

OCF Security Specification

VERSION 1.0.0 | June 2017



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org
Copyright OCF © 2017. All Rights Reserved.

Legal Disclaimer

4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

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

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

Copyright © 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.

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

CONTENTS

22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

- 1 Scope 13
- 2 Normative References 13
- 3 Terms, Definitions, Symbols and Abbreviations 14
 - 3.1 Terms and definitions 14
 - 3.2 Symbols and Abbreviations 16
- 4 Document Conventions and Organization 16
 - 4.1 Introduction 16
 - 4.2 Conventions 17
 - 4.3 Notation..... 17
 - 4.4 Data types 18
 - 4.5 Document structure 18
- 5 Security Overview 19
 - 5.1 Introduction 19
 - 5.2 Access Control 21
 - 5.2.1 General 21
 - 5.2.2 ACL Architecture 22
 - 5.2.3 Access Control Scoping Levels 25
 - 5.3 Onboarding Overview 26
 - 5.3.1 General 26
 - 5.3.2 OnBoarding Steps 28
 - 5.3.3 Establishing a Device Owner 30
 - 5.3.4 Provisioning for Normal Operation 30
 - 5.4 Provisioning..... 31
 - 5.4.1 General 31
 - 5.4.2 Provisioning a bootstrap service 31
 - 5.4.3 Provisioning other services 32
 - 5.4.4 Credential provisioning..... 32
 - 5.4.5 Role assignment and provisioning 32
 - 5.4.6 ACL provisioning 33
 - 5.5 Secure Resource Manager-(SRM) 33
 - 5.6 Credential Overview 34
- 6 Security for the Discovery Process 35
 - 6.1 Introduction 35
 - 6.2 Security Considerations for Discovery 35
- 7 Security Provisioning..... 38
 - 7.1 Device Identity..... 38
 - 7.1.1 Device Identity for Devices with UAID 38
 - 7.2 Device Ownership 40
 - 7.3 Device Ownership Transfer Methods 40
 - 7.3.1 OTM implementation requirements 40
 - 7.3.2 SharedKey Credential Calculation 42

65	7.3.3	Certificate Credential Generation	42
66	7.3.4	Just-Works Owner Transfer Method	42
67	7.3.5	Random PIN Based Owner Transfer Method	44
68	7.3.6	Manufacturer Certificate Based Owner Transfer Method.....	46
69	7.3.7	Vendor Specific Owner Transfer Methods.....	51
70	7.3.8	Establishing Owner Credentials.....	52
71	7.3.9	Security considerations regarding selecting an Ownership Transfer Method..	63
72	7.4	Provisioning.....	63
73	7.4.1	Provisioning Flows	63
74	7.5	Bootstrap Example	69
75	8	Device Onboarding State Definitions	70
76	8.1	Introduction	70
77	8.2	Device Onboarding-Reset State Definition.....	71
78	8.3	Device Ready-for-OTM State Definition	72
79	8.4	Device Ready-for-Provisioning State Definition.....	72
80	8.5	Device Ready-for-Normal-Operation State Definition	73
81	8.5	Device Soft Reset State Definition	73
82	9	Security Credential Management.....	76
83	9.1	Introduction	76
84	9.2	Credential Lifecycle	76
85	9.2.1	General.....	76
86	9.2.2	Creation	76
87	9.2.3	Deletion	76
88	9.2.4	Refresh	76
89	9.2.5	Revocation	77
90	9.3	Credential Types	77
91	9.3.1	General.....	77
92	9.3.2	Pair-wise Symmetric Key Credentials.....	77
93	9.3.3	Group Symmetric Key Credentials.....	77
94	9.3.4	Asymmetric Authentication Key Credentials.....	78
95	9.3.5	Asymmetric Key Encryption Key Credentials	78
96	9.3.6	Certificate Credentials.....	79
97	9.3.7	Password Credentials.....	79
98	9.4	Certificate Based Key Management	79
99	9.4.1	Overview	79
100	9.4.2	Certificate Format	80
101	9.4.3	CRL Format.....	85
102	9.4.4	Resource Model	86
103	9.4.5	Certificate Provisioning	87
104	9.4.6	CRL Provisioning	88
105	10	Device Authentication	91
106	10.1	General	91
107	10.2	Device Authentication with Symmetric Key Credentials.....	91
108	10.3	Device Authentication with Raw Asymmetric Key Credentials	91

109	10.4	Device Authentication with Certificates	91
110	10.4.1	General	91
111	10.4.2	Role Assertion with Certificates	92
112	11	Message Integrity and Confidentiality	94
113	11.1	General	94
114	11.2	Session Protection with DTLS.....	94
115	11.2.1	General	94
116	11.2.2	Unicast Session Semantics	94
117	11.3	Cipher Suites.....	94
118	11.3.1	General.....	94
119	11.3.2	Cipher Suites for Device Ownership Transfer	94
120	11.3.3	Cipher Suites for Symmetric Keys	95
121	11.3.4	Cipher Suites for Asymmetric Credentials.....	95
122	12	Access Control.....	97
123	12.1	ACL Generation and Management	97
124	12.2	ACL Evaluation and Enforcement	97
125	12.2.1	General.....	97
126	12.2.2	Host Reference Matching	97
127	12.2.3	Resource Type Matching.....	97
128	12.2.4	Interface Matching.....	97
129	12.2.5	Multiple Criteria Matching.....	97
130	12.2.6	Resource Wildcard Matching	98
131	12.2.7	Subject Matching using Wildcards	99
132	12.2.8	Subject Matching using Roles	99
133	12.2.9	ACL Evaluation	99
134	13	Security Resources	100
135	13.1	General	100
136	13.2	Device Owner Transfer Resource	101
137	13.3	Credential Resource	106
138	13.3.1	Introduction	106
139	13.3.2	Properties of the Credential Resource	112
140	13.3.3	Key Formatting.....	115
141	13.3.4	Credential Refresh Method Details	115
142	13.4	Certificate Revocation List.....	117
143	13.4.1	CRL Resource Definition	117
144	13.5	ACL Resources	117
145	13.5.1	General.....	117
146	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures.	117
147	13.5.3	ACL Resource	118
148	13.6	Access Manager ACL Resource	128
149	13.7	Signed ACL Resource	128
150	13.8	Provisioning Status Resource	128
151	13.9	Certificate Signing Request Resource	137
152	13.10	Roles resource	138

