PE**
716 Policy Engine

717 **3.2.31**
718 **PIN**
719 Personal Identification Number

720 **3.2.32**
721 **PPSK**
722 PIN-authenticated pre-shared key

723 **3.2.33**
724 **PRF**
725 Pseudo Random Function

726 **3.2.34**
727 **PSI**
728 Persistent Storage Interface

729 **3.2.35**
730 **PSK**
731 Pre Shared Key

732 **3.2.36**
733 **RBAC**
734 Role Based Access Control

735 **3.2.37**
736 **RM**
737 Resource Manager

738 **3.2.38**
739 **RNG**
740 Random Number Generator

741 **3.2.39**
742 **SBAC**
743 Subject Based Access Control

744 **3.2.40**
745 **SEE**
746 Secure Execution Environment

747 **3.2.41**
748 **SRM**
749 Secure Resource Manager

750 **3.2.42**
751 **SVR**
752 Security Virtual Resource

753 **3.2.43**
754 **SW**
755 Software

756 **3.2.44**
757 **UAID**
758 Unique Authenticable Identifier

759 **3.2.45**
760 **URI**
761 Uniform Resource Identifier

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

763 **3.2.46**
764 **VOD**
765 Virtual OCF Device

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

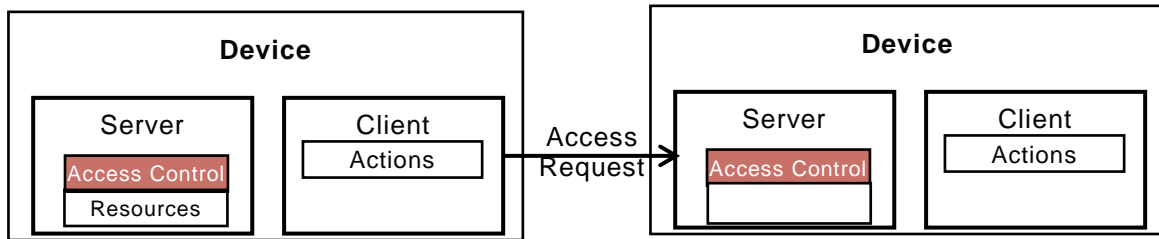
767 **4 Document Conventions and Organization**

768 **4.1 Conventions**

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

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

772 Figure 1 depicts interaction between OCF Devices.



773

774

Figure 1 – OCF Interaction

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

779 **4.2 Notation**

780 In this document, features are described as required, recommended, allowed or DEPRECATED as
 781 follows:

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

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

786 **Recommended (or should).**

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

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

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

796 **Conditionally allowed (CA)**

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

799 **Conditionally required (CR)**

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

803 **DEPRECATED**

804 Although these features are still described in this document, they should not be implemented except
 805 for backward compatibility. The occurrence of a deprecated feature during operation of an

806 implementation compliant with the current document has no effect on the implementation's
807 operation and does not produce any error conditions. Backward compatibility may require that a
808 feature is implemented and functions as specified but it shall never be used by implementations
809 compliant with this document.

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

811 Words that are emphasized are printed in italic.

812 **4.3 Data types**

813 See ISO/IEC 30118-1:2018.

814 **4.4 Document structure**

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

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

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

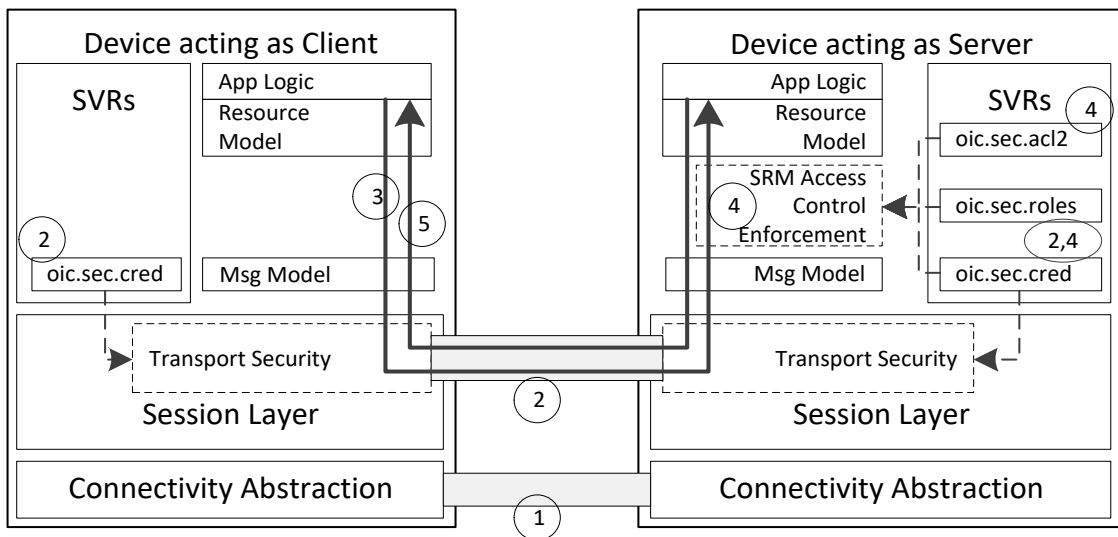
821

822 **5 Security Overview**

823 **5.1 Preamble**

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

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



832

833 **Figure 2 – OCF Layers**

- 834 1) The Client establishes a network connection to the Server (Device holding the Resources). The
 835 connectivity abstraction layer ensures the Devices are able to connect despite differences in
 836 connectivity options.
- 837 2) The Devices (e.g. Server and Client) exchange messages either with or without a mutually-
 838 authenticated secure channel between the two Devices.
- 839 a) The "/oic/sec/cred" Resource on each Devices holds the credentials used for mutual
 840 authentication and (when applicable) certificate validation.
- 841 b) Messages received over a secured channel are associated with a "deviceUUID". In the case
 842 of a certificate credential, the "deviceUUID" is in the certificate received from the other
 843 Device. In the case of a symmetric key credential, the "deviceUUID" is configured with the
 844 credential in the "/oic/sec/cred" Resource.
- 845 c) The Server can associate the Client with any number of roleid. In the case of mutual
 846 authentication using a certificate, the roleid (if any) are provided in role certificates; these
 847 are configured by the Client to the Server. In the case of a symmetric key, the allowed roleid
 848 (if any) are configured with the credential in the "/oic/sec/cred" Resource.

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

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

852 4) The Server receives the request.

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

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

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

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

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

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

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

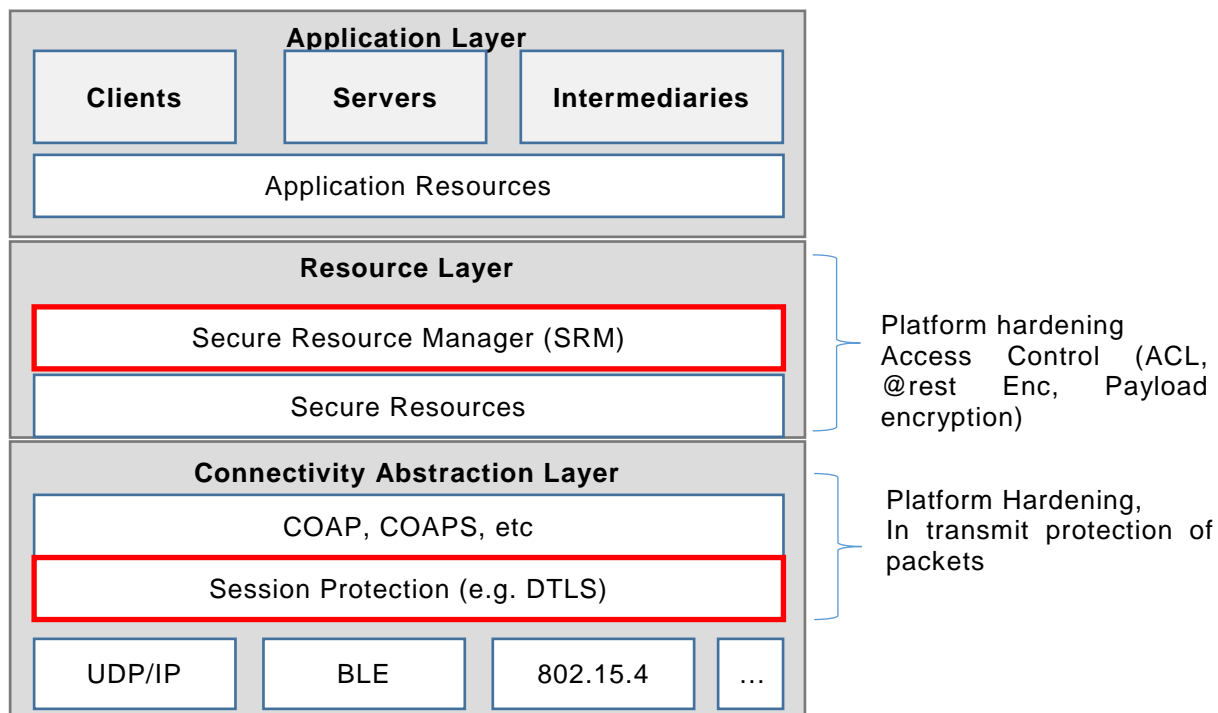
869 5) The Server sends a response back to the Client.

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

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

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

884 Figure 3 depicts OCF Security Enforcement Points.



885

886

887

Figure 3 – OCF Security Enforcement Points

888

5.2 Access Control

889

890

891

892

893

894

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.

895

896

897

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:

898

899

900

901

902

- 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.

903

904

905

906

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.

907

908

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.

909

910

911

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

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

914 Example Wildcard Matching Policy:

```
915 "aclist2": [  
916 {  
917   "subject": {"conntype": "anon-clear" },  
918   "resources": [  
919     { "wc": "*" }  
920   ],  
921   "permission": 31  
922 },  
923 {  
924   "subject": {"conntype": "auth-crypt" },  
925   "resources": [  
926     { "wc": "*" }  
927   ],  
928   "permission": 31  
929 },  
930 ]
```

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

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

939 **5.2.1 ACL Architecture**

940 **5.2.1.1 ACL Architecture General**

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

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

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

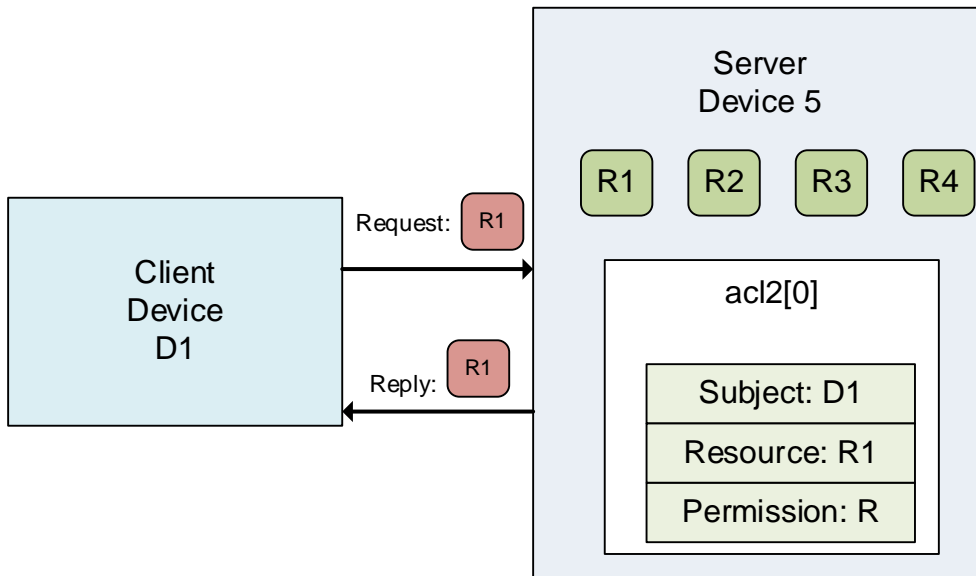
956 **5.2.1.2 Use of local ACLs**

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

959 The following use cases describe the operation of access control

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

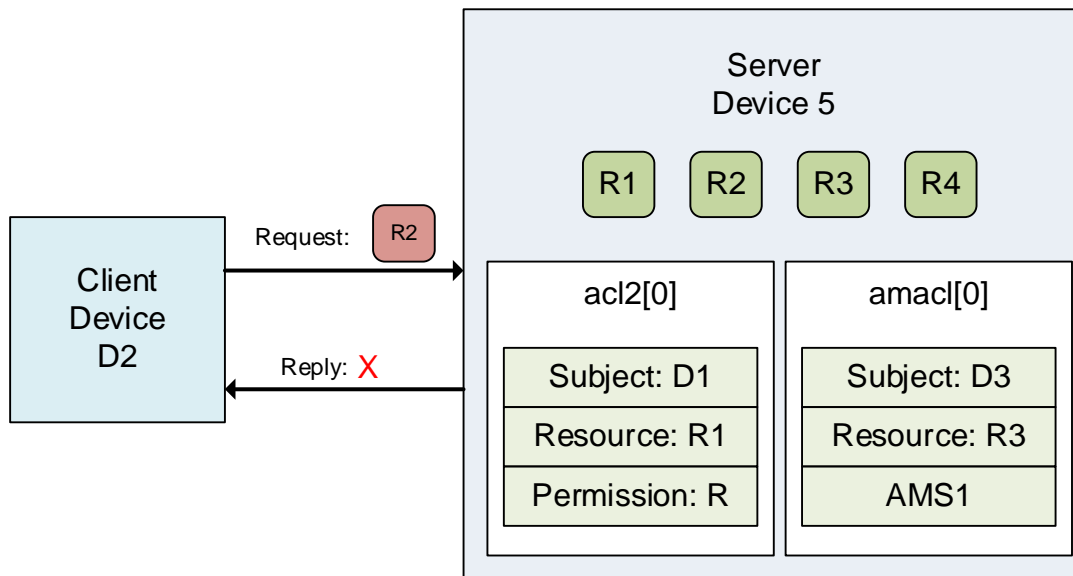
964



965

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

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



970

971 **Figure 5 – Use case 2: A policy for the requested Resource is missing**972 **5.2.1.3 Use of AMS**

973 AMS improves ACL policy management. However, they can become a central point of failure. Due to network latency overhead, ACL processing may be slower through an AMS.
974

975 AMS centralizes access control decisions, but Server Devices retain enforcement duties.

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

978 **5.2.2 Access Control Scoping Levels**

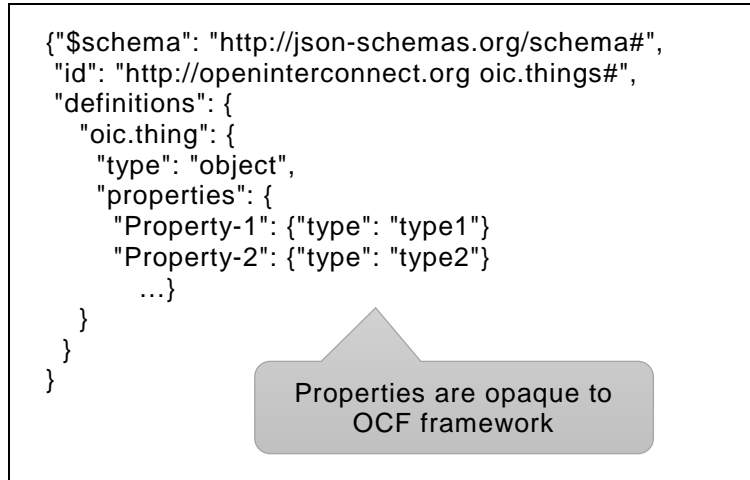
979 **Group Level Access** - Group scope means applying AC to the group of Devices that are grouped
980 for a specific context. Group Level Access means all group members have access to group data
981 but non-group members must be granted explicit access. Group level access is implemented using
982 Role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

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

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

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

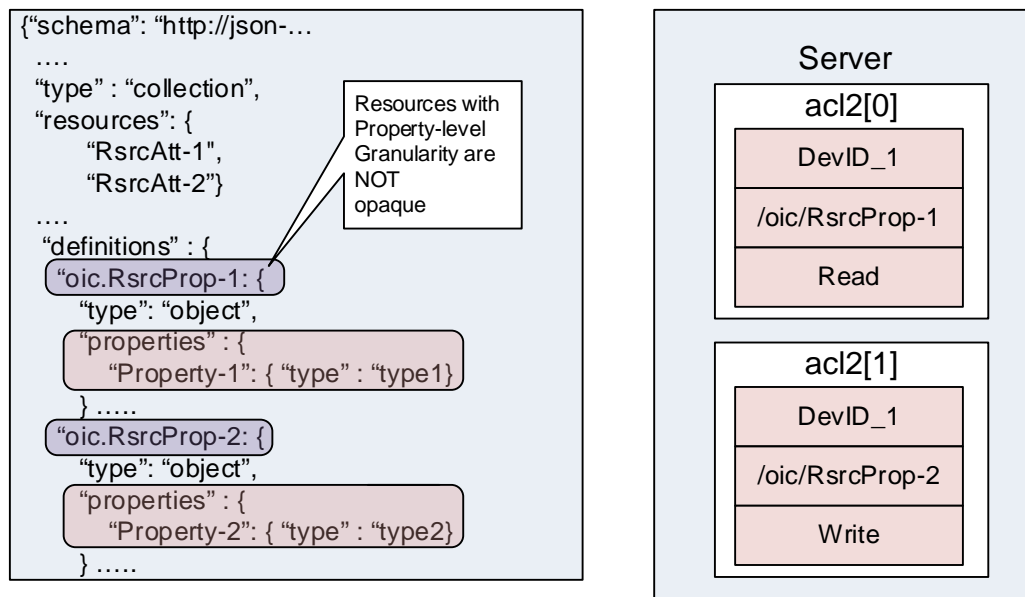
993 Controlling access to static Resources where it is impractical to redesign the Resource, it may
 994 appropriate to introduce a collection Resource that references the child Resources having separate
 995 access permissions. An example is shown Figure 6, where an "oic.thing" Resource has two
 996 properties: Property-1 and Property-2 that would require different permissions.



997

998 **Figure 6 – Example Resource definition with opaque Properties**

999 Currently, OCF framework treats property level information as opaque; therefore, different
 1000 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1
 1001 and write-only permission to Property-2). Thus, as shown in Figure 7, the "oic.thing" is split into
 1002 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be
 1003 achieved through use of Resource-level ACLs.



1004

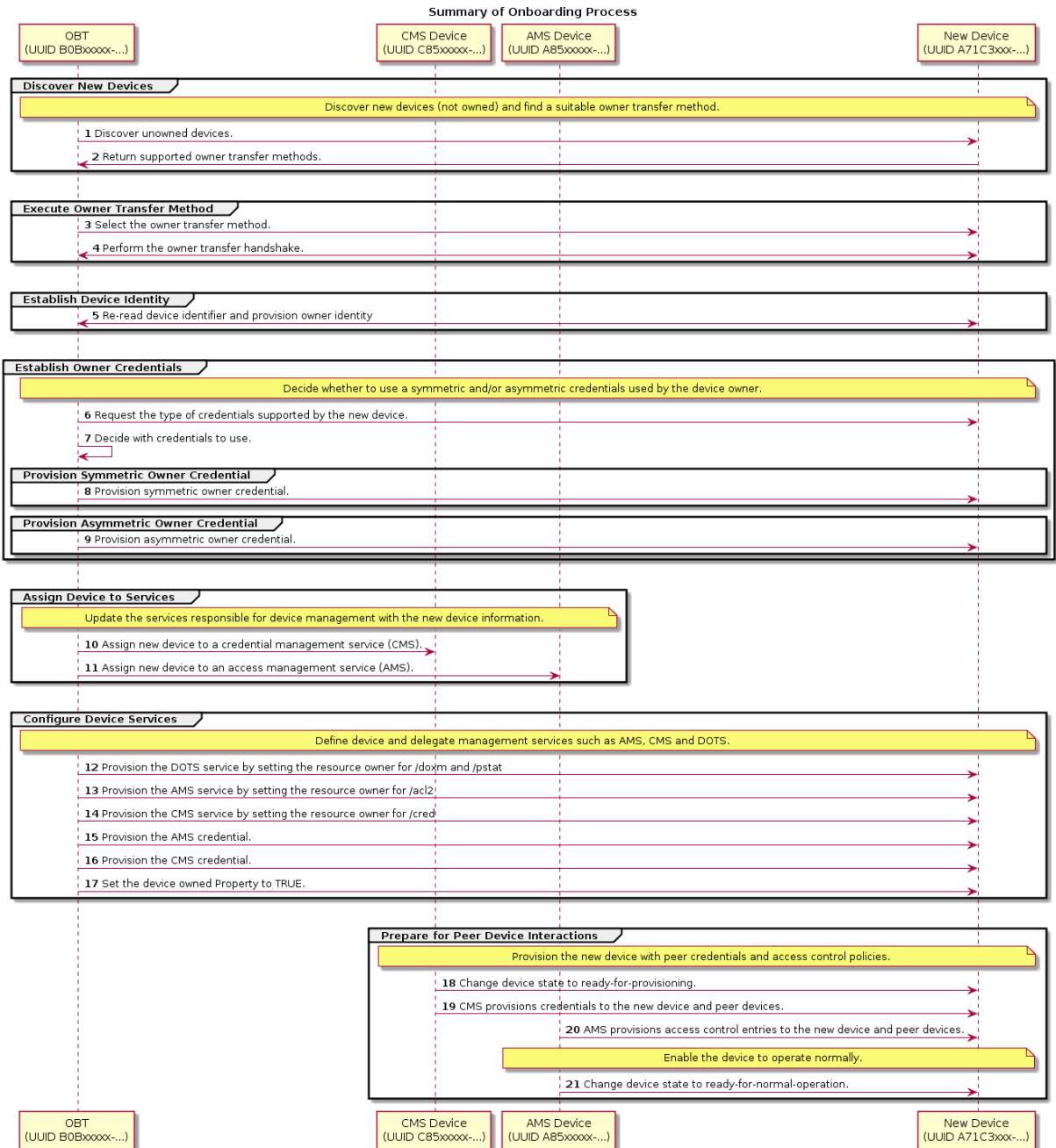
1005 **Figure 7 – Property Level Access Control**

1006 **5.3 Onboarding Overview**

1007 **5.3.1 Onboarding General**

1008 Before a Device becomes operational in an OCF environment and is able to interact with other
 1009 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to
 1010 configure the ownership where the legitimate user that owns/purchases the Device uses an
 1011 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to
 1012 establish ownership. Once ownership is established, the OBT becomes the mechanism through
 1013 which the Device can then be provisioned, at the end of which the Device becomes operational
 1014 and is able to interact with other Devices in an OCF environment.

1015 Figure 8 depicts Onboarding Overview.



1016

1017

Figure 8 – Onboarding Overview

1018 This clause explains the onboarding and security provisioning process but leaves the provisioning
1019 of non-security aspects to other OCF documents. In the context of security, all Devices are required
1020 to be provisioned with minimal security configuration that allows the Device to securely
1021 interact/communicate with other Devices in an OCF environment. This minimal security
1022 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified
1023 in 7.5.

1024 Onboarding and provisioning implementations could utilize services defined outside this document,
1025 it is expected that in using other services, trust between the device being onboarded and the
1026 various tools is not transitive. This implies that the device being onboarded will individually
1027 authenticate the credentials of each and every tool used during the onboarding process; that the
1028 tools not share credentials or imply a trust relationship where one has not been established.

1029 **5.3.2 Onboarding Steps**

1030 The flowchart in Figure 9 shows the typical steps that are involved during onboarding. Although
1031 onboarding may include a variety of non-security related steps, the diagram focus is mainly on the
1032 security related configuration to allow a new Device to function within an OCF environment.
1033 Onboarding typically begins with the Device becoming an Owned Device followed by configuring
1034 the Device for the environment that it will operate in. This would include setting information such
1035 as who can access the Device and what actions can be performed as well as what permissions the
1036 Device has for interacting with other Devices.

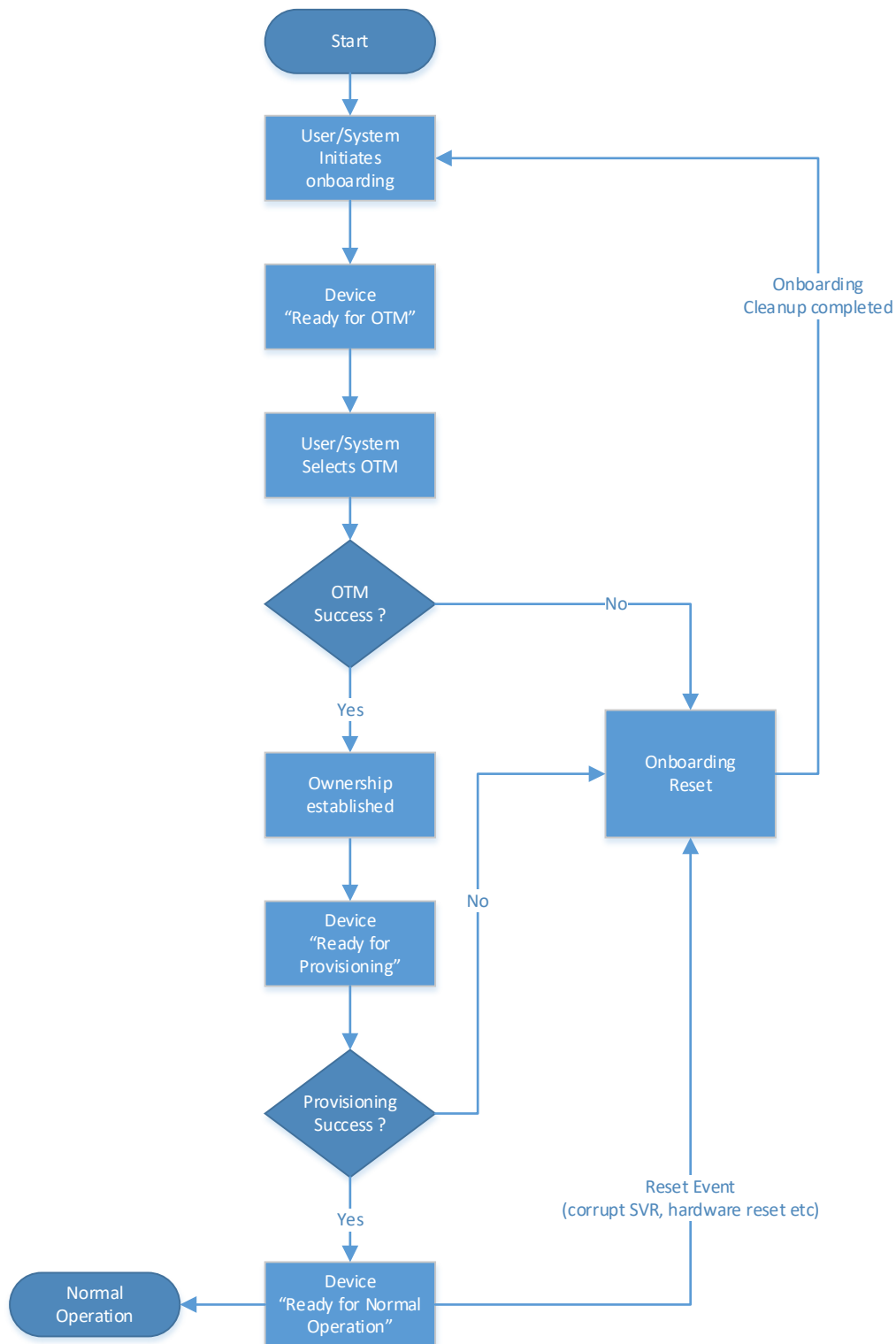


Figure 9 – OCF Onboarding Process

1037

1038

1039 5.3.3 Establishing a Device Owner

1040 The objective behind establishing Device ownership is to allow the legitimate user that
 1041 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

1042 through the use of a DOTS that includes the creation of an ownership context between the new
1043 Device and the DOTS and asserts operational control and management of the Device. The DOTS
1044 is hosted on an OBT.

1045 The DOTS uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term
1046 owner transfer is used since it is assumed that even for a new Device, the ownership is transferred
1047 from the manufacturer/provider of the Device to the buyer/legitimate user of the new Device.

1048 An OTM establishes a new owner (the operator of DOTS) that is authorized to manage the Device.
1049 Owner transfer establishes the following

- 1050 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"
1051 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during
1052 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property
1053 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the
1054 identity of Device as part of ownership transfer.
- 1055 – The Device owner establishes trust in the Device through the OTM.
- 1056 – Preparing the Device for provisioning by providing credentials that may be needed.

1057 **5.3.4 Provisioning for Normal Operation**

1058 Once the Device has the necessary information to initiate provisioning, the next step is to provision
1059 additional security configuration that allows the Device to become operational. This can include
1060 setting various parameters and may also involve multiple steps. Also provisioning of ACL's for the
1061 various Resources hosted by the Server on the Device is done at this time. The provisioning step
1062 is not limited to this stage only. Device provisioning can happen at multiple stages in the Device's
1063 operational lifecycle. However specific security related provisioning of Resource and Property state
1064 would likely happen at this stage at the end of which, each Device reaches the Onboarded Device
1065 "Ready for Normal Operation" State. The "Ready for Normal Operation" State is expected to be
1066 consistent and well defined regardless of the specific OTM used or regardless of the variability in
1067 what gets provisioned. However individual OTM mechanisms and provisioning steps may specify
1068 additional configuration of Resources and Property states. The minimal mandatory configuration
1069 required for a Device to be in "Ready for Normal Operation" state is specified in 8.

1070 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview – moved to** 1071 **OCF Cloud Security document**

1072 This clause is intentionally left blank.

1073 **5.3.6 OCF Compliance Management System**

1074 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that
1075 provides Certification status and information for OCF Devices.

1076 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:
1077 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1078 The OBT shall possess the Root Certificate needed to enable https connection to the URI
1079 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1080 The OBT should periodically refresh its copy of the CPL via the URI
1081 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security
1082 Domain owner policy requirements.

1083 **5.4 Provisioning**

1084 **5.4.1 Provisioning General**

1085 In general, provisioning may include processes during manufacturing and distribution of the Device
1086 as well as processes after the Device has been brought into its intended environment (parts of
Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 34

1087 onboarding process). In this document, security provisioning includes, processes after ownership
1088 transfer (even though some activities during ownership transfer and onboarding may lead to
1089 provisioning of some data in the Device) configuration of credentials for interacting with
1090 provisioning services, configuration of any security related Resources and credentials for dealing
1091 with any services that the Device need to contact later on.

1092 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to
1093 be provisioned with proper security credentials and parameters for regular operation. These
1094 parameters can include:

- 1095 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1096 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the
1097 same OBT, but may be part of AMS in future.

1098 Devices are aware of their security provisioning status. Self-awareness allows them to be proactive
1099 about provisioning or re-provisioning security Resources as needed to achieve the devices
1100 operational goals.

1101 **5.4.2 Provisioning other services**

1102 To be able to support the use of potentially different device management service hosts, each Device
1103 Secure Virtual Resource (SVR) has an associated Resource owner identified in the Resource's
1104 rowneruuid Property.

1105 The "rowneruuid" Property of the "/oic/sec/doxm" and "/oic/sec/pstat" resources identifies the
1106 DOTS.

1107 The "rowneruuid" Property of the "/oic/sec/cred" resource identifies the CMS.

1108 The "rowneruuid" Property of the "/oic/sec/acl2" resource identifies the AMS.

1109 The DOTS provisions credentials that enable secure connections between OCF Services and the
1110 new Device. The DOTS initiates client-directed provisioning by signaling the OCF Service.

1111 **5.4.3 Provisioning Credentials for Normal Operation**

1112 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1113 – Pairwise symmetric keys
- 1114 – Group symmetric keys
- 1115 – Certificates
- 1116 – Raw asymmetric keys

1117 The CMS securely provisions credentials for Device-to-Device interactions using the CMS
1118 credential provisioned by the DOTS.

1119 The following example describes how a Device updates a symmetric key credential involving a peer
1120 Device. The Device discovers the credential to be updated; for example, a secure connection
1121 attempt fails. The CMS returns an updated symmetric key credential. The CMS updates the
1122 corresponding symmetric key credential on the peer Device.

1123 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1124 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)
1125 asserted by the Client requesting the Resource and compare that role identifier(s) with the
1126 constraints described in the Server's ACLs Thus, a Client Device may need to be provisioned with
1127 one or more role credentials.

1128 Each Device holds the role information as a Property within the credential Resource.

1129 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a
 1130 certificate role credential.

1131 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned
 1132 for a client, access to a Resource is granted if it would be granted under any of the roles.

1133 **5.4.5 ACL provisioning**

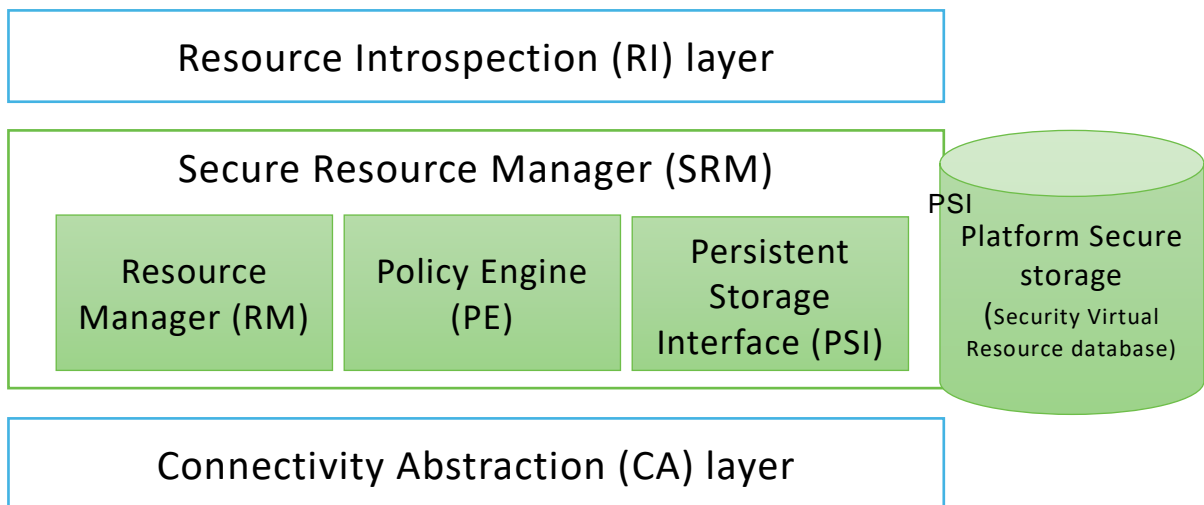
1134 ACL provisioning is performed over a secure connection between the AMS and its Devices. The
 1135 AMS provisions the ACL by updating the Device's ACL Resource.

1136 **5.5 Secure Resource Manager (SRM)**

1137 SRM plays a key role in the overall security operation. In short, SRM performs both management
 1138 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3
 1139 main functional elements:

- 1140 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using PSI)
 1141 as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3) Responding
 1142 to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a format that is
 1143 consistent with device-specific data store format. However, the RM will use JSON format to
 1144 marshal SVR data structures before being passed to PSI for storage, or travel off-device.
- 1145 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control
 1146 policies responds to the requests with either "ACCESS_GRANTED" or "ACCESS_DENIED". To
 1147 make the access decisions, the PE consults the appropriate ACL and looks for best Access
 1148 Control Entry (ACE) that can serve the request given the subject (Device or role) that was
 1149 authenticated by DTLS.
- 1150 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in
 1151 its own memory and storage. The SRM design is modular such that it may be implemented in
 1152 the Platform's secure execution environment; if available.

1153 Figure 10 depicts OCF's SRM Architecture.



1154

1155 **Figure 10 – OCF's SRM Architecture**

Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 36

1156 **5.6 Credential Overview**

1157 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional
1158 communication. Credentials can be symmetric or asymmetric. Each device stores secret and public
1159 parts of its own credentials where applicable, as well as credentials for other devices that have
1160 been provided by the DOTS or a CMS. These credentials are then used in the establishment of
1161 secure communication sessions (e.g. using DTLS) to validate the identities of the participating
1162 parties. Role credentials are used once an authenticated session is established, to assert one or
1163 more roles for a device.

1164

1165 **6 Security for the Discovery Process**

1166 **6.1 Preamble**

1167 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,
1168 called links) for the Resources hosted by the Server, complemented by attributes about those
1169 Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-1:2018)

1170 **6.2 Security Considerations for Discovery**

1171 When defining discovery process, care must be taken that only a minimum set of Resources are
1172 exposed to the discovering entity without violating security of sensitive information or privacy
1173 requirements of the application at hand. This includes both data included in the Resources, as well
1174 as the corresponding metadata.

1175 To achieve extensibility and scalability, this document does not provide a mandate on
1176 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on
1177 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any
1178 of the Resources.

1179 The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted
1180 Resources. (See 13.5)

1181 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery
1182 process itself needs to be secured. This document sets the following requirements for the discovery
1183 process:

- 1184 1) Providing integrity protection for discovered Resources.
- 1185 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1186 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)
1187 on the known "/oic/res" Resource.

1188 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a
1189 Server cannot determine the identity of the requester. In such cases, a Server that wants to
1190 authenticate the Client before responding can list the secure discovery URI (e.g.
1191 coaps://IP:PORT/oic/res) in the unsecured "/oic/res" Resource response. This means the secure
1192 discovery URI is by default discoverable by any Client. The Client will then be required to send a
1193 separate unicast request using DTLS to the secure discovery URI.

1194 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to
1195 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN
1196 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1197 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource
1198 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1199 {  
1200   "aclist2": [  
1201     {  
1202       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1203       "resources": [{"href": "/door"}],  
1204       "permission": 2, // RETRIEVE  
1205       "aceid": 1  
1206     }  
1207   ],
```

```

1208     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1209   }
1210   {
1211     "aclist2": [
1212       {
1213         "subject": {"authority": "owner", "role": "owner"}
1214         "resources": [{"href": "/door"}],
1215         "permission": 2, // RETRIEVE
1216         "aceid": 2
1217       }
1218     ],
1219     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1220   }
1221   {
1222     "aclist2": [
1223       {
1224         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1225         "resources": [{"href": "/door/lock"}],
1226         "permission": 4, // UPDATE
1227         "aceid": 3
1228       }
1229     ],
1230     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1231   }
1232   {
1233     "aclist2": [
1234       {
1235         "subject": {"conntype": "anon-clear"},
1236         "resources": [{"href": "/light"}],
1237         "permission": 2, // RETRIEVE
1238         "aceid": 4
1239       }
1240     ],
1241     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1242   }

```

1243 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when
1244 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will include
1245 the URI of the "/door" Resource metadata. Client "d2" will have access to both the Resources.
1246 ACE2 will prevent "d4" from update.

1247 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1248   [
1249     {
1250       "href": "/door",
1251       "rt": ["oic.r.door"],
1252       "if": ["oic.if.b", "oic.if.ll"],

```

```
1253     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1254   }
1255 ]
```

1256 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1257 [
1258   {
1259     "href": "/door",
1260     "rt": ["oic.r.door"],
1261     "if": ["oic.if.b", "oic.if.ll"],
1262     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1263   },
1264   {
1265     "href": "/door/lock",
1266     "rt": ["oic.r.lock"],
1267     "if": ["oic.if.b"],
1268     "type": ["application/json", "application/exi+xml"]
1269   }
1270 ]
```

1271 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

```
1272 [
1273   {
1274     "href": "/door/lock",
1275     "rt": ["oic.r.lock"],
1276     "if": ["oic.if.b"],
1277     "type": ["application/json", "application/exi+xml"],
1278     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1279   }
1280 ]
```

1281 Discovery results delivered to any device regarding d3's "/oic/res" Resource from the unsecure interface:

```
1283 [
1284   {
1285     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1286     "href": "/light",
1287     "rt": ["oic.r.light"],
1288     "if": ["oic.if.s"]
1289   }
1290 ]
1291
```


1292 **7 Security Provisioning**

1293 **7.1 Device Identity**

1294 **7.1.1 General Device Identity**

1295 Each Device, which is a logical device, is identified with a Device ID.

1296 Devices shall be identified by a Device ID value that is established as part of device onboarding.
1297 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. "urn:uuid"). Device IDs shall be
1298 unique within the scope of operation of the corresponding OCF Security Domain, and should be
1299 universally unique. The DOTS shall ensure Device ID of the new Device is unique within the scope
1300 of the owner's OCF Security Domain. The DOTS shall verify the chosen new device identifier does
1301 not conflict with Device IDs previously introduced into the OCF Security Domain.

1302 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"
1303 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying
1304 authentication credentials of a peer device.

1305 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of credentials,
1306 both its own and other Device credentials, in the "/oic/sec/cred" Resource. The device ID can be
1307 used to distinguish between a device's own credential, and credentials for other devices.
1308 Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the device.

1309 Device ID shall be:

- 1310 – Unique
- 1311 – Immutable
- 1312 – Verifiable

1313 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the
1314 device as described later in this clause.

1315 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The
1316 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in the
1317 device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1318 An OCF Platform may have a secure execution environment, which shall be used to secure unique
1319 identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to provide
1320 each Device with the appropriate and separate security.

1321 **7.1.2 Device Identity for Devices with UAID [Deprecated]**

1322 This clause is intentionally left blank.

1323 **7.2 Device Ownership**

1324 This is an informative clause. Devices are logical entities that are security endpoints that have an
1325 identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first
1326 initialized. Establishing device ownership is a process by which the device asserts its identity to
1327 the DOTS and the DOTS provisions an owner identity. This exchange results in the device changing
1328 its ownership state, thereby preventing a different DOTS from asserting administrative control over
1329 the device.

1330 The ownership transfer process starts with the OBT discovering a new device that is in Unowned
1331 state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new
1332 device. At the end of ownership transfer, the following is accomplished:

- 1333 1) The DOTS establishes a secure session with new device.

- 1334 2) Optionally asserts any of the following:
- 1335 a) Proximity (using PIN) of the OBT to the Platform.
 - 1336 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific
 - 1337 attributes.
- 1338 3) Determines the device identifier.
- 1339 4) Determines the device owner.
- 1340 5) Specifies the device owner (e.g. Device ID of the OBT).
- 1341 6) Provisions the device with owner's credentials.
- 1342 7) Sets the "Owned" state of the new device to TRUE.
- 1343 .

1344 **7.3 Device Ownership Transfer Methods**

1345 **7.3.1 OTM implementation requirements**

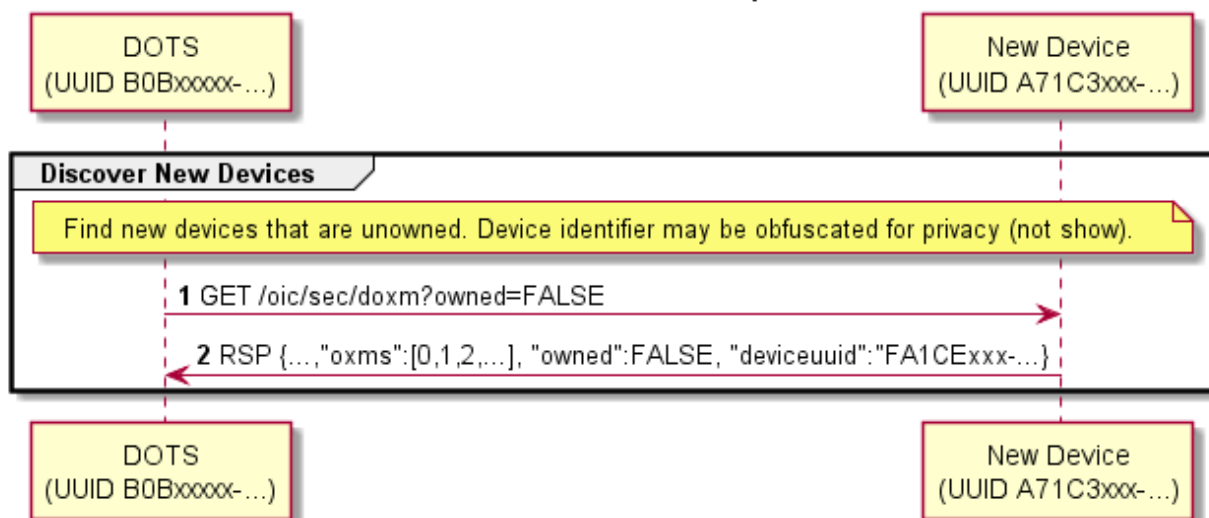
1346 This document provides specifications for several methods for ownership transfer. Implementation
1347 of each individual ownership transfer method is considered optional. However, each device shall
1348 implement at least one of the ownership transfer methods not including vendor specific methods.

1349 All OTMs included in this document are considered optional. Each vendor is required to choose
1350 and implement at least one of the OTMs specified in this document. The OCF, does however,
1351 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability
1352 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with
1353 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the preferred
1354 approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in designing
1355 vendor-specific OTMs.

1356 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer
1357 methods (OTM). The DOTS determines which OTM is most appropriate to onboard the new Device.
1358 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the
1359 "/oic/sec/doxm" Resource. The DOTS queries the Device's supported credential types using the
1360 "credtype" Property of the "/oic/sec/cred" Resource. The DOTS and CMS provision credentials
1361 according to the credential types supported.

1362 Figure 11 depicts new Device discovery sequence.

Discover New Devices Sequence



1363

1364

Figure 11 – Discover New Device Sequence

1365

1366

Table 1 – Discover New Device Details

Step	Description
1	The DOTS 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.

1367 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCS
 1368 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for
 1369 establishing trust in the new Device by the DOTS and optionally establishing trust in the OBT by
 1370 the new Device.

1371 The new device may have to perform some initialization steps at the beginning of an OTM. For
 1372 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN
 1373 value. The DOTS updates the oxmsel property of "/oic/sec/doxm" to the value corresponding to the
 1374 OTM being used, before performing other OTM steps. This update notifies the new device that
 1375 ownership transfer is starting.

1376 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT and
 1377 the OBT to authenticate to the new device.

1378 Additional provisioning steps may be performed subsequent to owner transfer success leveraging
 1379 the established OTM session.

1380 **7.3.2 SharedKey Credential Calculation**

1381 The SharedKey credential is derived using a PRF that accepts the key_block value resulting from
1382 the DTLS handshake used for onboarding. The new Device shall use the following calculation to
1383 ensure interoperability across vendor products (the DOTS performs the same calculation):

1384 SharedKey = PRF(Secret, Message);

1385 Where:

- 1386 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1387 - Secret is the key_block resulting from the DTLS handshake
 - 1388 ▪ See IETF RFC 5246 clause 6.3
 - 1389 ▪ The length of key_block depends on cipher suite.
 - 1390 • (e.g. 96 bytes for TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - 1391 40 bytes for TLS_PSK_WITH_AES_128_CCM_8)
- 1392 - Message is a concatenation of the following:
 - 1393 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
 - 1394 • See clause 13.2.4 for specific DoxmTypes
 - 1395 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
 - 1396 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
 - 1397 ▪ Device ID is new device's UUID Device ID
 - 1398 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1399 - SharedKey Length will be 32 octets.
 - 1400 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used.
 - 1401 DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

1402 **7.3.3 Certificate Credential Generation**

1403 The Certificate Credential will be used by Devices for secure bidirectional communication. The
1404 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used
1405 to mutually establish the authenticity of the Device.

1406 **7.3.4 Just-Works OTM**

1407 **7.3.4.1 Just-Works OTM General**

1408 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a
1409 secure connection through which a device should be provisioned for use within the owner's OCF
1410 Security Domain. Provisioning additional credentials and Resources is a typical step following
1411 ownership establishment. The pre-shared key is called SharedKey.

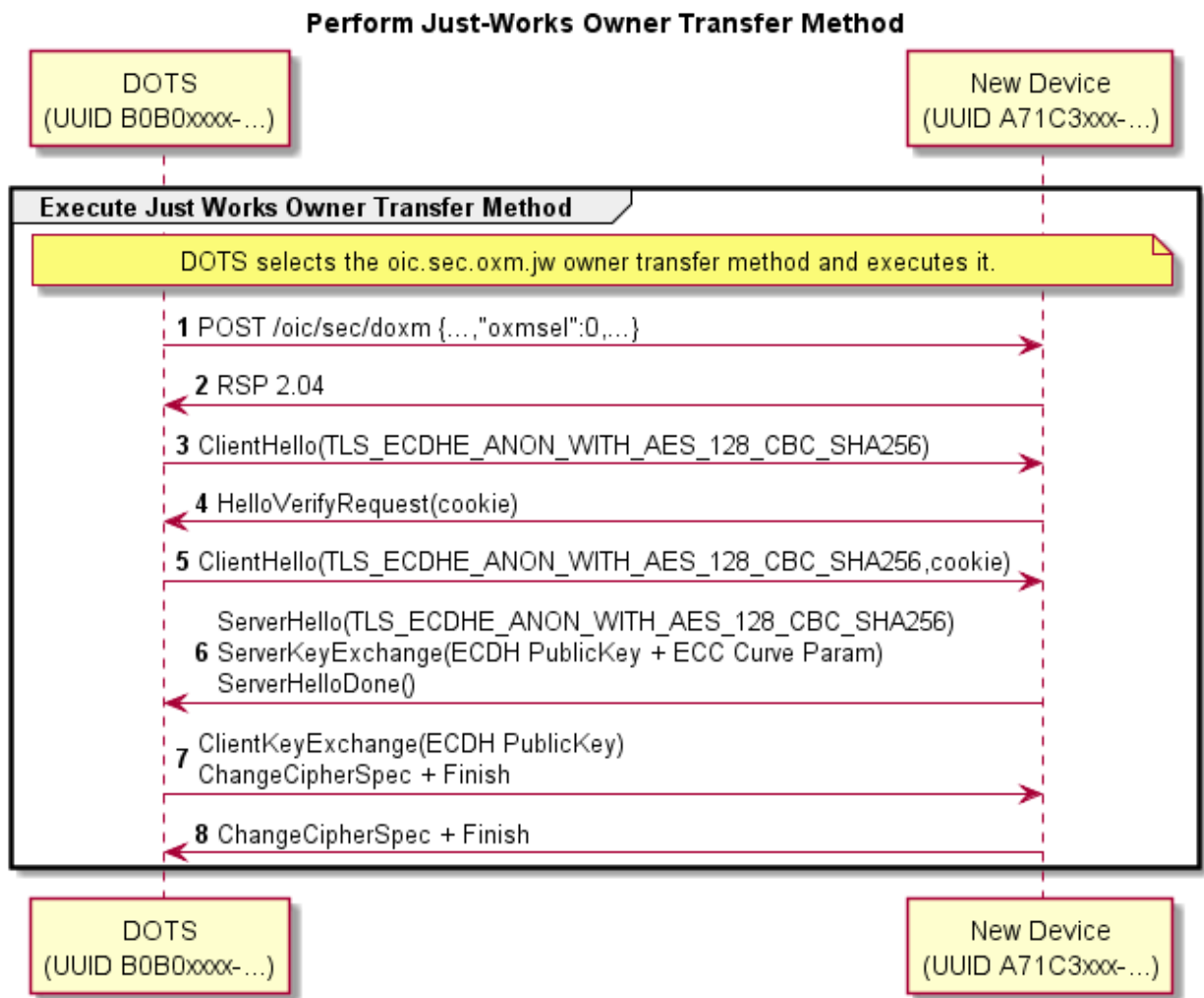
1412 The DOTS selects the Just-works OTM using the "oxmsel" Property of the "/oic/sec/doxm"
1413 Resource and establishes a DTLS session using a ciphersuite defined for the Just-works OTM.

1414 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1415 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
1416 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

1417 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for
1418 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1419 Just Works OTM sequence is shown in Figure 12 and steps described in Table 2.



1420
1421
1422
1423

Figure 12 – A Just Works OTM

Table 2 – A Just Works OTM Details

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

1424 **7.3.4.2 Security Considerations**

1425 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this
 1426 method presumes that both the DOTS and the new device perform the "just-works" method
 1427 assumes onboarding happens in a relatively safe environment absent of an attack device.

1428 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to
 1429 the device.

1430 The new device should use a temporal device ID prior to transitioning to an owned device while it
1431 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-
1432 temporal device ID that could differ from the temporal value during the secure session in which
1433 owner transfer exchange takes place. The DOTS verifies the asserted Device ID does not conflict
1434 with a Device ID already in use. If it is already in use the existing credentials are used to establish
1435 a secure session.

1436 An un-owned Device that also has established device credentials might be an indication of a
1437 corrupted or compromised device.

1438 **7.3.5 Random PIN Based OTM**

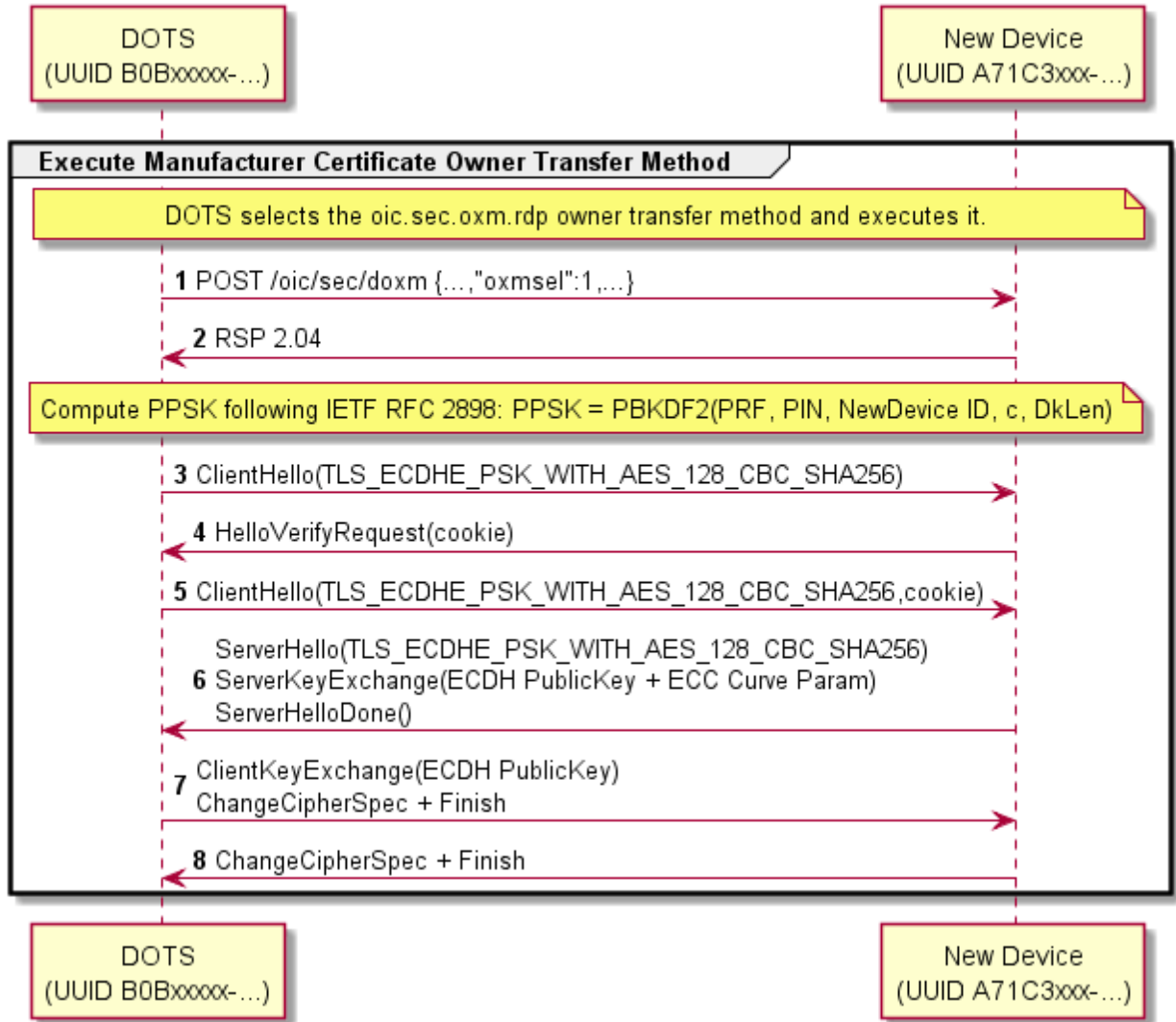
1439 **7.3.5.1 Random PIN OTM General**

1440 The Random PIN method establishes physical proximity between the new device and the OBT can
1441 prevent man-in-the-middle attacks. The Device generates a random number that is communicated
1442 to the DOTS over an Out of Band Communication Channel. The definition of an Out of Band
1443 Communication Channel is outside the scope of the definition of device OTMs. The DOTS and new
1444 Device use the PIN in a key exchange as evidence that someone authorized the transfer of
1445 ownership by having physical access to the new Device via the Out-of-Band Communication
1446 Channel.

1447 **7.3.5.2 Random PIN Owner Transfer Sequence**

1448 Random PIN-based OTM sequence is shown in Figure 13 and steps described in Table 3.

Perform Random PIN Device Owner Transfer Method



1449

1450

1451

1452

Figure 13 – Random PIN-based OTM

Table 3 – Random PIN-based OTM Details

Step	Description
1, 2	The DOTS 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 Communication Channel that establishes proximal context between the new device and the DOTS. The security principle is the attack device will be unable to intercept the PIN due to a lack of proximity.

1453 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF
1454 RFC 2898 and a PIN exchanged via an Out of Band Communication Channel to generate a pre-
1455 shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that
1456 accept a PSK.

1457 $PPSK = PBKDF2(PRF, PIN, Device\ ID, c, dkLen)$

1458 The PBKDF2 function has the following parameters:

1459 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.

1460 - PIN – obtained via Out of Band Communication Channel.

1461 - Device ID – UUID of the new device.

1462 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2

1463 - c – Iteration count initialized to 1000

1464 - dkLen – Desired length of the derived PSK in octets.

1465 **7.3.5.3 Security Considerations**

1466 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with
1467 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials
1468 provisioned as a part of onboarding. In particular, learning the provisioned symmetric key
1469 credentials allows an attacker to masquerade as the onboarded device.

1470 It is recommended that the entropy of the PIN be enough to withstand an online brute-force attack,
1471 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-9a-z), or
1472 a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack (MITM) is
1473 when the attacker is active on the network and can intercept and modify messages between the
1474 DOTS and device. In the MITM attack, the attacker must recover the PIN from the key exchange
1475 messages in "real time", i.e., before the peer's time out and abort the connection attempt. Having
1476 recovered the PIN, he can complete the authentication step of key exchange. The guidance given
1477 here calls for a minimum of 40 bits of entropy, however, the assurance this provides depends on
1478 the resources available to the attacker. Given the parallelizable nature of a brute force guessing
1479 attack, the attack enjoys a linear speedup as more cores/threads are added. A more conservative
1480 amount of entropy would be 64 bits. Since the Random PIN OTM requires using a DTLS ciphersuite
1481 that includes an ECDHE key exchange, the security of the Random PIN OTM is always at least
1482 equivalent to the security of the JustWorks OTM.

1483 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN. The
1484 rationale is to increase the cost of a brute force attack, by increasing the cost of each guess in the
1485 attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an effective way
1486 to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify the reduction,
1487 since an X-fold increase in time spent by the honest peers does not directly translate to an X-fold
1488 increase in time by the attacker. This asymmetry is because the attacker may use specialized
1489 implementations and hardware not available to honest peers. For this reason, when deciding how
1490 much entropy to use for a PIN, it is recommended that implementers assume PBKDF2 provides no
1491 security, and ensure the PIN has sufficient entropy.

1492 The Random PIN device OTM security depends on an assumption that a secure Out of Band
1493 Communication Channel for communicating a randomly generated PIN from the new device to the
1494 OBT exists. If the Out of Band Communication Channel leaks some or the entire PIN to an attacker,
1495 this reduces the entropy of the PIN, and the attacks described above apply. The Out of Band
1496 Communication Channel should be chosen such that it requires proximity between the DOTS and
1497 the new device. The attacker is assumed to not have compromised the Out of Band Communication
1498 Channel. As an example Out of Band Communication Channel, the device may display a PIN to be
1499 entered into the OBT software. Another example is for the device to encode the PIN as a 2D
1500 barcode and display it for a camera on the DOTS device to capture and decode.

1501 **7.3.6 Manufacturer Certificate Based OTM**

1502 **7.3.6.1 Manufacturer Certificate Based OTM General**

1503 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the
1504 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device
1505 and the DOTS.

1506 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust
1507 anchor.

1508 For some environments, policies or administrators, additional information about device
1509 characteristics may be sought. This list of additional attestations that OCF may or may not have
1510 tested (understanding that some attestations are incapable of testing or for which testing may be
1511 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3
1512 extension described in 9.4.2.2.6.

1513 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with
1514 certificate data to authenticate their identities with the DOTS in the process of bringing a new
1515 device into operation on an OCF Security Domain. The onboarding process involves several
1516 discrete steps:

1517 1) Pre-on-board conditions

1518 a) The credential element of the Device's credential Resource ("/oic/sec/cred") containing the
1519 manufacturer certificate shall be identified by the "credusage" Property containing the string
1520 "oic.sec.cred.mfgcert" to indicate that the credential contains a manufacturer certificate.

1521 b) The manufacturer certificate chain shall be contained in the identified credential element's
1522 "publicdata" Property.

1523 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1524 d) If the device requires authentication of the DOTS as part of ownership transfer, it is
1525 presumed that the DOTS has been registered and has obtained a certificate for its unique
1526 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.

1527 e) User has configured the DOTS app with network access info and account info (if any).

1528 2) The DOTS authenticates the Device using ECDSA to verify the signature. Additionally, the
1529 Device may authenticate the DOTS to verify the DOTS signature.

1530 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready
1531 for OTM state. If authentication succeeds, the Device shall establish an encrypted link with the
1532 DOTS in accordance with the negotiated cipher suite.

1533 **7.3.6.2 Certificate Profiles**

1534 See 9.4.2 for details.

1535 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

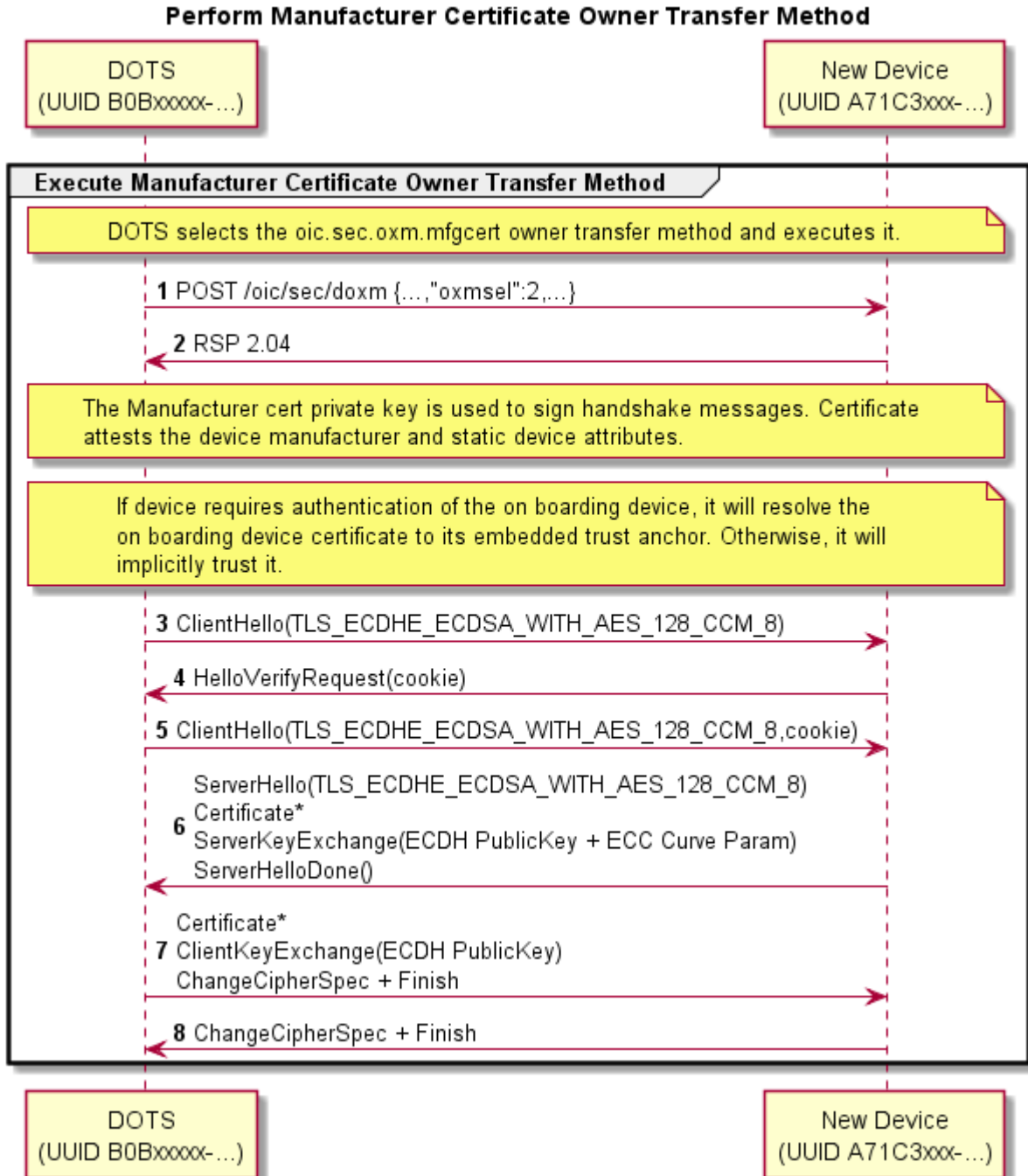
1536 In order for full, mutual authentication to occur between the device and the DOTS, both the device
1537 and DOTS must be able to trace back to a mutual Trust Anchor or Certificate Authority. This implies
1538 that OCF may need to obtain services from a Certificate Authority (e.g. Symantec, Verisign, etc.)
1539 to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors are derived.

1540 The DOTS authenticates the device during onboarding. However, the device is not required to
1541 authenticate the DOTS due to potential resource constraints on the device.

1542 In the case where the Device does NOT authenticate the DOTS software, there is the possibility of
1543 malicious DOTS software unwittingly deployed by users, or maliciously deployed by an adversary,
1544 which can compromise OCF Security Domain access credentials and/or personal information.

1545 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1546 Manufacturer Certificate Based OTM sequence is shown in Figure 14 and steps described in
 1547 Table 4.



1548

1549

Figure 14 – Manufacturer Certificate Based OTM Sequence

1550

1551

Table 4 – Manufacturer Certificate Based OTM Details

Step	Description
1, 2	The DOTS 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 DOTS certificate. The device's manufacturer certificate may contain data attesting to the Device hardening and security properties.

1552 **7.3.6.5 Security Considerations**

1553 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of
1554 assurance that the private key cannot be compromised.

1555 The Platform manufacturer issues the manufacturer certificate and attests the private key
1556 protection mechanism.

1557 **7.3.7 Vendor Specific OTMs**

1558 **7.3.7.1 Vendor Specific OTM General**

1559 The OCF anticipates situations where a vendor will need to implement an OTM that accommodates
1560 manufacturing or Device constraints. The Device OTM resource is extensible for this purpose.
1561 Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1562 – The OBT must determine which credential types are supported by the Device. This is
1563 accomplished by querying the Device's "/oic/sec/doxm" Resource to identify supported
1564 credential types.
- 1565 – The OBT provisions the Device with OC(s).
- 1566 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1567 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security
1568 Domain subsequent to ownership establishment.
- 1569 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks
1570 to be performed by a security service.

1571 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1572 Vendor-specific OTM sequence example is shown in Figure 15 and steps described in Table 5.

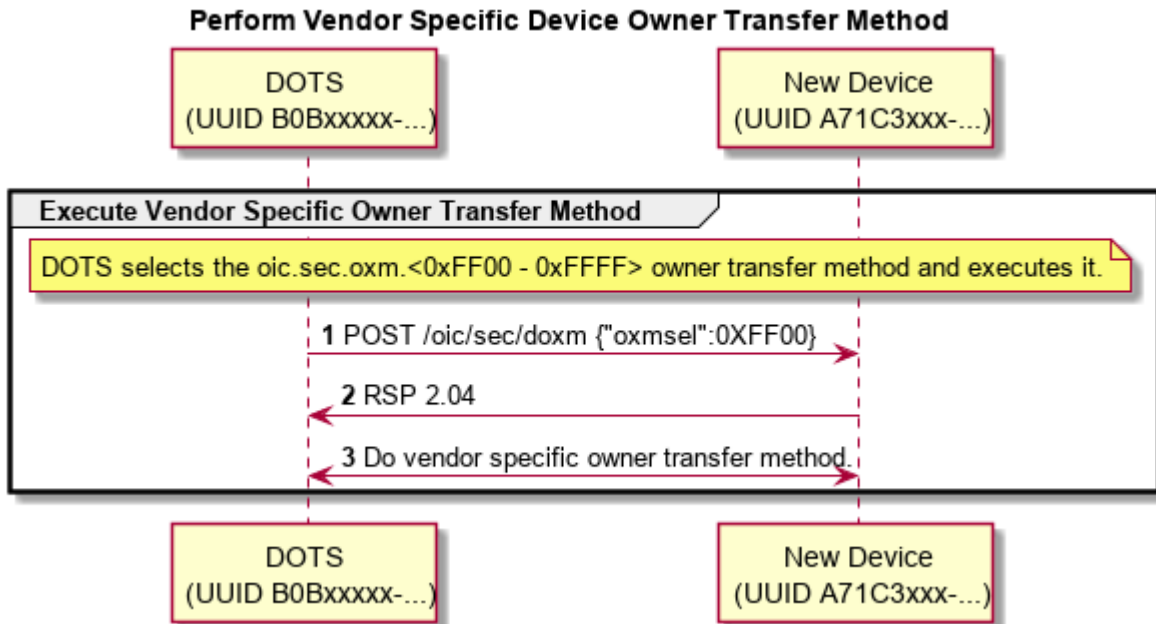


Figure 15 – Vendor-specific Owner Transfer Sequence

Table 5 – Vendor-specific Owner Transfer Details

Step	Description
1, 2	The DOTS selects a vendor-specific OTM.
3	The vendor-specific OTM is applied

7.3.7.3 Security Considerations

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

7.3.8 Establishing Owner Credentials

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

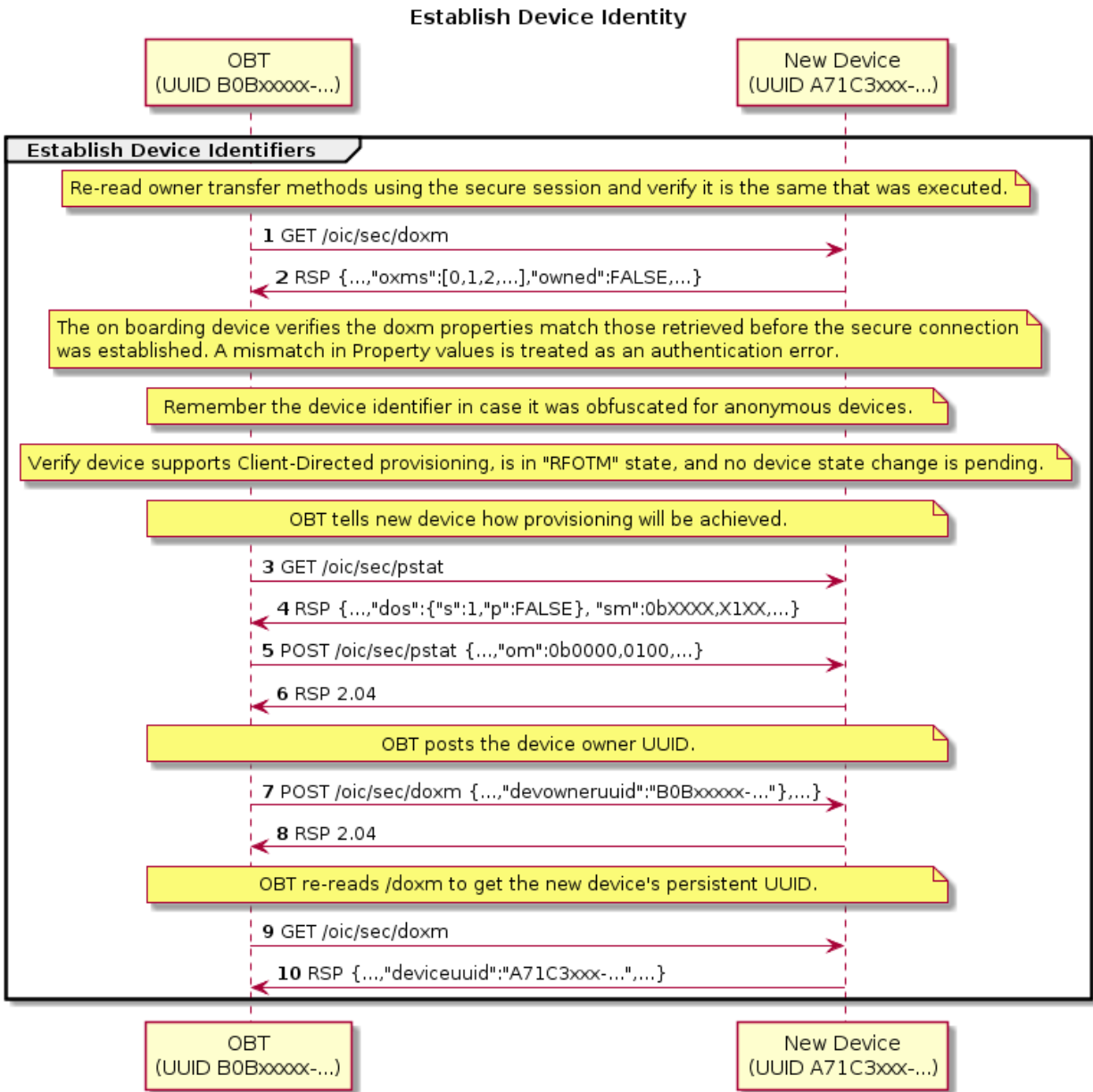
Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security Domain access information, provisioning functions, shared keys, or Kerberos tickets.

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

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

- 1) The OBT establishes the Device ID and Device owner uuid - See Figure 16 and Table 6.
- 2) The OBT then establishes Device's OC - See Figure 17 and Table 7. This can be either:
 - a) Symmetric credential - See Figure 18 and Table 8.
 - b) Asymmetric credential - See Figure 19 and Table 9.
- 3) Configure Device services - See Figure 20 and Table 10.

1593 4) Configure Device for peer to peer interaction - See Figure 21 and Table 11.
 1594



1595
 1596
 1597
 1598

Figure 16 – 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.

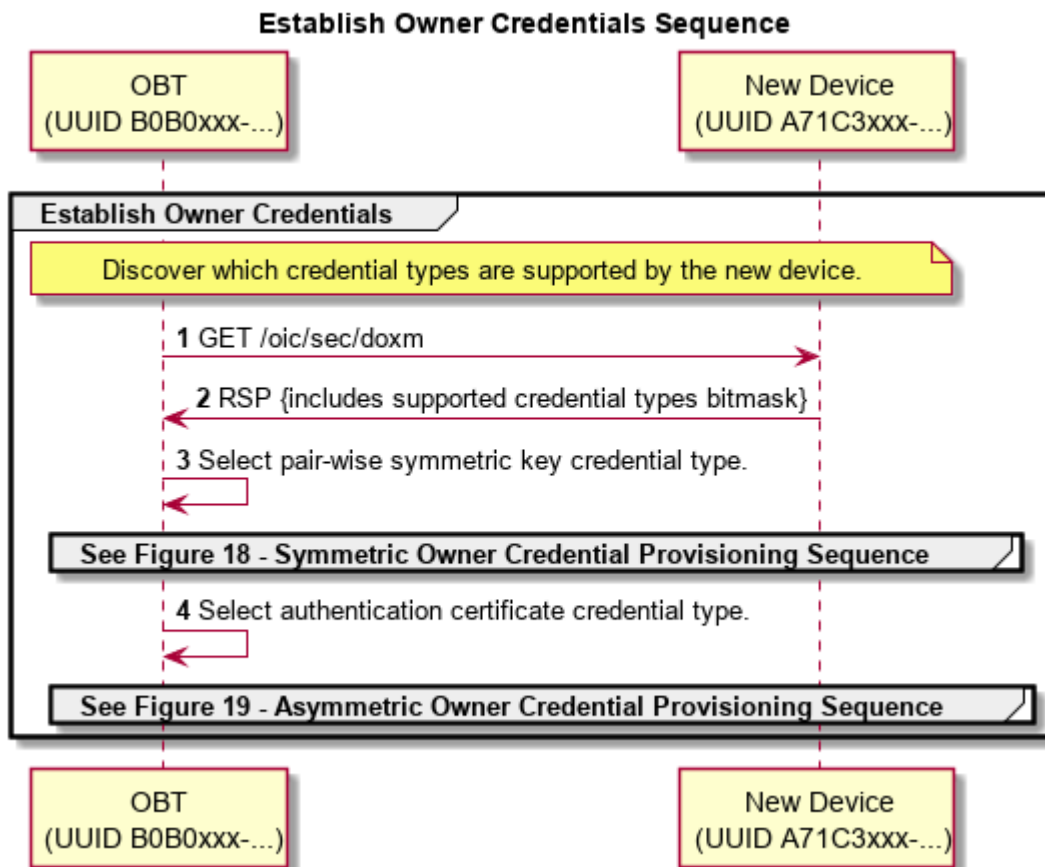


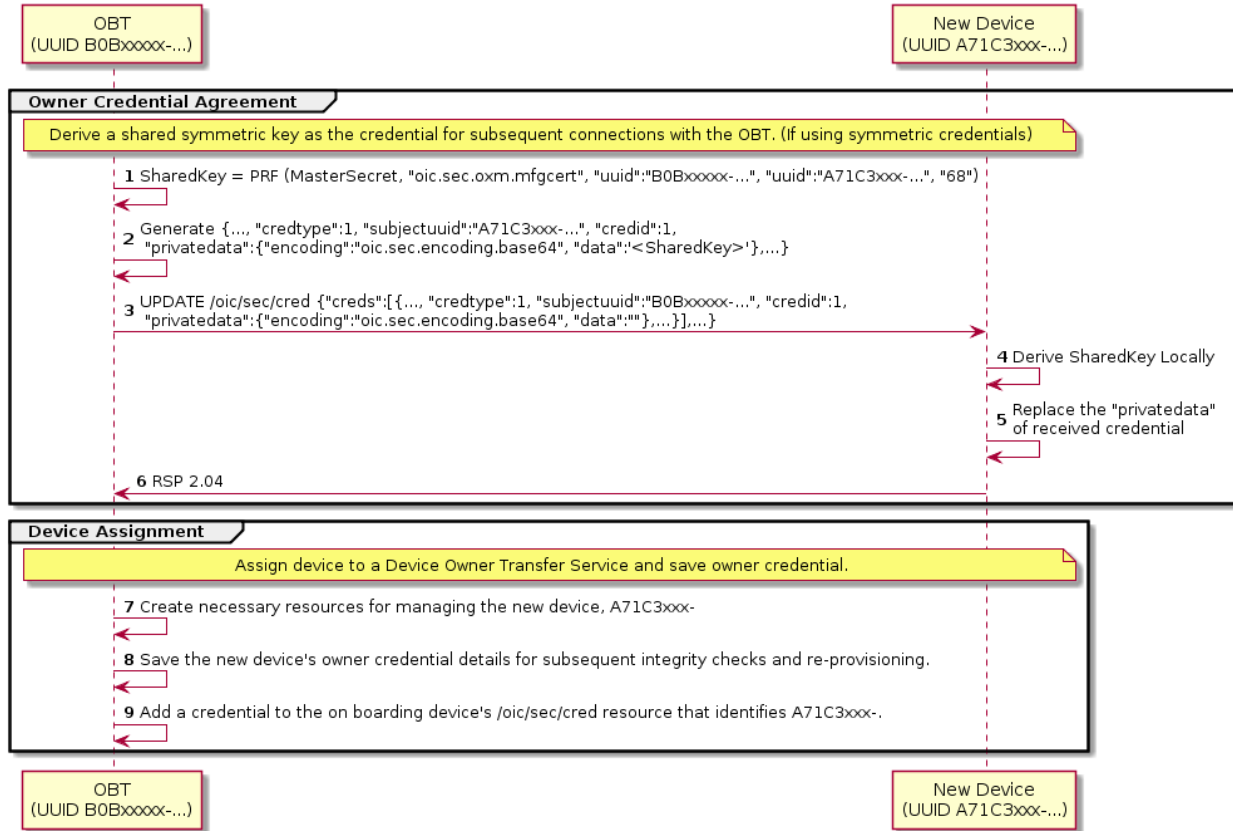
Figure 17 – Owner Credential Selection Provisioning Sequence

1599
1600
1601
1602

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



1603

1604

1605

1606

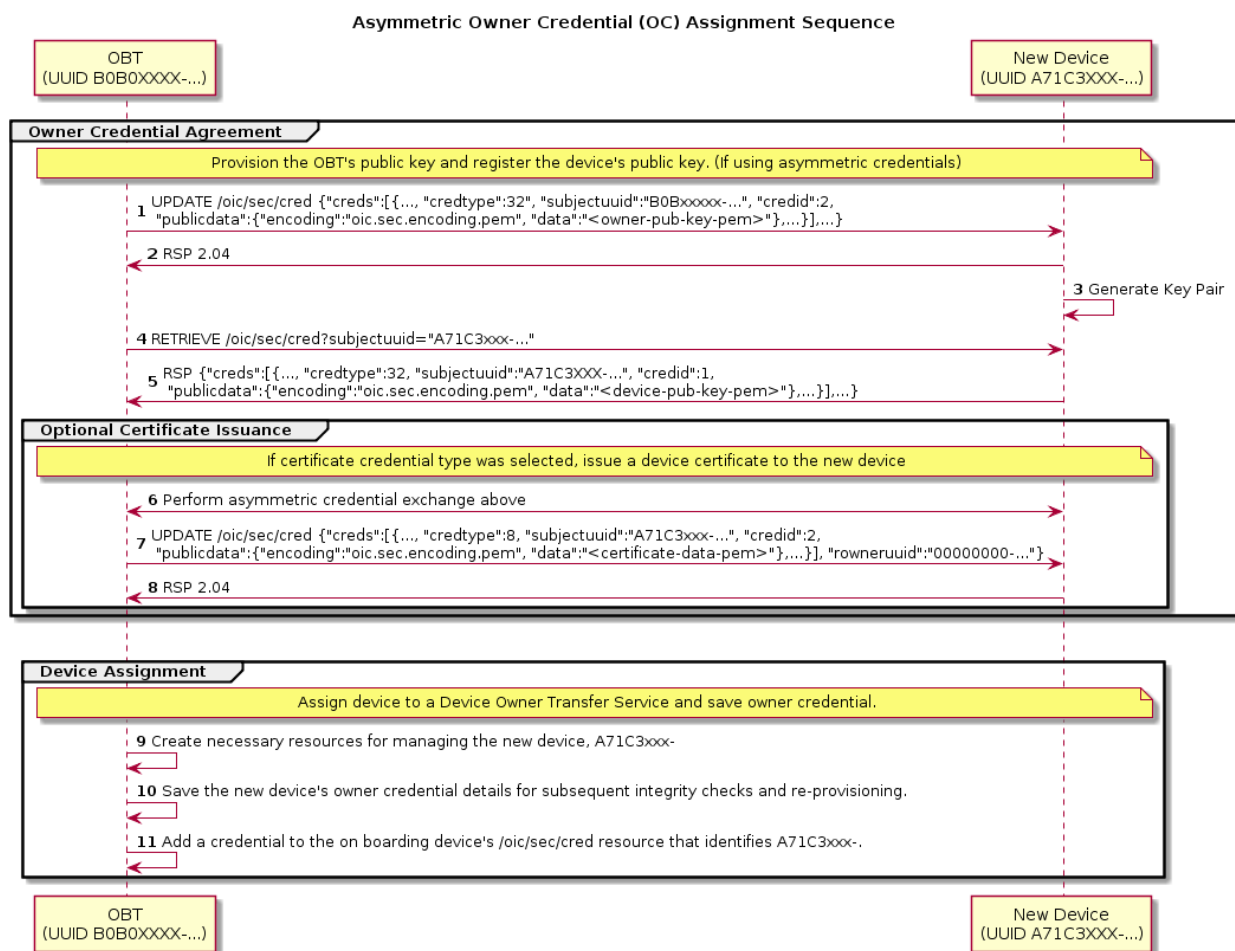
Figure 18 – Symmetric Owner Credential Provisioning Sequence

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.
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------

- 1607 In particular, if the OBT selects symmetric owner credentials:
- 1608 – The OBT generates a Shared Key using the SharedKey Credential Calculation method
 - 1609 described in 7.3.2.
 - 1610 – The OBT sends an empty key to the new Device's "/oic/sec/cred" Resource, identified as a
 - 1611 symmetric pair-wise key.
 - 1612 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
 - 1613 generate the Shared Key using the SharedKey Credential Calculation method described in 7.3.2
 - 1614 and store it with the owner credential.
 - 1615 – The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred"
 - 1616 Resource to authenticate the owner during subsequent connections.



1617

1618

Figure 19 – Asymmetric Owner Credential Provisioning Sequence

1619

1620

Table 9 – Asymmetric Owner Credential Assignment Details

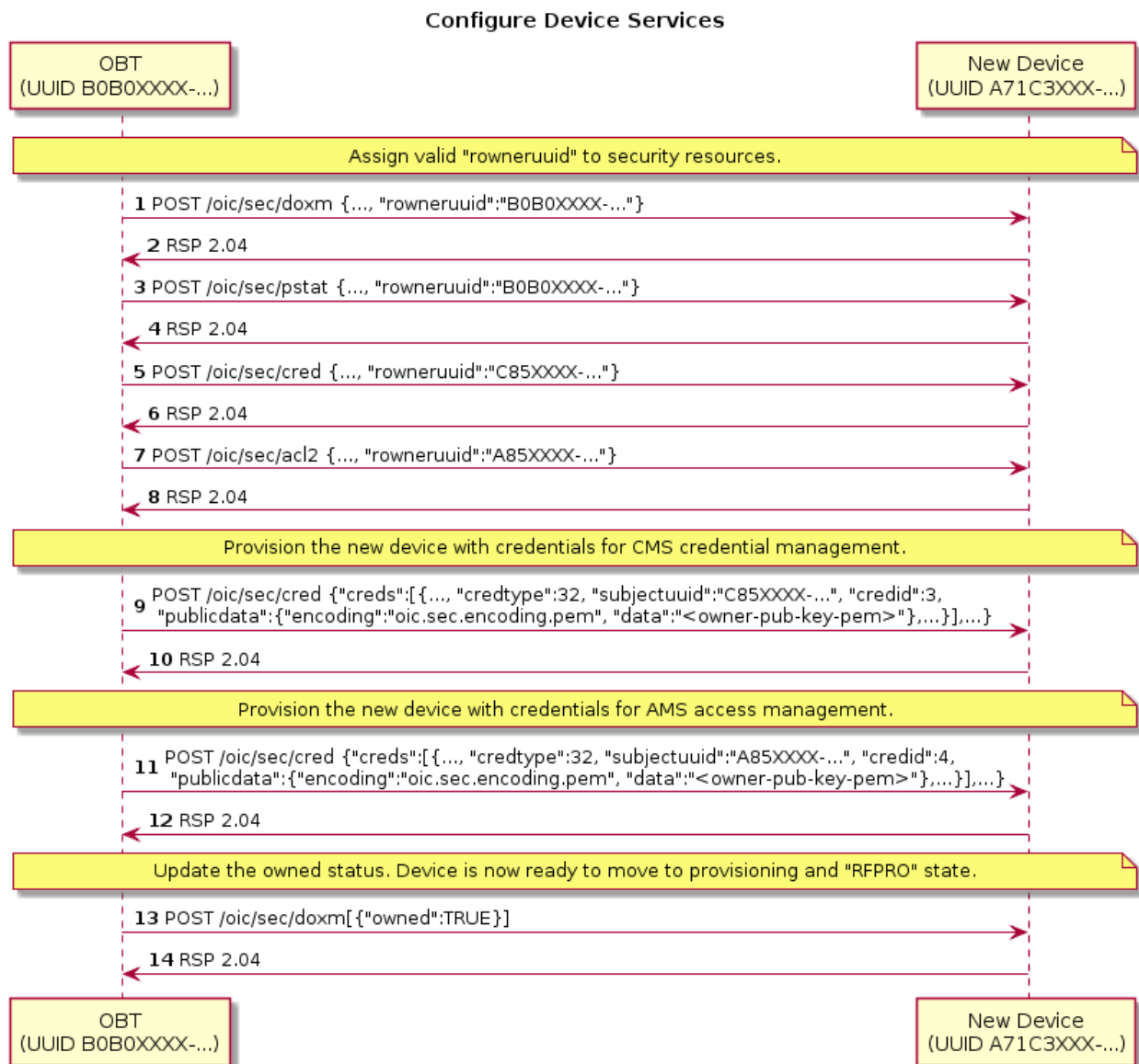
Step	Description
If an asymmetric or certificate owner credential type was selected by the OBT	
1, 2	The OBT creates an asymmetric type credential Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential 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.

1621 If the OBT selects asymmetric owner credentials:

- 1622 – The OBT adds its public key to the new Device's "/oic/sec/cred" Resource, identified as an
- 1623 Asymmetric Encryption Key.
- 1624 – The OBT queries the "/oic/sec/cred" Resource from the new Device, supplying the new Device's
- 1625 UUID via the SubjectID query parameter. In response, the new Device shall return the public
- 1626 Asymmetric Encryption Key

1627 If the OBT selects certificate owner credentials:

- 1628 – The OBT creates a certificate or certificate chain with the leaf certificate containing the public
- 1629 key returned by the new Device, signed by a mutually-trusted CA, and complying with the
- 1630 Certificate Credential Generation requirements defined in 7.3.3.
- 1631 – The OBT adds the newly-created certificate chain to the "/oic/sec/cred" Resource, identified as
- 1632 an Asymmetric Signing Key with Certificate.



1633

1634

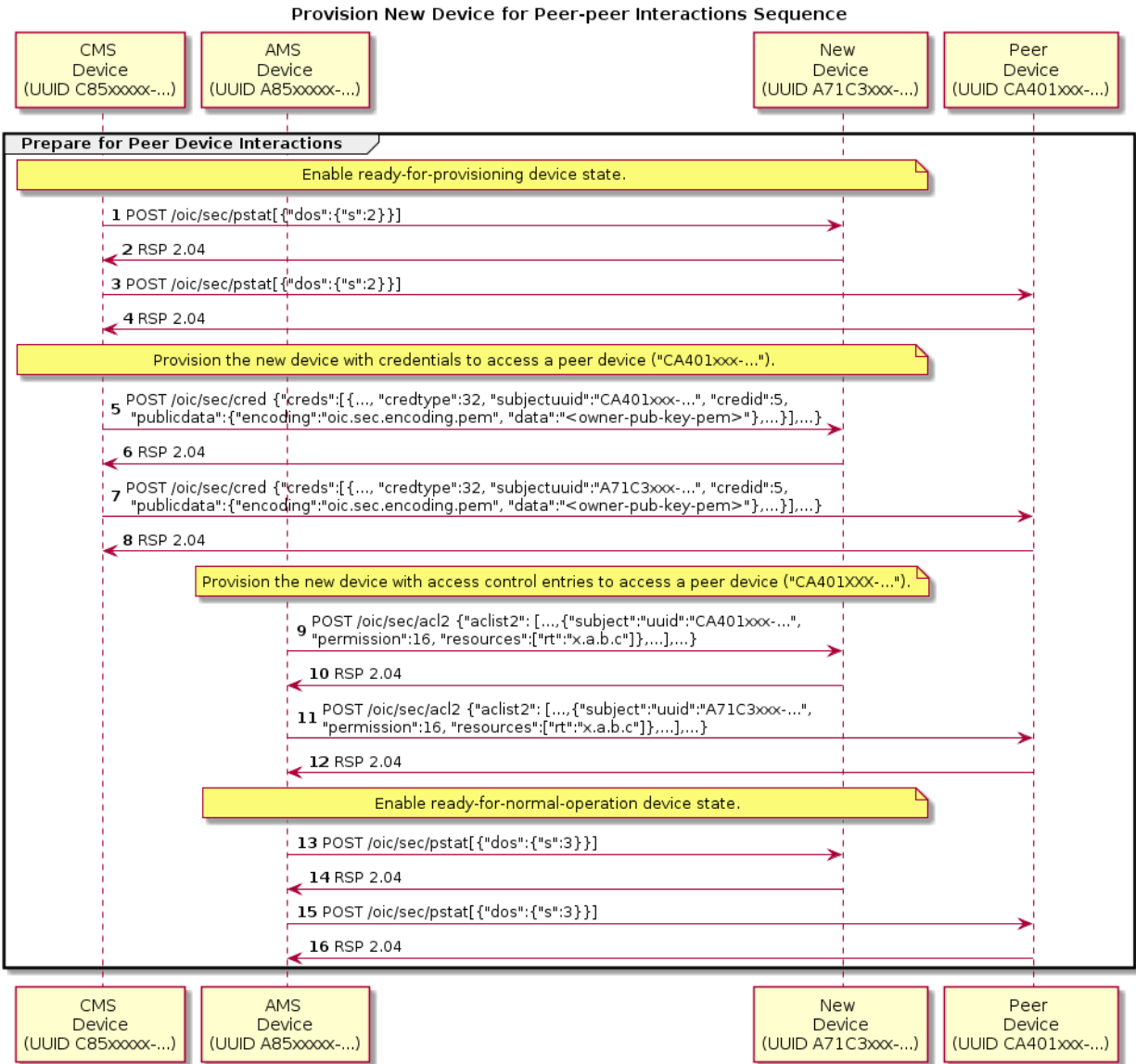
1635

1636

Figure 20 – 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.



1637

1638

Figure 21 – Provision New Device for Peer to Peer Interaction Sequence

1639

1640

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.

1641 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method -**
1642 **Moved to OCF Onboarding Tool document**

1643 This clause is intentionally left blank.

1644 **7.3.10 Security Profile Assignment**

1645 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results
1646 could be accessed from a manufacturer's certificate, OCF web server or other public repository.
1647 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device is
1648 authorized to possess and configures the Device with the subset of evaluated security profiles best
1649 suited for the OCF Security Domain owner's intended segmentation strategy.

1650 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles" Property
1651 of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification process.
1652 The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the values
1653 contained in the "supportedprofiles". The manufacturer default value shall be re-asserted when the
1654 Device transitions to RESET Device State.

1655 The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in
1656 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as
1657 directed by a Security Profile.

1658 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a subset
1659 of the OCF Security Profiles values the Device achieved as part of OCF Conformance testing. The
1660 DOTS may locate conformance results by inspecting manufacturer certificates supplied with the
1661 OCF Device by selecting the "credusage" Property of the "/oic/sec/cred" Resource having the value
1662 of "oic.sec.cred.mfgcert". The DOTS may further locate conformance results by visiting a well-
1663 known OCF web site URI corresponding to the ocfCPLAttributes extension fields (clause 9.4.2.2.7).
1664 The DOTS may select a subset of Security Profiles (from those evaluated by OCF conformance
1665 testing) based on a local policy.

1666 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries to
1667 allow DOTS access subsequent to onboarding.

1668 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the value
1669 that most correctly depicts the OCF Security Domain owner's intended Device deployment strategy.

1670 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID") to
1671 indicate the OCF Security Domain owner's intention to segment the OCF Security Domain
1672 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the
1673 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security
1674 Profiles when issuing role credentials.

1675 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control
1676 entries that include the role designation(s).

1677 **7.4 Provisioning**

1678 **7.4.1 Provisioning Flows**

1679 **7.4.1.1 Provisioning Flows General**

1680 As part of onboarding a new Device a secure channel is formed between the new Device and the
1681 OBT. Subsequent to the Device ownership status being changed to "owned", there is an opportunity
1682 to begin provisioning. The OBT provisions the support services that should be subsequently used
1683 to complete Device provisioning and on-going Device management.

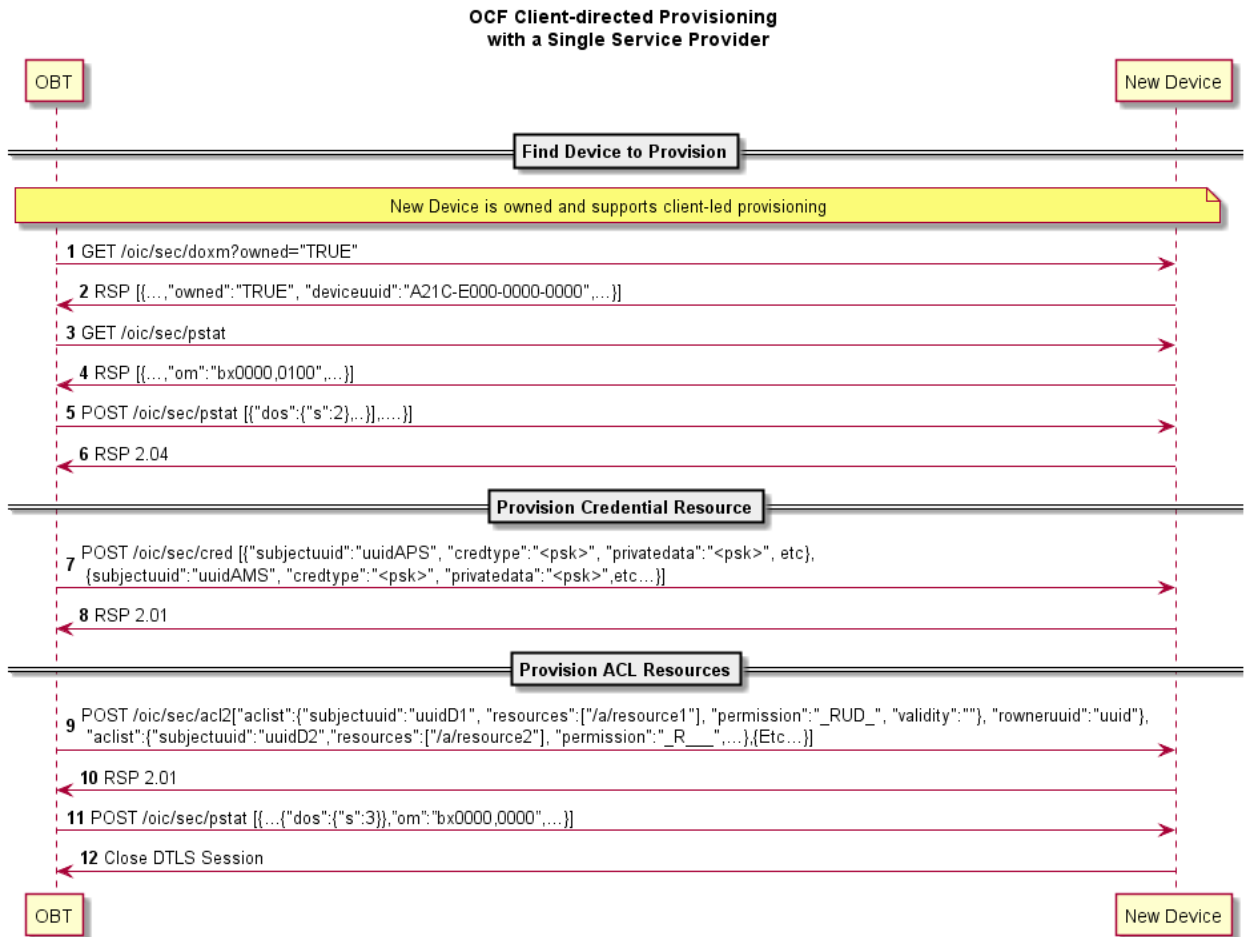
1684 The Device employs a Client-directed provisioning strategy. The "/oic/sec/pstat" Resource
1685 identifies the provisioning strategy and current provisioning status. The provisioning service should

1686 determine which provisioning strategy is most appropriate for the OCF Security Domain. See 13.8
 1687 for additional detail.

1688 **7.4.1.2 Client-directed Provisioning**

1689 Client-directed provisioning relies on a provisioning service that identifies Servers in need of
 1690 provisioning then performs all necessary provisioning duties.

1691 An example of Client-directed provisioning is shown in Figure 22 and steps described in Table 12.



1692

1693 **Figure 22 – Example of Client-directed provisioning**

1694

1695 **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	DOTS (on OBT) 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	CMS (on OBT)instantiates the "/oic/sec/cred" Resource. It contains credentials for the provisioned services and other Devices
9 - 10	AMS (on OBT) 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.

1696 **7.4.1.3 Server-directed Provisioning [DEPRECATED]**

1697 This clause is intentionally left blank.

1698 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**
1699 **[DEPRECATED]**

1700 This clause is intentionally left blank.

1701 **7.5 Device Provisioning for OCF Cloud – moved to OCF Cloud Security document**

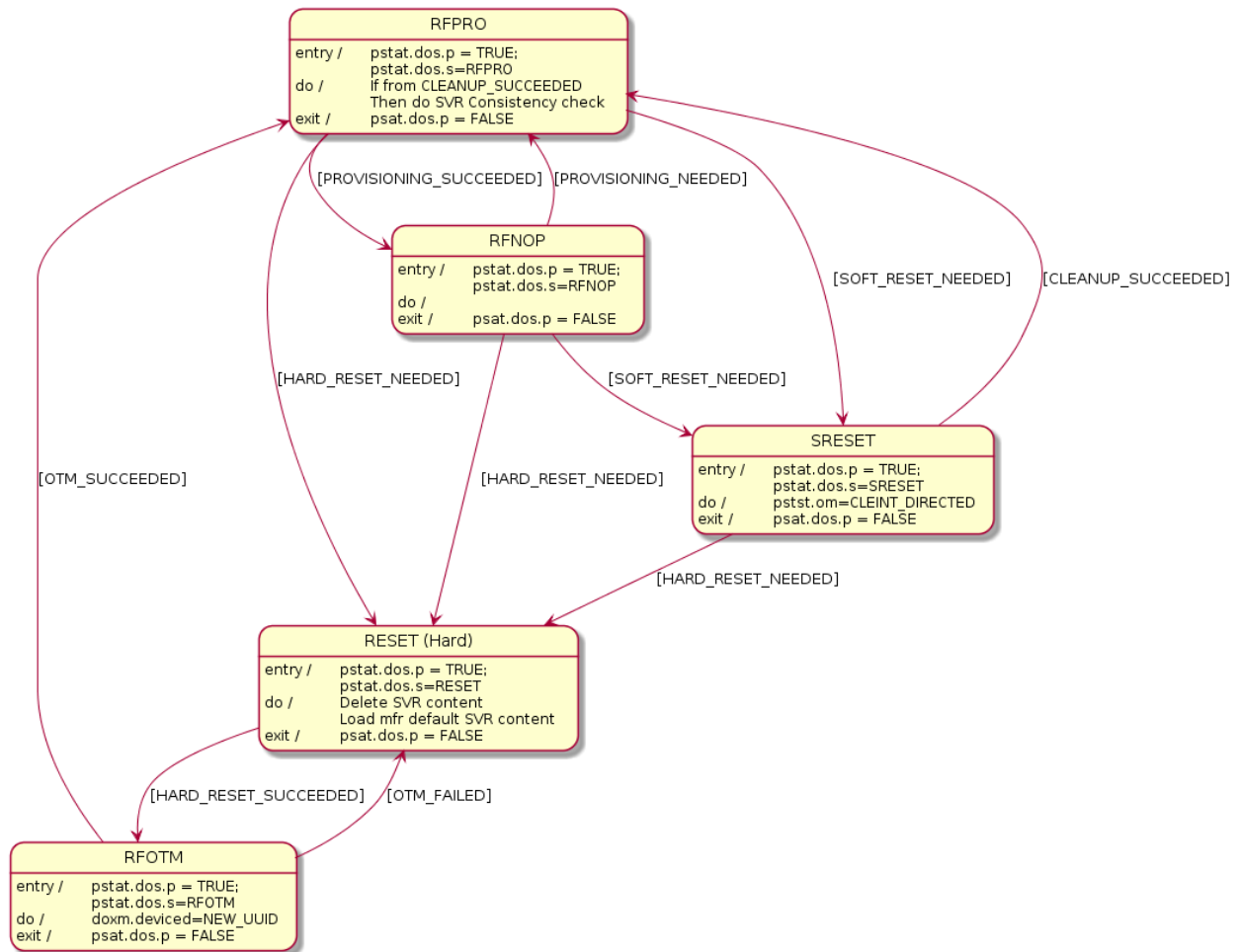
1702 This clause is intentionally left blank.

1703 **8 Device Onboarding State Definitions**

1704 **8.1 Device Onboarding General**

1705 As explained in 5.3, the process of onboarding completes after the ownership of the Device has
1706 been transferred and the Device has been provisioned with relevant configuration/services as
1707 explained in 5.4. The Figure 23 shows the various states a Device can be in during the Device
1708 lifecycle.

1709 The "/pstat.dos.s" Property is RW by the "/oic/sec/pstat" resource owner (e.g. "doxs" service) so
1710 that the resource owner can remotely update the Device state. When the Device is in RFNOP or
1711 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the
1712 Device state is SRESET the Device OC may be the only indication of authorization to access the
1713 Device. The Device owner may perform low-level consistency checks and re-provisioning to get
1714 the Device suitable for a transition to RFPRO.



1715

1716

Figure 23 – Device state model

1717 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the "Ready
 1718 for Normal Operation" state where it has all it needs in order to start interoperating with other
 1719 Devices. Clause 8.5 specifies the minimum mandatory configuration that a Device shall hold in
 1720 order to be considered as "Ready for Normal Operation".

1721 In the event of power loss or Device failure, the Device should remain in the same state that it was
 1722 in prior to the power loss / failure

1723 If a Device or resource owner OBSERVES "/pstat.dos.s", then transitions to SRESET will give early
 1724 warning notification of Devices that may require SVR consistency checking.

1725 In order for onboarding to function, the Device shall have the following Resources installed:

- 1726 1) "/oic/sec/doxm" Resource
- 1727 2) "/oic/sec/pstat" Resource
- 1728 3) "/oic/sec/cred" Resource

1729 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5
 1730 and 8.6.

1731 **8.2 Device Onboarding-Reset State Definition**

1732 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset
1733 also defines a state where the Device asset is ready to be transferred to another party.

1734 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces Platform
1735 reset. All Devices hosted on the same Platform transition their Device states to RESET when the
1736 Platform reset is asserted.

1737 The following Resources and their specific properties shall have the value as specified:

- 1738 – The "owned" Property of the "/oic/sec/doxm" Resource shall transition to FALSE.
- 1739 – The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 1740 – The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
1741 implemented.
- 1742 – The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
1743 default value.
- 1744 – The "deviceid" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
1745 default value, if this Property is implemented.
- 1746 – The "sct" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's default
1747 value.
- 1748 – The "oxmsel" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
1749 default value.
- 1750 – The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1751 – The "dos" Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RESET"
1752 state and dos.p shall equal "FALSE".
- 1753 – The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the
1754 manufacturer default value.
- 1755 – The "sm" (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set
1756 to the manufacturer default value.
- 1757 – The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and
1758 "/oic/sec/cred" Resources shall be nil UUID.
- 1759 – The "supportedprofiles" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer
1760 default value.
- 1761 – The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer
1762 default value.

1763 **8.3 Device Ready-for-OTM State Definition**

1764 The following Resources and their specific properties shall have the value as specified when the
1765 Device enters ready for ownership transfer:

- 1766 – The "owned" Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to
1767 TRUE.
- 1768 – The "devowner" Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
1769 implemented.
- 1770 – The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 1771 – The "deviceid" Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is
1772 implemented. The value of the di Property in "/oic/d" is undefined.
- 1773 – The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
1774 default value.

- 1775 – The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1776 – The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFOTM" state
1777 and dos.p shall equal "FALSE".
- 1778 – The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

1779 **8.4 Device Ready-for-Provisioning State Definition**

1780 The following Resources and their specific properties shall have the value as specified when the
1781 Device enters ready for provisioning:

- 1782 – The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 1783 – The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 1784 – The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be
1785 set to the value that was determined during RFOTM processing. Also the value of the "di"
1786 Property in "/oic/d" Resource shall be the same as the "deviceid" Property in the "/oic/sec/doxm"
1787 Resource.
- 1788 – The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM
1789 used during ownership transfer.
- 1790 – The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1791 – The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFPRO" state
1792 and "dos.p" shall equal "FALSE".
- 1793 – The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner
1794 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
1795 "rowneruuid" may result in an orphan Resource.
- 1796 – The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by
1797 "rowneruuid" and "devowneruuid" Properties.

1798 **8.5 Device Ready-for-Normal-Operation State Definition**

1799 The following Resources and their specific properties shall have the value as specified when the
1800 Device enters ready for normal operation:

- 1801 – The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 1802 – The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 1803 – The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be
1804 set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d" shall
1805 be the same as the deviceuuid.
- 1806 – The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM
1807 used during ownership transfer.
- 1808 – The "isop" Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once
1809 transition to RFNOP is otherwise complete.
- 1810 – The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFNOP" state
1811 and dos.p shall equal "FALSE".
- 1812 – The "rowneruuid" Property of every installed Resource shall be set to a valid resource owner
1813 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
1814 "rowneruuid" results in an orphan Resource.
- 1815 – The "/oic/sec/cred" Resource shall contain credentials for each service referenced by
1816 "rowneruuid" and "devowneruuid" Properties.

1817 **8.6 Device Soft Reset State Definition**

1818 The soft reset state is defined (e.g. "/pstat.dos.s" = SRESET) where entrance into this state means
1819 the Device is not operational but remains owned by the current owner. The Device may exit

1820 SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during original
1821 onboarding (but should not require use of an OTM /doxm.oxms).

1822 If the DOTS credential cannot be found or is determined to be corrupted, the Device state
1823 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to validate
1824 the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS Devices.

1825 When in SRESET, the following Resources and their specific Properties shall have the values as
1826 specified.

- 1827 – The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 1828 – The "devowneruid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 1829 – The "devowner" Property of the "/oic/sec/doxm" Resource shall be non-null, if this Property is
1830 implemented.
- 1831 – The "deviceuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 1832 – The "deviceid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 1833 – The "sct" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 1834 – The "oxmsel" Property of the "/oic/sec/doxm" Resource shall retain its value.
- 1835 – The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 1836 – The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- 1837 – The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-directed
1838 mode".
- 1839 – The "sm" (supported operational modes) Property of "/oic/sec/pstat" Resource may be updated
1840 by the Device owner (aka DOTS).
- 1841 – The "rowneruid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and
1842 "/oic/sec/cred" Resources may be reset by the Device owner (aka DOTS) and re-provisioned.
- 1843

1844 **9 Security Credential Management**

1845 **9.1 Preamble**

1846 This clause provides an overview of the credential types in OCF, along with details of credential
1847 use, provisioning and ongoing management.

1848 **9.2 Credential Lifecycle**

1849 **9.2.1 Credential Lifecycle General**

1850 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh and (4)
1851 revocation.

1852 **9.2.2 Creation**

1853 The CMS can provision credentials to the credential Resource onto the Device. The Device shall
1854 verify the CMS is authorized by matching the rowneruuid Property of the "/oic/sec/cred" resource
1855 to the DeviceID of the credential the CMS used to establish the secure connection.

1856 Credential Resources created using a CMS may involve specialized credential issuance protocols
1857 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate
1858 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of a
1859 provisioning action by a DOTS, CMS or AMS.

1860 **9.2.3 Deletion**

1861 The CMS can delete credentials from the credential Resource. The Device (e.g. the Device where
1862 the credential Resource is hosted) should delete credential Resources that have expired.

1863 An expired credential Resource may be deleted to manage memory and storage space.

1864 Deletion in OCF key management is equivalent to credential suspension.

1865 **9.2.4 Refresh**

1866 Credential refresh may be performed before it expires. The CMS performs credential refresh.

1867 The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may be
1868 applied when a credential is about to expire or is about to exceed a maximum threshold for bytes
1869 encrypted.

1870 A credential refresh method specifies the options available when performing key refresh. The
1871 Period Property informs when the credential should expire. The Device may proactively obtain a
1872 new credential using a credential refresh method using current unexpired credentials to refresh the
1873 existing credential. If the Device does not have an internal time source, the current time should be
1874 obtained from a CMS at regular intervals.

1875 If the onboarding established credentials are allowed to expire the DOTS shall re-onboard the
1876 Device to re-apply device owner transfer steps.

1877 All Devices shall support at least one credential refresh method.

1878 **9.2.5 Revocation**

1879 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where the
1880 revocation method involves provisioning of a revocation object that identifies a credential that is to
1881 be revoked prior to its normal expiration period, a credential Resource is created containing the
1882 revocation information that supersedes the originally issued credential. The revocation object
1883 expiration should match that of the revoked credential so that the revocation object is cleaned up
1884 upon expiry.

1885 It is conceptually reasonable to consider revocation applying to a credential or to a Device. Device
1886 revocation asserts all credentials associated with the revoked Device should be considered for
1887 revocation. Device revocation is necessary when a Device is lost, stolen or compromised. Deletion
1888 of credentials on a revoked Device might not be possible or reliable.

1889 **9.3 Credential Types**

1890 **9.3.1 Preamble**

1891 The "/oic/sec/cred" Resource maintains a credential type Property that supports several
1892 cryptographic keys and other information used for authentication and data protection. The
1893 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric
1894 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.
1895 PIN/password).

1896 **9.3.2 Pair-wise Symmetric Key Credentials**

1897 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The
1898 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

1899 Pair-wise keys could be established through ad-hoc key agreement protocols.

1900 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

1901 The PublicData Property may contain a token encrypted to the peer Device containing the pair-
1902 wise key.

1903 The OptionalData Property may contain revocation status.

1904 The Device implementer should apply hardened key storage techniques that ensure the
1905 PrivateData remains private.

1906 The Device implementer should apply appropriate integrity, confidentiality and access protection
1907 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1908 unauthorized modifications.

1909 **9.3.3 Group Symmetric Key Credentials**

1910 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are
1911 used for efficient sharing of data among group participants.

1912 Group keys do not provide authentication of Devices but only establish membership in a group.

1913 The CMS shall provision group symmetric key credentials to the group members. The CMS
1914 maintains the group memberships.

1915 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

1916 The PublicData Property may contain the group name.

1917 The OptionalData Property may contain revocation status.

1918 The Device implementer should apply hardened key storage techniques that ensure the
1919 PrivateData remains private.

1920 The Device implementer should apply appropriate integrity, confidentiality and access protection
1921 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1922 unauthorized modifications.

1923 **9.3.4 Asymmetric Authentication Key Credentials**

1924 **9.3.4.1 Asymmetric Authentication Key Credentials General**

1925 Asymmetric authentication key credentials contain either a public and private key pair or only a
1926 public key. The private key is used to sign Device authentication challenges. The public key is used
1927 to verify a device authentication challenge-response.

1928 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

1929 The PublicData Property contains the public key.

1930 The OptionalData Property may contain revocation status.

1931 The Device implementer should apply hardened key storage techniques that ensure the
1932 PrivateData remains private.

1933 Devices should generate asymmetric authentication key pairs internally to ensure the private key
1934 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material
1935 between Devices.

1936 The Device implementer should apply appropriate integrity, confidentiality and access protection
1937 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1938 unauthorized modifications.

1939 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

1940 Devices should employ industry-standard high-assurance techniques when allowing off-device key
1941 pair creation and provisioning. Use of such key pairs should be minimized, particularly if the key
1942 pair is immutable and cannot be changed or replaced after provisioning.

1943 When used as part of onboarding, these key pairs can be used to prove the Device possesses the
1944 manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept
1945 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the
1946 Device, and then provisions new OCF Security Domain credentials for use.

1947 **9.3.5 Asymmetric Key Encryption Key Credentials**

1948 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when
1949 distributing or storing the key.

1950 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

1951 The PublicData Property contains the public key.

1952 The OptionalData Property may contain revocation status.

1953 The Device implementer should apply hardened key storage techniques that ensure the
1954 PrivateData remains private.

1955 The Device implementer should apply appropriate integrity, confidentiality and access protection
1956 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1957 unauthorized modifications.

1958 **9.3.6 Certificate Credentials**

1959 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a CMS
1960 or an external certificate authority (CA).

1961 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

1962 The issued certificate is stored with the asymmetric key credential Resource.

1963 Other objects useful in managing certificate lifecycle such as certificate revocation status are
1964 associated with the credential Resource.

1965 Either an asymmetric key credential Resource or a self-signed certificate credential is used to
1966 terminate a path validation.

1967 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

1968 The PublicData Property contains the issued certificate.

1969 The OptionalData Property may contain revocation status.

1970 The Device implementer should apply hardened key storage techniques that ensure the
1971 PrivateData remains private.

1972 The Device implementer should apply appropriate integrity, confidentiality and access protection
1973 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1974 unauthorized modifications.

1975 **9.3.7 Password Credentials**

1976 Shared secret credentials are used to maintain a PIN or password that authorizes Device access
1977 to a foreign system or Device that doesn't support any other OCF credential types.

1978 The PrivateData Property in the "/oic/sec/cred" Resource contains the PIN, password and other
1979 values useful for changing and verifying the password.

1980 The PublicData Property may contain the user or account name if applicable.

1981 The OptionalData Property may contain revocation status.

1982 The Device implementer should apply hardened key storage techniques that ensure the
1983 PrivateData remains private.

1984 The Device implementer should apply appropriate integrity, confidentiality and access protection
1985 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent
1986 unauthorized modifications.

1987 **9.4 Certificate Based Key Management**

1988 **9.4.1 Overview**

1989 To achieve authentication and transport security during communications in OCF Security Domain,
1990 certificates containing public keys of communicating parties and private keys can be used.

1991 The certificate and private key may be issued by a local or remote certificate authority (CA). For
1992 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of identity.
1993 In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to validate
1994 proof of identity and validity.

1995 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,
1996 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in X.509
1997 are not supported so that the format intends to meet the constrained Device's requirement.

1998 As for the certificate and CRL management in the Server, the process of storing, retrieving and
1999 parsing Resources of the certificates and CRL will be performed at the security resource manager
2000 layer; the relevant interfaces may be exposed to the upper layer.

2001 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of
2002 certificates and CRL will be stored and managed in SVR database.

2003 The CMS manages the certificate lifecycle for certificates it issues. The DOTS assigns a CMS to a
2004 Device when it is newly onboarded.

2005 **9.4.2 X.509 Digital Certificate Profiles**

2006 **9.4.2.1 Digital Certificate Profile General**

2007 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in
2008 IETF RFC 5280.

2009 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications for
2010 those communities wishing to make use of X.509 technology. The X.509 v3 certificate format is
2011 described in detail, with additional information regarding the format and semantics of OCF specific
2012 extension(s). The supported standard certificate extensions are also listed.

2013 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this
2014 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are
2015 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,
2016 compliant entities shall ignore their contents.

2017 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as
2018 defined in ISO/IEC 8825-1 to encode certificates.

2019 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public
2020 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs
2021 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and
2022 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2023 The following clauses specify the supported standard and custom extensions for the OCF
2024 certificates profile.

2025 **9.4.2.2 Certificate Profile and Fields**

2026 **9.4.2.2.1 Root CA Certificate Profile**

2027 Table 13 describes X.509 v1 fields required for Root CA Certificates.

2028 **Table 13 – 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)
-------------------------	--------------------------------------------------------------------------------

2029 Table 14 describes X.509 v3 extensions required for Root CA Certificates.

2030 **Table 14 - 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)

2031 **9.4.2.2.2 Intermediate CA Certificate Profile**

2032 Table 15 describes X.509 v1 fields required for Intermediate CA Certificates.

2033 **Table 15 - 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)

2034 Table 16 describes X.509 v3 extensions required for Intermediate CA Certificates.

2035 **Table 16 – 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

2036 **9.4.2.2.3 End-Entity Black Certificate Profile**

2037 Table 17 describes X.509 v1 fields required for End-Entity Certificates used for Black security
2038 profile.

2039 **Table 17 – 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)

2040 Table 18 describes X.509 v3 extensions required for End-Entity Certificates.

2041 **Table 18 – X.509 v3 extensions for End-Entity Certificates**

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	digitalSignature (0) and keyAgreement(4) bits SHALL be the only bits enabled

basicConstraints	OPTIONAL	Non-Critical	cA = FALSE pathLenConstraint = not present
certificatePolicies	OPTIONAL	Non-critical	End-Entity certificates chaining to an OCF Root CA SHOULD contain at least one PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2) corresponding to the version of the OCF Certificate Policy under which it was issued. Additional manufacturer-specific CP OIDs may also be populated.
extendedKeyUsage	REQUIRED	Non-critical	The following extendedKeyUsage (EKU) OIDs SHALL both be present: <ul style="list-style-type: none"> • serverAuthentication - 1.3.6.1.5.5.7.3.1 • clientAuthentication - 1.3.6.1.5.5.7.3.2 Exactly ONE of the following OIDs SHALL be present: <ul style="list-style-type: none"> • Identity certificate - 1.3.6.1.4.1.44924.1.6 • Role certificate - 1.3.6.1.4.1.44924.1.7 End-Entity certificates SHALL NOT contain the anyExtendedKeyUsage OID (2.5.29.37.0)
subjectAlternativeName	REQUIRED UNDER CERTAIN CONDITIONS	Non-critical	The subjectAltName extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. When the extendedKeyUsage (EKU) extension contains the Identity Certificate OID (1.3.6.1.4.1.44924.1.6), the subjectAltName extension SHOULD NOT be present. If the EKU extension contains the Role Certificate OID (1.3.6.1.4.1.44924.1.7), the subjectAltName extension SHALL be present and populated as follows: Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name shall contain exactly one CN (Common Name) component, and zero

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

2042 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2043 The OCF Compliance Extension defines required parameters to correctly identify the type of Device,
2044 its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2045 The extension carries an "ocfVersion" field which provides the specific base version of the OCF
2046 documents the device implements. The "ocfVersion" field shall contain a sequence of three integers
2047 ("major", "minor", and "build"). For example, if an entity is certified to be compliant with OCF
2048 specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be set to
2049 "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote
2050 compliance to a specified base version of the OCF documents.

2051 The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more
2052 supported Security Profiles associated with the certificate in string form (UTF-8). All Security
2053 Profiles associated with the certificate should be identified by this field.

2054 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".
2055 The fields carry human-readable descriptions of the Device's name and manufacturer, respectively.

2056 The ASN.1 definition of the OCFCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined as
2057 follows:

```
2058 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2059                               private(4) enterprise(1) OCF(51414) }
2060
2061   id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2062
2063   id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2064
2065 ocfVersion ::= SEQUENCE {
2066     major INTEGER,
2067     --Major version number
2068     minor INTEGER,
2069     --Minor version number
2070     build INTEGER,
2071     --Build/Micro version number
2072 }
2073
2074 ocfCompliance ::= SEQUENCE {
2075     version ocfVersion,
2076     --Device/OCF version
2077     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2078     --Sequence of OCF Security Profile OID strings
2079     --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2080     deviceName UTF8String,
2081     --Name of the device
2082     deviceManufacturer UTF8String,
2083     --Human-Readable Manufacturer
2084     --of the device
2085 }
```

2086 9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension

2087 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for devices
2088 to signal to the network the access and network functionality they require to properly function.
2089 Access controls can be more easily achieved and deployed at scale when the MUD extension is
2090 used. The current draft of the MUD v3 extension at this time of writing is:

2091 <https://tools.ietf.org/html/rfc8520#section-11>

2092 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2093 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2094                       internet(1) security(5) mechanisms(5) pkix(7)
2095                       id-mod(0) id-mod-mudURLExtn2016(88) }
2096
2097 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2098 -- EXPORTS ALL --
2099 IMPORTS
2100     EXTENSION
2101     FROM PKIX-CommonTypes-2009
2102     { iso(1) identified-organization(3) dod(6) internet(1)
2103       security(5) mechanisms(5) pkix(7) id-mod(0)
2104       id-mod-pkixCommon-02(57) }
2105     id-pe
2106     FROM PKIX1Explicit-2009
2107     { iso(1) identified-organization(3) dod(6) internet(1)
2108       security(5) mechanisms(5) pkix(7) id-mod(0)
2109       id-mod-pkix1-explicit-02(51) } ;
2110 MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2111 ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2112                             IDENTIFIED BY id-pe-mud-url }
```

```
2113
2114         id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2115
2116         MUDURLSyntax ::= IA5String
2117
2118     END
```

2119 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2120 The OCF Security Claims Extension defines a list of OIDs representing security claims that the
2121 manufacturer/integrator is making as to the security posture of the device above those required by
2122 the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2123 The purpose of this extension is to allow for programmatic evaluation of assertions made about
2124 security to enable some platforms/policies/administrators to better understand what is being
2125 onboarded or challenged.

2126 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is defined
2127 as follows:

```
2128 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2129         private(4) enterprise(1) OCF(51414) }
2130
2131     id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2132
2133     id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2134
2135         claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2136         --Device claims that the boot process follows a procedure trusted
2137         --by the firmware and the BIOS
2138
2139         claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2140         --Device claims that credentials are stored in a specialized hardware
2141         --protection environment such as a Trusted Platform Module (TPM) or
2142         --similar mechanism.
2143
2144         ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2145
2146     ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
```

2147 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2148 The OCF Certified Product List Extension defines required parameters to utilize the OCF
2149 Compliance Management System Certified Product List (OCMS-CPL). This clause is only
2150 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to verify
2151 the compliance level of a device.

2152 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model and
2153 Version.

2154 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique
2155 PEN established in the IANA PEN list located at: [https://www.iana.org/assignments/enterprise-](https://www.iana.org/assignments/enterprise-numbers)
2156 [numbers](https://www.iana.org/assignments/enterprise-numbers). The 'cpl-at-IANAPen' field found in end-products shall be the same information as
2157 reported during OCF Certification.

2158 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field
2159 found in end-products shall be the same information as reported during OCF Certification.

2160 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found
2161 in end-products shall be the same information as reported during OCF Certification.

2162 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is defined
2163 as follows:

```
2164 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2165                               private(4) enterprise(1) OCF(51414) }
2166
2167 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2168
2169   id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2170
2171     cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2172     cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2173     cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2174
2175
2176   ocfCPLAttributes ::= SEQUENCE {
2177     cpl-at-IANAPen      UTF8String,
2178                       --Manufacturer's registered IANA Private Enterprise Number
2179     cpl-at-model       UTF8String,
2180                       --Device OCF Security Profile
2181     cpl-at-version     UTF8String
2182                       --Name of the device
2183   }
```

2184 9.4.2.3 Supported Certificate Extensions

2185 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the
2186 clause number from that RFC to include it by reference. Each extension is summarized here, and
2187 any modifications to the RFC definition are listed. Devices MUST implement and understand the
2188 extensions listed here; other extensions from the RFC are not included in this document and
2189 therefore are not required. 10.4 describes what Devices must implement when validating certificate
2190 chains, including processing of extensions, and actions to take when certain extensions are absent.

2191 – Authority Key Identifier (4.2.1.1)

2192 The Authority Key Identifier (AKI) extension provides a means of identifying the public key
2193 corresponding to the private key used to sign a certificate. This document makes the following
2194 modifications to the referenced definition of this extension:

2195 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier
2196 sequence are not permitted; only keyIdentifier is allowed. This results in the following grammar
2197 definition:

```
2198 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2199
2200 AuthorityKeyIdentifier ::= SEQUENCE {
2201   keyIdentifier          [0] KeyIdentifier
2202 }
2203 KeyIdentifier ::= OCTET STRING
```

2204 – Subject Key Identifier (4.2.1.2)

2205 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that
2206 contain a particular public key.

2207 This document makes the following modification to the referenced definition of this extension:

2208 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's
2209 SubjectPublicKeyInfo field or a method that generates unique values. This document
2210 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey
2211 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains must
2212 not assume any particular method of computing key identifiers, however, and must only base
2213 matching AKI's and SKI's in certification path constructions on key identifiers seen in certificates.

2214 – Subject Alternative Name

2215 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role
2216 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and
2217 interpreted as described below. When no EKU is present, or has another value, the
2218 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to
2219 encode one or more Role ID values in role certificates, binding the roles to the subject public
2220 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2221 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2222
2223 SubjectAltName ::= GeneralNames
2224
2225 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2226
2227 GeneralName ::= CHOICE {
2228     otherName                [0]     OtherName,
2229     rfc5322Name              [1]     IA5String,
2230     dNSName                  [2]     IA5String,
2231     x400Address              [3]     ORAddress,
2232     directoryName            [4]     Name,
2233     ediPartyName             [5]     EDIPartyName,
2234     uniformResourceIdentifier [6]     IA5String,
2235     iPAddress                [7]     OCTET STRING,
2236     registeredID             [8]     OBJECT IDENTIFIER }
2237
2238     EDIPartyName ::= SEQUENCE {
2239         nameAssigner          [0]     DirectoryString OPTIONAL,
2240         partyName             [1]     DirectoryString }
2241
```

2242 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a
2243 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name
2244 shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational
2245 Unit) components. The OU component, if present, shall specify the authority that defined the
2246 semantics of the role. If the OU component is absent, the certificate issuer has defined the role.
2247 The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may
2248 be present, but shall not be interpreted as roles. Therefore, if the certificate issuer includes
2249 non-role names in the subjectAltName extension, the extension should not be marked critical.

2250 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted
2251 character set [0-9a-z-A-z '()+, -./:=?].

2252 – Key Usage (4.2.1.3)

2253 The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing)
2254 of the key contained in the certificate. The usage restriction might be employed when a key that
2255 could be used for more than one operation is to be restricted.

2256 This document does not modify the referenced definition of this extension.

2257 – Basic Constraints (4.2.1.9)

2258 The basic constraints extension identifies whether the subject of the certificate is a CA and the
2259 maximum depth of valid certification paths that include this certificate. Without this extension,
2260 a certificate cannot be an issuer of other certificates.

2261 This document does not modify the referenced definition of this extension.

2262 – Extended Key Usage (4.2.1.12)

2263
2264 Extended Key Usage describes allowed purposes for which the certified public key may can be
2265 used. When a Device receives a certificate, it determines the purpose based on the context of
2266 the interaction in which the certificate is presented, and verifies the certificate can be used for
2267 that purpose.

2268 This document makes the following modifications to the referenced definition of this extension:
2269 CAs SHOULD mark this extension as critical.

2270 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).

2271

2272 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2273 – Identity certificate 1.3.6.1.4.1.44924.1.6

2274 – Role certificate 1.3.6.1.4.1.44924.1.7

2275 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2276 See 9.4.3.5 for details.

2277 **9.4.2.5 Encoding of Certificate**

2278 See 9.4.2 for details.

2279 **9.4.3 Certificate Revocation List (CRL) Profile**

2280 **9.4.3.1 CRL General**

2281 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use within
2282 OCF applications for those communities wishing to support revocation features in their PKIs.

2283 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in
2284 IETF RFC 5280 – Clause 5.1

2285 **9.4.3.2 CRL Profile and Fields**

2286 This clause intentionally left empty.

2287 **9.4.3.3 Encoding of CRL**

2288 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]
2289 should be used to encode CRL.

2290 **9.4.3.4 CRLs Supported Standard Extensions**

2291 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55] provide
2292 methods for associating additional attributes with CRLs. The following list of X.509 extensions
2293 should be supported in this certificate profile:

2294 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides
2295 a means of identifying the public key corresponding to the private key used to sign a CRL.
2296 Conforming CRL issuers should use the key identifier method, and shall include this extension
2297 in all CRLs issued

2298 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that
2299 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer

2300 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for
2301 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]
2302 [X9.55]. Although this document does not provide any recommendation about the use of specific
2303 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not marked
2304 critical.

2305 **9.4.3.5 Encryption Ciphers and TLS support**

2306 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support
2307 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite as defined in IETF RFC 7251 and may
2308 support additional ciphers as defined in the TLS v1.2 specifications.

2309 **9.4.4 Resource Model**

2310 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated
2311 with a separate crl Resource that is defined for maintaining the revocation list.

2312 The cred Resource contains the certificate information pertaining to the Device. The PublicData
2313 Property holds the device certificate and CA certificate chain. PrivateData Property holds the
2314 Device private key paired to the certificate. (See 13.3 for additional detail regarding the
2315 "/oic/sec/cred" Resource).

2316 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained
2317 through the CMS. The Device must consider revoked certificates as part of certificate path
2318 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain a
2319 full list, the Device must query the CMS for current revocation status. (See 13.4 for additional detail
2320 regarding the "/oic/sec/crl" Resource).

2321 **9.4.5 Certificate Provisioning**

2322 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices.

2323 The CA in the CMS retrieves a Device's public key and proof of possession of the private key,
2324 generates a Device's certificate signed by this CA certificate, and then the CMS transfers them to
2325 the Device including its CA certificate chain. Optionally, the CMS can also transfer one or more
2326 role certificates, which shall have the format described in clause 9.4.2. The subjectPublicKey of
2327 each role certificate shall match the subjectPublicKey in the Device certificate.

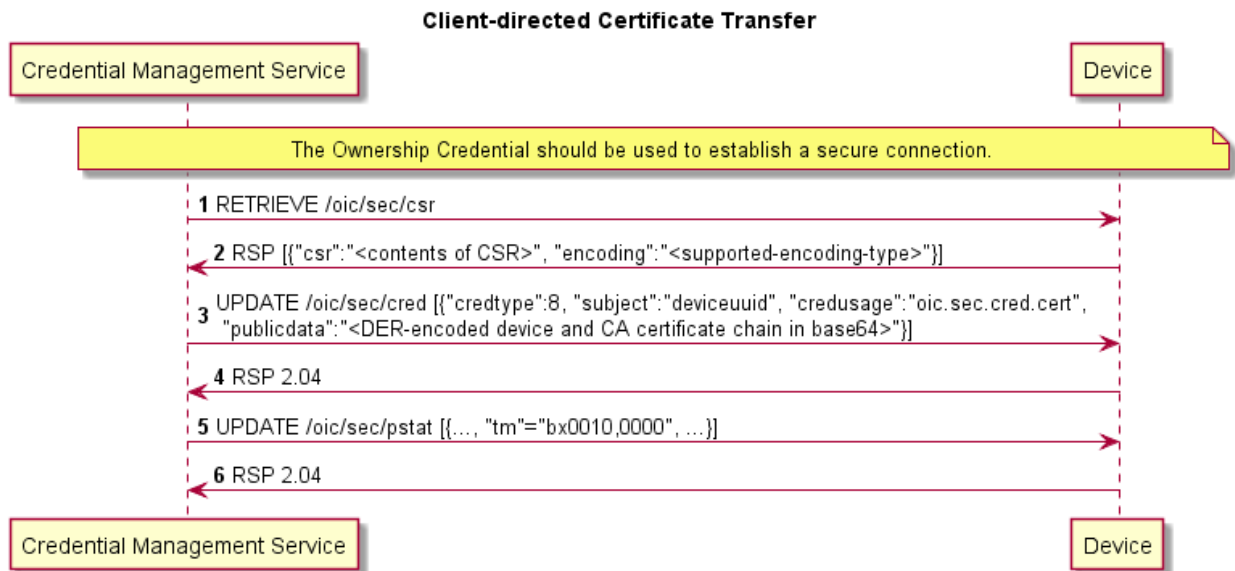
2328 In the sequence in Figure 24, the Certificate Signing Request (CSR) is defined by PKCS#10 in
2329 IETF RFC 2986, and is included here by reference.

2330 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 24.

2331 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device
2332 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.
2333 The Device determines the public key to present; this may be an already-provisioned key it has
2334 selected for use with authentication, or if none is present, it may generate a new key pair
2335 internally and provide the public part. The key pair shall be compatible with the allowed
2336 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF
2337 authentication.

2338 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on
2339 its own, then it is not capable of using certificates. The Device shall advertise this fact both by
2340 setting the 0x8 bit position in the sct Property of "/oic/sec/doxm" to 0, and return an error that
2341 the "/oic/sec/csr" resource does not exist.

2342 3) The CMS transfers the issued certificate and CA chain to the designated Device using the same
2343 credid, to maintain the association with the private key. The credential type ("oic.sec.cred")
2344 used to transfer certificates in Figure 24 is also used to transfer role certificates, by including
2345 multiple credentials in the POST from CMS to Device. Identity certificates shall be stored with
2346 the credusage Property set to "oic.sec.cred.cert" and role certificates shall be stored with the
2347 credusage Property set to "oic.sec.cred.rolecert".



2348

2349 **Figure 24 – Client-directed Certificate Transfer**

2350 **9.4.6 CRL Provisioning**

2351 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the function
 2352 to register revocation certificates, to sign CRL and to transfer it to Devices.

2353 The CMS sends the CRL to the Device.

2354 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2355 – change of issuer name
- 2356 – change of association between Devices and CA
- 2357 – certificate compromise
- 2358 – suspected compromise of the corresponding private key

2359 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In some
 2360 special cases, Devices may request CRL to a given CMS.

2361 There are two options to update and deliver CRL;

- 2362 – CMS pushes CRL to each Device
- 2363 – each Device periodically requests to update CRL

2364 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 25.

- 2365 1) The CMS may retrieve the CRL Resource Property.
- 2366 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.

2367

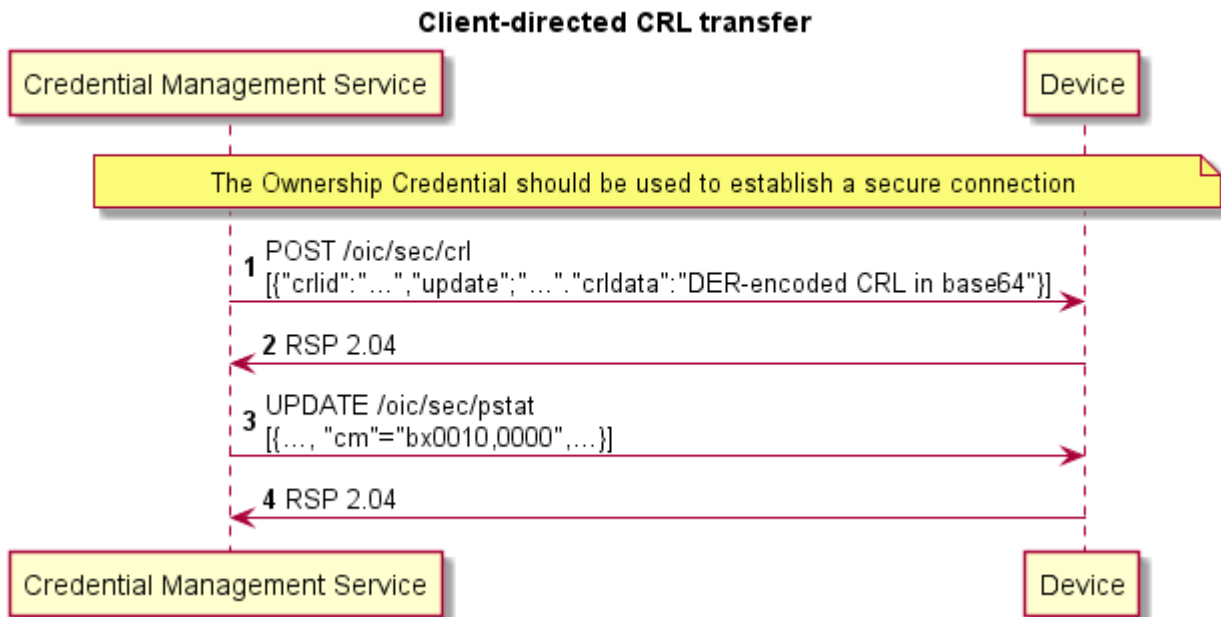


Figure 25 – Client-directed CRL Transfer

2369

2370

2371 **10 Device Authentication**

2372 **10.1 Device Authentication General**

2373 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the
2374 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or
2375 more roles that the server can use in access control decisions. Roles may be asserted when the
2376 Device authentication is done with certificates.

2377 **10.2 Device Authentication with Symmetric Key Credentials**

2378 When using symmetric keys to authenticate, the Server Device shall include the
2379 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client shall
2380 validate that it has a credential with the Subject ID set to the Server's Device ID, and a credential
2381 type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error or other
2382 suitable error.

2383 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that
2384 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a
2385 credential with the matching Subject ID and type. If it does not, the Server shall respond with an
2386 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS
2387 protocol, and both Client and Server shall compute the resulting premaster secret.

2388 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2389 When using raw asymmetric keys to authenticate, the Client and the Server shall include a suitable
2390 public key from a credential that is bound to their Device. Each Device shall verify that the provided
2391 public key matches the `PublicData` field of a credential they have, and use the corresponding
2392 Subject ID of the credential to identify the peer Device.

2393 **10.4 Device Authentication with Certificates**

2394 **10.4.1 Device Authentication with Certificates General**

2395 When using certificates to authenticate, the Client and Server shall each include their certificate
2396 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.
2397 Each Device shall validate the certificate chain presented by the peer Device. Each certificate
2398 signature shall be verified until a public key is found within the `"/oic/sec/cred"` Resource with the
2399 `"oic.sec.cred.trustca"` credusage. Credential Resource found in `"/oic/sec/cred"` is used to terminate
2400 certificate path validation. Also, the validity period and revocation status should be checked for all
2401 above certificates, but at this time a failure to obtain a certificate's revocation status (CRL or OCSP
2402 response) MAY continue to allow the use of the certificate if all other verification checks succeed.

2403 If available, revocation information should be used to verify the revocation status of the certificate.
2404 The URL referencing the revocation information should be retrieved from the certificate (via the
2405 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used
2406 to gather relevant revocation information like CRLs or OCSP responses.

2407 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2408 Devices must follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In
2409 particular:

- 2410 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is
2411 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate chain
2412 MUST be rejected. If the `pathLenConstraint` field is present, Devices will confirm the number of
2413 certificates between this certificate and the End-Entity certificate is less than or equal to
2414 `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity certificate can
2415 be issued by this certificate. If the `pathLenConstraint` field is absent, there is no limit to the
2416 chain length.

- 2417 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is present,
2418 and that the keyCertSign bit is asserted.
- 2419 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.
2420 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt
2421 validation with the public keys of possible issuer certificates whose subject name equals the
2422 issuer name of this certificate.
- 2423 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage (EKU)
2424 suitable to the purpose for which it is being presented. An End-Entity certificate which contains
2425 no EKU extension is not valid for any purpose and must be rejected. Any certificate which
2426 contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other valid
2427 EKUs are also present.
- 2428 – Devices MUST verify "transitive EKU" for certificate chains. Issuer certificates (any certificate
2429 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the certificate
2430 chain is being presented. An issuer certificate is valid for a purpose if it contains an EKU
2431 extension and the EKU OID for that purpose is listed in the extension, OR it does not have an
2432 EKU extension. An issuer certificate SHOULD contain an EKU extension and a complete list of
2433 EKUs for the purposes for which it is authorized to issue certificates. An issuer certificate
2434 without an EKU extension is valid for all purposes; this differs from End-Entity certificates
2435 without an EKU extension.
- 2436 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2437 If the Device does not recognize an extension, it must examine the `critical` field. If the field is
2438 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the
2439 certificate as if the extension were absent and proceed accordingly. This applies to all certificates
2440 in a chain.

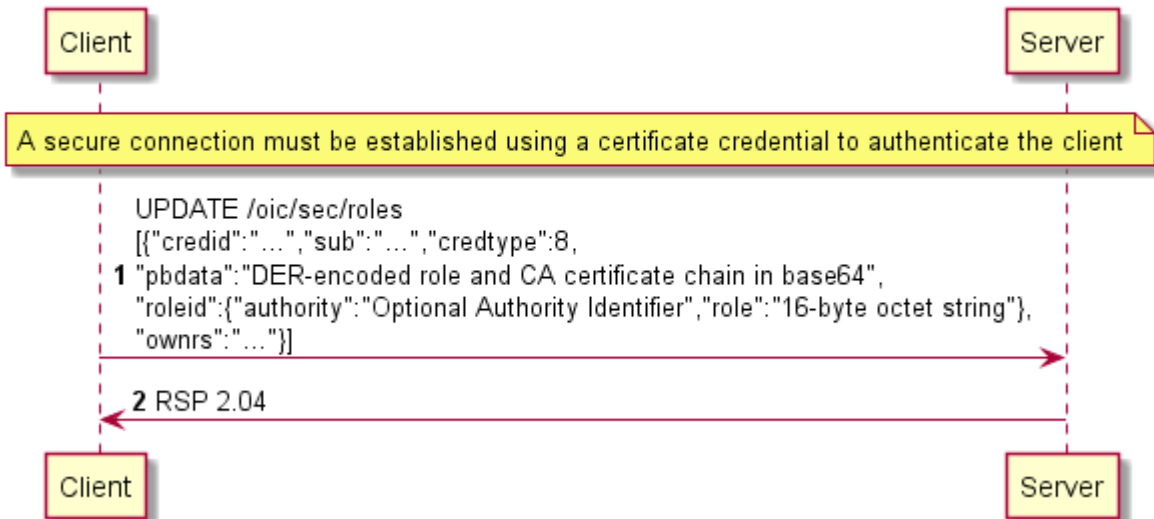
2441 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

2442 **10.4.2 Role Assertion with Certificates**

2443 This clause describes role assertion by a client to a server using a certificate role credential. If a
2444 server does not support the certificate credential type, clients should not attempt to assert roles
2445 with certificates.

2446 Following authentication with a certificate, a client may assert one or more roles by updating the
2447 server's roles resource with the role certificates it wants to use. The role credentials must be
2448 certificate credentials and shall include a certificate chain. The server shall validate each certificate
2449 chain as specified in clause 10.3. Additionally, the public key in the End-Entity certificate used for
2450 Device authentication must be identical to the public key in all role (End-Entity) certificates. Also,
2451 the subject distinguished name in the End-Entity authentication and role certificates must match.
2452 The roles asserted are encoded in the subjectAltName extension in the certificate. The
2453 subjectAltName field can have multiple values, allowing a single certificate to encode multiple roles
2454 that apply to the client. The server shall also check that the EKU extension of the role certificate(s)
2455 contains the value 1.3.6.1.4.1.44924.1.7 (see clause 9.4.2.2) indicating the certificate may be used
2456 to assert roles. Figure 26 describes how a client Device asserts roles to a server.

Asserting Certificate Role Credentials



2457

2458

Figure 26 – Asserting a role with a certificate role credential.

2459 Additional comments for Figure 26

- 2460 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If
 2461 the server does not support certificate credentials, it should return "501 Not Implemented"
- 2462 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration shall
 2463 not exceed the validity period of the role certificate. When fresh CRL information is obtained,
 2464 the certificates in "/oic/sec/roles" should be checked, and the role removed if the certificate is
 2465 revoked or expired.
- 2466 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role
 2467 by a client. It is recommended that servers use the validity period of the certificate as a duration,
 2468 effectively allowing the CMS to decide the duration.
- 2469 4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see
 2470 Table 24). They shall have credtype 8 (indicating certificates) and PrivateData field shall not
 2471 be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the
 2472 value in the certificate shall be used for validation. For example, if the Period field is set in the
 2473 credential, the server shall treat the validity period in the certificate as authoritative. Similar for
 2474 the roleid data (authority, role).
- 2475 5) Certificates shall be encoded as in Figure 24 (DER-encoded certificate chain in base64)
- 2476 6) Clients may GET the "/oic/sec/roles" resource to determine the roles that have been previously
 2477 asserted. An array of credential objects shall be returned. If there are no valid certificates
 2478 corresponding to the currently connected and authenticated Client's identity, then an empty
 2479 array (i.e. []) shall be returned.

2480 10.4.3 OCF PKI Roots

2481 This clause intentionally left empty.

2482 10.4.4 PKI Trust Store

2483 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store the
 2484 OCF Root CA certificates in the "oic/sec/cred" resource and SHOULD physically store this resource
 2485 in a hardened memory location where the certificates cannot be tampered with.

2486 **10.4.5 Path Validation and extension processing**

2487 Devices SHALL follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In
2488 addition, the following are best practices and SHALL be adhered to by any OCF-compliant
2489 application handling digital certificates

2490 – Validity Period checking

2491 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and
2492 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,
2493 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2494 – Revocation checking

2495 Relying applications SHOULD check the revocation status for all certificates, but at this time,
2496 an application MAY continue to allow the use of the certificate upon a failure to obtain a
2497 certificate's revocation status (CRL or OCSP response), if all other verification checks succeed.

2498 – basicConstraints

2499 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify that
2500 the basicConstraints extension is present, flagged critical, and that the cA boolean value in the
2501 extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2502 If the pathLenConstraint field is present, Devices will confirm the number of certificates between
2503 this certificate and the End-Entity certificate is less than or equal to pathLenConstraint. In
2504 particular, if pathLenConstraint is zero, only an End-Entity certificate can be issued by this
2505 certificate. If the pathLenConstraint field is absent, there is no limit to the chain length.

2506 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged
2507 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a
2508 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a
2509 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a
2510 null value.

2511 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-compliant
2512 Root CA certificates SHALL NOT contain a pathLenConstraint. This allows additional tiers of
2513 Intermediate CAs to be implemented in the future without changing the Root CA trust anchors,
2514 should such a requirement emerge.

2515 – keyUsage

2516 For all certificates, Devices shall verify that the key usage extension is present and flagged
2517 critical.

2518 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits SHALL
2519 be asserted.

2520 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be
2521 asserted.

2522 – extendedKeyUsage:

2523 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be
2524 rejected.

2525 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)
2526 are required for compatibility with various TLS implementations.

2527 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)
2528 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be
2529 present and End-Entity certificates with EKU extensions containing both OIDs SHALL be
2530 rejected.

2531 – certificatePolicies

2532 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one
2533 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)
2534 corresponding to the version of the OCF Certificate Policy under which it was issued. Additional
2535 manufacturer-specific CP OIDs may also be populated.

2536 **10.5 Device Authentication with OCF Cloud – moved to OCF Cloud Security document**

2537 This clause is intentionally left blank.

2538 .

2539

2540 **11 Message Integrity and Confidentiality**

2541 **11.1 Preamble**

2542 Secured communications between Clients and Servers are protected against eavesdropping,
2543 tampering, or message replay, using security mechanisms that provide message confidentiality and
2544 integrity.

2545 **11.2 Session Protection with DTLS**

2546 **11.2.1 DTLS Protection General**

2547 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices
2548 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See
2549 11.3 for a list of required and optional cipher suites for message communication.

2550 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1
2551 or lower.

2552 Multicast session semantics are not yet defined in this version of the security document.

2553 **11.2.2 Unicast Session Semantics**

2554 For unicast messages between a Client and a Server, both Devices shall authenticate each other.
2555 See clause 10 for details on Device Authentication.

2556 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.
2557 The sending Device shall encrypt and authenticate messages as defined by the selected cipher
2558 suite and the receiving Device shall verify and decrypt the messages before processing them.

2559 **11.2.3 Cloud Session Semantics – moved to OCF Cloud Security document**

2560 This clause is intentionally left blank.

2561 **11.3 Cipher Suites**

2562 **11.3.1 Cipher Suites General**

2563 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher
2564 suites allowed during ownership transfer and normal operation. The following RFCs provide
2565 additional information about the cipher suites used in OCF.

2566 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2567 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2568 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and
2569 PSKs

2570 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2571 **11.3.2 Cipher Suites for Device Ownership Transfer**

2572 **11.3.2.1 Just Works Method Cipher Suites**

2573 The Just Works OTM may use the following (D)TLS cipher suites.

2574 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,

2575 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

2576 All Devices supporting Just Works OTM shall implement:

2577 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (with the value 0xFF00)

2578 All Devices supporting Just Works OTM should implement:

2579 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256 (with the value 0xFF01)

2580 **11.3.2.2 Random PIN Method Cipher Suites**

2581 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2582 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2583 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2584 All Devices supporting Random Pin Based OTM shall implement:

2585 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

2586 **11.3.2.3 Certificate Method Cipher Suites**

2587 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2588 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

2589 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2590 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2591 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2592 Using the following curve:

2593 secp256r1 (See IETF RFC 4492)

2594 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2595 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2596 Devices supporting Manufacturer Certificate Based OTM should implement:

2597 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2598 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2599 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2600 **11.3.3 Cipher Suites for Symmetric Keys**

2601 The following cipher suites are defined for (D)TLS communication using PSKs:

2602 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2603 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2604 TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET Authentication tag *)

2605 TLS_PSK_WITH_AES_256_CCM_8,

2606 TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET Authentication tag *)

2607 TLS_PSK_WITH_AES_256_CCM,

2608 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2609 All Devices shall implement the following:

2610 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2611

2612 Devices should implement the following:

2613 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
2614 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,
2615 TLS_PSK_WITH_AES_128_CCM_8,
2616 TLS_PSK_WITH_AES_256_CCM_8,
2617 TLS_PSK_WITH_AES_128_CCM,
2618 TLS_PSK_WITH_AES_256_CCM

2619 **11.3.4 Cipher Suites for Asymmetric Credentials**

2620 The following cipher suites are defined for (D)TLS communication with asymmetric keys or
2621 certificates:

2622 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,
2623 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
2624 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
2625 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2626 Using the following curve:

2627 secp256r1 (See IETF RFC 4492)

2628 All Devices supporting Asymmetric Credentials shall implement:

2629 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2630 All Devices supporting Asymmetric Credentials should implement:

2631 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
2632 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
2633 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2634 **11.3.5 Cipher suites for OCF Cloud Credentials – moved to OCF Cloud Security document**

2635 This clause is intentionally left blank.

2636

2637 **12 Access Control**

2638 **12.1 ACL Generation and Management**

2639 This clause intentionally left empty.

2640 **12.2 ACL Evaluation and Enforcement**

2641 **12.2.1 ACL Evaluation and Enforcement General**

2642 The Server enforces access control over application Resources before exposing them to the
2643 requestor. The Security Layer in the Server authenticates the requestor when access is received
2644 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL
2645 entries that specify the requestor's identity, role or may match authenticated requestors using a
2646 subject wildcard.

2647 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a
2648 subject wildcard match of anonymous requestors.

2649 Access is denied if a requested Resource is not matched by an ACL entry.

2650 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources
2651 may be granted prior to provisioning of ACL Resources.

2652 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries (ACE2)
2653 that employ a Resource matching algorithm that uses an array of Resource references to match
2654 Resources to which the ACE2 access policy applies. Matching consists of comparing the values of
2655 the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources are matched
2656 in two ways:

- 2657 1) host reference ("href")
- 2658 2) resource wildcard ("wc").

2659 **12.2.2 Host Reference Matching**

2660 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for
2661 Resource matching.

2662 – The href Property shall be used to find an exact match of the Resource name if present.

2663 **12.2.3 Resource Wildcard Matching**

2664 When present, a wildcard (wc) expression shall be used to match multiple Resources using a
2665 wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

2666 A wildcard expression may be used to match multiple Resources using a wildcard Property
2667 contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined
2668 in Table 19.

2669 **Table 19 – 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.

2670 NOTE Discoverable resources appear in the "/oic/res" Resource, while non-discoverable resources may appear in other
2671 collection resources but do not appear in the /res collection.

2672 **12.2.4 Multiple Criteria Matching**

2673 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for
2674 each array element. For example, if a first array element of the "resources" Property contains
2675 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",
2676 then Resources that match either of the two "href" criteria shall be included in the set of matched
2677 Resources.

2678 Example 1 JSON for Resource matching

```
2679 {  
2680 //Matches Resources named "/x/door1" or "/x/door2"  
2681 "resources": [  
2682   {  
2683     "href": "/x/door1"  
2684   },  
2685   {  
2686     "href": "/x/door2"  
2687   },  
2688 ]  
2689 }
```

2690 Example 2 JSON for Resource matching

```
2691 {  
2692 // Matches all Resources  
2693 "resources": [  
2694   {  
2695     "wc": "*"   
2696   }  
2697 ]  
2698 }
```

2699 **12.2.5 Subject Matching using Wildcards**

2700 When the ACE subject is specified as the wildcard string "*" any requestor is matched. The OCF
2701 server may authenticate the OCF client, but is not required to.

2702 Examples: JSON for subject wildcard matching

```
2703 //matches all subjects that have authenticated and confidentiality protections in place.  
2704 "subject" : {  
2705   "conntype" : "auth-crypt"  
2706 }  
2707 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
2708 "subject" : {  
2709   "conntype" : "anon-clear"  
2710 }
```

2711 **12.2.6 Subject Matching using Roles**

2712 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 2713 1) The requestor authenticated with a symmetric key credential, and the role is present in the
2714 roleid Property of the credential's entry in the credential resource, or

2715 2) The requestor authenticated with a certificate, and a valid role certificate is present in the roles
2716 resource with the requestor's certificate's public key at the time of evaluation. Validating role
2717 certificates is defined in 10.3.1.

2718 **12.2.7 ACL Evaluation**

2719 **12.2.7.1 ACE2 matching algorithm**

2720 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 2721 1) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.
- 2722 2) Access shall be granted when all these criteria are met:
 - 2723 a) The requestor is matched by the ACE2 "subject" Property.
 - 2724 b) The requested Resource is matched by the ACE2 resources Property and the requested
2725 Resource shall exist on the local Server.
 - 2726 c) The "period" Property constraint shall be satisfied.
 - 2727 d) The "permission" Property constraint shall be applied.

2728 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching
2729 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then UNION
2730 (Perm1, Perm2)=CRUDN.

2731 The Server shall enforce access based on the effective permissions granted.

2732 Batch requests to Resource containing Links require additional considerations when accessing the
2733 linked Resources. ACL considerations for batch request to the Atomic Measurement Resource
2734 Type are provided in clause 12.2.7.2. ACL considerations for batch request to the Collection
2735 Resource Type are provided in clause 12.2.7.3.

2736 Clause 12.2.7.4 provides ACL considerations when a new Resource is created on a Server in
2737 response to a CREATE request.

2738 **12.2.7.2 (Currently blank)**

2739 This clause intentionally left empty.

2740 **12.2.7.3 ACL considerations for a batch OCF Interface request to a Collection**

2741 This clause addresses the additional authorization processes which take place when a Server
2742 receives a batch OCF Interface request from a Client to a Collection hosted on that Server,
2743 assuming there is an ACE matching the Collection which permits the original Client request. For
2744 the purposes of this clause, the Server hosting this Collection is called the "Collection host". The
2745 additional authorization process is dependent on whether the linked Resource is hosted on the
2746 Collection host or the linked Resource is hosted on another Server:

- 2747 – For each generated request to a linked Resource hosted on the Collection host, the Collection
2748 host shall apply the ACE2 matching algorithm in clause 12.2.7.1 to determine whether the linked
2749 Resource is permitted to process the generated request, with the following clarifications:
 - 2750 – The requestor in clause 12.2.7.1 shall be the Client which sent the original Client request.
 - 2751 – The requested Resource in clause 12.2.7.1 shall be the linked Resource, which shall be
2752 matched using at least one of:
 - 2753 – a Resource Wildcard matching the linked Resource, or
 - 2754 – an exact match of the local path of the linked Resource with a "href" Property in the
2755 "resources" array in the ACE2.
 - 2756 – an exact match of the full URI of the linked Resource with a "href" Property in the
2757 "resources" array in the ACE2.

2758 NOTE The full URI of a linked Resource is obtained by concatenating the "anchor" Property of the Link, if present, and
2759 the "href" Property of the Link. The local path can then be determined from the full URI.

2760 If the linked Resource is not permitted to process the generated request, then the Collection host
2761 shall treat such cases as a linked Resource which cannot process the request when composing the
2762 aggregated response to the original Client Request, as specified for the batch OCF Interface in the
2763 ISO/IEC 30118-1:2018.

2764 **12.2.7.4 ACL Considerations on creation of a new Resource**

2765 When a new Resource is created on a Server in response to a CREATE request, there might be
2766 no ACEs permitting access to the newly created Resource. The present clause describes how the
2767 Server autonomously modifies the "/oic/sec/acl2" Resource to provide some initial authorizations
2768 for accessing the newly created Resource. The purpose of this autonomous modification is to avoid
2769 relying on the AMS update the "/oic/sec/acl2" Resource after every new Resource is created.

2770 Subsequent to a Server creating a Collection inside another Collection in response to a CREATE
2771 request from a Client, and prior to sending a response to the Client:

- 2772 – If there is an ACE with "subject" containing the UUID of the Client, and "permissions" exactly
2773 matching the CREATE, RETRIEVE, UPDATE and DELETE operations, then the Server shall
2774 autonomously add an "href" entry to "resources" with the URI of the newly created Collection.
- 2775 – Otherwise, the Server shall autonomously add an ACE with "subject" containing the UUID
2776 of the Client, "resources" containing an "href" entry with the URI of the newly created
2777 Collection, and "permissions" exactly matching the CREATE, RETRIEVE, UPDATE and
2778 DELETE operations.

2779 Subsequent to a Server creating a non-Collection Resource inside another Collection in response
2780 to a CREATE request from a Client, and prior to sending a response to the Client:

- 2781 – If there is an ACE with "subject" containing the UUID of the Client, and "permissions" exactly
2782 matching the RETRIEVE, UPDATE and DELETE operations, then the Server shall
2783 autonomously add an "href" entry to "resources" with the URI of the newly created Resource.
- 2784 – Otherwise, the Server shall autonomously add an ACE with "subject" containing the UUID
2785 of the Client, "resources" containing an "href" entry with the URI of the newly created, and
2786 "permissions" exactly matching the RETRIEVE, UPDATE and DELETE operations.

2787

2788 **13 Security Resources**

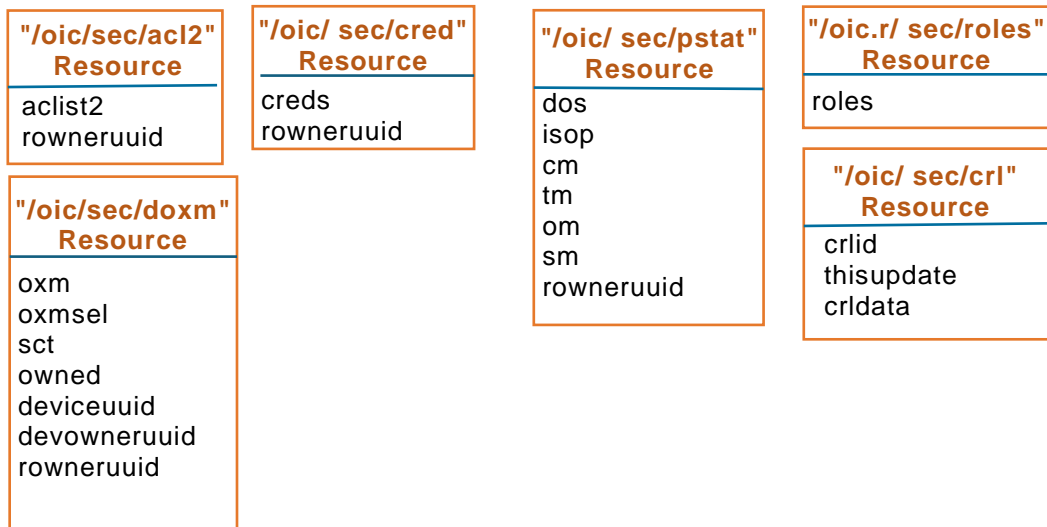
2789 **13.1 Security Resources General**

2790 OCF Security Resources are shown in Figure 27.

2791 "/oic/sec/cred" Resource and Properties are shown in Figure 28.

2792 "/oic/sec/acl2" Resource and Properties are shown in Figure 29.

2793



2794

Figure 27 – OCF Security Resources

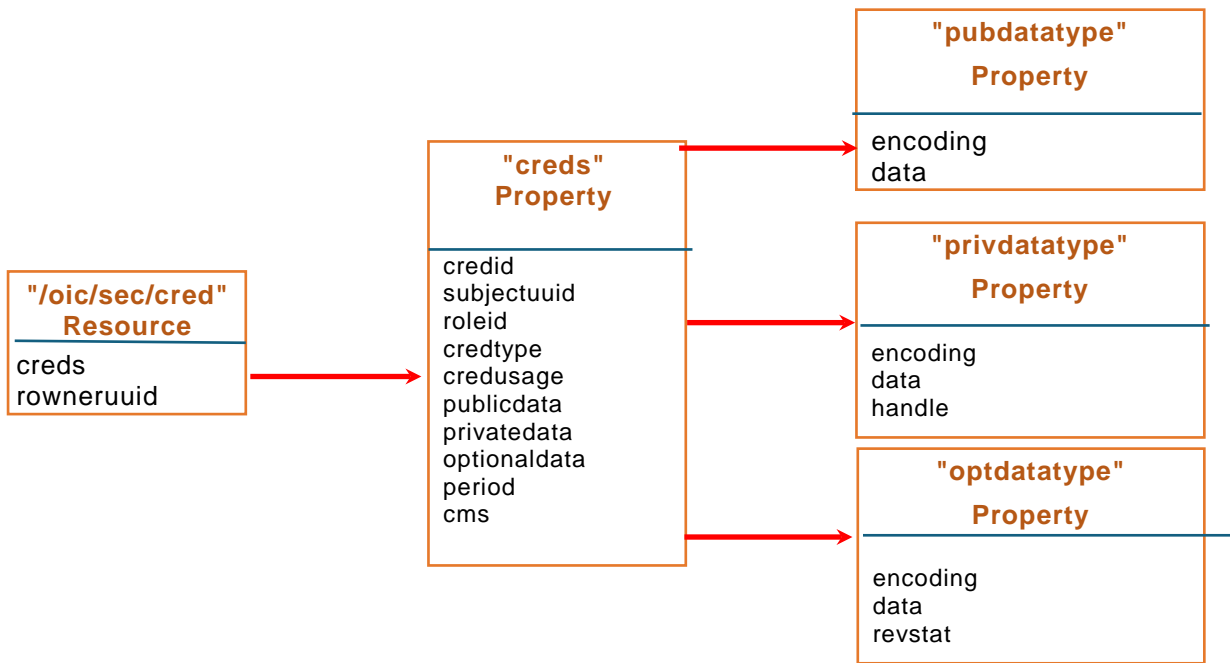


Figure 28 – "/oic/sec/cred" Resource and Properties

2795

2796

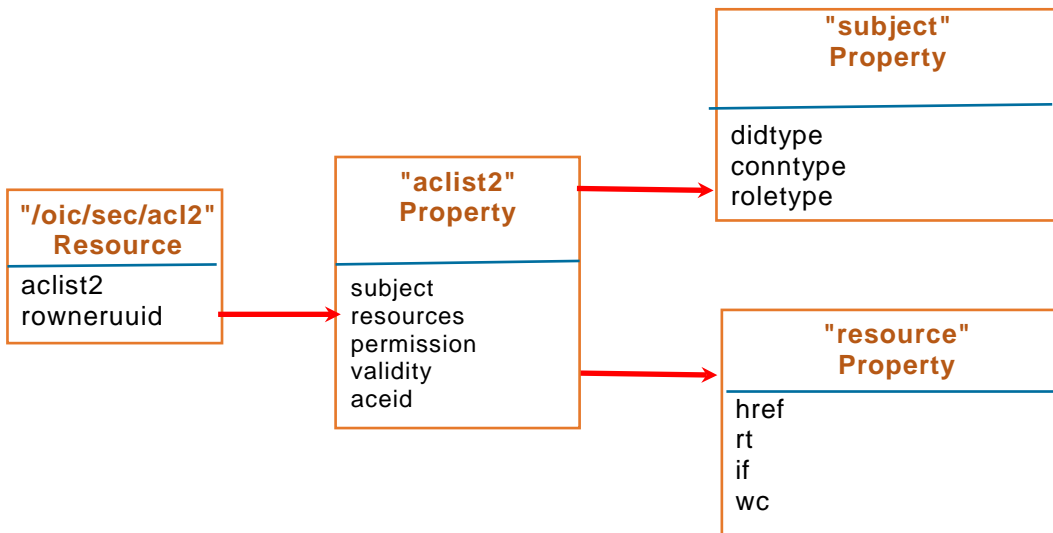


Figure 29 – "/oic/sec/acl2" Resource and Properties

2797

2798

2799 **13.2 Device Owner Transfer Resource**

2800 **13.2.1 Device Owner Transfer Resource General**

2801 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

2802 Resource discovery processing respects the CRUDN constraints supplied as part of the security
 2803 Resource definitions contained in this document.

2804 "/oic/sec/doxm" Resource is defined in Table 20.

2805 **Table 20 – 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

2806 Table 21 defines the Properties of the "/oic/sec/doxm" Resource.

2807 **Table 21 – 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.

2808 Table 22 defines the Properties of the "oic.sec.didtype".

2809 **Table 22 – 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

2810 The oxms Property contains a list of OTM where the entries appear in the order of preference. This
2811 Property contains the higher priority methods appearing before the lower priority methods. The
2812 DOTS queries this list at the time of onboarding and selects the most appropriate method.

2813 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

2814 <DoxmType> ::= <NSS>
2815 <NSS> ::= <Identifier> | { {<NID> "." } <NameSpaceQualifier> "." } <Method>
2816 <NID> ::= <Vendor-or-Organization>
2817 <Identifier> ::= INTEGER
2818 <NameSpaceQualifier> ::= String
2819 <Method> ::= String
2820 <Vendor-Organization> ::= String

```

2821 When an OTM successfully completes, the "owned" Property is set to "1" (TRUE). Consequently,
2822 subsequent attempts to take ownership of the Device will fail.

2823 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in the
2824 "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource is
2825 UPDATED to non-nil UUID value.

2826 The Device vendor shall determine that the Device identifier ("deviceuuid") is persistent (not
2827 updatable) or that it is non-persistent (updatable by the owner transfer service – aka. DOTS).

2828 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall
2829 fail with the error PROPERTY_NOT_FOUND.

2830 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is non-persistent, the request to
2831 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is
2832 RESET. If the UPDATE to deviceuuid Property of the "/oic/sec/doxm" Resource fails while in the
2833 RFOTM Device state the device state shall transition to RESET where the Server shall set the
2834 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g. "00000000-
2835 0000-0000-0000-000000000000").

2836 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the
2837 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the "deviceuuid"
2838 Property of the "/oic/sec/doxm" Resource each time the device enters RESET Device state. The
2839 temporary deviceuuid value is used while the device state is in the RESET state and while in the
2840 RFOTM device state until the DOTS establishes a secure OTM connection.

2841 The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall expose a persistent value (i.e. is
2842 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTS via
2843 an OCF Interface to the deviceuuid Property of the "/oic/sec/doxm" Resource during RFOTM Device
2844 state.).

2845 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a
2846 secure OTM connection and UPDATES the "devowneruuid" Property to a non-nil UUID value.
2847 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the "deviceuuid" Property of
2848 the "/oic/sec/doxm" Resource shall reveal the persistent or semi-persistent value to authenticated
2849 requestors and shall reveal the temporary non-repeated value to unauthenticated requestors.

2850 See 13.16 for additional details related to privacy sensitive considerations.

2851 **13.2.2 Persistent and Semi-Persistent Device Identifiers**

2852 The Device vendor determines whether a device identifier can be set by a configuration tool or
2853 whether it is immutable. If it is an immutable value this document refers to it as a persistent device
2854 identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four device
2855 identifiers that could be considered persistent or semi-persistent:

- 2856 1) "deviceuuid" Property of "/oic/sec/doxm" Resource
- 2857 2) "di" Property of "/oic/d" Resource
- 2858 3) "piid" Property of "/oic/d" Resource
- 2859 4) "pi" Property of "/oic/p" Resource

2860 **13.2.3 Onboarding Considerations for Device Identifier**

2861 The "deviceuuid" is used to onboard the Device. The other identifiers ("di", "piid" and "pi") are not
2862 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the
2863 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security Domain
2864 owner may have a preference for persistent or semi-persistent device identifiers. Detecting whether
2865 the Device is using persistent or semi-persistent deviceuuid can be achieved by attempting to
2866 update it.

2867 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is persistent, then an UPDATE request,
2868 at the appropriate time during onboarding shall fail with an appropriate error response.

2869 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device
 2870 state is RFOTM and when devowneruuid Property value of the "/oic/sec/doxm" Resource has a
 2871 non-nil UUID value.

2872 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is semi-persistent, subsequent to a
 2873 successful UPDATE request to change it; the Device shall remember the semi-persistent value
 2874 until the next successful UPDATE request or until the Device state transitions to RESET.

2875 See 13.16 for addition behaviour regarding "deviceuuid".

2876

2877 **13.2.4 OCF defined OTMs**

2878 Table 23 defines the Properties of the "oic.sec.doxmtype".

2879

Table 23 – Properties of the "oic.sec.doxmtype" type

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

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

2880 **13.3 Credential Resource**

2881 **13.3.1 Credential Resource General**

2882 The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and
 2883 support services as well as credentials used to verify Clients and support services.

2884 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared
 2885 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to
 2886 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

2887 In order to provide an interface which allows management of the "creds" Array Property, the
 2888 RETRIEVE, UPDATE and DELETE operations on the "/oic/sec/cred" Resource shall behave as
 2889 follows:

- 2890 1) A RETRIEVE shall return the full Resource representation, except that any write-only Properties
 2891 shall be omitted (e.g. private key data).
- 2892 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
 2893 UPDATE request, as follows:
 - 2894 a) If an UPDATE representation includes the "creds" array Property, then:
 - 2895 i) Supplied "creds" with a "credid" that matches an existing "credid" shall replace
 2896 completely the corresponding "cred" in the existing "creds" array.
 - 2897 ii) Supplied "creds" without a "credid" shall be appended to the existing "creds" array, and
 2898 a unique (to the cred Resource) "credid" shall be created and assigned to the new "cred"
 2899 by the Server. The "credid" of a deleted "cred" should not be reused, to improve the
 2900 determinism of the interface and reduce opportunity for race conditions.
 - 2901 iii) Supplied "creds" with a "credid" that does not match an existing "credid" shall be
 2902 appended to the existing "creds" array, using the supplied "credid".
 - 2903 iv) The rows in Table 25 corresponding to the "creds" array Property dictate the Device
 2904 States in which an UPDATE of the "creds" array Property is always rejected. If OCF
 2905 Device is in a Device State where the Access Mode in this row contains "R", then the
 2906 OCF Device shall reject all UPDATES of the "creds" array Property.
- 2907 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not remove
 2908 the "/oic/sec/cred" Resource.
- 2909 4) A DELETE with one or more "credid" query parameters shall remove the "cred"(s) with the
 2910 corresponding "credid"(s) from the "creds" array.
- 2911 5) The rows in Table 25 corresponding to the "creds" array Property dictate the Device States in
 2912 which a DELETE is always rejected. If OCF Device is in a Device State where the Access Mode
 2913 in this row contains "R", then the OCF Device shall reject all DELETES.

2914 NOTE The "/oic/sec/cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined
 2915 in ISO/IEC 30118-1:2018.

2916 "/oic/sec/cred" Resource is defined in Table 24.

2917 **Table 24 – Definition of the "/oic /sec/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

2918 Table 25 defines the Properties of the "/oic/sec/cred" Resource.

Table 25 – 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.

2920 All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end
2921 interaction.

2922 The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid Property.

2923 ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access
2924 modes.

2925 Table 26 defines the Properties of "oic.sec.creds".

Table 26 – Properties of the "oic.sec.creds" 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".

2927 Table 27 defines the Properties of "oic.sec.credusagetype".

2928 **Table 27: 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

2929 Table 28 defines the Properties of "oic.sec.pubdatatype".

2930 **Table 28 – 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

2931 Table 29 defines the Properties of "oic.sec.privdatatype".

2932 **Table 29 – 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

2933 Table 30 defines the Properties of "oic.sec.optdatatype".

2934 **Table 30 – 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

2935 Table 31 defines the Properties of "oic.sec.roletype".

2936 **Table 31 – 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.

2937 **13.3.2 Properties of the Credential Resource**

2938 **13.3.2.1 Credential ID**

2939 Credential ID ("credid") is a local reference to an entry in a "creds" Property array of the
2940 "/oic/sec/cred" Resource. The SRM generates it. The "credid" Property shall be used to
2941 disambiguate array elements of the "creds" Property.

2942 **13.3.2.2 Subject UUID**

2943 The "subjectuid" Property identifies the Device to which an entry in a "creds" Property array of the
2944 "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication
2945 challenge-response or to authenticate an authentication challenge.

2946 A "subjectuid" Property that matches the Server's own "deviceuid" Property, distinguishes the
2947 array entries in the "creds" Property that pertain to this Device.

2948 The "subjectuid" Property shall be used to identify a group to which a group key is used to protect
2949 shared data.

2950 When certificate chain is used during secure connection establishment, the "subjectuuid" Property
2951 shall also be used to verify the identity of the responder. The presented certificate chain shall be
2952 accepted, if there is a matching Credential entry on the Device that satisfies all of the following:

- 2953 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 2954 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity certificate
2955 in the presented chain. If Subject UUID of the entry is set as a wildcard "*", this condition is
2956 automatically satisfied.
- 2957 – Credential Usage of the entry is "oic.sec.cred.trustca".

2958 **13.3.2.3 Role ID**

2959 The roleid Property identifies a role that has been granted to the credential.

2960 **13.3.2.4 Credential Type**

2961 The "credtype" Property is used to interpret several of the other Property values whose contents
2962 can differ depending on credential type. These Properties include "publicdata", "privatedata" and
2963 "optionaldata". The "credtype" Property value of "0" ("no security mode") is reserved for testing and
2964 debugging circumstances. Production deployments shall not allow provisioning of credentials of
2965 type "0". The SRM should introduce checking code that prevents its use in production deployments.

2966 **13.3.2.5 Public Data**

2967 The "publicdata" Property contains information that provides additional context surrounding the
2968 issuance of the credential. For example, it might contain information included in a certificate or
2969 response data from a CMS. It might contain wrapped data.

2970 **13.3.2.6 Private Data**

2971 The "privatedata" Property contains secret information that is used to authenticate a Device, protect
2972 data or verify an authentication challenge-response.

2973 The "privatedata" Property shall not be disclosed outside of the SRM's trusted computing perimeter.
2974 A secure element (SE) or trusted execution environment (TEE) should be used to implement the
2975 SRM's trusted computing perimeter. The privatedata contents may be referenced using a handle;
2976 for example, if used with a secure storage sub-system.

2977 **13.3.2.7 Optional Data**

2978 The "optionaldata" Property contains information that is optionally supplied, but facilitates key
2979 management, scalability or performance optimization.