153	13.11	Security Virtual Resources (SVRs) and Access Policy	139
154	13.12	SVRs, Discoverability and Endpoints	139
155	13.13	Privacy Consideration for Core and SVRs.....	140
156	14	Core Interaction Patterns Security.....	142
157	14.1	Observer	142
158	14.2	Subscription/Notification	142
159	14.3	Groups	142
160	14.4	Publish-subscribe Patterns and Notification	142
161	15	Security Hardening Guidelines/ Execution Environment Security.....	143
162	15.1	General	143
163	15.2	Execution environment elements	143
164	15.2.1	General.....	143
165	15.2.2	Secure Storage	143
166	15.2.3	Secure execution engine	145
167	15.2.4	Trusted input/output paths.....	145
168	15.2.5	Secure clock	146
169	15.2.6	Approved algorithms	146
170	15.2.7	Hardware tamper protection	146
171	15.3	Secure Boot	147
172	15.3.1	Concept of software module authentication	147
173	15.3.2	Secure Boot process	148
174	15.3.3	Robustness requirements.....	149
175	15.4	Software Update.....	149
176	15.4.1	Overview:.....	149
177	15.4.2	Recognition of Current Differences.....	149
178	15.4.3	Software Version Validation	149
179	15.4.4	Software Update	149
180	15.4.5	Recommended Usage	150
181	15.5	Security Levels	150
182	16	Appendix A: Access Control Examples	151
183	16.1	Example OCF ACL Resource.....	151
184	16.2	Example Access Manager Service	151
185	17	Appendix B: Execution Environment Security Profiles	152
186	18	Appendix C: RAML Definition	153
187	A.1	OICSecurityAcIResource	153
188	A.1.1	Introduction	153
189	A.1.2	Example URI	153
190	A.1.3	Resource Type	153
191	A.1.4	RAML Definition	153
192	A.1.5	Property Definition	157
193	A.1.6	CRUDN behavior.....	157
194	A.2	OICSecurityAcI2Resource	157
195	A.2.1	Introduction	157
196	A.2.2	Example URI	157

197	A.2.3	Resource Type	158
198	A.2.4	RAML Definition	158
199	A.2.5	Property Definition	162
200	A.2.6	CRUDN behavior.....	162
201	A.2.7	Referenced JSON schemas.....	162
202	A.2.8	oic.sec.didtype.json.....	162
203	A.2.9	Property Definition	162
204	A.2.10	Schema Definition	162
205	A.2.11	oic.sec.ace2.json	162
206	A.2.12	Property Definition	162
207	A.2.13	Schema Definition	163
208	A.2.14	oic.sec.roletype.json.....	165
209	A.2.15	Property Definition	165
210	A.2.16	Schema Definition	165
211	A.2.17	oic.sec.time-pattern.json	165
212	A.2.18	Property Definition	165
213	A.2.19	Schema Definition	165
214	A.2.20	oic.sec.crudntype.json.....	166
215	A.2.21	Property Definition	166
216	A.2.22	Schema Definition	166
217	A.3	OICSecurityAmaclResource.....	167
218	A.3.1	Introduction	167
219	A.3.2	Example URI	167
220	A.3.3	Resource Type	167
221	A.3.4	RAML Definition	167
222	A.3.5	Property Definition	170
223	A.3.6	CRUDN behavior.....	170
224	A.4	OICSecuritySignedAclResource.....	170
225	A.4.1	Introduction	170
226	A.4.2	Example URI	170
227	A.4.3	Resource Type	170
228	A.4.4	RAML Definition	170
229	A.4.5	Property Definition	176
230	A.4.6	CRUDN behavior.....	176
231	A.4.7	Referenced JSON schemas.....	176
232	A.4.8	oic.sec.sigtype.json.....	176
233	A.4.9	Property Definition	176
234	A.4.10	Schema Definition	176
235	A.5	OICSecurityDoxmResource	177
236	A.5.1	Introduction	177
237	A.5.2	Example URI	177
238	A.5.3	Resource Type	177
239	A.5.4	RAML Definition	177
240	A.5.5	Property Definition	181

241	A.5.6	CRUDN behavior	182
242	A.5.7	Referenced JSON schemas.....	182
243	A.5.8	oic.sec.doxmtype.json	182
244	A.5.9	Property Definition	182
245	A.5.10	Schema Definition	182
246	A.5.11	oic.sec.credtype.json.....	182
247	A.5.12	Property Definition	182
248	A.5.13	Schema Definition	182
249	A.6	OICSecurityPstatResource	183
250	A.6.1	Introduction	183
251	A.6.2	Example URI	183
252	A.6.3	Resource Type	183
253	A.6.4	RAML Definition	183
254	A.6.5	Property Definition	187
255	A.6.6	CRUDN behavior	188
256	A.6.7	Referenced JSON schemas.....	188
257	A.6.8	oic.sec.dostype.json.....	188
258	A.6.9	Property Definition	188
259	A.6.10	Schema Definition	188
260	A.6.11	oic.sec.dpmttype.json.....	189
261	A.6.12	Property Definition	189
262	A.6.13	Schema Definition	189
263	A.6.14	oic.sec.pomttype.json.....	189
264	A.6.15	Property Definition	189
265	A.6.16	Schema Definition	190
266	A.6.17	190	
267	A.7	OICSecurityCredentialResource	190
268	A.7.1	Introduction	190
269	A.7.2	Example URI	190
270	A.7.3	Resource Type	190
271	A.7.4	RAML Definition	190
272	A.7.5	Property Definition	194
273	A.7.6	CRUDN behavior.....	194
274	A.7.7	Referenced JSON schemas.....	194
275	A.7.8	oic.sec.roletype.json.....	194
276	A.7.9	Property Definition	194
277	A.7.10	Schema Definition	195
278	A.7.11	oic.sec.credtype.json.....	195
279	A.7.12	Property Definition	195
280	A.7.13	Schema Definition	195
281	A.7.14	oic.sec.pubdatatype.json	196
282	A.7.15	Property Definition	196
283	A.7.16	Schema Definition	196
284	A.7.17	oic.sec.privdatatype.json	196

285	A.7.18	Property Definition	196
286	A.7.19	Schema Definition	197
287	A.7.20	oic.sec.optdatatype.json	197
288	A.7.21	Property Definition	197
289	A.7.22	Schema Definition	198
290	A.7.23	oic.sec.crmttype.json	198
291	A.7.24	Property Definition	198
292	A.7.25	Schema Definition	198
293	A.8	OICSecurityCsrResource	199
294	A.8.1	Introduction	199
295	A.8.2	Example URI	199
296	A.8.3	Resource Type	199
297	A.8.4	RAML Definition	199
298	A.8.5	Property Definition	200
299	A.8.6	CRUDN behavior	200
300	A.9	OICSecurityRolesResource	200
301	A.9.1	Introduction	200
302	A.9.2	Example URI	201
303	A.9.3	Resource Type	201
304	A.9.4	RAML Definition	201
305	A.9.5	Property Definition	204
306	A.9.6	CRUDN behavior	204
307	A.10	OICSecurityCrlResource	204
308	A.10.1	Introduction	204
309	A.10.2	Example URI	204
310	A.10.3	Resource Type	204
311	A.10.4	RAML Definition	204
312	A.10.5	Property Definition	207
313	A.10.6	CRUDN behavior	208
314			
315			