2980 **13.3.2.8 Period**

2981 The "period" Property identifies the validity period for the credential. If no validity period is specified,
2982 the credential lifetime is undetermined. Constrained devices that do not implement a date-time
2983 capability shall obtain current date-time information from its CMS.

2984 **13.3.2.9 Credential Refresh Method Type Definition [Deprecated]**

2985 This clause is intentionally left blank.

2986 **13.3.2.10 Credential Usage**

2987 Credential Usage indicates to the Device the circumstances in which a credential should be used.
2988 Five values are defined:

- 2989 – "oic.sec.cred.trustca": This certificate is a trust anchor for the purposes of certificate chain
2990 validation, as defined in 10.4. OCF Server SHALL remove any "/oic/sec/cred" entries with an
2991 "oic.sec.cred.trustca" credusage upon transitioning to RFOTM. OCF Servers SHALL use

2992 "/oic/sec/cred" entries that have an "oic.sec.cred.trustca" Value of "credusage" Property only
 2993 as trust anchors for post-onboarding (D)TLS session establishment in RFNOP state; these
 2994 entries are not to be used for onboarding (D)TLS sessions.

2995 – "oic.sec.cred.cert": This "credusage" is used for certificates for which the Device possesses the
 2996 private key and uses it for identity authentication in a secure session, as defined in clause 10.4.

2997 – "oic.sec.cred.rolecert": This "credusage" is used for certificates for which the Device possesses
 2998 the private key and uses to assert one or more roles, as defined in clause 10.4.2.

2999 – "oic.sec.cred.mfgtrustca": This certificate is a trust anchor for the purposes of the Manufacturer
 3000 Certificate Based OTM as defined in clause 7.3.6. OCF Servers SHALL use "/oic/sec/cred"
 3001 entries that have an "oic.sec.cred.mfgtrustca" Value of "credusage" Property only as trust
 3002 anchors for onboarding (D)TLS session establishment; these entries are not to be used for post-
 3003 onboarding (D)TLS sessions.

3004 – "oic.sec.cred.mfgcert": This certificate is used for certificates for which the Device possesses
 3005 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as
 3006 defined in clause 7.3.6.

3007 **13.3.2.11 Resource Owner**

3008 The Resource Owner Property allows credential provisioning to occur soon after Device onboarding
 3009 before access to support services has been established. It identifies the entity authorized to
 3010 manage the "/oic/sec/cred" Resource in response to Device recovery situations.

3011 **13.3.3 Key Formatting**

3012 **13.3.3.1 Symmetric Key Formatting**

3013 Symmetric keys shall have the format described in Table 32 and Table 33.

3014 **Table 32 – 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.

3015

3016 **Table 33 – 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.

3017 **13.3.3.2 Asymmetric Keys**

3018 Asymmetric key formatting is not available in this revision of the document.

3019 **13.3.3.3 Asymmetric Keys with Certificate**

3020 Key formatting is defined by certificate definition.

3021 **13.3.3.4 Passwords**

3022 Password formatting is not available in this revision of the document.

3023 **13.3.4 Credential Refresh Method Details [Deprecated]**

3024 This clause is intentionally left blank.

3025 **13.4 Certificate Revocation List**

3026 **13.4.1 CRL Resource Definition**

3027 Device certificates and private keys are kept in "cred" Resource. CRL is maintained and updated
 3028 with a separate "crl" Resource that is newly defined for maintaining the revocation list.

3029 "/oic/sec/crl" Resource is defined in Table 34.

3030 **Table 34 – Definition of the "oic /sec/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

3031 Table 35 defines the Properties of "oic.r.crl".

3032 **Table 35 – Properties of the "oic/sec/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

3033 **13.5 ACL Resources**

3034 **13.5.1 ACL Resources General**

3035 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be
 3036 expressed using "/oic/sec/acl2". The subject (e.g. "deviceuuid" of the Client) requesting access to
 3037 a Resource shall be authenticated prior to applying the ACL check. Resources that are available
 3038 to multiple Clients can be matched using a wildcard subject. All Resources accessible via the
 3039 unsecured communication endpoint shall be matched using a wildcard subject.

3040 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3041 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 36:

3042 **Table 36 – 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>

3043 The <Deviceld> token means the requestor must possess a credential that uses <UUID> as its
3044 identity in order to match the requestor to the <ACE> policy.

3045 The <RoleID> token means the requestor must possess a role credential with <Character> as its
3046 role in order to match the requestor to the <ACE> policy.

3047 The <Wildcard> token "*" means any requestor is matched to the <ACE> policy, with or without
3048 authentication.

3049 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the <ACE>
3050 policy to Resources.

3051 The <OIC_LINK> token contains values used to query existence of hosted Resources.

3052 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>
3053 and <ResourceRef> matching does not produce the empty set match.

3054 Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE ("D"),
3055 NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the
3056 respective permission is not granted.

3057 The empty set match result defaults to a condition where no access rights are granted.

3058 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.
3059 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively
3060 be granted and rescinded according to the pattern.

3061 13.5.3 ACL Resource

3062 An "acl2" is a list of type "ace2".

3063 In order to provide an interface which allows management of array elements of the "aclist2"
3064 Property associated with a "/oic/sec/acl2" Resource. The RETRIEVE, UPDATE and DELETE
3065 operations on the "/oic/sec/acl2" Resource SHALL behave as follows:

- 3066 1) A RETRIEVE shall return the full Resource representation.
- 3067 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
3068 UPDATE request, as follows:
 - 3069 a) If an UPDATE representation includes the array Property, then:
 - 3070 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace completely
3071 the corresponding ACE in the existing "aces2" array.
 - 3072 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array, and
3073 a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new ACE
3074 by the Server. The "aceid" of a deleted ACE should not be reused, to improve the
3075 determinism of the interface and reduce opportunity for race conditions.
 - 3076 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be
3077 appended to the existing "aces2" array, using the supplied "aceid".

3078 The rows in Table 39 defines the Properties of "oic.sec.acl2".

3079 iv) Table 39 corresponding to the "aclist2" array Property dictate the Device States in which
 3080 an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is in a
 3081 Device State where the Access Mode in this row contains "R", then the OCF Device
 3082 shall reject all UPDATES of the "aclist2" array Property.

3083 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not remove
 3084 the "oic/sec/ace2" Resource.

3085 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the
 3086 corresponding "aceid"(s) from the "aces2" array.

3087 The rows in Table 39 define the Properties of "/oic/sec/acl2" Resource.

3088 5) Table 39 corresponding to the "aclist2" array Property dictate the Device States in which a
 3089 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this
 3090 row contains "R", then the OCF Device shall reject all DELETES.

3091 NOTE The "/oic/sec/acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces
 3092 defined in ISO/IEC 30118-1:2018.

3093 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no
 3094 entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl2" does not
 3095 match the subject and the requested Resource.

3096 Table 37 defines the values of "oic.sec.crudntype".

3097 **Table 37 – Value Definition of the "oic.sec.crudntype" Property**

Value	Access Policy	Description	RemarksNotes
bx0000,0000 (0)	No permissions	No permissions	N/A
bx0000,0001 (1)	C	CREATE	N/A
bx0000,0010 (2)	R	RETRIEVE, OBSERVE, DISCOVER	The "R" permission bit covers both the Read permission and the Observe permission.
bx0000,0100 (4)	U	WRITE, UPDATE	N/A
bx0000,1000 (8)	D	DELETE	N/A
bx0001,0000 (16)	N	NOTIFY	The "N" permission bit is ignored in OCF 1.0, since "R" covers the Observe permission. It is documented for future versions

3098 "/oic/sec/acl2" Resource is defined in Table 24.

3099 **Table 38 – 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

3100 Table 39 defines the Properties of "oic.sec.acl2".

Table 39 – 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.

3102

3103 Table 40 defines the Properties of "oic.sec.ace2".

3104

Table 40 – "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.

3105 Table 41 defines the Properties of "oic.sec.ace2.resource-ref".

3106

Table 41 – "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 19.

3107 Table 42 defines the values of "oic.sec.ace2.resource-ref".

3108

Table 42 – 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

3109 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack
 3110 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's
 3111 request using policies contained in ACL resources.

3112 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified
 3113 Resource references include the device identifier in the href Property that identifies the remote
 3114 Resource Server that hosts the Resource. Partially qualified references mean that the local
 3115 Resource Server hosts the Resource. If a fully qualified resource reference is given, the
 3116 Intermediary enforcing access shall have a secure channel to the Resource Server and the
 3117 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource
 3118 access enforcement point.

3119 Resource Servers should include references to Device and ACL Resources where access
3120 enforcement is to be applied. However, access enforcement logic shall not depend on these
3121 references for access control processing as access to Server Resources will have already been
3122 granted.

3123 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and modify
3124 this Resource. This prevents non-terminating dependency on some other ACL Resource.
3125 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL
3126 Resource.

3127 An ACE2 entry is considered "currently valid" if the validity period of the ACE2 entry includes the
3128 time of the request. The validity period in the ACE2 may be a recurring time period (e.g., daily from
3129 1:00-2:00). Matching the resource(s) specified in a request to the resource Property of the ACE2
3130 is defined in clause 12.2. For example, one way they can match is if the Resource URI in the
3131 request exactly matches one of the resource references in the ACE2 entries.

3132 A request will match an ACE2 if any of the following are true:

3133 1) The ACE2 "subject" Property is of type "oic.sec.didtype" has a UUID value that matches the
3134 "deviceuuid" Property associated with the secure session;

3135 AND the Resource of the request matches one of the resources Property of the ACE2
3136 "oic.sec.ace2.resource-ref";

3137 AND the ACE2 is currently valid.

3138 2) The ACE2 "subject" Property is of type "oic.sec.conntype" and has the wildcard value that
3139 matches the currently established connection type;

3140 AND the resource of the request matches one of the resources Property of the ACE2
3141 "oic.sec.ace2.resource-ref";

3142 AND the ACE2 is currently valid.

3143 3) When Client authentication uses a certificate credential;

3144 AND one of the "roleid" values contained in the role certificate matches the "roleid" Property of
3145 the ACE2 "oic.sec.roletype";

3146 AND the role certificate public key matches the public key of the certificate used to establish
3147 the current secure session;

3148 AND the resource of the request matches one of the array elements of the "resources" Property
3149 of the ACE2 "oic.sec.ace2.resource-ref";

3150 AND the ACE2 is currently valid.

3151 4) When Client authentication uses a certificate credential;

3152 AND the CoAP payload query string of the request specifies a role, which is member of the set
3153 of roles contained in the role certificate;

3154 AND the roleid values contained in the role certificate matches the "roleid" Property of the ACE2
3155 "oic.sec.roletype";

3156 AND the role certificate public key matches the public key of the certificate used to establish
3157 the current secure session;

3158 AND the resource of the request matches one of the resources Property of the ACE2
3159 "oic.sec.ace2.resource-ref";

3160 AND the ACE2 is currently valid.

3161 5) When Client authentication uses a symmetric key credential;

3162 AND one of the "roleid" values associated with the symmetric key credential used in the secure
3163 session, matches the "roleid" Property of the ACE2 "oic.sec.roletype";

3164 AND the resource of the request matches one of the array elements of the "resources" Property
3165 of the ACE2 "oic.sec.ace2.resource-ref";

3166 AND the ACE2 is currently valid.

3167 6) When Client authentication uses a symmetric key credential;

3168 AND the CoAP payload query string of the request specifies a role, which is contained in the
3169 "oic.r.cred.creds.roleid" Property of the current secure session;

3170 AND CoAP payload query string of the request specifies a role that matches the "roleid"
3171 Property of the ACE2 "oic.sec.roletype";

3172 AND the resource of the request matches one of the array elements of the "resources" Property
3173 of the ACE2 "oic.sec.ace2.resource-ref";

3174 AND the ACE2 is currently valid.

3175 A request is granted if ANY of the 'matching' ACE2 entries contain the permission to allow the
3176 request. Otherwise, the request is denied.

3177 There is no way for an ACE2 entry to explicitly deny permission to a resource. Therefore, if one
3178 Device with a given role should have slightly different permissions than another Device with the
3179 same role, they must be provisioned with different roles.

3180 The Server is required to verify that any hosted Resource has authorized access by the Client
3181 requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the
3182 Resource request processing should be applied securely and efficiently. See Annex A for example.

3183 **13.6 Access Manager ACL Resource [Deprecated]**

3184 This clause is intentionally left blank.

3185 **13.7 Signed ACL Resource [Deprecated]**

3186 This clause is intentionally left blank.

3187 **13.8 Provisioning Status Resource**

3188 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning should
3189 be Client-directed or Server-directed. Client-directed provisioning relies on a Client device to
3190 determine what, how and when Server Resources should be instantiated and updated. Server-
3191 directed provisioning relies on the Server to seek provisioning when conditions dictate. Furthermore,
3192 the "/oic/sec/cred" Resource should be provisioned at ownership transfer with credentials
3193 necessary to open a secure connection with appropriate support service.

3194 "/oic/sec/pstat" Resource is defined in Table 43.

3195 **Table 43 – Definition of the "/oic/sec/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

3196 Table 44 defines the Properties of "/oic/sec/pstat".

Table 44 – Properties of the "/oic/sec/pstat" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	N/A	Yes	RW		Device Onboarding State
Is Device Operational	isop	Boolean	T F	Yes	R	RESET	Server shall set to FALSE
					R	RFOTM	Server shall set to FALSE
					R	RFPRO	Server shall set to FALSE
					R	RFNOP	Server shall set to TRUE
					R	SRESET	Server shall set to FALSE
Current Mode	cm	oic.sec.dpmttype	bitmask	Yes	R		Current Mode
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes	RW		Target Mode
Operational Mode	om	oic.sec.pomttype	bitmask	Yes	R	RESET	Server shall set to manufacturer default.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by DOTS.
Supported Mode	sm	oic.sec.pomttype	bitmask	Yes	R	All states	Supported provisioning services operation modes
Device UUID	deviceuuid	String	uuid	Yes	RW	All states	[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOTS should configure the rowneruid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruid does not refer to a valid DOTS the Server shall transition to RESET Device state.

Table 45 – 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

3201 In all Device states:

- 3202 – The Device permits an authenticated and authorised Client to change the Device state of a
- 3203 Device by updating pstat.dos.s to the desired value. The allowed Device state transitions are
- 3204 defined in Figure 23.
- 3205 – Prior to updating "pstat.dos.s", the Client configures the Device to meet entry conditions for the
- 3206 new Device state. The SVR definitions define the entity (Client or Server) expected to perform
- 3207 the specific SVR configuration change to meet the entry conditions. Once the Client has
- 3208 configured the aspects for which the Client is responsible, it can update "pstat.dos.s". The
- 3209 Server then makes any changes for which the Server is responsible, including updating required
- 3210 SVR values, and set pstat.dos.s to the new value.
- 3211 – The "pstat.dos.p" Property is read-only by all Clients.
- 3212 – The Server sets "pstat.dos.p" to TRUE before beginning the process of updating "pstat.dos.s",
- 3213 and sets it back to FALSE when the "pstat.dos.s" change is completed.
- 3214 Any requests to update "pstat.dos.s" while "pstat.dos.p" is TRUE are denied.

3215 When Device state is RESET:

- 3216 – All SVR content is removed and reset to manufacturer default values.
- 3217 – The default manufacturer Device state is RESET.
- 3218 – NCRs are reset to manufacturer default values.
- 3219 – NCRs are inaccessible.
- 3220 – After successfully processing RESET the SRM transitions to RFOTM by setting "pstat.dos.s" to
- 3221 RFOTM.

3222 When Device state is RFOTM:

- 3223 – NCRs are inaccessible.
- 3224 – Before OTM is successful, the deviceuuid Property of "/oic/sec/doxm" Resource shall be set to
- 3225 a temporary non-repeated value as defined in clauses 13.2 and 13.16.
- 3226 – Before OTM is successful, the "pstat.dos.s" is read-only by unauthenticated requestors
- 3227 – After the OTM is successful, the "pstat.dos.s" is read-write by authorized requestors.

- 3228 – The negotiated Device OC is used to create an authenticated session over which the DOTS
3229 directs the Device state to transition to RFPRO.
- 3230 – If an authenticated session cannot be established the ownership transfer session should be
3231 disconnected and SRM sets back the Device state to RESET state.
- 3232 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the
3233 SRM asserts the OTM failed, should be disconnected, and transitions to RESET
3234 ("/pstat.dos.s"=RESET).
- 3235 – The DOTS UPDATES the "devowneruuid" Property in the "/oic/sec/doxm" Resource to a non-
3236 nil UUID value. The DOTS (or other authorized client) can update it multiple times while in
3237 RFOTM. It is not updatable while in other device states except when the Device state returns
3238 to RFOTM through RESET.
- 3239 – The DOTS can have additional provisioning tasks to perform while in RFOTM. When done, the
3240 DOTS UPDATES the "owned" Property in the "/oic/sec/doxm" Resource to "true".
- 3241 When Device state is RFPRO:
- 3242 – The "pstat.dos.s" is read-only by unauthorized requestors and read-write by authorized
3243 requestors.
- 3244 – NCRs are inaccessible, except for Easy Setup Resources, if supported.
- 3245 – The OCF Server may re-create NCRs.
- 3246 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
- 3247 – An authorized Client may perform consistency checks on SVRs to determine which shall be re-
3248 provisioned.
- 3249 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if
3250 the Device has already transitioned from SRESET but consistency checks continue to fail.
- 3251 – The authorized Client sets the "/pstat.dos.s"=RFNOP.
- 3252 When Device state is RFNOP:
- 3253 – The "/pstat.dos.s" Property is read-only by unauthorized requestors and read-write by
3254 authorized requestors.
- 3255 – NCRs, SVRs and core Resources are accessible following normal access processing.
- 3256 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or
3257 RESET.
- 3258 When Device state is SRESET:
- 3259 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to
3260 access or reference them.
- 3261 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
3262 include devowneruuid Property of the "/oic/sec/doxm" Resource,
3263 "creds":{[...,{"subjectuuid":<devowneruuid>},...]} Property of the "/oic/sec/cred" Resource and
3264 "pstat.dos.s" "/oic/sec/pstat" Resource.
- 3265 – The certificates that identify and authorize the Device owner are sufficient to re-create
3266 minimalist "/oic/sec/cred" and "/oic/sec/doxm" Resources enabling Device owner control of
3267 SRESET. If the SRM can't establish these Resources, then it will transition to RESET state.
- 3268 – An authorized Client performs SVR consistency checks. The authorized Client can provision
3269 SVRs as needed to ensure they are available for continued provisioning in RFPRO or for normal
3270 functioning in RFNOP.
- 3271 – The authorized Device owner can avoid entering RESET state and RFOTM by UPDATING
3272 "pstat.dos.s" with RFPRO or RFNOP values.

3273 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to Device
 3274 owner privileges only.

3275 – The SRM asserts a Client-directed operational mode (e.g. "/pstat.om"=CLIENT_DIRECTED).

3276 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.
 3277 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning
 3278 mode without selecting any particular value.

3279 "oic.sec.dpmttype" is defined in Table 46.

3280 **Table 46 – 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

3281 Table 47 and Table 48 define the values of "oic.sec.dpmttype".

3282 **Table 47 – 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)

3283 **Table 48 – 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

3284 The *provisioning operation mode* type is an 8-bit mask enumerating the various provisioning
 3285 operation modes.

3286 "oic.sec.pomtype" is defined in Table 49.

3287 **Table 49 – 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

3288 Table 50 defines the values of "oic.sec.pomtype".

Table 50 – 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

3290 13.9 Certificate Signing Request Resource

3291 The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be
 3292 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC
 3293 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of
 3294 "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr"
 3295 Resource.

3296 "/oic/sec/csr" Resource is defined in Table 51.

3297 **Table 51 – Definition of the "/oic/sec/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

3298 Table 52 defines the Properties of "/oic/sec/csr".

3299 **Table 52 – 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

3300 The Device chooses which public key to use, and may optionally generate a new key pair for this
 3301 purpose.

3302 In the CSR, the Common Name component of the Subject Name shall contain a string of the format
 3303 "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122. The

3304 Common Name, and other components of the Subject Name, may contain other data. If the Device
3305 chooses to include additional information in the Common Name component, it shall delimit it from
3306 the UUID field by white space, a comma, or a semicolon.

3307 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to generate
3308 a new key pair, the Device may begin generation of a key pair as a result of a RETRIEVE of this
3309 resource. If the Device cannot immediately respond to the RETRIEVE request due to time required
3310 to generate a key pair, the Device shall return an "operation pending" error. This indicates to the
3311 Client that the Device is not yet ready to respond, but will be able at a later time. The Client should
3312 retry the request after a short delay.

3313 **13.10 Roles Resource**

3314 The roles Resource maintains roles that have been asserted with role certificates, as described in
3315 clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role
3316 certificate. Servers shall only grant access to the roles information associated with the public key
3317 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.
3318 See 10.4.2 for how role certificates are validated.

3319 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS session
3320 with a Client, if is not already created. The roles Resource shall only expose a secured OCF
3321 Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long as the
3322 (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least until the
3323 certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of clause
3324 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A Server
3325 should regularly inspect the contents of the roles resource and purge contents based on a policy it
3326 determines based on its resource constraints. For example, expired certificates, and certificates
3327 from Clients that have not been heard from for some arbitrary period of time could be candidates
3328 for purging.

3329 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In
3330 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall behave
3331 as follows. Unlisted operations are implementation specific and not reliable.

- 3332 1) A RETRIEVE request shall return all previously asserted roles associated with the currently
3333 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query
3334 parameter is not supported; all previously asserted roles associated with the currently
3335 connected and authenticated Client's identity are returned.
- 3336 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties
3337 included in the array as follows:
 - 3338 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the
3339 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid" value.
 - 3340 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,
3341 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed
3342 response and a duplicate entry shall not be added to the array.
 - 3343 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored
3344 by the Server. The Server shall assign a unique "credid" value for every entry of the "roles"
3345 array.
- 3346 3) A DELETE request without a "credid" query parameter shall remove all entries from the
3347 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated
3348 Client's identity.
- 3349 4) A DELETE request with a "credid" query parameter shall remove only the entries of the
3350 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated
3351 Client's identity and where the corresponding "credid" matches the entry.

3352 NOTE The "/oic/sec/roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces
 3353 defined in ISO/IEC 30118-1:2018.

3354 "/oic/sec/roles" Resource is defined in Table 53.

3355 **Table 53 – Definition of the "/oic/sec/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

3356 Table 54 defines the Properties of "/oic/sec/roles".

3357 **Table 54 – Properties of the "/oic/sec/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

3358 Because "/oic/sec/roles" shares the "oic.sec.cred" schema with "/oic/sec/cred", "subjectuoid" is a required Property.
 3359 However, "subjectuoid" is not used in a role certificate. Therefore, a Device may ignore the "subjectuoid" Property if the
 3360 Property is contained in an UPDATE request to the "/oic/sec/roles" Resource.

3361 **13.11 Account Resource – moved to OCF Cloud Security document**

3362 This clause is intentionally left blank.

3363 **13.12 Account Session Resource – moved to OCF Cloud Security document**

3364 This clause is intentionally left blank.

3365 **13.13 Account Token Refresh Resource – moved to OCF Cloud Security document**

3366 This clause is intentionally left blank.

3367 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3368 The SVRs expose the security-related Properties of the Device.

3369 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to unauthenticated
 3370 (anonymous) Clients could create privacy or security concerns.

3371 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for
 3372 the "/oic/sec/doxm" Resource to anonymous requesters, so that the Device can be discovered and
 3373 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the
 3374 "/oic/sec/doxm" Resource to anonymous requesters, to preserve privacy.

3375 **13.15 SVRs, Discoverability and OCF Endpoints**

3376 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy Parameter
 3377 clause 7.8.2.1.2).

3378 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS) (reference
 3379 ISO/IEC 30118-1:2018, clause 10).

3380 The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM
 3381 (reference ISO/IEC 30118-1:2018, clause 10).

3382 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

3383 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

3384 Unique identifiers are a privacy consideration due to their potential for being used as a tracking
3385 mechanism. These include the following Resources and Properties:

- 3386 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3387 – "/oic/p" Resource containing the "pi" Property.
- 3388 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3389 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous
3390 requestors. This implies any Client Device, including those with malicious intent, are able to reliably
3391 obtain identifiers useful for building a log of activity correlated with a specific Platform and Device.

3392 There are two strategies for privacy protection of Devices:

- 3393 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers
- 3394 2) Limit identifier persistence to make it impractical for tracking use.

3395 Both techniques can be used effectively together to limit exposure to privacy attacks.

3396 1) A Platform / Device manufacturer should specify a default ACL policy that restricts anonymous
3397 requestors from accessing unique identifiers. An OCF Security Domain owner should modify
3398 the ACL policy to grant access to authenticated Devices who, presumably, do not present a
3399 privacy threat.

3400 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the
3401 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and
3402 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated
3403 identifiers shall be:

- 3404 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 3405 b) Generated by a function that is pre-image resistant, second pre-image resistant and collision
3406 resistant

3407 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier used
3408 to begin the onboarding process. However, attackers could obtain the value too and use it for
3409 Device tracking throughout the Device's lifetime.

3410 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the
3411 deviceuuid Property of the "/oic/sec/doxm" Resource to unauthenticated "/oic/res" and
3412 "/oic/sec/doxm" Resource RETRIEVE requests when the devowneruuid Property of "/oic/sec/doxm"
3413 Resource is the nil-UUID. The Server shall expose a new temporary non-repeated deviceuuid
3414 Property of the "/oic/sec/doxm" Resource when the device state transitions to RESET. This ensures
3415 the deviceuuid Property of the "/oic/sec/doxm" cannot be used to track across multiple owners.

3416 The devowneruuid Property of "/oic/sec/doxm" Resource is initialized to the nil-UUID upon entering
3417 RESET; which is retained until being set to a non-nil-UUID value during RFOTM device state. The
3418 device shall supply a temporary, non-repeated deviceuuid Property of "/oic/sec/doxm" Resource to
3419 RETRIEVE requests on "/oic/sec/doxm" and "/oic/res" Resources while devowneruuid Property of
3420 "/oic/sec/doxm" Resource is the nil-UUID. During the OTM process the DOTS UPDATING
3421 devowneruuid Property of the "/oic/sec/doxm" Resource to a non-nil UUID value is the trigger for
3422 the Device to expose its persistent or semi-persistent device identifier. Therefore, the Device shall
3423 supply deviceuuid Property of "/oic/sec/doxm" Resource in response to RETRIEVE requests while
3424 the devowneruuid Property of the "/oic/sec/doxm" Resource is a non-nil-UUID value.

3425 The DOTS or AMS can also provision an ACL policy that restricts access to the "/oic/sec/doxm"
 3426 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent
 3427 device identifier via the deviceuuid Property value of the "/oic/sec/doxm" Resource.

3428 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent
 3429 or semi-persistent identifier using the "/oic/sec/cred" Resource to first establish an authenticated
 3430 connection. This is achieved by first provisioning a "/oic/sec/cred" Resource entry that contains the
 3431 Server's deviceuuid Property value of the "/oic/sec/doxm" Resource.

3432 The "di" Property in the "/oic/d" Resource shall mirror that of the deviceuuid Property of the
 3433 "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the
 3434 "/oic/d" resource such that only authenticated Clients are able to obtain the "di" Property of "/oic/d"
 3435 Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

3436 Servers should expose a temporary, non-repeated, piid Property of "/oic/p" Resource Value upon
 3437 entering RESET Device state. Servers shall expose a persistent value via the "piid" Property of
 3438 "/oic/p" Property when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. An ACL
 3439 policy on the "/oic/d" Resource should protect the "piid" Property of "/oic/p" Resource from being
 3440 disclosed to unauthenticated requestors.

3441 Servers shall expose a temporary, non-repeated, "pi" Property value upon entering RESET Device
 3442 state. Servers shall expose a persistent or semi-persistent platform identifier value via the "pi"
 3443 Property of the "/oic/p" Resource when onboarding sets "devowneruuid" Property to a non-nil-UUID
 3444 value. An ACL policy on the "/oic/p" Resource should protect the "pi" Property from being disclosed
 3445 to unauthenticated requestors.

3446 Table 55 depicts Core Resource Properties Access Modes given various Device States.

3447 **Table 55 – 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 should construct a temporary random UUID (The temporary value shall not overwrite the persistent pi internally). Server switches 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.

3448 Four identifiers are thought to be privacy sensitive:

- 3449 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3450 – "/oic/p" Resource containing the "pi" Property.
- 3451 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3452 There are three strategies for privacy protection of Devices:

- 3453 1) Apply access control to restrict read access to Resources containing unique identifiers. This
3454 ensures privacy sensitive identifiers do not leave the Device.
- 3455 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy sensitive
3456 identifiers are less effective for tracking and correlation.
- 3457 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can
3458 do so.

3459 These techniques can be used to limit exposure to privacy attacks. For example:

- 3460 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent
3461 identifiers can be created.
- 3462 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to
3463 facilitate onboarding.
- 3464 – Persistent and semi-persistent identifiers can be encrypted before sending them to another
3465 Device.

3466 A temporary, non-repeated identifier shall be:

- 3467 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 3468 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision
3469 resistant

3470 NOTE This requirement is met through a vendor attestation certification mechanism.

3471 **13.16.2 Privacy Protecting the Device Identifiers**

3472 The "di" Property Value of the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of
3473 the "/oic/sec/doxm" Resource. The Device should use a new, temporary non-repeated identifier in
3474 place of the "deviceuuid" Property Value of "/oic/sec/doxm" Resource upon entering the RESET
3475 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID
3476 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value of
3477 the "/oic/sec/doxm" Resource after the DOTS sets the "devowneruuid" Property to a non-nil-UUID
3478 value. The temporary identifier should not change more frequently than once per Device state
3479 transition to RESET.

3480 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3481 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"
3482 Property values:
 - 3483 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di") Property
3484 value only if responding to an authenticated requestor and the "deviceuuid" (or "di") value
3485 is confidentiality protected .
 - 3486 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if
3487 responding to an unauthenticated requestor.
- 3488 – The AMS can provision an ACL policy on the "/oic/sec/doxm" and "/oic/d" resources to further
3489 protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

3490 See 13.2 for deviceuuid Property lifecycle requirements.

3491 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary
3492 discovery requests. This is achieved by provisioning a "/oic/sec/cred" Resource entry that contains the Server's
3493 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

3494 **13.16.3 Privacy Protecting the Protocol Independent Device Identifier**

3495 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property
3496 Value of "/oic/d" Resource upon entering the RESET Device state. If a temporary, non-repeated
3497 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID

3498 value. The Device should use its persistent "piid" Property value after the DOTS sets the
 3499 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change more
 3500 frequently than once per Device state transition to RESET.

3501 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3502 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
 - 3503 – The Device should include its persistent "piid" Property value only if responding to an
 - 3504 authenticated requestor and the "piid" value is confidentiality protected.
 - 3505 – The Device should include a temporary non-repeated "piid" Property value if responding to
 - 3506 an unauthenticated requestor.
- 3507 – The AMS can provision an ACL policy on the "/oic/d" Resource to further protect the piid
- 3508 Property of "/oic/p" Resource from being disclosed unnecessarily.

3509 **13.16.4 Privacy Protecting the Platform Identifier**

3510 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property Value
 3511 of the "/oic/p" Resource upon entering the RESET Device state. This value should be exposed
 3512 while the "devowneruuid" Property has a nil UUID value. The Device should use its persistent (or
 3513 semi-persistent) "pi" Property value after the DOTS sets the "devowneruuid" Property to a non-nil-
 3514 UUID value. The temporary identifier should not change more frequently than once per Device state
 3515 transition to RESET.

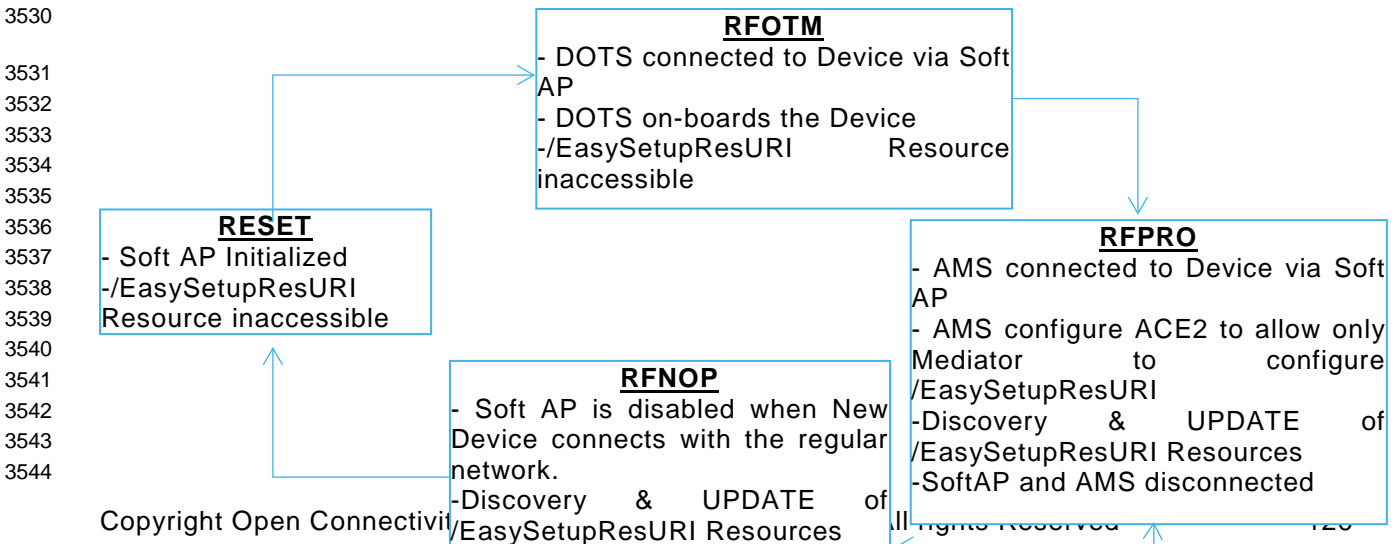
3516 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 3517 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
 - 3518 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
 - 3519 responding to an authenticated requestor and the "pi" value is confidentiality protected.
 - 3520 – The Device should include a temporary non-repeated "pi" Property value if responding to
 - 3521 an unauthenticated requestor.
- 3522 – The AMS can provision an ACL policy on the "/oic/p" Resource to protect the pi Property from
- 3523 being disclosed unnecessarily.

3524 **13.17 Easy Setup Resource Device State**

3525 This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined
 3526 in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of
 3527 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup
 3528 Devices.

3529 Figure 30 shows an example of Soft AP and Easy Setup Resource in different Device states.



3545
3546

3547 **Figure 30 – Example of Soft AP and Easy Setup Resource in different Device states**

3548 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's
3549 state.

3550 While it is reasonable for a user to expect that power cycling a new Device will turn on the Soft AP
3551 for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it is a
3552 security risk to make this the default behaviour of a device that remains unenrolled beyond a
3553 reasonable period after first boot.

3554 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

3555 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes
3556 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for Easy
3557 Setup.

3558 – If a new Device tried and failed to complete Easy Setup Enrolment immediately following the
3559 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically for
3560 another 30 minutes upon being power cycled, provided that the power cycle occurs within 3
3561 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup Soft
3562 AP directly without a factory reset, it is not necessary to turn it back on if it was on immediately
3563 prior to power cycle, because the user obviously knows how to initiate the process manually.

3564 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft
3565 AP should not turn back on for Easy Setup until another factory reset occurs, or the user initiates
3566 the Easy Setup Soft AP directly.

3567 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new
3568 Device to connect to the Enroller.

3569 – The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the
3570 Enroller.

3571 – Once a new Device has successfully connected to the Enroller, it shall not turn the Easy Setup
3572 Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or the user
3573 initiates the Easy Setup Soft AP directly.

3574 – Just Works OTM shall not be enabled on Devices which support Easy Setup.

3575 – The Soft AP shall be secured (e.g. shall not expose an open AP).

3576 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase
3577 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on
3578 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.

3579 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the
3580 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by an
3581 attacker with knowledge of the Device type, model, manufacturer, or any other information
3582 discoverable through Device's exposed interfaces.

3583 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the
3584 "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the
3585 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller, for
3586 use in connecting the Enrollee to the Enroller.

3587 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the
3588 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.

3589 The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET
3590 state. After ownership transfer process is completed with the DOTS, and the Device enters in
3591 RFPRO Device state, the "/example/EasySetupResURI" may be Discoverable.

3592 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership
3593 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used by
3594 AMS for "/oic/sec/acl2" Resource provisioning in RFPRO state. The CoAPS session authentication
3595 and encryption is already defined in the Security spec.

3596 In RFPRO state, AMS is expected to configure ACL2 Resource on the Device with ACE2 for
3597 following Resources to be only configurable by the Mediator with permission to UPDATE or
3598 RETRIEVE access:

- 3599 – "/example/EasySetupResURI"
- 3600 – "/example/WifiConfResURI"
- 3601 – "/example/DevConfResURI"