Figures

317	Figure 1 – OCF Interaction	17
318	Figure 2 – OCF Layers	19
319	Figure 3 – OCF Security Enforcement Points.....	21
320	Figure 4 – Use case-1 showing simple ACL enforcement.....	23
321	Figure 5 – Use case 2: A policy for the requested Resource is missing.....	23
322	Figure 6 – Use case-3 showing Access Manager Service supported ACL	24
323	Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS	25
324	Figure 8 – Example Resource definition with opaque Properties	26
325	Figure 9 – Property Level Access Control	26
326	Figure 10 - Onboarding Overview	27
327	Figure 11 – OCF Onboarding Process	29
328	Figure 12 – OCF's SRM Architecture	33
329	Figure 13 - Discover New Device Sequence	41
330	Figure 14 – A Just Works Owner Transfer Method.....	43
331	Figure 15 – Random PIN-based Owner Transfer Method	45
332	Figure 16 – Manufacturer Certificate Hierarchy.....	48
333	Figure 17 – Manufacturer Certificate Based Owner Transfer Method Sequence.....	50
334	Figure 18 – Vendor-specific Owner Transfer Sequence	52
335	Figure 19 - Establish Device Identity Flow	55
336	Figure 20 – Owner Credential Selection Provisioning Sequence	57
337	Figure 21 - Symmetric Owner Credential Provisioning Sequence.....	58
338	Figure 22 - Asymmetric Ownership Credential Provisioning Sequence.....	59
339	Figure 23 - Configure Device Services.....	61
340	Figure 24 - Provision New Device for Peer to Peer Interaction Sequence	62
341	Figure 25 – Example of Client-directed provisioning	64
342	Figure 26 – Example of Server-directed provisioning using a single provisioning service.....	66
343	Figure 27 – Example of Server-directed provisioning involving multiple support services.....	68
344	Figure 28 – Device state model	70
345	Figure 29 – OBT Sanity Check Sequence in SRESET.....	74
346	Figure 30 – Certificate Management Architecture	80
347	Figure 31 – Client-directed Certificate Transfer	88
348	Figure 32 – Client-directed CRL Transfer	89
349	Figure 33 – Server-directed CRL Transfer	90
350	Figure 34 – Asserting a role with a certificate role credential.....	93
351	Figure 35 – OCF Security Resources.....	100
352	Figure 36 – oic.r.cred Resource and Properties	101
353	Figure 37 – oic.r.acl2 Resource and Properties	101
354	Figure 38 – oic.r.amacl Resource and Properties	101

355	Figure 39 – oic.secr.sacl Resource and Properties	101
356	Figure 40 – Software Module Authentication.....	147
357	Figure 41 – Verification Software Module	148
358	Figure 42 – Software Module Authenticity.....	148
359		

Tables

360		
361	Table 1 – Symbols and abbreviations	16
362	Table 2 - Discover New Device Details	41
363	Table 3 – A Just Works Owner Transfer Method Details	43
364	Table 4 – Random PIN-based Owner Transfer Method Details	45
365	Table 5 – Manufacturer Certificate Based Owner Transfer Method Details.....	51
366	Table 6 – Vendor-specific Owner Transfer Details	52
367	Table 7 - Establish Device Identity Details.....	56
368	Table 8 - Owner Credential Selection Details.....	58
369	Table 9 - Symmetric Owner Credential Assignment Details.....	58
370	Table 10 – Asymmetric Owner Credential Assignment Details	59
371	Table 11 - Configure Device Services Detail.....	62
372	Table 12 - Provision New Device for Peer to Peer Details.....	63
373	Table 13 – Steps describing Client -directed provisioning	65
374	Table 14 – Steps for Server-directed provisioning using a single provisioning service	67
375	Table 15 – Steps for Server-directed provisioning involving multiple support services	69
376	Table 16 – Comparison between OCF and X.509 certificate fields	82
377	Table 17 – Comparison between OCF and X.509 CRL fields	86
378	Table 18 – ACE2 Wildcard Matching Strings Description	98
379	Table 19 – Definition of the oic.r.doxm Resource	102
380	Table 20 – Properties of the oic.r.doxm Resource	104
381	Table 21 - Properties of the oic.sec.didtype Property.....	104
382	Table 22 – Properties of the oic.sec.doxmtype Property	106
383	Table 23 – Definition of the oic.r.cred Resource	108
384	Table 24 – Properties of the oic.r.cred Resource	108
385	Table 25 – Properties of the oic.sec.cred Property	111
386	Table 26 – Properties of the oic.sec.pubdatatype Property	111
387	Table 27 – Properties of the oic.sec.privdatatype Property	112
388	Table 28 – Properties of the oic.sec.optdatatype Property	112
389	Table 29 – Definition of the oic.sec.roletype Property	112
390	Table 30 – Value Definition of the oic.sec.crmtype Property	114
391	Table 31 – 128-bit symmetric key	115
392	Table 32 – 256-bit symmetric key	115
393	Table 33 – Definition of the oic.r.crl Resource	117
394	Table 34 – Properties of the oic.r.crl Resource	117
395	Table 35 – BNF Definition of OCF ACL.....	118
396	Table 36 – Definition of the oic.r.acl Resource.....	120
397	Table 37 – Properties of the oic.r.acl Resource	121
398	Table 38 – Properties of the oic.r.ace Property	122

399	Table 39 – Value Definition of the oic.sec.crudntype Property	122
400	Table 40 – Definition of the oic.sec.acl2 Resource	122
401	Table 41 – Properties of the oic.sec.acl2 Resource	123
402	Table 42 – oic.sec.ace2 data type definition.	124
403	Table 43 – oic.sec.ace2.resource-ref data type definition.	124
404	Table 44 – Value definition oic.sec.conntype Property	124
405	Table 45 – Definition of the oic.r.amacl Resource	128
406	Table 46 – Properties of the oic.r.amacl Resource	128
407	Table 47 – Definition of the oic.r.sacl Resource	128
408	Table 48 – Properties of the oic.r.sacl Resource	128
409	Table 49 – Properties of the oic.sec.sigtype Property	128
410	Table 50 – Definition of the oic.r.pstat Resource	130
411	Table 51 – Properties of the oic.r.pstat Resource	132
412	Table 52 – Properties of the oic.sec.dostype Property	133
413	Table 53 – Definition of the oic.sec.dpmtype Property	136
414	Table 54 – Value Definition of the oic.sec.dpmtype Property (Low-Byte)	136
415	Table 55 – Value Definition of the oic.sec.dpmtype Property (High-Byte)	136
416	Table 56 – Definition of the oic.sec.pomtype Property	137
417	Table 57 – Value Definition of the oic.sec.pomtype Property	137
418	Table 58 – Definition of the oic.r.csr Resource	138
419	Table 59 – Properties of the oic.r.csr Resource	138
420	Table 60 – Definition of the oic.r.roles Resource	139
421	Table 61 – Properties of the oic.r.roles Resource	139
422	Table 62 – Core Resource Properties state	141
423	Table 63 – Examples of Sensitive Data	144
424	Table 64 – OCF Security Profile	152
425	Table 65 – OCF SVR RAML	153
426		
427		