3602 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
3603 {  
3604     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
3605     "resources": [  
3606         { "href": "/example/EasySetupResURI" },  
3607         { "href": "/example/WiFiConfResURI" },  
3608         { "href": "/example/DevConfResURI" },  
3609     ],  
3610     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
3611 }
```

3612 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to
3613 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

3614 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE
3615 these Resources. The Mediator may UPDATE /EasySetupResURI resources in RFNOP Device
3616 state.

3617 A Mediator shall be hosted on an OCF Device.

3618 **14 Security Hardening Guidelines/ Execution Environment Security**

3619 **14.1 Preamble**

3620 This is an informative clause. Many TGs in OCF have security considerations for their protocols
3621 and environments. These security considerations are addressed through security mechanisms
3622 specified in the security documents for OCF. However, effectiveness of these mechanisms depends
3623 on security robustness of the underlying hardware and software Platform. This clause defines the
3624 components required for execution environment security.

3625 **14.2 Execution Environment Elements**

3626 **14.2.1 Execution Environment Elements General**

3627 Execution environment within a computing Device has many components. To perform security
3628 functions in a robustness manner, each of these components has to be secured as a separate
3629 dimension. For instance, an execution environment performing AES cannot be considered secure
3630 if the input path entering keys into the execution engine is not secured, even though the partitions
3631 of the CPU, performing the AES encryption, operate in isolation from other processes. Different
3632 dimensions referred to as elements of the execution environment are listed below. To qualify as a
3633 secure execution environment (SEE), the corresponding SEE element must qualify as secure.

- 3634 – (Secure) Storage
- 3635 – (Secure) Execution engine
- 3636 – (Trusted) Input/output paths
- 3637 – (Secure) Time Source/clock
- 3638 – (Random) number generator
- 3639 – (Approved) cryptographic algorithms
- 3640 – Hardware Tamper (protection)

3641 NOTE Software security practices (such as those covered by OWASP) are outside scope of this document, as
3642 development of secure code is a practice to be followed by the open source development community. This document will
3643 however address the underlying Platform assistance required for executing software. Examples are secure boot and
3644 secure software upgrade.

3645 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,
3646 14.2.7.

3647 **14.2.2 Secure Storage**

3648 **14.2.2.1 Secure Storage General**

3649 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive
3650 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,
3651 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive
3652 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both its
3653 integrity and confidentiality be maintained.

3654 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive
3655 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either
3656 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication and
3657 encryption, it must maintain its integrity against intentional or accidental alteration.

3658 A partial list of Sensitive Data is outlined in Table 56:

Table 56 – Examples of Sensitive Data

Data	Integrity protection	Confidentiality protection
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
Access Token	Yes	Yes

3660 Exact method of protection for secure storage is implementation specific, but typically combinations
3661 of hardware and software methods are used.

3662 **14.2.2.2 Hardware Secure Storage**

3663 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric
3664 and asymmetric private keys, access credentials, and personal private data. Hardware secure
3665 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes
3666 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

3667 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides
3668 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or electronic
3669 attacks. It is not necessary to prevent the attacks themselves, but an attempted attack should not
3670 result in an unauthorized entity successfully retrieving Sensitive Data.

3671 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data
3672 from attacks that include but are not limited to:

- 3673 1) Physical decapping of chip packages to optically read NVRAM contents
- 3674 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 3675 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit patterns
3676 of Critical Sensitive Data
- 3677 4) Use of malicious software or firmware to read memory contents at rest or in transit within a
3678 microcontroller
- 3679 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

3680 **14.2.2.3 Software Storage**

3681 It is generally NOT recommended to rely solely on software and unsecured memory to store
3682 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and encryption
3683 keys should be housed in hardware secure storage whenever possible.

3684 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable
3685 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

3686 **14.2.2.4 Additional Security Guidelines and Best Practices**

3687 Some general practices that can help ensure that Sensitive Data is not compromised by various
3688 forms of security attacks:

- 3689 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG
3690 used for authentication challenges can substantially degrade security strength. For this reason,
3691 it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source be used
3692 for all authentication challenges.
- 3693 2) Secure download and boot – To prevent the loading and execution of malicious software, where
3694 it is practical, it is recommended that Secure Download and Secure Boot methods that
3695 authenticate a binary's source as well as its contents be used.
- 3696 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered
3697 insecure and shall not be used for any security-related function:
 - 3698 a) SHA-1
 - 3699 b) MD5
 - 3700 c) RC4
 - 3701 d) RSA 1024
- 3702 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is
3703 stored in Secure Storage, any use of that data that requires its transmission out of that Secure
3704 Storage should be encrypted to prevent eavesdropping by malicious software within an
3705 MCU/MPU.
- 3706 5) It is recommended to avoid using wildcard in Subject Id ("*"), when setting up "/oic/sec/cred"
3707 Resource entries, since this opens up an identity spoofing opportunity.
- 3708 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device
3709 meets security requirements for its intended uses. As an example, IoTivity is a reference
3710 implementation intended to be used as a basis for a product, but IoTivity has not undergone
3711 3rd party security review, penetration testing, etc. Any Device based on IoTivity should undergo
3712 appropriate penetration testing and security review prior to sale or deployment.
- 3713 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to
3714 consumers. Changes should be made to a public and accessible website. Expectations should
3715 be clear as to what will be supported and for how long the Device vendor expects to support
3716 security updates to the software, operating system, drivers, networking, firmware and hardware
3717 of the device.
- 3718 8) Device vendor has not implemented test or debug interfaces on the Device which are operable
3719 or which can be enabled which might present an attack vector on the Device which circumvents
3720 the interface-level security or access policies of the Device.
- 3721 9) Device vendor understands that if an application running on the Device has access to
3722 cryptographic elements such as the private keys or Ownership Credential, then those elements
3723 have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or a Device
3724 with access to the Internet beyond the local network, the execution of critical functions should
3725 take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 3726 10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management or
3727 access, or other security-critical function, should be sufficiently unique (do not duplicate
3728 passphrases. The creation of these passphrases or PINS should not be algorithmically
3729 deterministic nor should they use insufficient entropy in their creation.
- 3730 11) Ensure that there are no remaining "VENDOR_TODO" items in the source code.

3731 12) If the implementation of this document uses the "Just Works" onboarding method, understand
3732 that there is a man-in-the-middle vulnerability during the onboarding process where a malicious
3733 party could intercept messages between the device being onboarded and the OBT and could
3734 persist, acting as an intermediary with access to message traffic, during the lifetime of that
3735 onboarded device. The recommended best practice would be to use an alternate ownership
3736 transfer method (OTM) instead of "Just Works".

3737 13) It is recommended that at least one static and dynamic analysis tool¹ be applied to any
3738 proposed major production release of the software before its release, and any vulnerabilities
3739 resolved.

3740 **14.2.3 Secure execution engine**

3741 Execution engine is the part of computing Platform that processes security functions, such as
3742 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine requires
3743 the following

3744 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This
3745 includes isolation of CPU caches, and all of execution elements that needed to be considered
3746 as part of trusted (crypto) boundary.

3747 – Isolation of data paths into and out of execution engine. For instance, both unencrypted but
3748 sensitive data prior to encryption or after decryption, or cryptographic keys used for
3749 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

3750 **14.2.4 Trusted input/output paths**

3751 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be protected.
3752 This includes paths into and out secure execution engine and secure memory.

3753 Path protection can be both hardware based (e.g. use of a privileged bus) or software based (using
3754 encryption over an untrusted bus).

3755 **14.2.5 Secure clock**

3756 Many security functions depend on time-sensitive credentials. Examples are time stamped
3757 Kerberos tickets, OAuth tokens, X.509 certificates, OSCP response, software upgrades, etc. Lack
3758 of secure source of clock can mean an attacker can modify the system clock and fool the validation
3759 mechanism. Thus an SEE needs to provide a secure source of time that is protected from tampering.
3760 Trustworthiness from security robustness standpoint is not the same as accuracy. Protocols such
3761 as NTP can provide rather accurate time sources from the network, but are not immune to attacks.
3762 A secure time source on the other hand can be off by seconds or minutes depending on the time-
3763 sensitivity of the corresponding security mechanism. Secure time source can be external as long
3764 as it is signed by a trusted source and the signature validation in the local Device is a trusted
3765 process (e.g. backed by secure boot).

3766 **14.2.6 Approved algorithms**

3767 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and
3768 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by
3769 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only
3770 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic
3771 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are
3772 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms (even
3773 if they deemed stronger by some parties) must be considered non-approved.

3774 The set of algorithms to be considered for approval are algorithms for

3775 – Hash functions

¹ A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

- 3776 – Signature algorithms
 - 3777 – Encryption algorithms
 - 3778 – Key exchange algorithms
 - 3779 – Pseudo Random functions (PRF) used for key derivation
- 3780 This list will be included in this or a separate security robustness rules document and must be
3781 followed for all security specifications within OCF.

3782 **14.2.7 Hardware tamper protection**

3783 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not
3784 requirements) regarding tamper protection for cryptographic module

- 3785 – Production-grade (lowest level): this means components that include conformal sealing coating
3786 applied over the module's circuitry to protect against environmental or other physical damage.
3787 This does not however require zeroization of secret material during physical maintenance. This
3788 definition is borrowed from FIPS 140-2 security level 1.
- 3789 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,
3790 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from FIPS
3791 140-2 security level 2.
- 3792 – Tamper resistance (highest level), this means there is a response to physical tempering that
3793 typically includes zeroization of sensitive material on the module. This definition is borrowed
3794 from FIPS 140-2 security level 3.

3795 It is difficult of specify quantitative certification test cases for accreditation of these levels. Content
3796 protection regimes usually talk about different tools (widely available, specialized and professional
3797 tools) used to circumvent the hardware protections put in place by manufacturing. If needed, OCF
3798 can follow that model, if and when OCF engage in distributing sensitive key material (e.g. PKI) to
3799 its members.

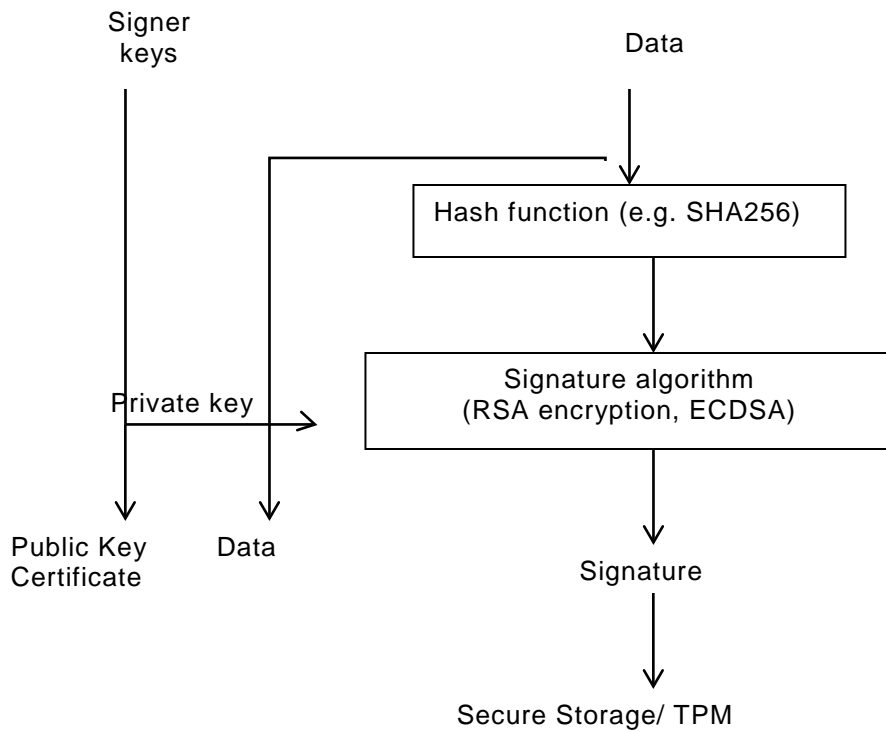
3800 **14.3 Secure Boot**

3801 **14.3.1 Concept of software module authentication**

3802 In order to ensure that all components of a Device are operating properly and have not been
3803 tampered with, it is best to ensure that the Device is booted properly. There may be multiple stages
3804 of boot. The end result is an application running on top an operating system that takes advantage
3805 of memory, CPU and peripherals through drivers.

3806 The general concept is that each software module is invoked only after cryptographic integrity
3807 verification is complete. The integrity verification relies on the software module having been hashed
3808 (e.g. SHA_1, SHA_256) and then signed with a cryptographic signature algorithm with (e.g. RSA),
3809 with a key that only a signing authority has access to.

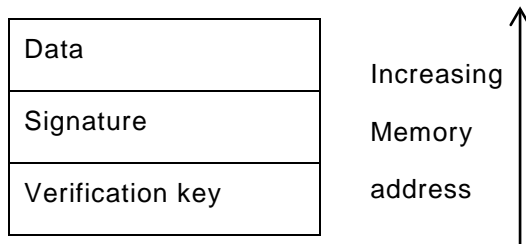
3810 Figure 31 depicts software module authentication.



3811 **Figure 31 – Software Module Authentication**

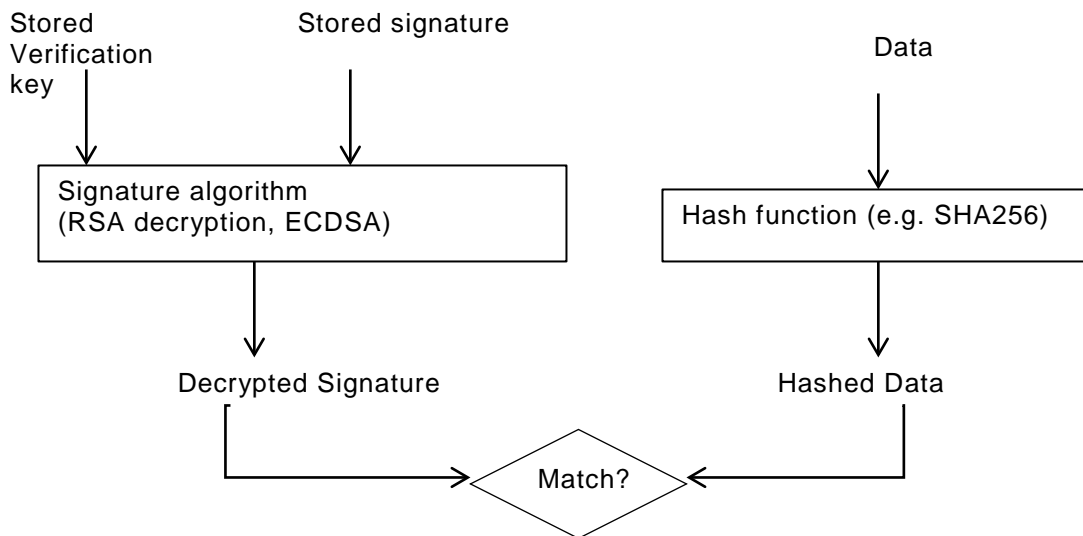
3812 After the data is signed with the signer’s signing key (a private key), the verification key (the public
 3813 key corresponding to the private signing key) is provided for later verification. For lower level
 3814 software modules, such as bootloaders, the signatures and verification keys are inserted inside
 3815 tamper proof memory, such as one-time programmable memory or TPM. For higher level software
 3816 modules, such as application software, the signing is typically performed according to the PKCS#7
 3817 format IETF RFC 2315, where the signedData format includes both indications for signature
 3818 algorithm, hash algorithm as well as the signature verification key (or certificate). Secure boot does
 3819 not require use of PKCS#7 format.

3820 Figure 32 depicts verification software module.



3821 **Figure 32 – Verification Software Module**

3822 As shown in Figure 33. the verification module first decrypts the signature with the verification key
 3823 (public key of the signer). The verification module also calculates a hash of the data and then
 3824 compares the decrypted signature (the original) with the hash of data (actual) and if the two values
 3825 match, the software module is authentic.



3826 **Figure 33 – Software Module Authenticity**

3826

3827 **14.3.2 Secure Boot process**

3828 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/
 3829 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to
 3830 find out where the boot code is and then run the boot code (second-stage boot loader). The second
 3831 stage bootloader is typically the process that loads the operating system (Kernel) and transfers the
 3832 execution to the where the Kernel code is. Once the Kernel starts, it may load external Kernel
 3833 modules and drivers.

3834 When performing a secure boot, it is required that the integrity of each boot loader is verified before
 3835 executing the boot loader stage. As mentioned, while the signature and verification key for the
 3836 lowest level bootloader is typically stored in tamper-proof memory, the signature and verification
 3837 key for higher levels should be embedded (but attached in an easily accessible manner) in the data
 3838 structures software.

3839 **14.3.3 Robustness Requirements**

3840 **14.3.3.1 Robustness General**

3841 To qualify as high robustness secure boot process, the signature and hash algorithms shall be one
 3842 of the approved algorithms, the signature values and the keys used for verification shall be stored
 3843 in secure storage and the algorithms shall run inside a secure execution environment and the keys
 3844 shall be provided the SEE over trusted path.

3845 **14.3.3.2 Next steps**

3846 Develop a list of approved algorithms and data formats

3847 **14.4 Attestation**

3848 **14.5 Software Update**

3849 **14.5.1 Overview:**

3850 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;
 3851 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and
 3852 end-of-life stages for the Device remain outstanding. It is possible for the Device to require update

3853 during any of these stages, although the most likely times are during onboarding, regular operation
 3854 and maintenance. The aspects of the software include, but are not limited to, firmware, operating
 3855 system, networking stack, application code, drivers, etc.

3856 **14.5.2 Recognition of Current Differences**

3857 Different manufacturers approach software update utilizing a collection of tools and strategies:
 3858 over-the-air or wired USB connections, full or partial replacement of existing software, signed and
 3859 verified code, attestation of the delivery package, verification of the source of the code, package
 3860 structures for the software, etc.

3861 It is recommended that manufacturers review their processes and technologies for compliance with
 3862 industry best-practices that a thorough security review of these takes place and that periodic review
 3863 continue after the initial architecture has been established.

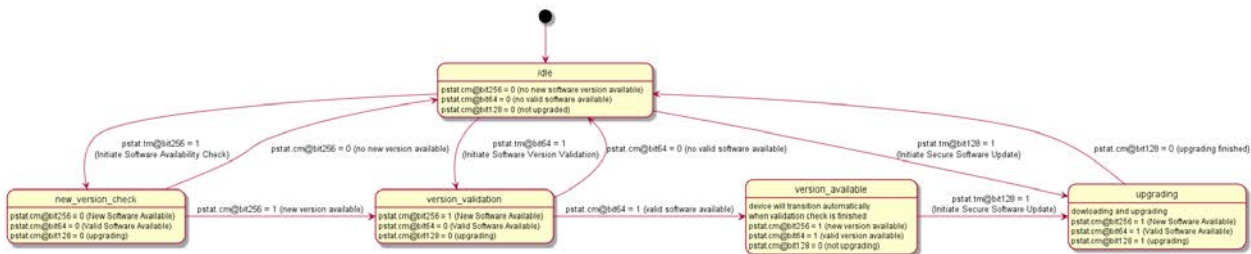
3864 This document applies to software updates as recommended to be implemented by OCF Devices;
 3865 it does not have any bearing on the above-mentioned alternative proprietary software update
 3866 mechanisms. The described steps are being triggered by an OCF Client, the actual implementation
 3867 of the steps and how the software package is downloaded and upgraded is vendor specific.

3868 The triggers that can be invoked from OCF clients can perform:

- 3869 1) Check if new software is available
- 3870 2) Download and verify the integrity of the software package
- 3871 3) Install the verified software package

3872 The triggers are not sequenced, each trigger can be invoked individually.

3873 The state of the transitions of software update is in Figure 34.



3874
 3875 **Figure 34 – State transitioning diagram for software download**

3876
 3877 **Table 57 – 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

3878
 3879 **14.5.2.1 Checking availability of new software**
 3880 Setting the Initiate Software Availability Check bit in the "/oic/sec/pstat.tm" Property (see Table 44
 3881 of clause 13.8) indicates a request to initiate the process to check if new software is available, e.g.
 3882 the process whereby the Device checks if a newer software version is available on the external

3883 endpoint. Once the Device has determined if an newer software version is available, it sets the
3884 Initiate Software Availability Check bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE), indicating
3885 that new software is available or to 0 (FALSE) if no newer software version is available, See also
3886 Table 57 where the bits in property TM indicates that the action is initiated and the CM bits are
3887 indicating the result of the action. The Device receiving this trigger is not downloading and not
3888 validating the software to determine if new software is available. The version check is determined
3889 by the current software version and the software version on the external endpoint. The
3890 determination if a software package is newer is vendor defined.

3891 **14.5.3 Software Version Validation**

3892 Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see Table 44
3893 defines the Properties of "/oic/sec/pstat").

3894 Table 44 of 13.8) indicates a request to initiate the software version validation process, the process
3895 whereby the Device validates the software (including firmware, operating system, Device drivers,
3896 networking stack, etc.) against a trusted source to see if, at the conclusion of the check, the
3897 software update process will need to be triggered (see clause 14.5.4). When the Initiate Software
3898 Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the
3899 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a
3900 software version check. Once the Device has determined if a valid software is available, it sets the
3901 Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if an update
3902 is available or 0 (FALSE) if no update is available. To signal completion of the Software Version
3903 Validation process, the Device sets the Initiate Software Version Validation bit in the
3904 "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation bit of
3905 "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation process. The
3906 Software Version Validation process can download the software from the external endpoint to verify
3907 the integrity of the software package.

3908 **14.5.4 Software Update**

3909 Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 44 of
3910 clause 13.8) indicates a request to initiate the software update process. When the Initiate Secure
3911 Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the
3912 Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a
3913 software update process. Once the Device has completed the software update process, it sets the
3914 Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if/when the
3915 software was successfully updated or 0 (FALSE) if no update was performed. To signal completion
3916 of the Secure Software Update process, the Device sets the Initiate Secure Software Update bit in
3917 the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure Software Update bit of
3918 "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the update process.

3919 **14.5.4.1 State of Device after software update**

3920 The state of all resources implemented in the Device should be the same as after boot, meaning
3921 that the software update is not resetting user data and retaining a correct state.

3922 User data of a Device is defined as:

- 3923 – Retain the SVR states, e.g. the on boarded state, registered clients.
- 3924 – Retain all created resources
- 3925 – Retain all stored data of a resource
- 3926 – For example the preferences stored for the brewing resource ("oic.r.brewing").

3927 **14.5.5 Recommended Usage**

3928 The Initiate Secure Software Update bit of "/oic/sec/pstat.tm" should only be set by a Client after
3929 the Initiate Software Version Validation check is complete.

3930 The process of updating Device software may involve state changes that affect the Device
3931 Operational State ("/oic/sec/pstat.dos"). Devices with an interest in the Device(s) being updated
3932 should monitor "/oic/sec/pstat.dos" and be prepared for pending software update(s) to affect Device
3933 state(s) prior to completion of the update.

3934 The Device itself may indicate that it is autonomously initiating a software version check/update or
3935 that a check/update is complete by setting the "pstat.tm" and "pstat.cm" Initiate Software Version
3936 Validation and Secure Software Update bits when starting or completing the version check or
3937 update process. As is the case with a Client-initiated update, Clients can be notified that an
3938 autonomous version check or software update is pending and/or complete by observing pstat
3939 resource changes.

3940 The "oic.r.softwareupdate" Resource Type specifies additional features to control the software
3941 update process see core specification.

3942 **14.6 Non-OCF Endpoint interoperability**

3943 **14.7 Security Levels**

3944 Security Levels are a way to differentiate Devices based on their security criteria. This need for
3945 differentiation is based on the requirements from different verticals such as industrial and health
3946 care and may extend into smart home. This differentiation is distinct from Device classification
3947 (e.g. IETF RFC 7228)

3948 These categories of security differentiation may include, but is not limited to:

- 3949 1) Security Hardening
- 3950 2) Identity Attestation
- 3951 3) Certificate/Trust
- 3952 4) Onboarding Technique
- 3953 5) Regulatory Compliance
 - 3954 a) Data at rest
 - 3955 b) Data in transit
- 3956 6) Cipher Suites – Crypto Algorithms & Curves
- 3957 7) Key Length
- 3958 8) Secure Boot/Update

3959 In the future security levels can be used to define interoperability.

3960 The following applies to the OCF Security Specification 1.1:

3961 The current document does not define any other level beyond Security Level 0. All Devices will be
3962 designated as Level 0. Future versions may define additional levels.

3963 Additional comments:

- 3964 – The definition of a given security level will remain unchanged between versions of the document.
- 3965 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 3966 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of the
3967 higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and a later
3968 document version defines a security level 1, the Device could be evaluated and classified as
3969 level 1 if it meets level 1 requirements).
- 3970 – The security levels may need to be visible to the end user.

3971 **14.8 Security Profiles**

3972 **14.8.1 Security Profiles General**

3973 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This need
3974 for differentiation is based on the requirements from different verticals such as industrial and health
3975 care and may extend into smart home. This differentiation is distinct from device classification (e.g.
3976 IETF RFC 7228)

3977 These categories of security differentiation may include, but is not limited to:

- 3978 1) Security Hardening and assurances criteria
3979 2) Identity Attestation
3980 3) Certificate/Trust
3981 4) Onboarding Technique
3982 5) Regulatory Compliance
3983 a) Data at rest
3984 b) Data in transit
3985 6) Cipher Suites – Crypto Algorithms & Curves
3986 7) Key Length
3987 8) Secure Boot/Update

3988 Each Security Profile definition must specify the version or versions of the OCF Security
3989 Specification(s) that form a baseline set of normative requirements. The profile definition may
3990 include security requirements that supersede baseline requirements (not to relax security
3991 requirements).

3992 Security Profiles have the following properties:

- 3993 – A given profile definition is not specific to the version of the document that defines it. For
3994 example, the profile may remain constant for subsequent OCF Security Specification versions.
3995 – A specific OCF Device and platform combination may be used to satisfy the security profile.
3996 – Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles
3997 simultaneously.
3998 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found to
3999 satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the document,
4000 and a later document version defines a security profile Black, the device could be evaluated
4001 and classified as profile Black if it meets profile Black requirements).
4002 – A machine-readable representation of compliance results specifically describing profiles
4003 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or
4004 manifest may contain security profiles attributes).

4005 **14.8.2 Identification of Security Profiles (Normative)**

4006 **14.8.2.1 Security Profiles in Prior Documents**

4007 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles
4008 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in
4009 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use
4010 the OCF Security Specification version to characterize expected security behaviour.

4011 **14.8.2.2 Security Profile Resource Definition**

4012 The "/oic/sec/sp" Resource is used by the OCF Device to show which OCF Security Profiles the
4013 OCF Device is capable of supporting and which are authorized for use by the OCF Security Domain

4014 owner. Properties of the Resource identify which OCF Security Profile is currently operational. The
 4015 ocfSecurityProfileOID value type shall represent OID values and may reference an entry in the form
 4016 of strings (UTF-8).

4017 "/oic/sec/sp" Resource is defined in Table 58.

4018 **Table 58 – 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

4019 Table 59 defines the Properties of "/oic/sec/sp" Resource.

4020 **Table 59 – 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")

4021 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or
 4022 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4023 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
 4024 private(4) enterprise(1) OCF(51414) }

4025
 4026 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4027
 4028 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4029
 4030 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }

4031 --The Security Profile is not specified

4032 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }

4033 --This specifies the OCF Baseline Security Profile(s)

4034 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }

4035 --This specifies the OCF Black Security Profile(s)

4036 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }

4037 --This specified the OCF Blue Security Profile(s)

4038 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }

4039 --This specifies the OCF Purple Security Profile(s)

4040
 4041 --versioned Security Profiles

4042 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)

4043 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4044 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}

4045 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4046 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}

4047 --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"

4048 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}

4049 --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"

4050 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}

4051 --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"

4052
 4053 ocfSecurityProfileOID ::= UTF8String

4054

4055 **14.8.3 Security Profiles**

4056 **14.8.3.1 Security Profiles General**

4057 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to
4058 the Security Profile clauses for additional details).

4059 The OCF Conformance criteria may require vendor attestation that establishes the expected
4060 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific
4061 requirements).

4062 **14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)**

4063 The Security Profile "sp-unspecified-v0" is reserved for future use.

4064 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4065 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where
4066 the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the
4067 "supportedprofiles" Property of the "/oic/sec/sp" Resource.

4068 It indicates the OCF Device satisfies the normative security requirements for this document.

4069 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-
4070 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other
4071 profiles.

4072 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to
4073 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

4074 **14.8.3.4 Security Profile Black (sp-black-v0)**

4075 **14.8.3.4.1 Black Profile General**

4076 The need for Security Profile Black v0 is to support devices and manufacturers who wish to certify
4077 their devices meeting this specific set of security criteria. A Device may satisfy the Black
4078 requirements as well as requirements of other profiles, the Black Security Profile is not necessarily
4079 mutually exclusive with other Security Profiles unless those requirements conflict with the explicit
4080 requirements of the Black Security Profile.

4081 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4082 Security Profile Black devices could include any device a manufacturer wishes to certify at this
4083 profile, but healthcare devices and industrial devices with additional security requirements are the
4084 initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or devices
4085 with exceptional profiles of trust bestowed upon them, may wish to certify at this profile; these types
4086 of devices may include, but are not limited to the following:

4087 – Bridges (Mapping devices between ecosystems handling virtual devices from different
4088 ecosystems)

4089 – Resource Directories (Devices trusted to manage OCF Security Domain resources)

4090 – Remote Access (Devices which have external access but can also act within the OCF Security
4091 Domain)

4092 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)

4093 – Industrial Devices (Devices with advanced management, security and attestation requirements)

4094 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4095 Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security
4096 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4097 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4098 – Support for AES 128 encryption for data at rest and in transit.
- 4099 – Hardening minimums: manufacturer assertion of secure credential storage
- 4100 – In – in enumerated item #10 "The "/oic/sec/cred" Resource should contain credential(s) if
4101 required by the selected OTM" is changed to require the credential be stored: "The
4102 "/oic/sec/cred" Resource shall contain credential(s)."
- 4103 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4104 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by
4105 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4106 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-
4107 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4108 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to
4109 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4110 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework
4111 described in the supporting documents:

- 4112 – Certificate Profile (See 9.4.2)
- 4113 – Certificate Policy (see Certificate Policy document:
4114 <https://openconnectivity.org/specs/OCF%20Certificate%20Policy.pdf>)

4115 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4116 **14.8.3.5.1 Blue Profile General**

4117 The Security Profile Blue is used when manufacturers issue platform certificates for platforms
4118 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is
4119 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF
4120 Security Domain owners evaluate manufacturer supplied certificates and attributed data to
4121 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.
4122 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may
4123 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4124 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting
4125 Criteria defined by OCF.

4126 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4127 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from
4128 OCF Device vendor and where platform vendors may implement trusted platforms that may conform
4129 to industry standards defining trusted platforms. The OCF Security Profile Blue specifies
4130 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance criteria
4131 produced by OCF conformance operations. The OCF Security Domain owner evaluates these data
4132 when an OCF Device is onboarded into the OCF Security Domain. Based on this evaluation the
4133 OCF Security Domain owner determines which Security Profile may be applied during OCF Device
4134 operation. All OCF Device types may be considered for evaluation using the OCF Security Profile
4135 Blue.

4136 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4137 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for this
4138 document version are satisfied and the following additional criteria are satisfied.

4139 OCF Blue profile defines the following OCF Device quality assurances:

- 4140 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF Device
4141 was hosted on one or more platforms that satisfies OCF Blue platform security assurances and
4142 platform security and privacy functionality requirements.
- 4143 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF and
4144 published by OCF in a machine readable format.
- 4145 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned
4146 signing key.
- 4147 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4148 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by the
4149 ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4150 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in
4151 its certificate.
- 4152 – The DOTS is expected to perform a lookup of the certification status of the OCF Device using
4153 the OCF CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the
4154 extension's "securityprofiles" field.

4155 OCF Blue profile defines the following OCF Device security functionality:

- 4156 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage
4157 functions are hardened by the platform.
- 4158 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials using
4159 the "/oic/sec/cred" Resource where the "credusage" Property contains the value
4160 "oic.sec.cred.mfgcert".
- 4161 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an IEEE802.1AR
4162 IDevID and should verify the "TCG Endorsement Key Credential". All TCG-defined
4163 manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the
4164 "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to
4165 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4166 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See
4167 NIST SP 800-57).
- 4168 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST SP
4169 800-57).
- 4170 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST
4171 SP 800-57).
- 4172 – OCF Device(s) should protect the "/oic/sec/cred" resource using the platform provided secure
4173 storage.
- 4174 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned
4175 certificates) using platform provided secure storage.
- 4176 – OCF Device(s) should check certificate revocation status for locally issued certificates.
- 4177 – The DOTS is expected to check certificate revocation status for all certificates in manufacturer
4178 certificate path(s) if available. If a certificate is revoked, certificate validation fails and the
4179 connection is refused. The DOTS may disregard revocation status results if unavailable.

4180 OCF Blue profile defines the following platform security assurances:

- 4181 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage
4182 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL
4183 Level 2.

- 4184 – Platforms implementing trusted platform functionality should be evaluated with a minimum
4185 Common Criteria EAL Level 1.
- 4186 OCF Blue profile defines the following platform security and privacy functionality:
- 4187 – The Platform shall implement cryptographic service provider (CSP) functionality.
 - 4188 – Platform CSP functionality shall include cryptographic algorithms, random number generation,
4189 secure time.
 - 4190 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST SP
4191 800-57).
 - 4192 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See
4193 NIST SP 800-57).
 - 4194 – Platforms hosting OCF Device(s) should implement a platform identifier following IEEE802.1AR
4195 or Trusted Computing Group(TCG) specifications.
 - 4196 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF Device(s)
4197 should supply TCG-defined manufacture certificates; also known as "TCG Endorsement Key
4198 Credential" (which complies with IETF RFC 5280) and "TCG Platform Credential" (which
4199 complies with IETF RFC 5755).
- 4200 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,
4201 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.
- 4202 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to
4203 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.
- 4204 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"
4205 Property to one of the other values found in the "supportedprofiles" Property.
- 4206 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**
- 4207 Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause
4208 14.8.2 must support following minimum requirements
- 4209 – Hardening minimums: secure credential storage, software integrity validation, secure update.
 - 4210 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension
4211 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-
4212 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"
 - 4213 – The OCF Device shall include a X.509v3 OCFPLAttributes Extension (clause 9.4.2.2.7) in its
4214 End-Entity Certificate when manufacturer certificate is used.
- 4215 Security Profile Purple has following optional security hardening requirements that the device can
4216 additionally support.
- 4217 – Hardening additions: secure boot, hardware backed secure storage
 - 4218 – The OCF Device shall include a X.509v3 OCFSecurityClaims Extension (clause 9.4.2.2.6) in its
4219 End-Entity Certificate and it shall include corresponding OIDs to the hardening additions
4220 implemented and attested by the vendor. If there is no additional support for hardening
4221 requirements, X.509v3 OCFSecurityClaims Extension shall be omitted.
- 4222 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism
4223 for security critical executables such as cryptographic modules or secure service applications, and
4224 they should be validated before the execution. The key used for validating the integrity must be
4225 pinned at the least to the validating software module.
- 4226 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4227 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM) to
4228 be executed by the processor on power-on, and secure boot parameters to be provisioned by
4229 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the
4230 security critical executables and stop the boot process if any integrity of them is compromised.

4231 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile
4232 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic
4233 attacks.

4234 More details on security hardening guidelines for software integrity validation, secure boot, secure
4235 update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4236 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA Vetting
4237 Criteria defined by OCF.

4238 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-purple-
4239 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other profiles.

4240 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to
4241 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

4242 **15 Device Type Specific Requirements**

4243 **15.1 Bridging Security**

4244 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4245 The Bridge shall go through OCF ownership transfer as any other onboarder would.

4246 The software of an Bridge shall be field updatable. (This requirement need not be tested but can
4247 be certified via a vendor declaration.)

4248 Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be provisioned
4249 as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged Devices are
4250 treated the same way as physical Devices. They are entities that have to be provisioned in their
4251 network.

4252 Each VOD shall implement the behaviour required by ISO/IEC 30118-1:2018 and this document.
4253 Each VOD shall perform authentication, access control, and encryption according to the security
4254 settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security
4255 requirements of the Bridged Protocol.

4256 In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall
4257 use appropriate ecosystem-specific security options for communication between the Virtual Bridged
4258 Devices instantiated by the Bridge and Bridged Devices. This security shall include mutual
4259 authentication, and encryption and integrity protection of messages in the bridged ecosystem.

4260 A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see
4261 clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which
4262 instantiated that VOD.

4263 A VOD may authenticate itself to the OCF Cloud (see clause 0) using the Manufacturer Certificate
4264 and corresponding private key of the Bridge which instantiated that VOD.

4265 A Bridge and the VODs created by that Bridge shall operate as independent Devices, with the
4266 following exceptions:

4267 – If a Bridge creates a VOD while the Bridge is in an Unowned State, then the VOD shall be
4268 created in an Unowned State.

4269 – An Unowned VOD shall not accept DTLS connection attempts nor TLS connection attempts nor
4270 any other requests, including discovery requests, while the Bridge (that created that VOD) is
4271 Unowned.

4272 – At any time when a Bridge is transitioning from Owned to Unowned State, all Unowned VODs
4273 (created by that Bridge prior to the transition) shall drop any existing TLS and/or DTLS
4274 connections.

4275 – At any time when a Bridge is transitioning from Unowned to Owned State, the Bridge shall
4276 trigger all Unowned VODs (created by that Bridge prior to the transition) to become accessible
4277 in RFOTM state, with internal state as if the VOD has just transitioned from RESET to RFOTM.

4278 – If a Bridge creates a VOD while the Bridge is in an Owned State, then the VOD shall become
4279 accessible in RFOTM state, with internal state as if the VOD has just transitioned from RESET
4280 to RFOTM.

4281 Table 60 intends to clarify this behaviour.

4282
4283

Table 60 – 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

4284 The "vods" Property of the "oic.r.vodlist" Resource on a Bridge reflects the details of all currently
4285 Owned VODs which have been created by that Bridge since the most recent hardware reset (if any)
4286 of the Bridge Platform (which removes all the created VODs), regardless of whether the VODs have
4287 the same owner as the Bridge or not. The entries in the "vods" Property are added and removed
4288 according to the following criteria:

- 4289 – Whenever a VOD created by a Bridge transitions from being Unowned to being Owned, then
4290 an entry for that VOD shall be added to the "vods" Property of the "oic.r.vodlist" Resource of
4291 that Bridge.
- 4292 – Whenever a VOD created by a Bridge transitions from being Owned to being Unowned, then
4293 entry for that VOD shall be removed from the "vods" Property of the "oic.r.vodlist" Resource of
4294 that Bridge. If that Bridge is currently in Unowned state, then the "oic.r.vodlist" Resource is not
4295 accessible, and the entry for that VOD shall be removed from the "vods" Property before or
4296 during the transition of that Bridge to the Owned state.
- 4297 – All other modifications of the list are not allowed.

4298 A Bridge shall only expose a secure OCF Endpoint for the "oic.r.vodlist" Resource.

4299 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4300 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4301 For AllJoyn translator, an authenticated and authorized Client shall be able to block the
4302 communication of all OCF Devices with all Bridged Devices that don't communicate securely with
4303 the Bridge, by using the Bridge Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-
4304 3:2018

4305 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4306 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4307 communicate securely with the Bridge.

4308 **15.1.2.3 Additional Security Requirements specific to the oneM2M Protocols**

4309 The Bridge shall implement oneM2M application access control as defined in the oneM2M Release
4310 3 Specifications.

4311 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4312 communicate securely with the Bridge.

4313 **15.1.2.4 Additional Security Requirements specific to the U+ Protocol**

4314 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4315 communicate securely with the Bridge.

4316 **15.1.2.5 Additional Security Requirements specific to the Z-Wave Protocol**

4317 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4318 communicate securely with the Bridge.

4319 **15.1.2.6 Additional Security Requirements specific to the Zigbee Protocol**

4320 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't
4321 communicate securely with the Bridge.

4322

4323

4324

4325

4326

4327

4328

4329

4330

4331

4332

4333

4334

4335

4336

4337

4338

4339

4340

4341

4342

4343 .

4344
4345
4346

Annex A (informative) Access Control Examples

4347

Example OCF ACL Resource

4348 Figure A-1 shows how a "/oic/sec/acl2" Resource could be configured to enforce an example
4349 access policy on the Server.

```
4350 {  
4351   "aclist2": [  
4352     {  
4353       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve, Update,  
4354       Delete and Notify)  
4355       "subject": {"uuid": "XXX-...-XX01"},  
4356       "resources": [  
4357         {"href": "/oic/sh/light/1"},  
4358         {"href": "/oic/sh/temp/0"}  
4359     ],  
4360     "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC  
4361     "validity": [  
4362       // The period starting at 18:00:00 UTC, on January 1, 2015 and  
4363       // ending at 07:00:00 UTC on January 2, 2015  
4364       "period": ["20150101T180000Z/20150102T070000Z"],  
4365       // Repeats the {period} every week until the last day of Jan. 2015.  
4366       "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]  
4367     ],  
4368     "aceid": 1  
4369   }  
4370 ],  
4371   // An ACL provisioning and management service should be identified as  
4372   // the resource owner  
4373   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"  
4374 }  
4375
```

Figure A-1 – Example "/oic/sec/acl2" Resource

4376
4377
4378

Annex B (Informative) Execution Environment Security Profiles

4379 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security
4380 robustness requirements meeting all IOT applications and services will not serve the needs of OCF,
4381 and security profiles of varying degree of robustness (trustworthiness), cost and complexity have
4382 to be defined. To address a large ecosystem of vendors, the profiles can only be defined as
4383 requirements and the exact solutions meeting those requirements are specific to the vendors' open
4384 or proprietary implementations, and thus in most part outside scope of this document.

4385 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228
4386 (Terminology for constrained node networks) methodology, we limit the number of security profiles
4387 to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities criteria for
4388 each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4389 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are
4390 either capable of no security functionality or easily breakable security that depend on environmental
4391 (e.g. availability of human) factors to perform security functions. This means the class 0 will not be
4392 equipped with an SEE.

4393

Table B.1 – OCF Security Profile

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4394 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of
4395 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,
4396 memory, storage).

4397
4398
4399

Annex C (normative) Resource Type definitions

4400 C.1 List of Resource Type definitions

4401 Table C.1 contains the list of defined security resources in this document.

4402 **Table C.1 – Alphabetized list of security resources**

Friendly Name (informative)	Resource Type (rt)	Clause
Access Control List 2	oic.r.acl2	C.2
Certificate Revocation	oic.r.crl	C.4
Certificate Signing Request	oic.r.csr	C.5
Credential	oic.r.cred	C.3
Device owner transfer method	oic.r.doxm	C.6
Device Provisioning Status	oic.r.pstat	C.7
Roles	oic.r.roles	C.8
Security Profile	oic.r.sp	C.9
Account	oic.r.account	Moved to OCF Cloud Security document
Account Session	oic.r.session	Moved to OCF Cloud Security document
Account Token Refresh	oic.r.tokenrefresh	Moved to OCF Cloud Security document

4403 C.2 Access Control List-2

4404 C.2.1 Introduction

4405 This Resource specifies the local access control list.
4406 When used without query parameters, all the ACE entries are returned.
4407 When used with a query parameter, only the ACEs matching the specified
4408 parameter are returned.
4409

4410 C.2.2 Well-known URI

4411 /oic/sec/acl2

4412 C.2.3 Resource type

4413 The Resource Type is defined as: "oic.r.acl2".

4414 C.2.4 OpenAPI 2.0 definition

```
4415 {  
4416   "swagger": "2.0",  
4417   "info": {  
4418     "title": "Access Control List-2",  
4419     "version": "20190111",  
4420     "license": {  
4421       "name": "OCF Data Model License",  
4422       "url":  
4423       "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
4424       CENSE.md",  
4425       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
```

```

4426 reserved."
4427     },
4428     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4429     },
4430     "schemes": ["http"],
4431     "consumes": ["application/json"],
4432     "produces": ["application/json"],
4433     "paths": {
4434         "/oic/sec/acl2" : {
4435             "get": {
4436                 "description": "This Resource specifies the local access control list.\nWhen used without
4437 query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
4438 matching the specified\nparameter are returned.\n",
4439                 "parameters": [
4440                     {"$ref": "#/parameters/interface"},
4441                     {"$ref": "#/parameters/ace-filtered"}
4442                 ],
4443                 "responses": {
4444                     "200": {
4445                         "description": "",
4446                         "x-example":
4447                             {
4448                                 "rt": ["oic.r.acl2"],
4449                                 "aclist2": [
4450                                     {
4451                                         "aceid": 1,
4452                                         "subject": {
4453                                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4454                                             "role": "SOME_STRING"
4455                                         },
4456                                         "resources": [
4457                                             {
4458                                                 "href": "/light",
4459                                                 "rt": ["oic.r.light"],
4460                                                 "if": ["oic.if.baseline", "oic.if.a"]
4461                                             },
4462                                             {
4463                                                 "href": "/door",
4464                                                 "rt": ["oic.r.door"],
4465                                                 "if": ["oic.if.baseline", "oic.if.a"]
4466                                             }
4467                                         ],
4468                                         "permission": 24
4469                                     }
4470                                 ],
4471                                 "aceid": 2,
4472                                 "subject": {
4473                                     "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4474                                 },
4475                                 "resources": [
4476                                     {
4477                                         "href": "/light",
4478                                         "rt": ["oic.r.light"],
4479                                         "if": ["oic.if.baseline", "oic.if.a"]
4480                                     },
4481                                     {
4482                                         "href": "/door",
4483                                         "rt": ["oic.r.door"],
4484                                         "if": ["oic.if.baseline", "oic.if.a"]
4485                                     }
4486                                 ],
4487                                 "permission": 24
4488                             },
4489                             {
4490                                 "aceid": 3,
4491                                 "subject": {"conntype": "anon-clear"},
4492                                 "resources": [
4493                                     {
4494                                         "href": "/light",
4495                                         "rt": ["oic.r.light"],
4496                                         "if": ["oic.if.baseline", "oic.if.a"]

```



```

4497         },
4498         {
4499             "href": "/door",
4500             "rt": ["oic.r.door"],
4501             "if": ["oic.if.baseline", "oic.if.a"]
4502         }
4503     ],
4504     "permission": 16,
4505     "validity": [
4506         {
4507             "period": "20160101T180000Z/20170102T070000Z",
4508             "recurrence": [ "DSTART:XXXXX",
4509 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4510         },
4511         {
4512             "period": "20160101T180000Z/PT5H30M",
4513             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4514         }
4515     ]
4516 },
4517 ],
4518 "owneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
4519 },
4520 "schema": { "$ref": "#/definitions/Acl2" }
4521 },
4522 "400": {
4523     "description": "The request is invalid."
4524 }
4525 }
4526 },
4527 "post": {
4528     "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
4529 update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
4530 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
4531 the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
4532     "parameters": [
4533         { "$ref": "#/parameters/interface" },
4534         { "$ref": "#/parameters/ace-filtered" },
4535     ],
4536     "name": "body",
4537     "in": "body",
4538     "required": true,
4539     "schema": { "$ref": "#/definitions/Acl2-Update" },
4540     "x-example":
4541     {
4542         "aclist2": [
4543             {
4544                 "aceid": 1,
4545                 "subject": {
4546                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4547                     "role": "SOME_STRING"
4548                 },
4549                 "resources": [
4550                     {
4551                         "href": "/light",
4552                         "rt": ["oic.r.light"],
4553                         "if": ["oic.if.baseline", "oic.if.a"]
4554                     },
4555                     {
4556                         "href": "/door",
4557                         "rt": ["oic.r.door"],
4558                         "if": ["oic.if.baseline", "oic.if.a"]
4559                     }
4560                 ],
4561                 "permission": 24
4562             },
4563             {
4564                 "aceid": 3,
4565                 "subject": {
4566                     "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4567                 },

```

```

4568         "resources": [
4569             {
4570                 "href": "/light",
4571                 "rt": ["oic.r.light"],
4572                 "if": ["oic.if.baseline", "oic.if.a"]
4573             },
4574             {
4575                 "href": "/door",
4576                 "rt": ["oic.r.door"],
4577                 "if": ["oic.if.baseline", "oic.if.a"]
4578             }
4579         ],
4580         "permission": 24
4581     }
4582 ],
4583     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4584 }
4585 }
4586 ],
4587 "responses": {
4588     "400": {
4589         "description": "The request is invalid."
4590     },
4591     "201": {
4592         "description": "The ACL entry is created."
4593     },
4594     "204": {
4595         "description": "The ACL entry is updated."
4596     }
4597 },
4598 },
4599 "delete": {
4600     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
4601 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
4602 the\nspecified parameter are deleted.\n",
4603     "parameters": [
4604         {"$ref": "#/parameters/interface"},
4605         {"$ref": "#/parameters/ace-filtered"}
4606     ],
4607     "responses": {
4608         "200": {
4609             "description": "The matching ACEs or the entire ACL Resource has been successfully
4610 deleted."
4611         },
4612         "400": {
4613             "description": "The request is invalid."
4614         }
4615     }
4616 },
4617 },
4618 },
4619 "parameters": {
4620     "interface": {
4621         "in": "query",
4622         "name": "if",
4623         "type": "string",
4624         "enum": ["oic.if.baseline"]
4625     },
4626     "ace-filtered": {
4627         "in": "query",
4628         "name": "aceid",
4629         "required": false,
4630         "type": "integer",
4631         "description": "Only applies to the ACE with the specified aceid.",
4632         "x-example": 2112
4633     }
4634 },
4635 "definitions": {
4636     "Acl2": {
4637         "properties": {
4638             "rowneruuid": {

```

```

4639         "description": "The value identifies the unique Resource owner\nFormat pattern according
4640 to IETF RFC 4122.",
4641         "pattern": "[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4642 9]{12}$",
4643         "type": "string"
4644     },
4645     "rt" : {
4646         "description": "Resource Type of the Resource.",
4647         "items": {
4648             "maxLength": 64,
4649             "type": "string",
4650             "enum": ["oic.r.acl2"]
4651         },
4652         "minItems": 1,
4653         "maxItems": 1,
4654         "readOnly": true,
4655         "type": "array"
4656     },
4657     "aclist2" : {
4658         "description": "Access Control Entries in the ACL Resource.",
4659         "items": {
4660             "properties": {
4661                 "aceid": {
4662                     "description": "An identifier for the ACE that is unique within the ACL. In cases
4663 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
4664                     "minimum": 1,
4665                     "type": "integer"
4666                 },
4667                 "permission": {
4668                     "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
4669 permissions.",
4670                     "x-detail-desc": [
4671                         "0 - No permissions",
4672                         "1 - Create permission is granted",
4673                         "2 - Read, observe, discover permission is granted",
4674                         "4 - Write, update permission is granted",
4675                         "8 - Delete permission is granted",
4676                         "16 - Notify permission is granted"
4677                     ],
4678                     "maximum": 31,
4679                     "minimum": 0,
4680                     "type": "integer"
4681                 },
4682                 "resources": {
4683                     "description": "References the application's Resources to which a security policy
4684 applies.",
4685                     "items": {
4686                         "description": "Each Resource must have at least one of these properties set.",
4687                         "properties": {
4688                             "href": {
4689                                 "description": "When present, the ACE only applies when the href matches\nThis
4690 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
4691                                 "format": "uri",
4692                                 "maxLength": 256,
4693                                 "type": "string"
4694                             },
4695                             "if": {
4696                                 "description": "When present, the ACE only applies when the if (interface)
4697 matches\nThe interface set supported by this Resource.",
4698                                 "items": {
4699                                     "enum": [
4700                                         "oic.if.baseline",
4701                                         "oic.if.ll",
4702                                         "oic.if.b",
4703                                         "oic.if.rw",
4704                                         "oic.if.r",
4705                                         "oic.if.a",
4706                                         "oic.if.s"
4707                                     ],
4708                                     "type": "string"
4709                                 },

```