428 **1 Scope**

429 This specification defines security objectives, philosophy, resources and mechanism that impacts
430 OCF base layers of the OCF Core Specification. The OCF Core Specification contains informative
431 security content. The OCF Security specification contains security normative content and may
432 contain informative content related to the OCF base or other OCF specifications.

433 **2 Normative References**

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

437 OCF Core Specification, version 1.1, Open Connectivity Foundation, October 11, 2016. Latest
438 version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf.

439 OCF Device Specification, version 1.1, Open Connectivity Foundation, October 11, 2016. Latest
440 version available at: https://openconnectivity.org/specs/OCF_Device_Specification.pdf.

441 OCF Resource Type Specification, version 1.1, Open Connectivity Foundation, October 11,
442 2016. Latest version available at:
443 https://openconnectivity.org/specs/OCF_Resource_Type_Specification.pdf.

444 JSON SCHEMA, draft version 4, JSON Schema defines the media type "application/schema+json",
445 a JSON based format for defining the structure of JSON data. JSON Schema provides a contract
446 for what JSON data is required for a given application and how to interact with it. JSON Schema
447 is intended to define validation, documentation, hyperlink navigation, and interaction control of
448 JSON Available at: <http://json-schema.org/latest/json-schema-core.html>.

449 RAML, Restful API modelling language version 0.8. Available at: <http://raml.org/spec.html>.

450 OCF Security Resource Definitions, *API Definition Language for OCF Security Components*,
451 Release OCF-v1.0.0
452 <https://github.com/openconnectivityfoundation/security>

453

454

455 **3 Terms, Definitions, Symbols and Abbreviations**

456 Terms, definitions, symbols and abbreviations used in this specification are defined by the OCF
457 Core Specification. Terms specific to normative security mechanism are defined in this document
458 in context.

459 This section restates terminology that is defined elsewhere, in this document or in other OCF
460 specifications as a convenience for the reader. It is considered non-normative.

461 **3.1 Terms and definitions**

462 **3.1.1**

463 **Access Manager Service**

464 The Access Manager Service dynamically constructs ACL Resources in response to a Device
465 Resource request. An Access Manager Service can evaluate access policies remotely and supply
466 the result to a Server which allows or denies a pending access request.

467 **3.1.2**

468 **ACL Provisioning Service**

469 A name and Resource Type (oic.sec.aps) given to a Device that is authorized to provision ACL
470 Resources.

471 **3.1.3**

472 **Bootstrap Service**

473 A Device that implements a service of type oic.sec.bss

474 **3.1.4**

475 **Client**

476 Note 1 to entry: The details are defined in OCF Core Specification.

477 **3.1.5**

478 **Credential Management Service**

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

481 **3.1.6**

482 **Device**

483 Note 1 to entry: The details are defined in OCF Core Specification.

484 **3.1.7**

485 **Device Class**

486 As defined in RFC 7228. RFC 7228 defines classes of constrained devices that distinguish when
487 the OCF small footprint stack is used vs. a large footprint stack. Class 2 and below is for small
488 footprint stacks.

489 **3.1.8**

490 **Device ID**

491 A stack instance identifier.

492 **3.1.9**

493 **Entity**

494 Note 1 to entry: The details are defined in OCF Core Specification.

495 **3.1.10**

496 **Interface**

497 Note 1 to entry: The details are defined in OCF Core Specification.

498 **3.1.11**
499 **Intermediary**
500 A Device that implements both Client and Server roles and may perform protocol translation, virtual
501 device to physical device mapping or Resource translation

502 **3.1.12**
503 **OCF Cipher Suite**
504 A set of algorithms and parameters that define the cryptographic functionality of a Device. The
505 OCF Cipher Suite includes the definition of the public key group operations, signatures, and
506 specific hashing and encoding used to support the public key.

507 **3.1.13**
508 **Onboarding Tool**
509 A logical entity within a specific IoT network that establishes ownership for a specific device and
510 helps bring the device into operational state within that network

511 **3.1.14**
512 **Out of Band Method**
513 Any mechanism for delivery of a secret from one party to another, not specified by OCF

514 **3.1.15**
515 **Owner Credential**
516 Credential, provisioned by an Onboarding Tool to a Device during onboarding, for the purposes of
517 mutual authentication of the Device and Onboarding Tool during subsequent interactions

518 **3.1.16**
519 **Platform ID**
520 Note 1 to entry: The details are defined in OCF Core Specification.

521 **3.1.17**
522 **Property**
523 Note 1 to entry: The details are defined in OCF Core Specification.

524 **3.1.18**
525 **Resource**
526 Note 1 to entry: The details are defined in OCF Core Specification.

527 **3.1.19**
528 **Role (network context)**
529 Stereotyped behavior of a Device; one of [Client, Server or Intermediary]

530 **3.1.20**
531 **Role (Security context)**
532 A Property of an OCF credentials Resource that names a role that a Device may assert when
533 attempting access to Device Resources. Access policies may differ for Client if access is attempted
534 through a role vs. the device UUID. This document assumes the security context unless otherwise
535 stated.

536 **3.1.21**
537 **Secure Resource Manager**
538 A module in the OCF Core that implements security functionality that includes management of
539 security Resources such as ACLs, credentials and Device owner transfer state.

540 **3.1.22**
 541 **Security Virtual Resource**
 542 An SVR is a resource supporting security features.

543 **3.1.23**
 544 **Server**
 545 Note 1 to entry: The details are defined in OCF Core Specification.

546 **3.1.24**
 547 **Trust Anchor**
 548 A well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.
 549 a Device and an onboarding tool) can assume trust

550 **3.1.25**
 551 **Unique Authenticable Identifier**
 552 A unique identifier created from the hash of a public key and associated OCF Cipher Suite that is
 553 used to create the DeviceID. The ownership of a UAID may be authenticated by peer Devices.