```

4710         "minItems": 1,
4711         "type": "array"
4712     },
4713     "rt": {
4714         "description": "When present, the ACE only applies when the rt (Resource type)
4715 matches\nResource Type of the Resource.",
4716         "items": {
4717             "maxLength": 64,
4718             "type": "string"
4719         },
4720         "minItems": 1,
4721         "type": "array"
4722     },
4723     "wc": {
4724         "description": "A wildcard matching policy.",
4725         "pattern": "^[~*]$",
4726         "type": "string"
4727     }
4728 },
4729 "type": "object"
4730 },
4731 "type": "array"
4732 },
4733 "subject": {
4734     "anyOf": [
4735         {
4736             "description": "This is the Device identifier.",
4737             "properties": {
4738                 "uuid": {
4739                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4740 4122.",
4741                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
4742 fA-F0-9]{12}$",
4743                     "type": "string"
4744                 }
4745             },
4746             "required": [
4747                 "uuid"
4748             ],
4749             "type": "object"
4750         },
4751         {
4752             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
4753 <Authority> refers to the local entity or Device.",
4754             "properties": {
4755                 "authority": {
4756                     "description": "The Authority component of the entity being identified. A
4757 NULL <Authority> refers to the local entity or Device.",
4758                     "type": "string"
4759                 },
4760                 "role": {
4761                     "description": "The ID of the role being identified.",
4762                     "type": "string"
4763                 }
4764             },
4765             "required": [
4766                 "role"
4767             ],
4768             "type": "object"
4769         },
4770         {
4771             "properties": {
4772                 "conntype": {
4773                     "description": "This property allows an ACE to be matched based on the
4774 connection or message type.",
4775                     "x-detail-desc": [
4776                         "auth-crypt - ACE applies if the Client is authenticated and the data
4777 channel or message is encrypted and integrity protected",
4778                         "anon-clear - ACE applies if the Client is not authenticated and the data
4779 channel or message is not encrypted but may be integrity protected"
4780                     ],

```

```

4781         "enum": [
4782             "auth-crypt",
4783             "anon-clear"
4784         ],
4785         "type": "string"
4786     }
4787 },
4788     "required": [
4789         "conntype"
4790     ],
4791     "type": "object"
4792 }
4793 ]
4794 },
4795 "validity": {
4796     "description": "validity is an array of time-pattern objects.",
4797     "items": {
4798         "description": "The time-pattern contains a period and recurrence expressed in
4799 RFC5545 syntax.",
4800         "properties": {
4801             "period": {
4802                 "description": "String represents a period using the RFC5545 Period.",
4803                 "type": "string"
4804             },
4805             "recurrence": {
4806                 "description": "String array represents a recurrence rule using the RFC5545
4807 Recurrence.",
4808                 "items": {
4809                     "type": "string"
4810                 },
4811                 "type": "array"
4812             }
4813         },
4814         "required": [
4815             "period"
4816         ],
4817         "type": "object"
4818     },
4819     "type": "array"
4820 }
4821 },
4822 "required": [
4823     "aceid",
4824     "resources",
4825     "permission",
4826     "subject"
4827 ],
4828 "type": "object"
4829 },
4830 "type": "array"
4831 },
4832 "n": {
4833     "$ref":
4834 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4835 schema.json#/definitions/n"
4836 },
4837 "id": {
4838     "$ref":
4839 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4840 schema.json#/definitions/id"
4841 },
4842 "if" : {
4843     "description": "The interface set supported by this Resource.",
4844     "items": {
4845         "enum": [
4846             "oic.if.baseline"
4847         ],
4848         "type": "string"
4849     },
4850     "minItems": 1,
4851     "maxItems": 1,

```

```

4852         "readOnly": true,
4853         "type": "array"
4854     },
4855 },
4856 "type" : "object",
4857 "required": ["acllist2", "rowneruuid"]
4858 },
4859 "Acl2-Update" : {
4860     "properties": {
4861         "rowneruuid" : {
4862             "description": "The value identifies the unique Resource owner\n Format pattern according
4863 to IETF RFC 4122.",
4864             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4865 9]{12}$",
4866             "type": "string"
4867         },
4868         "acllist2" : {
4869             "description": "Access Control Entries in the ACL Resource.",
4870             "items": {
4871                 "properties": {
4872                     "aceid": {
4873                         "description": "An identifier for the ACE that is unique within the ACL. In cases
4874 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
4875                         "minimum": 1,
4876                         "type": "integer"
4877                     },
4878                     "permission": {
4879                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
4880 permissions.",
4881                         "x-detail-desc": [
4882                             "0 - No permissions",
4883                             "1 - Create permission is granted",
4884                             "2 - Read, observe, discover permission is granted",
4885                             "4 - Write, update permission is granted",
4886                             "8 - Delete permission is granted",
4887                             "16 - Notify permission is granted"
4888                         ],
4889                         "maximum": 31,
4890                         "minimum": 0,
4891                         "type": "integer"
4892                     },
4893                     "resources": {
4894                         "description": "References the application's Resources to which a security policy
4895 applies.",
4896                         "items": {
4897                             "description": "Each Resource must have at least one of these properties set.",
4898                             "properties": {
4899                                 "href": {
4900                                     "description": "When present, the ACE only applies when the href matches\nThis
4901 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
4902                                     "format": "uri",
4903                                     "maxLength": 256,
4904                                     "type": "string"
4905                                 },
4906                                 "if": {
4907                                     "description": "When present, the ACE only applies when the if (interface)
4908 matches\nThe interface set supported by this Resource.",
4909                                     "items": {
4910                                         "enum": [
4911                                             "oic.if.baseline",
4912                                             "oic.if.ll",
4913                                             "oic.if.b",
4914                                             "oic.if.rw",
4915                                             "oic.if.r",
4916                                             "oic.if.a",
4917                                             "oic.if.s"
4918                                         ],
4919                                         "type": "string"
4920                                     },
4921                                     "minItems": 1,
4922                                     "type": "array"

```

```

4923     },
4924     "rt": {
4925         "description": "When present, the ACE only applies when the rt (Resource type)
4926 matches\nResource Type of the Resource.",
4927         "items": {
4928             "maxLength": 64,
4929             "type": "string"
4930         },
4931         "minItems": 1,
4932         "type": "array"
4933     },
4934     "wc": {
4935         "description": "A wildcard matching policy.",
4936         "x-detail-desc": [
4937             "+ - Matches all discoverable Resources",
4938             "- - Matches all non-discoverable Resources",
4939             "* - Matches all Resources"
4940         ],
4941         "enum": [
4942             "+",
4943             "-",
4944             "*"
4945         ],
4946         "type": "string"
4947     }
4948 },
4949 "type": "object"
4950 },
4951 "type": "array"
4952 },
4953 "subject": {
4954     "anyOf": [
4955         {
4956             "description": "This is the Device identifier.",
4957             "properties": {
4958                 "uuid": {
4959                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4960 4122.",
4961                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
4962 fA-F0-9]{12}$",
4963                     "type": "string"
4964                 }
4965             },
4966             "required": [
4967                 "uuid"
4968             ],
4969             "type": "object"
4970         },
4971         {
4972             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
4973 <Authority> refers to the local entity or Device.",
4974             "properties": {
4975                 "authority": {
4976                     "description": "The Authority component of the entity being identified. A
4977 NULL <Authority> refers to the local entity or Device.",
4978                     "type": "string"
4979                 },
4980                 "role": {
4981                     "description": "The ID of the role being identified.",
4982                     "type": "string"
4983                 }
4984             },
4985             "required": [
4986                 "role"
4987             ],
4988             "type": "object"
4989         },
4990     ],
4991     "properties": {
4992         "conntype": {
4993             "description": "This property allows an ACE to be matched based on the

```

```

4994 connection or message type.",
4995         "x-detail-desc": [
4996             "auth-crypt - ACE applies if the Client is authenticated and the data
4997 channel or message is encrypted and integrity protected",
4998             "anon-clear - ACE applies if the Client is not authenticated and the data
4999 channel or message is not encrypted but may be integrity protected"
5000         ],
5001         "enum": [
5002             "auth-crypt",
5003             "anon-clear"
5004         ],
5005         "type": "string"
5006     }
5007 },
5008     "required": [
5009         "conntype"
5010     ],
5011     "type": "object"
5012 }
5013 ]
5014 },
5015 "validity": {
5016     "description": "validity is an array of time-pattern objects.",
5017     "items": {
5018         "description": "The time-pattern contains a period and recurrence expressed in
5019 RFC5545 syntax.",
5020         "properties": {
5021             "period": {
5022                 "description": "String represents a period using the RFC5545 Period.",
5023                 "type": "string"
5024             },
5025             "recurrence": {
5026                 "description": "String array represents a recurrence rule using the RFC5545
5027 Recurrence.",
5028                 "items": {
5029                     "type": "string"
5030                 },
5031                 "type": "array"
5032             }
5033         },
5034         "required": [
5035             "period"
5036         ],
5037         "type": "object"
5038     },
5039     "type": "array"
5040 }
5041 },
5042 "required": [
5043     "resources",
5044     "permission",
5045     "subject"
5046 ],
5047 "type": "object"
5048 },
5049 "type": "array"
5050 }
5051 },
5052 "type" : "object"
5053 }
5054 }
5055 }
5056

```

5057 C.2.5 Property definition

5058 Table C-1 defines the Properties that are part of the "oic.r.acl2" Resource Type.

5059 **Table C-1 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".**

Property name	Value type	Mandatory	Access mode	Description
---------------	------------	-----------	-------------	-------------

rowneruuid	string	Yes	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
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	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rowneruuid	string	No	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
aclist2	array: see schema	No	Read Write	Access Control Entries in the ACL Resource.

5060 **C.2.6 CRUDN behaviour**

5061 Table C-2 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

5062 **Table C-2 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5063 **C.3 Credential**

5064 **C.3.1 Introduction**

5065 This Resource specifies credentials a Device may use to establish secure communication.

5066 Retrieves the credential data.

5067 When used without query parameters, all the credential entries are returned.

5068 When used with a query parameter, only the credentials matching the specified
5069 parameter are returned.

5070 Note that write-only credential data will not be returned.
5071
5072

5073 **C.3.2 Well-known URI**

5074 /oic/sec/cred

5075 **C.3.3 Resource type**

5076 The Resource Type is defined as: "oic.r.cred".

5077 C.3.4 OpenAPI 2.0 definition

```
5078 {
5079   "swagger": "2.0",
5080   "info": {
5081     "title": "Credential",
5082     "version": "v1.0-20181031",
5083     "license": {
5084       "name": "OCF Data Model License",
5085       "url":
5086         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5087         CENSE.md",
5088       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5089       reserved."
5090     },
5091     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5092   },
5093   "schemes": ["http"],
5094   "consumes": ["application/json"],
5095   "produces": ["application/json"],
5096   "paths": {
5097     "/oic/sec/cred" : {
5098       "get": {
5099         "description": "This Resource specifies credentials a Device may use to establish secure
5100         communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
5101         credential entries are returned.\nWhen used with a query parameter, only the credentials matching
5102         the specified\nparameter are returned.\n\nNote that write-only credential data will not be
5103         returned.\n",
5104         "parameters": [
5105           {"$ref": "#/parameters/interface"}
5106           ,{"$ref": "#/parameters/cred-filtered-credid"}
5107           ,{"$ref": "#/parameters/cred-filtered-subjectuuid"}
5108         ],
5109         "responses": {
5110           "200": {
5111             "description": "",
5112             "x-example":
5113               {
5114                 "rt": ["oic.r.cred"],
5115                 "creds": [
5116                   {
5117                     "credid": 55,
5118                     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5119                     "roleid": {
5120                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5121                       "role": "SOME_STRING"
5122                     },
5123                     "credtype": 32,
5124                     "publicdata": {
5125                       "encoding": "oic.sec.encoding.base64",
5126                       "data": "BASE-64-ENCODED-VALUE"
5127                     },
5128                     "privatedata": {
5129                       "encoding": "oic.sec.encoding.base64",
5130                       "data": "BASE-64-ENCODED-VALUE",
5131                       "handle": 4
5132                     },
5133                     "optionaldata": {
5134                       "revstat": false,
5135                       "encoding": "oic.sec.encoding.base64",
5136                       "data": "BASE-64-ENCODED-VALUE"
5137                     },
5138                     "period": "20160101T180000Z/20170102T070000Z",
5139                     "crms": [ "oic.sec.crm.pk10" ]
5140                   },
5141                   {
5142                     "credid": 56,
5143                     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5144                     "roleid": {
5145                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5146                       "role": "SOME_STRING"
5147                     }
5148                   }
5149                 ]
5150             }
5151           }
5152         }
5153       }
5154     }
5155   }
5156 }
```

```

5147         },
5148         "credtype": 1,
5149         "publicdata": {
5150             "encoding": "oic.sec.encoding.base64",
5151             "data": "BASE-64-ENCODED-VALUE"
5152         },
5153         "privatedata": {
5154             "encoding": "oic.sec.encoding.base64",
5155             "data": "BASE-64-ENCODED-VALUE",
5156             "handle": 4
5157         },
5158         "optionaldata": {
5159             "revstat": false,
5160             "encoding": "oic.sec.encoding.base64",
5161             "data": "BASE-64-ENCODED-VALUE"
5162         },
5163         "period": "20160101T180000Z/20170102T070000Z",
5164         "crms": [ "oic.sec.crm.pk10" ]
5165     }
5166 ],
5167     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5168 }
5169 ,
5170     "schema": { "$ref": "#/definitions/Cred" }
5171 },
5172 "400": {
5173     "description": "The request is invalid."
5174 }
5175 }
5176 },
5177 "post": {
5178     "description": "Updates the credential Resource with the provided
5179 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
5180 Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
5181 Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
5182 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
5183 Resource.\n",
5184     "parameters": [
5185         { "$ref": "#/parameters/interface" },
5186         {
5187             "name": "body",
5188             "in": "body",
5189             "required": true,
5190             "schema": { "$ref": "#/definitions/Cred-Update" },
5191             "x-example":
5192                 {
5193                     "creds": [
5194                         {
5195                             "credid": 55,
5196                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5197                             "roleid": {
5198                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5199                                 "role": "SOME_STRING"
5200                             },
5201                             "credtype": 32,
5202                             "publicdata": {
5203                                 "encoding": "oic.sec.encoding.base64",
5204                                 "data": "BASE-64-ENCODED-VALUE"
5205                             },
5206                             "privatedata": {
5207                                 "encoding": "oic.sec.encoding.base64",
5208                                 "data": "BASE-64-ENCODED-VALUE",
5209                                 "handle": 4
5210                             },
5211                             "optionaldata": {
5212                                 "revstat": false,
5213                                 "encoding": "oic.sec.encoding.base64",
5214                                 "data": "BASE-64-ENCODED-VALUE"
5215                             },
5216                             "period": "20160101T180000Z/20170102T070000Z",
5217                             "crms": [ "oic.sec.crm.pk10" ]

```

```

5218         },
5219         {
5220             "credid": 56,
5221             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5222             "roleid": {
5223                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5224                 "role": "SOME_STRING"
5225             },
5226             "credtype": 1,
5227             "publicdata": {
5228                 "encoding": "oic.sec.encoding.base64",
5229                 "data": "BASE-64-ENCODED-VALUE"
5230             },
5231             "privatedata": {
5232                 "encoding": "oic.sec.encoding.base64",
5233                 "data": "BASE-64-ENCODED-VALUE",
5234                 "handle": 4
5235             },
5236             "optionaldata": {
5237                 "revstat": false,
5238                 "encoding": "oic.sec.encoding.base64",
5239                 "data": "BASE-64-ENCODED-VALUE"
5240             },
5241             "period": "20160101T180000Z/20170102T070000Z",
5242             "crms": [ "oic.sec.crm.pk10" ]
5243         }
5244     ],
5245     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5246 }
5247 }
5248 ],
5249 "responses": {
5250     "400": {
5251         "description": "The request is invalid."
5252     },
5253     "201": {
5254         "description": "The credential entry is created."
5255     },
5256     "204": {
5257         "description": "The credential entry is updated."
5258     }
5259 },
5260 },
5261 "delete": {
5262     "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
5263 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
5264 matching\nthe query parameter are deleted.\n",
5265     "parameters": [
5266         {"$ref": "#/parameters/interface"},
5267         {"$ref": "#/parameters/cred-filtered-credid"},
5268         {"$ref": "#/parameters/cred-filtered-subjectuuid"}
5269     ],
5270     "responses": {
5271         "400": {
5272             "description": "The request is invalid."
5273         },
5274         "204": {
5275             "description": "The specific credential(s) or the the entire credential Resource has
5276 been successfully deleted."
5277         }
5278     }
5279 }
5280 }
5281 },
5282 "parameters": {
5283     "interface": {
5284         "in": "query",
5285         "name": "if",
5286         "type": "string",
5287         "enum": ["oic.if.baseline"]
5288     },

```

```

5289     "cred-filtered-credid" : {
5290         "in" : "query",
5291         "name" : "credid",
5292         "required" : false,
5293         "type" : "integer",
5294         "description" : "Only applies to the credential with the specified credid.",
5295         "x-example" : 2112
5296     },
5297     "cred-filtered-subjectuuid" : {
5298         "in" : "query",
5299         "name" : "subjectuuid",
5300         "required" : false,
5301         "type" : "string",
5302         "description" : "Only applies to credentials with the specified subject UUID.",
5303         "x-example" : "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5304     }
5305 },
5306 "definitions": {
5307     "Cred" : {
5308         "properties": {
5309             "rowneruuid" : {
5310                 "description": "Format pattern according to IETF RFC 4122.",
5311                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5312                 "type": "string"
5313             },
5314             "rt" : {
5315                 "description": "Resource Type of the Resource.",
5316                 "items": {
5317                     "maxLength": 64,
5318                     "type": "string",
5319                     "enum": ["oic.r.cred"]
5320                 },
5321                 "minItems": 1,
5322                 "readOnly": true,
5323                 "type": "array",
5324                 "uniqueItems": true
5325             },
5326             "n" : {
5327                 "$ref":
5328                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-schema.json#/definitions/n"
5329             },
5330             "id" : {
5331                 "$ref":
5332                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-schema.json#/definitions/id"
5333             },
5334             "creds" : {
5335                 "description": "List of credentials available at this Resource.",
5336                 "items": {
5337                     "properties": {
5338                         "credid" : {
5339                             "description": "Local reference to a credential Resource.",
5340                             "type": "integer"
5341                         },
5342                         "credtype" : {
5343                             "description": "Representation of this credential's type\nCredential Types - Cred
5344 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
5345 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
5346 password32 - Asymmetric encryption key.",
5347                             "maximum": 63,
5348                             "minimum": 0,
5349                             "type": "integer"
5350                         },
5351                         "credusage" : {
5352                             "description": "A string that provides hints about how/where the cred is used\nThe
5353 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
5354 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
5355 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
5356                             "enum": [

```

```

5360         "oic.sec.cred.trustca",
5361         "oic.sec.cred.cert",
5362         "oic.sec.cred.rolecert",
5363         "oic.sec.cred.mfgtrustca",
5364         "oic.sec.cred.mfgcert"
5365     ],
5366     "type": "string"
5367 },
5368 "crms": {
5369     "description": "The refresh methods that may be used to update this credential.",
5370     "items": {
5371         "description": "Each enum represents a method by which the credentials are
5372 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
5373 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
5374 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
5375 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
5376         "enum": [
5377             "oic.sec.crm.pro",
5378             "oic.sec.crm.psk",
5379             "oic.sec.crm.rdp",
5380             "oic.sec.crm.skdc",
5381             "oic.sec.crm.pk10"
5382         ],
5383         "type": "string"
5384     },
5385     "type": "array",
5386     "uniqueItems" : true
5387 },
5388 "optionaldata": {
5389     "description": "Credential revocation status information\nOptional credential
5390 contents describes revocation status for this credential.",
5391     "properties": {
5392         "data": {
5393             "description": "The encoded structure.",
5394             "type": "string"
5395         },
5396         "encoding": {
5397             "description": "A string specifying the encoding format of the data contained in
5398 the optdata.",
5399             "x-detail-desc": [
5400                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
5401                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
5402                 "oic.sec.encoding.base64 - Base64 encoded object.",
5403                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
5404                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
5405                 "oic.sec.encoding.raw - Raw hex encoded data."
5406             ],
5407             "enum": [
5408                 "oic.sec.encoding.jwt",
5409                 "oic.sec.encoding.cwt",
5410                 "oic.sec.encoding.base64",
5411                 "oic.sec.encoding.pem",
5412                 "oic.sec.encoding.der",
5413                 "oic.sec.encoding.raw"
5414             ],
5415             "type": "string"
5416         },
5417         "revstat": {
5418             "description": "Revocation status flag - true = revoked.",
5419             "type": "boolean"
5420         }
5421     },
5422     "required": [
5423         "revstat"
5424     ],
5425     "type": "object"
5426 },
5427 "period": {
5428     "description": "String with RFC5545 Period.",
5429     "type": "string"
5430 },

```

```

5431         "privatedata": {
5432             "description": "Private credential information\nCredential Resource non-public
5433 contents.",
5434             "properties": {
5435                 "data": {
5436                     "description": "The encoded value.",
5437                     "maxLength": 3072,
5438                     "type": "string"
5439                 },
5440                 "encoding": {
5441                     "description": "A string specifying the encoding format of the data contained in
5442 the privdata\nnoic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\nnoic.sec.encoding.cwt -
5443 RFC CBOR web token (CWT) encoding\nnoic.sec.encoding.base64 - Base64 encoded
5444 object\nnoic.sec.encoding.uri - URI reference\nnoic.sec.encoding.handle - Data is contained in a
5445 storage sub-system referenced using a handle\nnoic.sec.encoding.raw - Raw hex encoded data.",
5446                     "enum": [
5447                         "oic.sec.encoding.jwt",
5448                         "oic.sec.encoding.cwt",
5449                         "oic.sec.encoding.base64",
5450                         "oic.sec.encoding.uri",
5451                         "oic.sec.encoding.handle",
5452                         "oic.sec.encoding.raw"
5453                     ],
5454                     "type": "string"
5455                 },
5456                 "handle": {
5457                     "description": "Handle to a key storage Resource.",
5458                     "type": "integer"
5459                 }
5460             },
5461             "required": [
5462                 "encoding"
5463             ],
5464             "type": "object"
5465         },
5466         "publicdata": {
5467             "description": "Public credential information.",
5468             "properties": {
5469                 "data": {
5470                     "description": "The encoded value.",
5471                     "maxLength": 3072,
5472                     "type": "string"
5473                 },
5474                 "encoding": {
5475                     "description": "A string specifying the encoding format of the data contained in
5476 the pubdata\nnoic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\nnoic.sec.encoding.cwt -
5477 RFC CBOR web token (CWT) encoding\nnoic.sec.encoding.base64 - Base64 encoded
5478 object\nnoic.sec.encoding.uri - URI reference\nnoic.sec.encoding.pem - Encoding for PEM encoded
5479 certificate or chain\nnoic.sec.encoding.der - Encoding for DER encoded
5480 certificate\nnoic.sec.encoding.raw - Raw hex encoded data.",
5481                     "enum": [
5482                         "oic.sec.encoding.jwt",
5483                         "oic.sec.encoding.cwt",
5484                         "oic.sec.encoding.base64",
5485                         "oic.sec.encoding.uri",
5486                         "oic.sec.encoding.pem",
5487                         "oic.sec.encoding.der",
5488                         "oic.sec.encoding.raw"
5489                     ],
5490                     "type": "string"
5491                 }
5492             },
5493             "type": "object"
5494         },
5495         "roleid": {
5496             "description": "The role this credential possesses\nSecurity role specified as an
5497 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
5498             "properties": {
5499                 "authority": {
5500                     "description": "The Authority component of the entity being identified. A NULL
5501 <Authority> refers to the local entity or Device.",

```

```

5502         "type": "string"
5503     },
5504     "role": {
5505         "description": "The ID of the role being identified.",
5506         "type": "string"
5507     }
5508 },
5509     "required": [
5510         "role"
5511     ],
5512     "type": "object"
5513 },
5514     "subjectuuid": {
5515         "anyOf": [
5516             {
5517                 "description": "The id of the Device, which the cred entry applies to or \"*\
5518 for wildcard identity.",
5519                 "pattern": "^\\*$",
5520                 "type": "string"
5521             },
5522             {
5523                 "description": "Format pattern according to IETF RFC 4122.",
5524                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
5525 F0-9]{12}$",
5526                 "type": "string"
5527             }
5528         ]
5529     }
5530 },
5531     "type": "object"
5532 },
5533     "type": "array"
5534 },
5535     "if" : {
5536         "description": "The interface set supported by this Resource.",
5537         "items": {
5538             "enum": [
5539                 "oic.if.baseline"
5540             ],
5541             "type": "string"
5542         },
5543         "minItems": 1,
5544         "readOnly": true,
5545         "type": "array"
5546     }
5547 },
5548     "type" : "object",
5549     "required": ["creds", "rowneruuid"]
5550 },
5551     "Cred-Update" : {
5552         "properties": {
5553             "rowneruuid" : {
5554                 "description": "Format pattern according to IETF RFC 4122.",
5555                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5556 9]{12}$",
5557                 "type": "string"
5558             },
5559             "creds" : {
5560                 "description": "List of credentials available at this Resource.",
5561                 "items": {
5562                     "properties": {
5563                         "credid": {
5564                             "description": "Local reference to a credential Resource.",
5565                             "type": "integer"
5566                         },
5567                         "credtype": {
5568                             "description": "Representation of this credential's type\nCredential Types - Cred
5569 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
5570 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
5571 password32 - Asymmetric encryption key.",
5572                             "maximum": 63,

```



```

5573         "minimum": 0,
5574         "type": "integer"
5575     },
5576     "credusage": {
5577         "description": "A string that provides hints about how/where the cred is used\nThe
5578 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
5579 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
5580 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
5581         "enum": [
5582             "oic.sec.cred.trustca",
5583             "oic.sec.cred.cert",
5584             "oic.sec.cred.rolecert",
5585             "oic.sec.cred.mfgtrustca",
5586             "oic.sec.cred.mfgcert"
5587         ],
5588         "type": "string"
5589     },
5590     "crms": {
5591         "description": "The refresh methods that may be used to update this credential.",
5592         "items": {
5593             "description": "Each enum represents a method by which the credentials are
5594 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
5595 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
5596 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
5597 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
5598             "enum": [
5599                 "oic.sec.crm.pro",
5600                 "oic.sec.crm.psk",
5601                 "oic.sec.crm.rdp",
5602                 "oic.sec.crm.skdc",
5603                 "oic.sec.crm.pk10"
5604             ],
5605             "type": "string"
5606         },
5607         "type": "array"
5608     },
5609     "optionaldata": {
5610         "description": "Credential revocation status information\nOptional credential
5611 contents describes revocation status for this credential.",
5612         "properties": {
5613             "data": {
5614                 "description": "The encoded structure.",
5615                 "type": "string"
5616             },
5617             "encoding": {
5618                 "description": "A string specifying the encoding format of the data contained in
5619 the optdata.",
5620                 "x-detail-desc": [
5621                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
5622                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
5623                     "oic.sec.encoding.base64 - Base64 encoded object.",
5624                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
5625                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
5626                     "oic.sec.encoding.raw - Raw hex encoded data."
5627                 ],
5628                 "enum": [
5629                     "oic.sec.encoding.jwt",
5630                     "oic.sec.encoding.cwt",
5631                     "oic.sec.encoding.base64",
5632                     "oic.sec.encoding.pem",
5633                     "oic.sec.encoding.der",
5634                     "oic.sec.encoding.raw"
5635                 ],
5636                 "type": "string"
5637             },
5638             "revstat": {
5639                 "description": "Revocation status flag - true = revoked.",
5640                 "type": "boolean"
5641             }
5642         },
5643         "required": [

```

```

5644         "revstat"
5645     ],
5646     "type" : "object"
5647 },
5648 "period": {
5649     "description": "String with RFC5545 Period.",
5650     "type": "string"
5651 },
5652 "privatedata": {
5653     "description": "Private credential information\nCredential Resource non-public
5654 contents.",
5655     "properties": {
5656         "data": {
5657             "description": "The encoded value.",
5658             "maxLength": 3072,
5659             "type": "string"
5660         },
5661         "encoding": {
5662             "description": "A string specifying the encoding format of the data contained in
5663 the privdata.",
5664             "x-detail-desc": [
5665                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
5666                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
5667                 "oic.sec.encoding.base64 - Base64 encoded object.",
5668                 "oic.sec.encoding.uri - URI reference.",
5669                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
5670 referenced using a handle.",
5671                 "oic.sec.encoding.raw - Raw hex encoded data."
5672             ],
5673             "enum": [
5674                 "oic.sec.encoding.jwt",
5675                 "oic.sec.encoding.cwt",
5676                 "oic.sec.encoding.base64",
5677                 "oic.sec.encoding.uri",
5678                 "oic.sec.encoding.handle",
5679                 "oic.sec.encoding.raw"
5680             ],
5681             "type": "string"
5682         },
5683         "handle": {
5684             "description": "Handle to a key storage Resource.",
5685             "type": "integer"
5686         }
5687     },
5688     "required": [
5689         "encoding"
5690     ],
5691     "type": "object"
5692 },
5693 "publicdata": {
5694     "properties": {
5695         "data": {
5696             "description": "The encoded value.",
5697             "maxLength": 3072,
5698             "type": "string"
5699         },
5700         "encoding": {
5701             "description": "Public credential information\nA string specifying the encoding
5702 format of the data contained in the pubdata.",
5703             "x-detail-desc": [
5704                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
5705                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
5706                 "oic.sec.encoding.base64 - Base64 encoded object.",
5707                 "oic.sec.encoding.uri - URI reference.",
5708                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
5709                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
5710                 "oic.sec.encoding.raw - Raw hex encoded data."
5711             ],
5712             "enum": [
5713                 "oic.sec.encoding.jwt",
5714                 "oic.sec.encoding.cwt",

```

```

5715         "oic.sec.encoding.base64",
5716         "oic.sec.encoding.uri",
5717         "oic.sec.encoding.pem",
5718         "oic.sec.encoding.der",
5719         "oic.sec.encoding.raw"
5720     ],
5721     "type": "string"
5722 }
5723 },
5724 "type": "object"
5725 },
5726 "roleid": {
5727     "description": "The role this credential possesses\nSecurity role specified as an
5728 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
5729     "properties": {
5730         "authority": {
5731             "description": "The Authority component of the entity being identified. A NULL
5732 <Authority> refers to the local entity or Device.",
5733             "type": "string"
5734         },
5735         "role": {
5736             "description": "The ID of the role being identified.",
5737             "type": "string"
5738         }
5739     },
5740     "required": [
5741         "role"
5742     ],
5743     "type": "object"
5744 },
5745 "subjectuuid": {
5746     "anyOf": [
5747         {
5748             "description": "The id of the Device, which the cred entry applies to or \"*\n
5749 for wildcard identity.",
5750             "pattern": "^\\*$",
5751             "type": "string"
5752         },
5753         {
5754             "description": "Format pattern according to IETF RFC 4122.",
5755             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
5756 F0-9]{12}$",
5757             "type": "string"
5758         }
5759     ]
5760 }
5761 },
5762 "type": "object"
5763 },
5764 "type": "array"
5765 },
5766 "if" :
5767 {
5768     "description": "The interface set supported by this Resource.",
5769     "items": {
5770         "enum": [
5771             "oic.if.baseline"
5772         ],
5773         "type": "string"
5774     },
5775     "minItems": 1,
5776     "readOnly": true,
5777     "type": "array"
5778 }
5779 },
5780 "type" : "object"
5781 }
5782 }
5783 }
5784

```

5785 **C.3.5 Property definition**

5786 Table C-3 defines the Properties that are part of the "oic.r.cred" Resource Type.

5787 **Table C-3 – The Property definitions of the Resource with type "rt" = "oic.r.cred".**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
creds	array: see schema	Yes	Read Write	List of credentials available at this Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rowneruuid	string	No	Read Write	Format pattern according to IETF RFC 4122.
creds	array: see schema	No	Read Write	List of credentials available at this Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

5788 **C.3.6 CRUDN behaviour**

5789 Table C-4 defines the CRUDN operations that are supported on the "oic.r.cred" Resource Type.

5790 **Table C-4 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5791 **C.4 Certificate Revocation**

5792 **C.4.1 Introduction**

5793 This Resource specifies certificate revocation lists as X.509 objects.

5794

5795 **C.4.2 Well-known URI**

5796 /oic/sec/crl

5797 **C.4.3 Resource type**

5798 The Resource Type is defined as: "oic.r.crl".

5799 C.4.4 OpenAPI 2.0 definition

```
5800 {
5801   "swagger": "2.0",
5802   "info": {
5803     "title": "Certificate Revocation",
5804     "version": "v1.0-20150819",
5805     "license": {
5806       "name": "OCF Data Model License",
5807       "url":
5808         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5809         CENSE.md",
5810       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5811         reserved."
5812     },
5813     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5814   },
5815   "schemes": ["http"],
5816   "consumes": ["application/json"],
5817   "produces": ["application/json"],
5818   "paths": {
5819     "/oic/sec/crl" : {
5820       "get": {
5821         "description": "This Resource specifies certificate revocation lists as X.509 objects.\n",
5822         "parameters": [
5823           {"$ref": "#/parameters/interface"}
5824         ],
5825         "responses": {
5826           "200": {
5827             "description": "",
5828             "x-example":
5829               {
5830                 "rt": ["oic.r.crl"],
5831                 "crlid": 1,
5832                 "thisupdate": "2016-04-12T23:20:50.52Z",
5833                 "crldata": "Base64ENCODEDCRL"
5834               },
5835             "schema": { "$ref": "#/definitions/Crl" }
5836           }
5837         }
5838       },
5839       "post": {
5840         "description": "Updates the CRL data.\n",
5841         "parameters": [
5842           {"$ref": "#/parameters/interface"},
5843           {
5844             "name": "body",
5845             "in": "body",
5846             "required": true,
5847             "schema": { "$ref": "#/definitions/Crl-Update" },
5848             "x-example":
5849               {
5850                 "crlid": 1,
5851                 "thisupdate": "2016-04-12T23:20:50.52Z",
5852                 "crldata": "Base64ENCODEDCRL"
5853               }
5854           }
5855         ],
5856         "responses": {
5857           "400": {
5858             "description": "The request is invalid."
5859           },
5860           "204": {
5861             "description": "The CRL entry is updated."
5862           }
5863         }
5864       }
5865     }
5866   },
5867   "parameters": {
5868     "interface" : {
```

```

5869     "in" : "query",
5870     "name" : "if",
5871     "type" : "string",
5872     "enum" : ["oic.if.baseline"]
5873   }
5874 },
5875 "definitions": {
5876   "Crl" : {
5877     "properties": {
5878       "rt" : {
5879         "description": "Resource Type of the Resource.",
5880         "items": {
5881           "maxLength": 64,
5882           "type": "string",
5883           "enum": ["oic.r.crl"]
5884         },
5885         "minItems": 1,
5886         "readOnly": true,
5887         "type": "array"
5888       },
5889       "n": {
5890         "$ref":
5891 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5892 schema.json#/definitions/n"
5893       },
5894       "id": {
5895         "$ref":
5896 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5897 schema.json#/definitions/id"
5898       },
5899       "crldata" : {
5900         "description": "Base64 BER encoded CRL data.",
5901         "type": "string"
5902       },
5903       "crlid" : {
5904         "description": "Local reference to a CRL Resource.",
5905         "type": "integer"
5906       },
5907       "thisupdate" : {
5908         "description": "UTC time of last CRL update.",
5909         "type": "string"
5910       },
5911       "if" : {
5912         "description": "The interface set supported by this Resource.",
5913         "items": {
5914           "enum": [
5915             "oic.if.baseline"
5916           ],
5917           "type": "string"
5918         },
5919         "minItems": 1,
5920         "readOnly": true,
5921         "type": "array"
5922       }
5923     },
5924     "type": "object",
5925     "required": ["crlid", "thisupdate", "crldata"]
5926   }
5927 },
5928 "Crl-Update": {
5929   "properties": {
5930     "crldata": {
5931       "description": "Base64 BER encoded CRL data.",
5932       "type": "string"
5933     },
5934     "crlid": {
5935       "description": "Local reference to a CRL Resource.",
5936       "type": "integer"
5937     },
5938     "thisupdate": {
5939       "description": "UTC time of last CRL update.",

```

```

5940         "type": "string"
5941     },
5942 },
5943     "type" : "object"
5944 }
5945 }
5946 }
5947

```

5948 C.4.5 Property definition

5949 Table C-5 defines the Properties that are part of the "oic.r.crl" Resource Type.

5950 **Table C-5 – The Property definitions of the Resource with type "rt" = "oic.r.crl".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
crldata	string	Yes	Read Write	Base64 BER encoded CRL data.
crlid	integer	Yes	Read Write	Local reference to a CRL Resource.
thisupdate	string	Yes	Read Write	UTC time of last CRL update.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
crldata	string		Read Write	Base64 BER encoded CRL data.
crlid	integer		Read Write	Local reference to a CRL Resource.
thisupdate	string		Read Write	UTC time of last CRL update.

5951 C.4.6 CRUDN behaviour

5952 Table C-6 defines the CRUDN operations that are supported on the "oic.r.crl" Resource Type.

5953 **Table C-6 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".**

Create	Read	Update	Delete	Notify
	get	post		observe

5954 C.5 Certificate Signing Request

5955 C.5.1 Introduction

5956 This Resource specifies a Certificate Signing Request.

5957

5958 C.5.2 Well-known URI

5959 /oic/sec/csr

5960 C.5.3 Resource type

5961 The Resource Type is defined as: "oic.r.csr".

5962 C.5.4 OpenAPI 2.0 definition

```
5963 {
5964   "swagger": "2.0",
5965   "info": {
5966     "title": "Certificate Signing Request",
5967     "version": "v1.0-20150819",
5968     "license": {
5969       "name": "OCF Data Model License",
5970       "url":
5971 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5972 CENSE.md",
5973     "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5974 reserved."
5975   },
5976   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5977 },
5978 "schemes": ["http"],
5979 "consumes": ["application/json"],
5980 "produces": ["application/json"],
5981 "paths": {
5982   "/oic/sec/csr" : {
5983     "get": {
5984       "description": "This Resource specifies a Certificate Signing Request.\n",
5985       "parameters": [
5986         {"$ref": "#/parameters/interface"}
5987       ],
5988       "responses": {
5989         "200": {
5990           "description": "",
5991           "x-example":
5992             {
5993               "rt": ["oic.r.csr"],
5994               "encoding" : "oic.sec.encoding.pem",
5995               "csr": "PEMENCODEDCSR"
5996             },
5997           "schema": { "$ref": "#/definitions/Csr" }
5998         },
5999         "404": {
6000           "description": "The Device does not support certificates and generating CSRs."
6001         },
6002         "503": {
6003           "description": "The Device is not yet ready to return a response. Try again later."
6004         }
6005       }
6006     }
6007   }
6008 },
6009 "parameters": {
6010   "interface" : {
6011     "in" : "query",
6012     "name" : "if",
6013     "type" : "string",
6014     "enum" : ["oic.if.baseline"]
6015   }
6016 },
6017 "definitions": {
6018   "Csr" : {
6019     "properties": {
6020       "rt" : {
6021         "description": "Resource Type of the Resource.",
6022         "items": {
6023           "maxLength": 64,
6024           "type": "string",
6025           "enum": ["oic.r.csr"]
6026         },
6027         "minItems": 1,
```



```

6028         "readOnly": true,
6029         "type": "array"
6030     },
6031     "encoding": {
6032         "description": "A string specifying the encoding format of the data contained in CSR.",
6033         "x-detail-desc": [
6034             "oic.sec.encoding.pem - Encoding for PEM encoded CSR.",
6035             "oic.sec.encoding.der - Encoding for DER encoded CSR."
6036         ],
6037         "enum": [
6038             "oic.sec.encoding.pem",
6039             "oic.sec.encoding.der"
6040         ],
6041         "readOnly": true,
6042         "type": "string"
6043     },
6044     "n": {
6045         "$ref":
6046         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6047         schema.json#/definitions/n"
6048     },
6049     "id": {
6050         "$ref":
6051         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6052         schema.json#/definitions/id"
6053     },
6054     "csr": {
6055         "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
6056         "maxLength": 3072,
6057         "readOnly": true,
6058         "type": "string"
6059     },
6060     "if": {
6061         "description": "The interface set supported by this Resource.",
6062         "items": {
6063             "enum": [
6064                 "oic.if.baseline"
6065             ],
6066             "type": "string"
6067         },
6068         "minItems": 1,
6069         "readOnly": true,
6070         "type": "array"
6071     }
6072 },
6073 "type" : "object",
6074 "required": ["csr", "encoding"]
6075 }
6076 }
6077 }
6078

```

6079 C.5.5 Property definition

6080 Table C-7 defines the Properties that are part of the "oic.r.csr" Resource Type.

6081 **Table C-7 – The Property definitions of the Resource with type "rt" = "oic.r.csr".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in CSR.

n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

6082 **C.5.6 CRUDN behaviour**

6083 Table C-8 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

6084 **Table C-8 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".**

Create	Read	Update	Delete	Notify
	get			observe

6085 **C.6 Device Owner Transfer Method**

6086 **C.6.1 Introduction**

6087 This Resource specifies properties needed to establish a Device owner.

6088

6089 **C.6.2 Well-known URI**

6090 /oic/sec/doxm

6091 **C.6.3 Resource type**

6092 The Resource Type is defined as: "oic.r.doxm".

6093 **C.6.4 OpenAPI 2.0 definition**

```

6094 {
6095   "swagger": "2.0",
6096   "info": {
6097     "title": "Device Owner Transfer Method",
6098     "version": "v1.0-20181001",
6099     "license": {
6100       "name": "OCF Data Model License",
6101       "url":
6102         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6103         CENSE.md",
6104       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6105         reserved."
6106     },
6107     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6108   },
6109   "schemes": ["http"],
6110   "consumes": ["application/json"],
6111   "produces": ["application/json"],
6112   "paths": {
6113     "/oic/sec/doxm" : {
6114       "get": {
6115         "description": "This Resource specifies properties needed to establish a Device owner.\n",
6116         "parameters": [
6117           {"$ref": "#/parameters/interface"}
6118         ],

```

```

6119     "responses": {
6120         "200": {
6121             "description": "",
6122             "x-example":
6123                 {
6124                     "rt": ["oic.r.doxm"],
6125                     "oxms": [ 0, 2, 3 ],
6126                     "oxmsel": 0,
6127                     "sct": 16,
6128                     "owned": true,
6129                     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
6130                     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6131                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6132                 }
6133             ,
6134             "schema": { "$ref": "#/definitions/Doxm" }
6135         },
6136         "400": {
6137             "description": "The request is invalid."
6138         }
6139     }
6140 },
6141 "post": {
6142     "description": "Updates the DOXM Resource data.\n",
6143     "parameters": [
6144         { "$ref": "#/parameters/interface" },
6145         {
6146             "name": "body",
6147             "in": "body",
6148             "required": true,
6149             "schema": { "$ref": "#/definitions/Doxm-Update" },
6150             "x-example":
6151                 {
6152                     "oxmsel": 0,
6153                     "owned": true,
6154                     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
6155                     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6156                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6157                 }
6158         }
6159     ],
6160     "responses": {
6161         "400": {
6162             "description": "The request is invalid."
6163         },
6164         "204": {
6165             "description": "The DOXM entry is updated."
6166         }
6167     }
6168 }
6169 },
6170 "parameters": {
6171     "interface": {
6172         "in": "query",
6173         "name": "if",
6174         "type": "string",
6175         "enum": ["oic.if.baseline"]
6176     }
6177 },
6178 "definitions": {
6179     "Doxm": {
6180         "properties": {
6181             "rowneruuid": {
6182                 "description": "Format pattern according to IETF RFC 4122.",
6183                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
6184                 "type": "string"
6185             },
6186             "oxms": {
6187                 "description": "List of supported owner transfer methods.",
6188             }
6189         }

```

```

6190         "items": {
6191             "description": "The Device owner transfer methods that may be selected at Device on-
6192 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
6193 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
6194 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
6195 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
6196 (deprecated).",
6197             "type": "integer"
6198         },
6199         "readOnly": true,
6200         "type": "array"
6201     },
6202     "devowneruuid": {
6203         "description": "Format pattern according to IETF RFC 4122.",
6204         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6205 9]{12}$",
6206         "type": "string"
6207     },
6208     "deviceuuid": {
6209         "description": "The uuid formatted identity of the Device\nFormat pattern according to
6210 IETF RFC 4122.",
6211         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6212 9]{12}$",
6213         "type": "string"
6214     },
6215     "owned": {
6216         "description": "Ownership status flag.",
6217         "type": "boolean"
6218     },
6219     "n": {
6220         "$ref":
6221 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6222 schema.json#/definitions/n"
6223     },
6224     "id": {
6225         "$ref":
6226 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6227 schema.json#/definitions/id"
6228     },
6229     "oxmsel": {
6230         "description": "The selected owner transfer method used during on-boarding\nThe Device
6231 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
6232 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
6233 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
6234 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
6235 method (oic.sec.doxm.dcap) (deprecated).",
6236         "type": "integer"
6237     },
6238     "sct": {
6239         "description": "Bitmask encoding of supported credential types\nCredential Types -
6240 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6241 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6242 password32 - Asymmetric encryption key.",
6243         "maximum": 63,
6244         "minimum": 0,
6245         "type": "integer",
6246         "readOnly": true
6247     },
6248     "rt": {
6249         "description": "Resource Type of the Resource.",
6250         "items": {
6251             "maxLength": 64,
6252             "type": "string",
6253             "enum": ["oic.r.doxm"]
6254         },
6255         "minItems": 1,
6256         "readOnly": true,
6257         "type": "array"
6258     },
6259     "if": {
6260         "description": "The interface set supported by this Resource.",

```