554 **3.2 Symbols and Abbreviations**

Symbol	Description
ACE	Access Control Entry
ACL	Access Control List
AMS	Access Manager Service
APS	ACL Provisioning Service
BSS	Bootstrap Service
CMS	Credential Management Service
CRUDN	CREATE, RETREIVE, UPDATE, DELETE, NOTIFY
CSR	Certificate Signing Request
ECDSA	Elliptic Curve Digital Signature Algorithm
EPC	Embedded Platform Credential
DPKP	Dynamic Public Key Pair
OC	Owner Credential
OCSP	Online Certificate Status Protocol
OBT	Onboarding Tool
PIN	Personal Identification Number
PSI	Persistent Storage Interface
RNG	Random Number Generator
SACL	Signed Access Control List
SE	Secure Element
SRM	Secure Resource Manager
SVR	Security Virtual Resource
TEE	Trusted Execution Environment
UAID	Unique Authenticable Identifier

555 **Table 1 – Symbols and abbreviations**

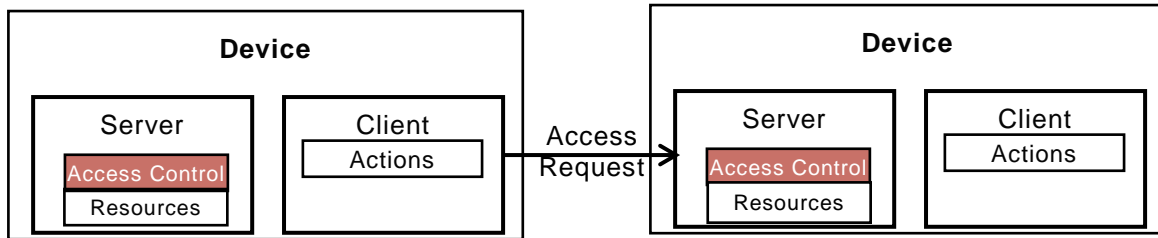
556 **4 Document Conventions and Organization**

557 **4.1 Introduction**

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

560 For the purposes of this document, the terms and definitions given in OCF Core Specification apply.

561 **4.2 Conventions**



562 **Figure 1 – OCF Interaction**

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

567 **4.3 Notation**

568 In this document, features are described as required, recommended, allowed or DEPRECATED as
569 follows:

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

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

574 **Recommended (or should).**

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

582 **Allowed (or allowed).**

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

586 **Conditionally allowed (CA)**

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

589 **Conditionally required (CR)**

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

593 **DEPRECATED**

594 Although these features are still described in this specification, they should not be implemented
595 except for backward compatibility. The occurrence of a deprecated feature during operation of
596 an implementation compliant with the current specification has no effect on the
597 implementation's operation and does not produce any error conditions. Backward compatibility

598 may require that a feature is implemented and functions as specified but it shall never be used
599 by implementations compliant with this specification.

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

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

602 **4.4 Data types**

603 See OCF Core Specification.

604 **4.5 Document structure**

605 Informative sections may be found in the Overview sections, while normative sections fall outside
606 of those sections.

607 The Security specification may use RAML as a specification language and JSON Schemas as
608 payload definitions for all CRUDN actions. The mapping of the CRUDN actions is specified in the
609 OCF Core Specification.

610

611 **5 Security Overview**

612 **5.1 Introduction**

613 This is an informative section. The goal for the OCF security architecture is to protect the
 614 Resources and all aspects of HW and SW that are used to support the protection of Resource.
 615 From OCF perspective, a Device is a logical entity that conforms to the OCF specifications. In an
 616 interaction between the Devices, the Device acting as the Server holds and controls the Resources
 617 and provides the Device acting as a Client with access to those Resources, subject to a set of
 618 security mechanisms. The Platform, hosting the Device may provide security hardening that will
 619 be required for ensuring robustness of the variety of operations described in this specification.

620 The security theory of operation is described in the following steps.

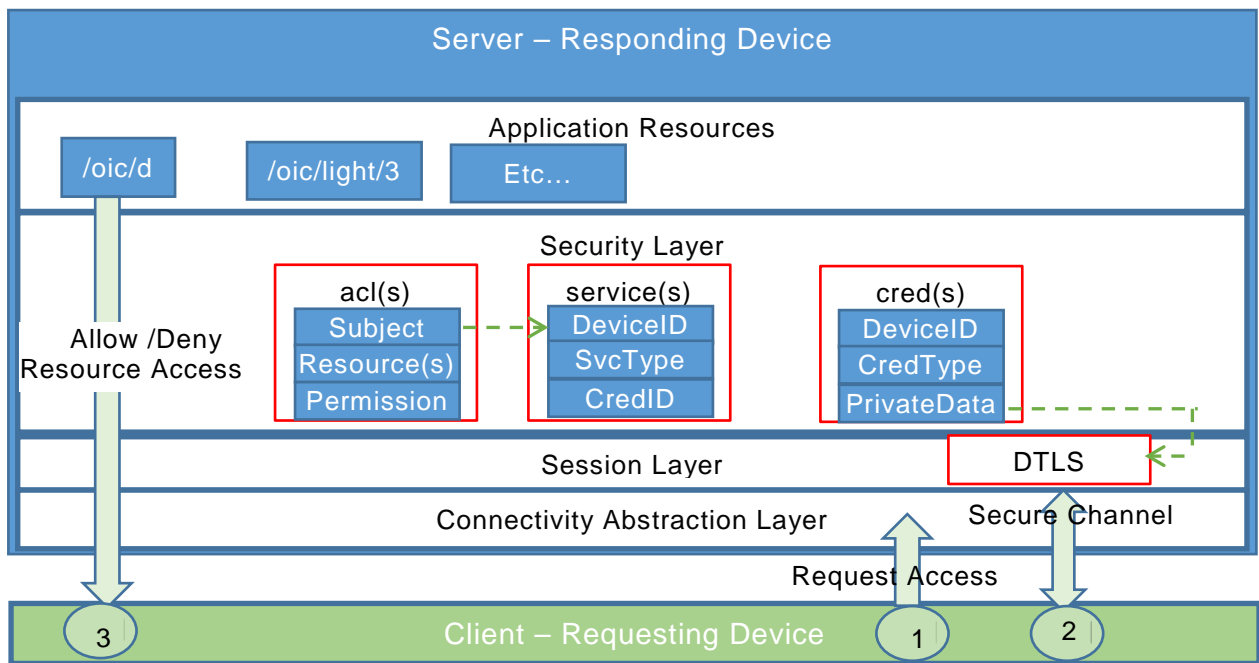


Figure 2 – OCF Layers

- 621
- 622 1. The Client establishes a network connection to the Server (Device holding the Resources).
 623 The connectivity abstraction layer ensures the Devices are able to connect despite differences
 624 in connectivity options.
- 625 2. The Devices (e.g. Server and Client) exchange messages either with or without a mutually-
 626 authenticated secure channel between the two Devices.
- 627
- 628 • The oic.sec.cred Resource on each Devices holds the credentials used for mutual
 629 authentication and (when applicable) certificate validation.
 - 630 • Messages received over a secured channel are associated with a deviceUUID. In the case
 631 of a certificate credential, the deviceUUID is in the certificate received from the other
 632 Device. In the case of a symmetric key credential, the deviceUUID is configured with the
 633 credential in the oic.sec.cred Resource. There should be a binding between the device
 634 context and the Platform implementing the Device.
 - 635 • The Server can associate the Client with any number of allowed roleid. In the case of
 636 mutual authentication using a certificate, the allowed roleid (if any) are provided in role
 637 certificates; these are configured by the Client to the Server. In the case of a symmetric
 638 key, the allowed roleid (if any) are configured with the credential in the oic.sec.cred.
 - 639 • Requests received by a Server over an unsecured channel are treated as anonymous and
 not associated with any deviceUUID or roleid.