```

6261         "items": {
6262             "enum": [
6263                 "oic.if.baseline"
6264             ],
6265             "type": "string"
6266         },
6267         "minItems": 1,
6268         "readOnly": true,
6269         "type": "array"
6270     }
6271 },
6272 "type" : "object",
6273 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
6274 },
6275 "Doxm-Update" : {
6276     "properties": {
6277         "rowneruuid": {
6278             "description": "Format pattern according to IETF RFC 4122.",
6279             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6280 9]{12}$",
6281             "type": "string"
6282         },
6283         "devowneruuid": {
6284             "description": "Format pattern according to IETF RFC 4122.",
6285             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6286 9]{12}$",
6287             "type": "string"
6288         },
6289         "deviceuuid": {
6290             "description": "The uuid formatted identity of the Device\nFormat pattern according to
6291 IETF RFC 4122.",
6292             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6293 9]{12}$",
6294             "type": "string"
6295         },
6296         "owned": {
6297             "description": "Ownership status flag.",
6298             "type": "boolean"
6299         },
6300         "oxmsel": {
6301             "description": "The selected owner transfer method used during on-boarding\nThe Device
6302 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
6303 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
6304 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
6305 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
6306 method (oic.sec.doxm.dcap) (deprecated).",
6307             "type": "integer"
6308         }
6309     },
6310     "type" : "object"
6311 }
6312 }
6313 }
6314

```

6315 C.6.5 Property definition

6316 Table C-9 defines the Properties that are part of the "oic.r.doxm" Resource Type.

6317 **Table C-9 – The Property definitions of the Resource with type "rt" = "oic.r.doxm".**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
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.
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the Device Format pattern according to IETF RFC 4122.
owned	boolean	Yes	Read Write	Ownership status flag.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
oxmsel	integer	Yes	Read Write	The selected owner transfer method used during on-boarding. The Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).
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.

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.
rowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
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.
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).

6318 **C.6.6 CRUDN behaviour**

6319 Table C-10 defines the CRUDN operations that are supported on the "oic.r.doxm" Resource Type.

6320 **Table C-10 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".**

Create	Read	Update	Delete	Notify
	get	post		observe

6321 **C.7 Device Provisioning Status**

6322 **C.7.1 Introduction**

6323 This Resource specifies Device provisioning status.

6324

6325 **C.7.2 Well-known URI**

6326 /oic/sec/pstat

6327 **C.7.3 Resource type**

6328 The Resource Type is defined as: "oic.r.pstat".

6329 **C.7.4 OpenAPI 2.0 definition**

```
6330 {
6331   "swagger": "2.0",
6332   "info": {
6333     "title": "Device Provisioning Status",
6334     "version": "v1.0-20191001",
6335     "license": {
6336       "name": "OCF Data Model License",
6337       "url":
6338         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6339         CENSE.md",
6340       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6341         reserved."
6342     },
6343     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6344   },
6345   "schemes": ["http"],
6346   "consumes": ["application/json"],
6347   "produces": ["application/json"],
6348   "paths": {
6349     "/oic/sec/pstat" : {
6350       "get": {
6351         "description": "This Resource specifies Device provisioning status.\n",
6352         "parameters": [
6353           {"$ref": "#/parameters/interface"}
6354         ],
6355         "responses": {
6356           "200": {
6357             "description": "",
6358             "x-example":
6359               {
6360                 "rt": ["oic.r.pstat"],
6361                 "dos": {"s": 3, "p": true},
6362                 "isop": true,
6363                 "cm": 8,
6364                 "tm": 60,
6365                 "om": 2,
6366                 "sm": 7,
6367                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
6368               },
6369             "schema": { "$ref": "#/definitions/Pstat" }
6370           },
6371           "400": {
6372             "description": "The request is invalid."
6373           }
6374         }
6375       },
6376       "post": {
6377         "description": "Sets or updates Device provisioning status data.\n",
6378         "parameters": [
6379           {"$ref": "#/parameters/interface"},
6380           {
6381             "name": "body",
6382             "in": "body",
```



```

6383         "required": true,
6384         "schema": { "$ref": "#/definitions/Pstat-Update" },
6385         "x-example":
6386             {
6387                 "dos": {"s": 3},
6388                 "tm": 60,
6389                 "om": 2,
6390                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
6391             }
6392     },
6393     ],
6394     "responses": {
6395         "400": {
6396             "description": "The request is invalid."
6397         },
6398         "204": {
6399             "description": "The PSTAT entry is updated."
6400         }
6401     }
6402 },
6403 },
6404 },
6405 "parameters": {
6406     "interface": {
6407         "in": "query",
6408         "name": "if",
6409         "type": "string",
6410         "enum": ["oic.if.baseline"]
6411     }
6412 },
6413 "definitions": {
6414     "Pstat": {
6415         "properties": {
6416             "rowneruuid": {
6417                 "description": "The UUID formatted identity of the Resource owner\nFormat pattern
6418 according to IETF RFC 4122.",
6419                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6420 9]{12}$",
6421                 "type": "string"
6422             },
6423             "rt": {
6424                 "description": "Resource Type of the Resource.",
6425                 "items": {
6426                     "maxLength": 64,
6427                     "type": "string",
6428                     "enum": ["oic.r.pstat"]
6429                 },
6430                 "minItems": 1,
6431                 "readOnly": true,
6432                 "type": "array"
6433             },
6434             "om": {
6435                 "description": "Current operational mode\nDevice provisioning operation may be server
6436 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
6437 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
6438 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
6439 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
6440                 "maximum": 7,
6441                 "minimum": 1,
6442                 "type": "integer"
6443             },
6444             "cm": {
6445                 "description": "Current Device provisioning mode\nDevice provisioning mode maintains a
6446 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
6447 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
6448 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
6449 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
6450 Software Version Validation128 - Initiate Secure Software Update.",
6451                 "maximum": 255,
6452                 "minimum": 0,
6453                 "type": "integer",

```

```

6454         "readOnly": true
6455     },
6456     "n": {
6457         "$ref":
6458         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6459         schema.json#/definitions/n"
6460     },
6461     "id": {
6462         "$ref":
6463         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6464         schema.json#/definitions/id"
6465     },
6466     "isop": {
6467         "description": "true indicates Device is operational.",
6468         "readOnly": true,
6469         "type": "boolean"
6470     },
6471     "tm": {
6472         "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
6473         bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
6474         in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
6475         - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
6476         services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
6477         Software Version Validation128 - Initiate Secure Software Update.",
6478         "maximum": 255,
6479         "minimum": 0,
6480         "type": "integer"
6481     },
6482     "sm": {
6483         "description": "Supported operational modes\nDevice provisioning operation may be server
6484         directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
6485         and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
6486         services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
6487         - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
6488         "maximum": 7,
6489         "minimum": 1,
6490         "type": "integer",
6491         "readOnly": true
6492     },
6493     "dos": {
6494         "description": "Device on-boarding state\nDevice operation state machine.",
6495         "properties": {
6496             "p": {
6497                 "default": true,
6498                 "description": "'p' is TRUE when the 's' state is pending until all necessary changes
6499                 to Device Resources are complete.",
6500                 "readOnly": true,
6501                 "type": "boolean"
6502             },
6503             "s": {
6504                 "description": "The current or pending operational state.",
6505                 "x-detail-desc": [
6506                     "0 - RESET - Device reset state.",
6507                     "1 - RFOTM - Ready for Device owner transfer method state.",
6508                     "2 - RFPPO - Ready for Device provisioning state.",
6509                     "3 - RFNOP - Ready for Device normal operation state.",
6510                     "4 - SRESET - The Device is in a soft reset state."
6511                 ],
6512                 "maximum": 4,
6513                 "minimum": 0,
6514                 "type": "integer"
6515             }
6516         },
6517         "required": [
6518             "s"
6519         ],
6520         "type": "object"
6521     },
6522     "if" : {
6523         "description": "The interface set supported by this Resource.",
6524         "items": {

```

```

6525         "enum": [
6526             "oic.if.baseline"
6527         ],
6528         "type": "string"
6529     },
6530     "minItems": 1,
6531     "readOnly": true,
6532     "type": "array"
6533 }
6534 },
6535 "type": "object",
6536 "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
6537 },
6538 "Pstat-Update" : {
6539     "properties": {
6540         "rowneruuid": {
6541             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
6542 according to IETF RFC 4122.",
6543             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6544 9]{12}$",
6545             "type": "string"
6546         },
6547         "om": {
6548             "description": "Current operational mode\nDevice provisioning operation may be server
6549 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
6550 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
6551 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
6552 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
6553             "maximum": 7,
6554             "minimum": 1,
6555             "type": "integer"
6556         },
6557         "tm": {
6558             "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
6559 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
6560 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
6561 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
6562 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
6563 Software Version Validation128 - Initiate Secure Software Update.",
6564             "maximum": 255,
6565             "minimum": 0,
6566             "type": "integer"
6567         },
6568         "dos": {
6569             "description": "Device on-boarding state\nDevice operation state machine.",
6570             "properties": {
6571                 "p": {
6572                     "default": true,
6573                     "description": "'p' is TRUE when the 's' state is pending until all necessary changes
6574 to Device Resources are complete.",
6575                     "readOnly": true,
6576                     "type": "boolean"
6577                 },
6578                 "s": {
6579                     "description": "The current or pending operational state.",
6580                     "x-detail-desc": [
6581                         "0 - RESET - Device reset state.",
6582                         "1 - RFOTM - Ready for Device owner transfer method state.",
6583                         "2 - RFPPO - Ready for Device provisioning state.",
6584                         "3 - RFNOP - Ready for Device normal operation state.",
6585                         "4 - SRESET - The Device is in a soft reset state."
6586                     ],
6587                     "maximum": 4,
6588                     "minimum": 0,
6589                     "type": "integer"
6590                 }
6591             },
6592             "required": [
6593                 "s"
6594             ],
6595             "type": "object"

```

```

6596     }
6597     },
6598     "type" : "object"
6599   }
6600 }
6601 }
6602

```

6603 **C.7.5 Property definition**

6604 Table C-11 defines the Properties that are part of the "oic.r.pstat" Resource Type.

6605 **Table C-11 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner. Format pattern according to IETF RFC 4122.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
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 modes: 1 - Server-directed utilizing multiple provisioning services 2 - Server-directed utilizing a single provisioning service 4 - Client-directed provisioning 8 - Unused 16 - Unused 32 - Unused 64 - Unused 128 - Unused.
cm	integer	Yes	Read Only	Current Device provisioning

				<p>mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.</p>
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
isop	boolean	Yes	Read Only	true indicates Device is operational.
tm	integer	Yes	Read Write	Target Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a

				<p>Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.</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-</p>

				directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
dos	object: see schema	Yes	Read Write	Device on-boarding state machine. Device operation state machine.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rowneruuid	string	No	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.
om	integer	No	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes 1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
tm	integer	No	Read Write	Target Device provisioning mode

				Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
dos	object: schema see	No	Read Write	Device on-boarding state Device operation state machine.

6606 **C.7.6 CRUDN behaviour**

6607 Table C-12 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

6608 **Table C-12 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".**

Create	Read	Update	Delete	Notify
	get	post		observe

6609 **C.8 Asserted Roles**

6610 **C.8.1 Introduction**

6611 This Resource specifies roles that have been asserted.

6612

6613 **C.8.2 Well-known URI**

6614 /oic/sec/roles

6615 **C.8.3 Resource type**

6616 The Resource Type is defined as: "oic.r.roles".

6617 **C.8.4 OpenAPI 2.0 definition**

```
6618 {
6619   "swagger": "2.0",
6620   "info": {
6621     "title": "Asserted Roles",
6622     "version": "v1.0-20170323",
6623     "license": {
6624       "name": "OCF Data Model License",
6625       "url":
6626         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6627         CENSE.md",
6628       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6629         reserved."
6630     },
6631     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6632   },
6633   "schemes": ["http"],
6634   "consumes": ["application/json"],
6635   "produces": ["application/json"],
6636   "paths": {
6637     "/oic/sec/roles" : {
6638       "get": {
6639         "description": "This Resource specifies roles that have been asserted.\n",
6640         "parameters": [
6641           {"$ref": "#/parameters/interface"}
6642         ],
6643         "responses": {
6644           "200": {
6645             "description": "",
6646             "x-example":
6647               {
6648                 "roles" :[
6649                   {
6650                     "credid":1,
6651                     "credtype":8,
6652                     "subjectuuid": "00000000-0000-0000-0000-000000000000",
6653                     "publicdata":
6654                       {
6655                         "encoding": "oic.sec.encoding.pem",
6656                         "data": "PEMENCODEDROLECERT"
6657                       },
6658                     "optionaldata":
6659                       {
6660                         "revstat": false,
6661                         "encoding": "oic.sec.encoding.pem",
6662                         "data": "PEMENCODEDISSUERCERT"
6663                       }
6664                   },
6665                   {
6666                     "credid":2,
6667                     "credtype":8,
6668                     "subjectuuid": "00000000-0000-0000-0000-000000000000",
6669                     "publicdata":
6670                       {
6671                         "encoding": "oic.sec.encoding.pem",
6672                         "data": "PEMENCODEDROLECERT"
6673                       },
6674                     "optionaldata":
6675                       {
6676                         "revstat": false,
6677                         "encoding": "oic.sec.encoding.pem",
6678                         "data": "PEMENCODEDISSUERCERT"

```

```

6679         }
6680     },
6681     ],
6682     "rt":["oic.r.roles"],
6683     "if":["oic.if.baseline"]
6684 }
6685 },
6686     "schema": { "$ref": "#/definitions/Roles" }
6687 },
6688     "400": {
6689         "description" : "The request is invalid."
6690     }
6691 }
6692 ],
6693 "post": {
6694     "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
6695 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
6696 same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
6697 ignored, the Resource assigns its own.\n",
6698     "parameters": [
6699         { "$ref": "#/parameters/interface",
6700         {
6701             "name": "body",
6702             "in": "body",
6703             "required": true,
6704             "schema": { "$ref": "#/definitions/Roles-update" },
6705             "x-example":
6706             {
6707                 "roles" :[
6708                     {
6709                         "credid":1,
6710                         "credtype":8,
6711                         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6712                         "publicdata":
6713                         {
6714                             "encoding":"oic.sec.encoding.pem",
6715                             "data":"PEMENCODEDROLECERT"
6716                         },
6717                         "optionaldata":
6718                         {
6719                             "revstat": false,
6720                             "encoding":"oic.sec.encoding.pem",
6721                             "data":"PEMENCODEDISSUERCERT"
6722                         }
6723                     },
6724                     {
6725                         "credid":2,
6726                         "credtype":8,
6727                         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6728                         "publicdata":
6729                         {
6730                             "encoding":"oic.sec.encoding.pem",
6731                             "data":"PEMENCODEDROLECERT"
6732                         },
6733                         "optionaldata":
6734                         {
6735                             "revstat": false,
6736                             "encoding":"oic.sec.encoding.pem",
6737                             "data":"PEMENCODEDISSUERCERT"
6738                         }
6739                     }
6740                 ]
6741             }
6742         }
6743     ],
6744     "responses": {
6745         "400": {
6746             "description" : "The request is invalid."
6747         },
6748         "204": {
6749             "description" : "The roles entry is updated."

```

```

6750     }
6751   },
6752 },
6753 "delete": {
6754   "description": "Deletes roles Resource entries.\nWhen DELETE is used without query
6755 parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
6756 entries matching\nthe query parameter are deleted.\n",
6757   "parameters": [
6758     {"$ref": "#/parameters/interface"},
6759     {"$ref": "#/parameters/roles-filtered"}
6760   ],
6761   "responses": {
6762     "200": {
6763       "description": "The specified or all roles Resource entries have been successfully
6764 deleted."
6765     },
6766     "400": {
6767       "description": "The request is invalid."
6768     }
6769   }
6770 }
6771 },
6772 },
6773 "parameters": {
6774   "interface" : {
6775     "in" : "query",
6776     "name" : "if",
6777     "type" : "string",
6778     "enum" : ["oic.if.baseline"]
6779   },
6780   "roles-filtered" : {
6781     "in" : "query",
6782     "name" : "credid",
6783     "required" : false,
6784     "type" : "integer",
6785     "description" : "Only applies to the credential with the specified credid.",
6786     "x-example" : 2112
6787   }
6788 },
6789 "definitions": {
6790   "Roles" : {
6791     "properties": {
6792       "rt": {
6793         "description": "Resource Type of the Resource.",
6794         "items": {
6795           "maxLength": 64,
6796           "type": "string",
6797           "enum": ["oic.r.roles"]
6798         },
6799         "minItems": 1,
6800         "readOnly": true,
6801         "type": "array"
6802       },
6803       "n": {
6804         "$ref":
6805 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6806 schema.json#/definitions/n"
6807       },
6808       "id": {
6809         "$ref":
6810 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6811 schema.json#/definitions/id"
6812       },
6813       "roles": {
6814         "description": "List of role certificates.",
6815         "items": {
6816           "properties": {
6817             "credid": {
6818               "description": "Local reference to a credential Resource.",
6819               "type": "integer"
6820             }

```

```

6821         "credtype": {
6822             "description": "Representation of this credential's type\nCredential Types - Cred
6823 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6824 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6825 password32 - Asymmetric encryption key.",
6826             "maximum": 63,
6827             "minimum": 0,
6828             "type": "integer"
6829         },
6830         "credusage": {
6831             "description": "A string that provides hints about how/where the cred is used\nThe
6832 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6833 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6834 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6835             "enum": [
6836                 "oic.sec.cred.trustca",
6837                 "oic.sec.cred.cert",
6838                 "oic.sec.cred.rolecert",
6839                 "oic.sec.cred.mfgtrustca",
6840                 "oic.sec.cred.mfgcert"
6841             ],
6842             "type": "string"
6843         },
6844         "crms": {
6845             "description": "The refresh methods that may be used to update this credential.",
6846             "items": {
6847                 "description": "Each enum represents a method by which the credentials are
6848 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6849 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6850 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6851 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6852                 "enum": [
6853                     "oic.sec.crm.pro",
6854                     "oic.sec.crm.psk",
6855                     "oic.sec.crm.rdp",
6856                     "oic.sec.crm.skdc",
6857                     "oic.sec.crm.pk10"
6858                 ],
6859                 "type": "string"
6860             },
6861             "type": "array"
6862         },
6863         "optionaldata": {
6864             "description": "Credential revocation status information\nOptional credential
6865 contents describes revocation status for this credential.",
6866             "properties": {
6867                 "data": {
6868                     "description": "This is the encoded structure.",
6869                     "type": "string"
6870                 },
6871                 "encoding": {
6872                     "description": "A string specifying the encoding format of the data contained in
6873 the optdata.",
6874                     "x-detail-desc": [
6875                         "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6876                         "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6877                         "oic.sec.encoding.base64 - Base64 encoded object.",
6878                         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6879                         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6880                         "oic.sec.encoding.raw - Raw hex encoded data."
6881                     ],
6882                     "enum": [
6883                         "oic.sec.encoding.jwt",
6884                         "oic.sec.encoding.cwt",
6885                         "oic.sec.encoding.base64",
6886                         "oic.sec.encoding.pem",
6887                         "oic.sec.encoding.der",
6888                         "oic.sec.encoding.raw"
6889                     ],
6890                     "type": "string"
6891                 },

```

```

6892         "revstat": {
6893             "description": "Revocation status flag - true = revoked.",
6894             "type": "boolean"
6895         }
6896     },
6897     "required": [
6898         "revstat"
6899     ],
6900     "type": "object"
6901 },
6902 "period": {
6903     "description": "String with RFC5545 Period.",
6904     "type": "string"
6905 },
6906 "privatedata": {
6907     "description": "Private credential information\nCredential Resource non-public
6908 contents.",
6909     "properties": {
6910         "data": {
6911             "description": "The encoded value.",
6912             "maxLength": 3072,
6913             "type": "string"
6914         },
6915         "encoding": {
6916             "description": "A string specifying the encoding format of the data contained in
6917 the privdata.",
6918             "x-detail-desc": [
6919                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6920                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6921                 "oic.sec.encoding.base64 - Base64 encoded object.",
6922                 "oic.sec.encoding.uri - URI reference.",
6923                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
6924 referenced using a handle.",
6925                 "oic.sec.encoding.raw - Raw hex encoded data."
6926             ],
6927             "enum": [
6928                 "oic.sec.encoding.jwt",
6929                 "oic.sec.encoding.cwt",
6930                 "oic.sec.encoding.base64",
6931                 "oic.sec.encoding.uri",
6932                 "oic.sec.encoding.handle",
6933                 "oic.sec.encoding.raw"
6934             ],
6935             "type": "string"
6936         },
6937         "handle": {
6938             "description": "Handle to a key storage Resource.",
6939             "type": "integer"
6940         }
6941     },
6942     "required": [
6943         "encoding"
6944     ],
6945     "type": "object"
6946 },
6947 "publicdata": {
6948     "description": "Public credential information.",
6949     "properties": {
6950         "data": {
6951             "description": "This is the encoded value.",
6952             "maxLength": 3072,
6953             "type": "string"
6954         },
6955         "encoding": {
6956             "description": "A string specifying the encoding format of the data contained in
6957 the pubdata.",
6958             "x-detail-desc": [
6959                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6960                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6961                 "oic.sec.encoding.base64 - Base64 encoded object.",
6962                 "oic.sec.encoding.uri - URI reference."

```

```

6963         "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6964         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6965         "oic.sec.encoding.raw - Raw hex encoded data."
6966     ],
6967     "enum": [
6968         "oic.sec.encoding.jwt",
6969         "oic.sec.encoding.cwt",
6970         "oic.sec.encoding.base64",
6971         "oic.sec.encoding.uri",
6972         "oic.sec.encoding.pem",
6973         "oic.sec.encoding.der",
6974         "oic.sec.encoding.raw"
6975     ],
6976     "type": "string"
6977 },
6978 },
6979 "type": "object"
6980 },
6981 "roleid": {
6982     "description": "The role this credential possesses\nSecurity role specified as an
6983 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6984     "properties": {
6985         "authority": {
6986             "description": "The Authority component of the entity being identified. A NULL
6987 <Authority> refers to the local entity or Device.",
6988             "type": "string"
6989         },
6990         "role": {
6991             "description": "The ID of the role being identified.",
6992             "type": "string"
6993         }
6994     },
6995     "required": [
6996         "role"
6997     ],
6998     "type": "object"
6999 },
7000 "subjectuuid": {
7001     "anyOf": [
7002         {
7003             "description": "The id of the Device, which the cred entry applies to or \"*\n
7004 for wildcard identity.",
7005             "pattern": "^\\*$",
7006             "type": "string"
7007         },
7008         {
7009             "description": "Format pattern according to IETF RFC 4122.",
7010             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7011 F0-9]{12}$",
7012             "type": "string"
7013         }
7014     ]
7015 },
7016 },
7017 "type": "object"
7018 },
7019 "type": "array"
7020 },
7021 "if": {
7022     "description": "The interface set supported by this Resource.",
7023     "items": {
7024         "enum": [
7025             "oic.if.baseline"
7026         ],
7027         "type": "string"
7028     },
7029     "minItems": 1,
7030     "readOnly": true,
7031     "type": "array"
7032 },
7033 },

```

```

7034     "type" : "object",
7035     "required": ["roles"]
7036 },
7037 "Roles-update" : {
7038     "properties": {
7039         "roles": {
7040             "description": "List of role certificates.",
7041             "items": {
7042                 "properties": {
7043                     "credid": {
7044                         "description": "Local reference to a credential Resource.",
7045                         "type": "integer"
7046                     },
7047                     "credtype": {
7048                         "description": "Representation of this credential's type\nCredential Types - Cred
7049 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7050 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7051 password32 - Asymmetric encryption key.",
7052                         "maximum": 63,
7053                         "minimum": 0,
7054                         "type": "integer"
7055                     },
7056                     "credusage": {
7057                         "description": "A string that provides hints about how/where the cred is used\nThe
7058 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7059 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7060 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7061                         "enum": [
7062                             "oic.sec.cred.trustca",
7063                             "oic.sec.cred.cert",
7064                             "oic.sec.cred.rolecert",
7065                             "oic.sec.cred.mfgtrustca",
7066                             "oic.sec.cred.mfgcert"
7067                         ],
7068                         "type": "string"
7069                     },
7070                     "crms": {
7071                         "description": "The refresh methods that may be used to update this credential.",
7072                         "items": {
7073                             "description": "Each enum represents a method by which the credentials are
7074 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7075 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7076 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7077 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7078                             "enum": [
7079                                 "oic.sec.crm.pro",
7080                                 "oic.sec.crm.psk",
7081                                 "oic.sec.crm.rdp",
7082                                 "oic.sec.crm.skdc",
7083                                 "oic.sec.crm.pk10"
7084                             ],
7085                             "type": "string"
7086                         },
7087                         "type": "array"
7088                     },
7089                     "optionaldata": {
7090                         "description": "Credential revocation status information\nOptional credential
7091 contents describes revocation status for this credential.",
7092                         "properties": {
7093                             "data": {
7094                                 "description": "This is the encoded structure.",
7095                                 "type": "string"
7096                             },
7097                             "encoding": {
7098                                 "description": "A string specifying the encoding format of the data contained in
7099 the optdata.",
7100                                 "x-detail-desc": [
7101                                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7102                                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7103                                     "oic.sec.encoding.base64 - Base64 encoded object.",
7104                                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",

```

```

7105         "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7106         "oic.sec.encoding.raw - Raw hex encoded data."
7107     ],
7108     "enum": [
7109         "oic.sec.encoding.jwt",
7110         "oic.sec.encoding.cwt",
7111         "oic.sec.encoding.base64",
7112         "oic.sec.encoding.pem",
7113         "oic.sec.encoding.der",
7114         "oic.sec.encoding.raw"
7115     ],
7116     "type": "string"
7117 },
7118 "revstat": {
7119     "description": "Revocation status flag - true = revoked.",
7120     "type": "boolean"
7121 }
7122 },
7123 "required": [
7124     "revstat"
7125 ],
7126 "type": "object"
7127 },
7128 "period": {
7129     "description": "String with RFC5545 Period.",
7130     "type": "string"
7131 },
7132 "privatedata": {
7133     "description": "Private credential information\nCredential Resource non-public
7134 contents.",
7135     "properties": {
7136         "data": {
7137             "description": "The encoded value.",
7138             "maxLength": 3072,
7139             "type": "string"
7140         },
7141         "encoding": {
7142             "description": "A string specifying the encoding format of the data contained in
7143 the privdata.",
7144             "x-detail-desc": [
7145                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7146                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7147                 "oic.sec.encoding.base64 - Base64 encoded object.",
7148                 "oic.sec.encoding.uri - URI reference.",
7149                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
7150 referenced using a handle.",
7151                 "oic.sec.encoding.raw - Raw hex encoded data."
7152             ],
7153             "enum": [
7154                 "oic.sec.encoding.jwt",
7155                 "oic.sec.encoding.cwt",
7156                 "oic.sec.encoding.base64",
7157                 "oic.sec.encoding.uri",
7158                 "oic.sec.encoding.handle",
7159                 "oic.sec.encoding.raw"
7160             ],
7161             "type": "string"
7162         },
7163         "handle": {
7164             "description": "Handle to a key storage Resource.",
7165             "type": "integer"
7166         }
7167     },
7168     "required": [
7169         "encoding"
7170     ],
7171     "type": "object"
7172 },
7173 "publicdata": {
7174     "description": "Public credential information.",
7175     "properties": {

```



```

7176         "data": {
7177             "description": "The encoded value.",
7178             "maxLength": 3072,
7179             "type": "string"
7180         },
7181         "encoding": {
7182             "description": "A string specifying the encoding format of the data contained in
7183 the pubdata.",
7184             "x-detail-desc": [
7185                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7186                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7187                 "oic.sec.encoding.base64 - Base64 encoded object.",
7188                 "oic.sec.encoding.uri - URI reference.",
7189                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7190                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7191                 "oic.sec.encoding.raw - Raw hex encoded data."
7192             ],
7193             "enum": [
7194                 "oic.sec.encoding.jwt",
7195                 "oic.sec.encoding.cwt",
7196                 "oic.sec.encoding.base64",
7197                 "oic.sec.encoding.uri",
7198                 "oic.sec.encoding.pem",
7199                 "oic.sec.encoding.der",
7200                 "oic.sec.encoding.raw"
7201             ],
7202             "type": "string"
7203         }
7204     },
7205     "type": "object"
7206 },
7207 "roleid": {
7208     "description": "The role this credential possesses\nSecurity role specified as an
7209 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
7210     "properties": {
7211         "authority": {
7212             "description": "The Authority component of the entity being identified. A NULL
7213 <Authority> refers to the local entity or Device.",
7214             "type": "string"
7215         },
7216         "role": {
7217             "description": "The ID of the role being identified.",
7218             "type": "string"
7219         }
7220     },
7221     "required": [
7222         "role"
7223     ],
7224     "type": "object"
7225 },
7226 "subjectuuid": {
7227     "anyOf": [
7228         {
7229             "description": "The id of the Device, which the cred entry applies to or \"*\n
7230 for wildcard identity.",
7231             "pattern": "^\\*$",
7232             "type": "string"
7233         },
7234         {
7235             "description": "Format pattern according to IETF RFC 4122.",
7236             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7237 F0-9]{12}$",
7238             "type": "string"
7239         }
7240     ]
7241 }
7242 },
7243 "type": "object"
7244 },
7245 "type": "array"
7246 }

```

```

7247     },
7248     "type" : "object",
7249     "required": ["roles"]
7250   }
7251 }
7252 }
7253

```

7254 C.8.5 Property definition

7255 Table C-13 defines the Properties that are part of the "oic.r.roles" Resource Type.

7256 **Table C-13 – The Property definitions of the Resource with type "rt" = "oic.r.roles".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
roles	array: see schema	Yes	Read Write	List of role certificates.
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.

7257 C.8.6 CRUDN behaviour

7258 Table C-14 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

7259 **Table C-14 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

7260 C.9 Security Profile

7261 C.9.1 Introduction

7262 Resource specifying supported and active security profile(s).

7263

7264 C.9.2 Well-known URI

7265 /oic/sec/sp

7266 C.9.3 Resource type

7267 The Resource Type is defined as: "oic.r.sp".

7268 C.9.4 OpenAPI 2.0 definition

```

7269 {
7270   "swagger": "2.0",
7271   "info": {
7272     "title": "Security Profile",
7273     "version": "v1.0-20190208",
7274     "license": {
7275       "name": "OCF Data Model License",
7276       "url":
7277         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7278         CENSE.md",

```

```

7279         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7280 reserved."
7281     },
7282     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7283 },
7284 "schemes": ["http"],
7285 "consumes": ["application/json"],
7286 "produces": ["application/json"],
7287 "paths": {
7288     "/oic/sec/sp" : {
7289         "get": {
7290             "description": "Resource specifying supported and active security profile(s).\n",
7291             "parameters": [
7292                 {"$ref": "#/parameters/interface"}
7293             ],
7294             "responses": {
7295                 "200": {
7296                     "description": "",
7297                     "x-example":
7298                         {
7299                             "rt": ["oic.r.sp"],
7300                             "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
7301                             "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
7302                         },
7303                     "schema": { "$ref": "#/definitions/SP" }
7304                 },
7305                 "400": {
7306                     "description": "The request is invalid."
7307                 }
7308             }
7309         },
7310         "post": {
7311             "description": "Sets or updates Device provisioning status data.\n",
7312             "parameters": [
7313                 {"$ref": "#/parameters/interface"},
7314                 {
7315                     "name": "body",
7316                     "in": "body",
7317                     "required": true,
7318                     "schema": { "$ref": "#/definitions/SP-Update" },
7319                     "x-example":
7320                         {
7321                             "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
7322                             "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
7323                         }
7324                 }
7325             ],
7326             "responses": {
7327                 "200": {
7328                     "description": "",
7329                     "x-example":
7330                         {
7331                             "rt": ["oic.r.sp"],
7332                             "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
7333                             "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
7334                         },
7335                     "schema": { "$ref": "#/definitions/SP" }
7336                 },
7337                 "400": {
7338                     "description": "The request is invalid."
7339                 }
7340             }
7341         }
7342     }
7343 },
7344 "parameters": {
7345     "interface" : {
7346         "in" : "query",
7347         "name" : "if",
7348         "type" : "string",
7349         "enum" : ["oic.if.baseline"]

```

```

7350     }
7351   },
7352   "definitions": {
7353     "SP" : {
7354       "properties": {
7355         "rt": {
7356           "description": "Resource Type of the Resource.",
7357           "items": {
7358             "maxLength": 64,
7359             "type": "string",
7360             "enum": ["oic.r.sp"]
7361           },
7362           "minItems": 1,
7363           "readOnly": true,
7364           "type": "array"
7365         },
7366         "n": {
7367           "$ref":
7368             "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7369             schema.json#/definitions/n"
7370         },
7371         "id": {
7372           "$ref":
7373             "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7374             schema.json#/definitions/id"
7375         },
7376         "currentprofile": {
7377           "description": "Security Profile currently active.",
7378           "type": "string"
7379         },
7380         "supportedprofiles": {
7381           "description": "Array of supported Security Profiles.",
7382           "items": {
7383             "type": "string"
7384           },
7385           "type": "array"
7386         },
7387         "if": {
7388           "description": "The interface set supported by this Resource.",
7389           "items": {
7390             "enum": [
7391               "oic.if.baseline"
7392             ],
7393             "type": "string"
7394           },
7395           "minItems": 1,
7396           "readOnly": true,
7397           "type": "array"
7398         }
7399       },
7400       "type": "object",
7401       "required": ["supportedprofiles", "currentprofile"]
7402     },
7403     "SP-Update" : {
7404       "properties": {
7405         "currentprofile": {
7406           "description": "Security Profile currently active.",
7407           "type": "string"
7408         },
7409         "supportedprofiles": {
7410           "description": "Array of supported Security Profiles.",
7411           "items": {
7412             "type": "string"
7413           },
7414           "type": "array"
7415         }
7416       },
7417       "type": "object"
7418     }
7419   }

```

7420 }
7421

7422 C.9.5 Property definition

7423 Table C-15 defines the Properties that are part of the "oic.r.sp" Resource Type.

7424 **Table C-15 – The Property definitions of the Resource with type "rt" = "oic.r.sp".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	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.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
currentprofile	string		Read Write	Security Profile currently active.
supportedprofiles	array: see schema		Read Write	Array of supported Security Profiles.

7425 C.9.6 CRUDN behaviour

7426 Table C-16 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

7427 **Table C-16 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".**

Create	Read	Update	Delete	Notify
	get	post		observe

7428

7429
7430
7431
7432

Annex D (informative)

OID definitions

7433 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to
7434 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of
7435 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
7436 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
7437     private(4) enterprise(1) OCF(51414) }
7438
7439 -- OCF Security specific OIDs
7440
7441 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
7442 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
7443
7444 -- OCF Security Categories
7445
7446 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
7447 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
7448
7449 -- OCF Security Profiles
7450
7451 sp-undefined ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
7452 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
7453 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
7454 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
7455 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
7456
7457 sp-undefined-v0 ::= ocfSecurityProfileOID (id-sp-undefined 0)
7458 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
7459 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
7460 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
7461 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
7462
7463 ocfSecurityProfileOID ::= UTF8String
7464
7465 -- OCF Security Certificate Policies
7466
7467 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
7468
7469 -- OCF X.509v3 Extensions
7470
7471 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
7472 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
7473 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
7474 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
7475
7476 ocfVersion ::= SEQUENCE {
7477     major    INTEGER,
7478     minor    INTEGER,
7479     build    INTEGER}
7480
7481 ocfCompliance ::= SEQUENCE {
7482     version          ocfVersion,
7483     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
7484     deviceName       UTF8String,
7485     deviceManufacturer UTF8String}
7486
7487 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
7488 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
7489
7490 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
7491
7492 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
7493
7494 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
7495 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
7496 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
7497
7498 ocfCPLAttributes ::= SEQUENCE {
7499     cpl-at-IANAPen UTF8String,
7500     cpl-at-model UTF8String,
7501     cpl-at-version UTF8String}
```

**Annex E
(informative)**

Security considerations specific to Bridged Protocols

7502
7503
7504
7505

7506 The text in this Annex is provided for information only. This Annex has no normative impact. This
7507 information is applicable at the time of initial publication and may become out of date.

7508 **E.1 Security Considerations specific to the AllJoyn Protocol**

7509 This clause intentionally left empty.

7510 **E.2 Security Considerations specific to the Bluetooth LE Protocol**

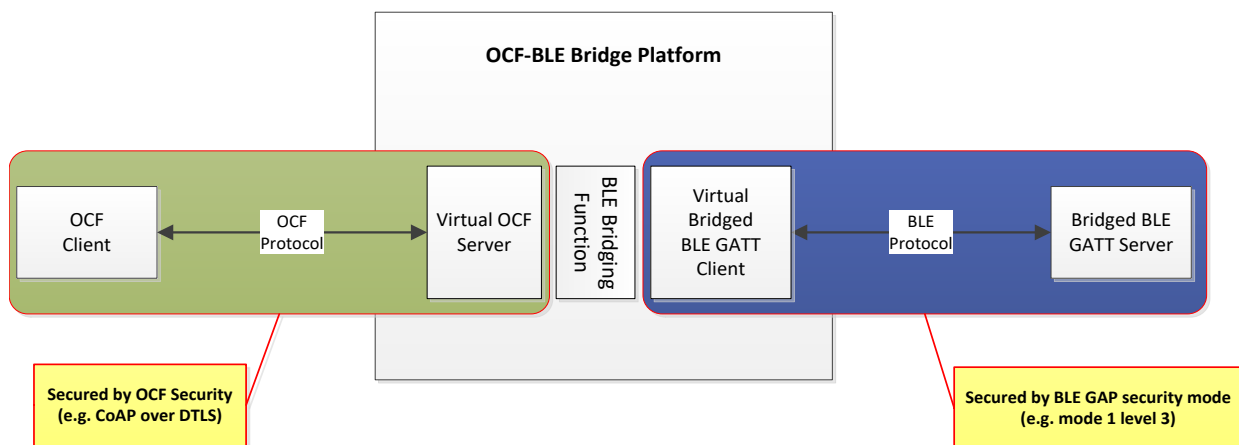
7511 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security mode
7512 has several security levels (see Table E.1)

7513 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF
7514 perspective. The appropriate selection of security mode and level is left to the vendor.

7515 **Table E.1 GAP security mode**

GAP security mode	security level
Security mode 1	1 (no security)
	2 (Unauthenticated pairing with encryption)
	3 (Authenticated pairing with encryption)
	4 (Authenticated LE Secure Connections pairing with encryption)
Security mode 2	1 (Unauthenticated pairing with data signing)
	2 (Authenticated pairing with data signing)

7516 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are
7517 secured by their own security.



7518

Figure E-1 Security Considerations for BLE Bridge

7519

7520 **E.3 Security Considerations specific to the oneM2M Protocol**

7521 This clause intentionally left empty.

7522 **E.4 Security Considerations specific to the U+ Protocol**

7523 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

7524 **Table E.2 TLS 1.2 Cipher Suites used by U+**

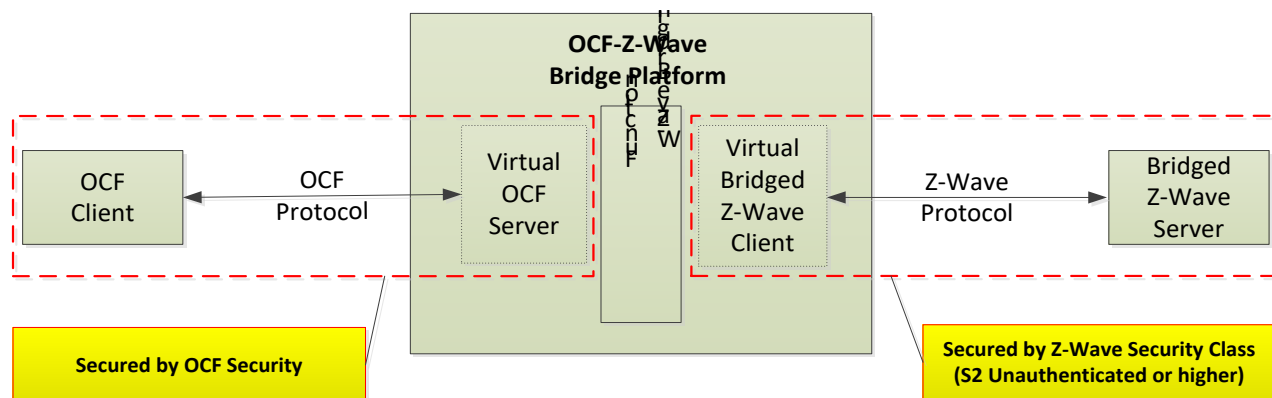
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

7525 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

7526 **E.5 Security Considerations specific to the Z-Wave Protocol**

7527 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2 Security
 7528 Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for communication
 7529 with a Virtual Bridged Client would typically be considered secure from an OCF perspective. The
 7530 appropriate selection for S2 Security Class and Class Name is left to the vendor.

7531 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their
 7532 own security.



7533

7534

Figure E-2 Security Considerations for Z-Wave Bridge

7535 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2
7536 Unauthenticated provides the following advantages from the security perspective;

- 7537 – The unique device specific key for every secure device enables validation of device identity and
7538 prevents man-in-the-middle compromises to security
- 7539 – The Secure cryptographic key exchange methods during inclusion achieves high level of
7540 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 7541 – Out of band key exchange for product authentication which is combined with device specific
7542 key prevents eavesdropping and man-in-the-middle attack vectors.

7543 See Table E.3 for a summary of Z-Wave Security Classes.

7544

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

7545 On the other hand, S0 Security Class has the vulnerability of security during inclusion by
7546 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the
7547 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices
7548 might be no longer secure in that case.

7549 E.6 Security Considerations specific to the Zigbee Protocol

7550 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the
7551 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0
7552 stack, "nwkSecurityLevel", represents the security level of a device.

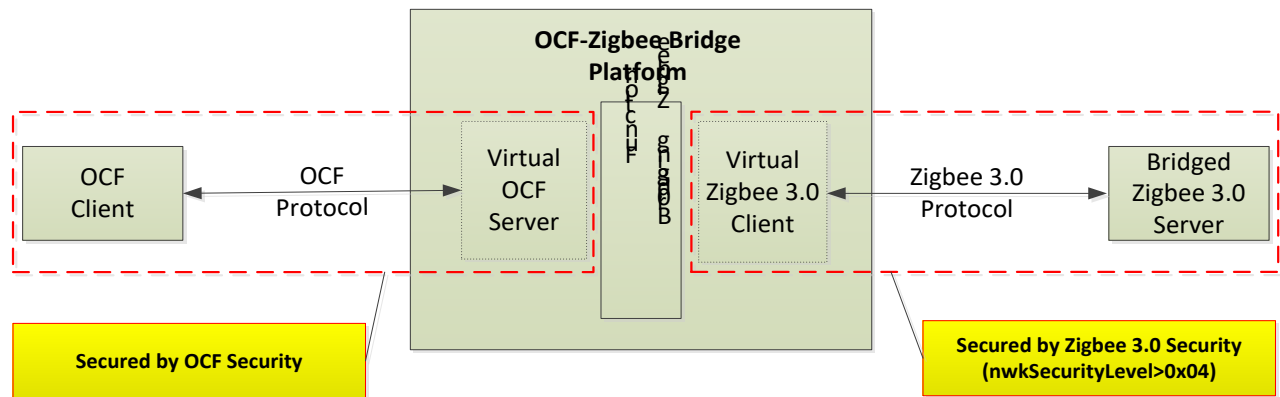
7553 The security level nwkSecurityLevel > 0x04 provides message integrity code (MIC) and/or AES128-
 7554 CCM encryption (ENC). Zigbee Servers using nwkSecurityLevel > 0x04 would typically be
 7555 considered secure from an OCF perspective. The appropriate selection for nwkSecurityLevel is left
 7556 to the vendor.

7557 See Table E.4 for a summary of the Zigbee Security Levels.

7558 **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)

7559 Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are
 7560 secured by their own security.



7561

7562

Figure E-3 Security Considerations for Zigbee Bridge