- 640 3. The Client submits a request to the Server.
- 641 • If the request is to be sent over the secure channel, then the Client can either explicitly
- 642 assert specific roleid by including 'role' options in the request, or implicitly assert all roleid
- 643 associated with the Client by including no 'role' options.
- 644 4. The Server receives the request.
- 645 a. If the request is received over an unsecured channel, the Server treats the request as
- 646 anonymous and no deviceUUID or roleid are associated with the request.
- 647 b. If the request is received over a secure channel, then the Server associates the
- 648 deviceUUID, and the Server either validates any explicitly asserted roleids by matching to
- 649 an allowed roleid of the Client, or implicitly asserts all valid roleid of the Client.
- 650 c. The Server then consults the Access Control List (ACL), and looks for an ACL entry
- 651 matching the following criteria:
- 652 • The requested Resource matches a Resource reference in the ACE
- 653 • The requested operation is allowed by the "permissions" of the ACE, and
- 654 • The "subjectUUID" contains either a special wildcard value matching all Devices or,
- 655 if the Device is not anonymous, the subject matches the Client Deviceid or a valid
- 656 asserted roleID. In certain cases, the requester may assert a role, if privileged
- 657 access is required.
- 658 If there is a matching ACE, then access to the Resource is allowed; otherwise access is
- 659 denied. Access is enforced by the Server's Secure Resource manager (SRM).
- 660

661 Resource protection includes protection of data both while at rest and during transit. It should be

662 noted that, aside from access control mechanisms, OCF security specification does not include

663 specification of secure storage of Resources, while stored at Servers. However, at rest protection

664 for security Resources is expected to be provided through a combination of secure storage and

665 access control. Secure storage can be accomplished through use of hardware security or

666 encryption of data at rest. The exact implementation of secure storage is subject to a set of

667 hardening requirements that are specified in Section 15 and may be subject to certification

668 guidelines.

669 Data in transit protection, on the other hand, will be specified fully as a normative part of this

670 specification. In transit protection may be afforded at

- 671 1. Resource layer through mechanisms such as JSON Web Encryption (JWE) and JSON Web
- 672 Signatures (JWS) that allow payload protection independent of underlying transport security.
- 673 This may be a necessary for transport mechanisms that cannot take advantage of DTLS for
- 674 payload protection.
- 675 2. At transport layer through use of mechanisms such as DTLS. It should be noted that DTLS will
- 676 provide packet by packet protection, rather than protection for the payload as whole. For
- 677 instance, if the integrity of the entire payload as a whole is required, separate signature
- 678 mechanisms must have already been in place before passing the packet down to the transport
- 679 layer.

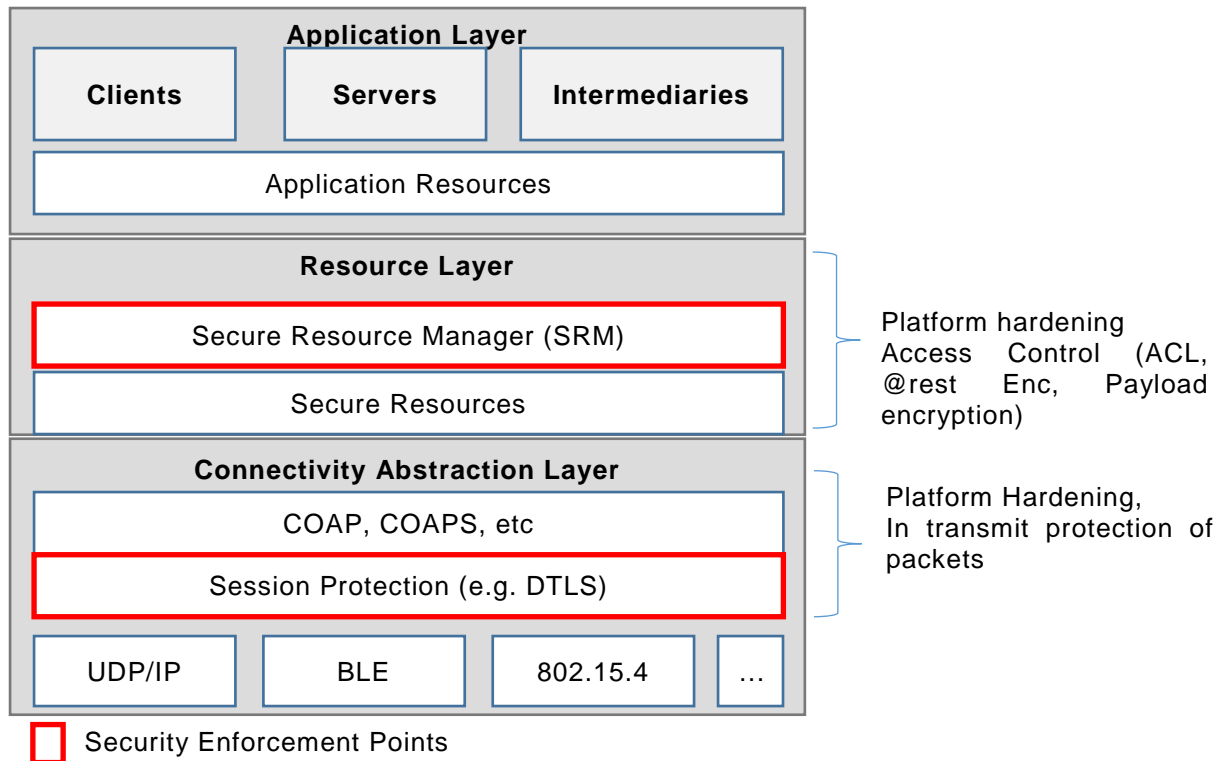


Figure 3 – OCF Security Enforcement Points

680

681 **5.2 Access Control**

682 **5.2.1 General**

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

689 Implementation of access control relies on a-priori definition of a set of access policies for the
 690 Resource. The policies may be stored by a local ACL or an Access Manager Service (AMS) in form
 691 of Access Control Entries (ACE), where each ACE defines permissions required to access a
 692 specific Resource along with an optional validity period for the granted permission. Two types of
 693 access control mechanisms can be applied:

- 694 • Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity
 695 of requestor) of the requesting entity against the subject included in the policy defined for
 696 Resource. Asserting the identity of the requestor requires an authentication process.
- 697 • Role-based Access Control (RBAC), where each ACE will match a role required by policy
 698 for the Resource to a role taken by the entity requesting access. Asserting the role of the
 699 requestor requires proper authorization.

700 In the OCF access control model, each Resource instance is required to have an associated
 701 access control policy. This means, each Device acting as Server, needs to have an ACL for each
 702 Resource it is protecting. Lack of an ACE that matches, it results in the Resource being
 703 inaccessible.

704 The ACE must match both the subject (i.e. OCF Client) and the Resource requested for the ACE
705 to apply. There are multiple ways a subject could be matched, (1) device id, (2) role or (3) wildcard.
706 The way in which the client connects to the server may be relevant context for making access
707 control decisions. Wildcard matching on authenticated vs. unauthenticated and encrypted vs.
708 unencrypted connection allows an access policy to be broadly applied to subject classes.

709 Example Wildcard Matching Policy:

```
710 "aclist2": [  
711   {  
712     "subject": {"conntype" : "anon-clear" },  
713     "resources": [  
714       { "wc": "*" }  
715     ],  
716     "permission": 31  
717   },  
718   {  
719     "subject": {"conntype" : "auth-crypt" },  
720     "resources": [  
721       { "wc": "*" }  
722     ],  
723     "permission": 31  
724   },  
725 ]
```

726 Details of the format for ACL are defined in Section 12. The ACL is composed of one or more
727 ACEs. The ACL defines the access control policy for the Devices.

728 It should be noted that the ACL Resource requires the same security protection as other sensitive
729 Resources, when it comes to both storage and handling by SRM and PSI. Thus hardening of an
730 underlying Platform (HW and SW) must be considered for protection of ACLs and as explained
731 below ACLs may have different scoping levels and thus hardening needs to be specially considered
732 for each scoping level. For instance a physical device may host multiple Device implementations
733 and thus secure storage, usage and isolation of ACLs for different Servers on the same Device
734 needs to be considered.

735 5.2.2 ACL Architecture

736 5.2.2.1 General

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

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

746 Each ACE contains the permission set that will be applied for a given Resource requestor.
747 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY
748 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more

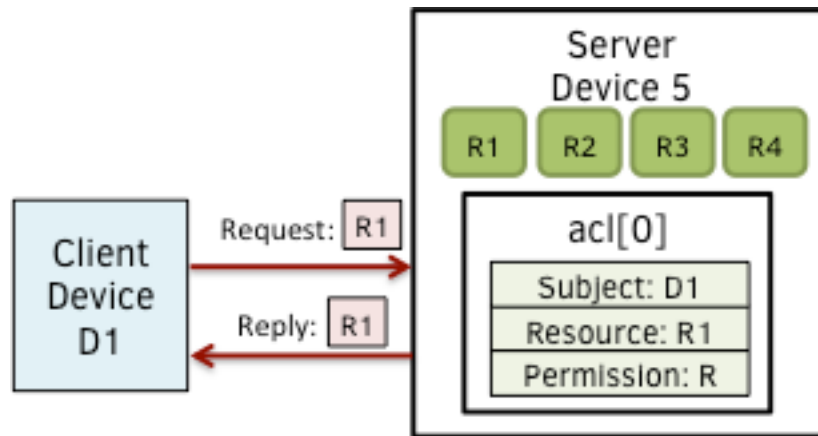
749 roles. Devices may acquire elevated access permissions when asserting a role. For example, an
 750 ADMINISTRATOR role might expose additional Resources and Interfaces not normally accessible.

751 **5.2.2.2 Use of local ACLs**

752 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control
 753 processing than remote ACL processing by an Access Management Service (AMS) as described
 754 below.

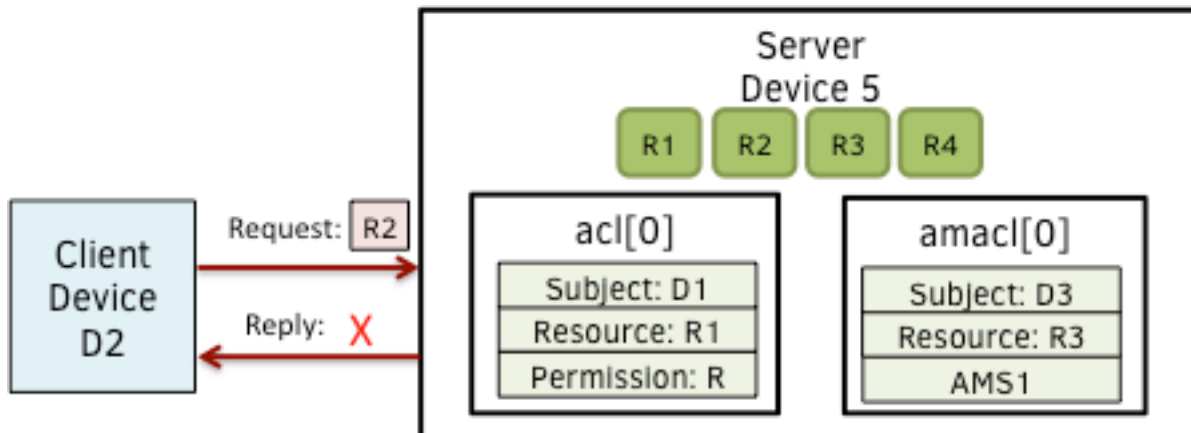
755 The following use cases describe the operation of access control

756 Use Case 1: Server Device hosts 4 Resources (R1, R2, R3 and R4). Client Device D1 requests
 757 access to Resource R1 hosted at Server Device 5. ACL[0] corresponds to Resource R1 below and
 758 includes D1 as an authorized subject. Thus, Device D1 receives access to Resource R1 because
 759 the local ACL /oic/sec/acl/0 matches the request.



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

761 Use Case 2: Client Device D2 access is denied because no local ACL match is found for subject
 762 D2 pertaining Resource R2 and no AMS policy is found.
 763



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

765 **5.2.2.3 Use of Access Manager Service**

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

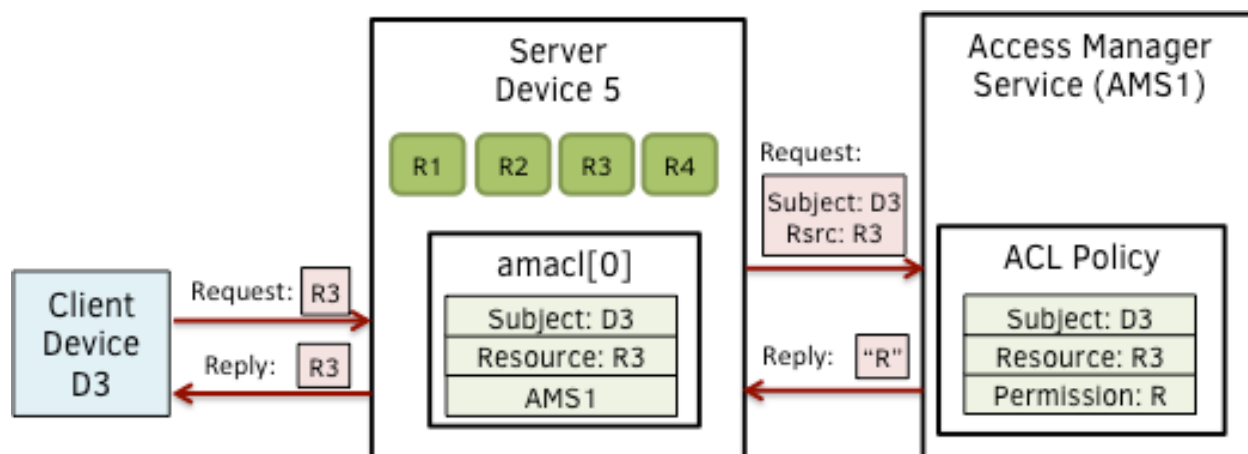
769 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The
 770 Server shall determine which ACL mechanism to use for which Resource set. The /oic/sec/amacl
 771 Resource is an ACL structure that specifies which Resources will use an AMS to resolve access
 772 decisions.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

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

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

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

781 Use Case 3: Device D3 requests and receives access to Resource R3 with permission Perm1
782 because the /oic/sec/amacl/0 matches a policy to consult the Access Manager Server AMS1
783 service



784

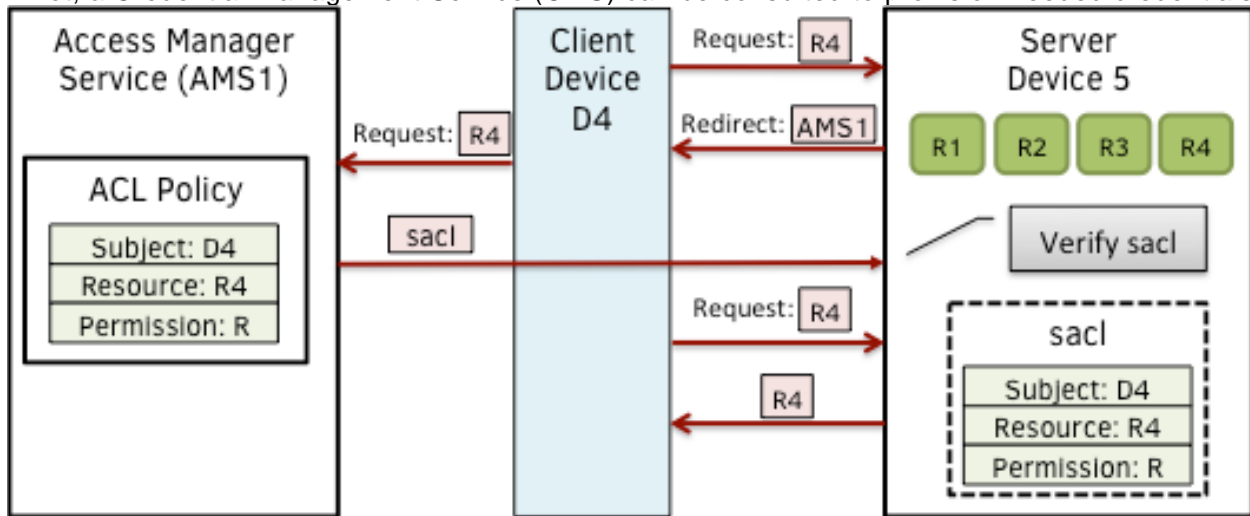
785

Figure 6 – Use case-3 showing Access Manager Service supported ACL

786 Use Case 4: Client Device D4 requests access to Resource R4 from Server Device 5, which fails
787 to find a matching ACE and redirects the Client Device D4 to AMS1 by returning an error identifying
788 AMS1 as a /oic/sec/sacl Resource issuer. Device D4 obtains Sacl1 signed by AMS1 and forwards
789 the SACL to Server D5. D5 verifies the signature in the /oic/sec/sacl Resource and evaluates the
790 ACE policy that grants Perm2 access.

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

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



799

800

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

801

5.2.3 Access Control Scoping Levels

802

803

804

805

806

Group Level Access - Group scope means applying AC to the group of Devices that are grouped for a specific context. Group credentials may be used when encrypting data to the group or authenticating individual Device members into the group. Group Level Access means all group members have access to group data but non-group members must be granted explicit access. Group level access may also be specified using wildcard matching.

807

808

809

810

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

811

812

813

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

814

815

816

817

818

819

Property Level Access - Property level scope means applying AC only to a property that is part of a parent Resource. This is to provide a finer granularity for AC to Resources that may require different permissions for different properties. Property level access control is achieved by creating a Resource that contains a single property. This technique allows the Resource level access control mechanisms to be used to enforce access at a finer level of granularity than would otherwise be possible.

820

821

822

823

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

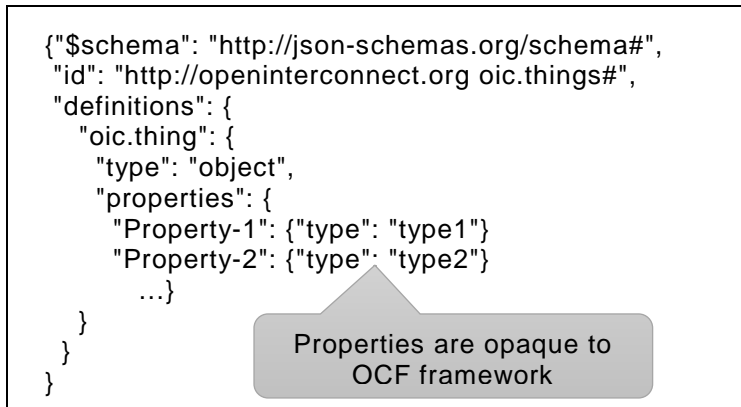


Figure 8 – Example Resource definition with opaque Properties

824

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

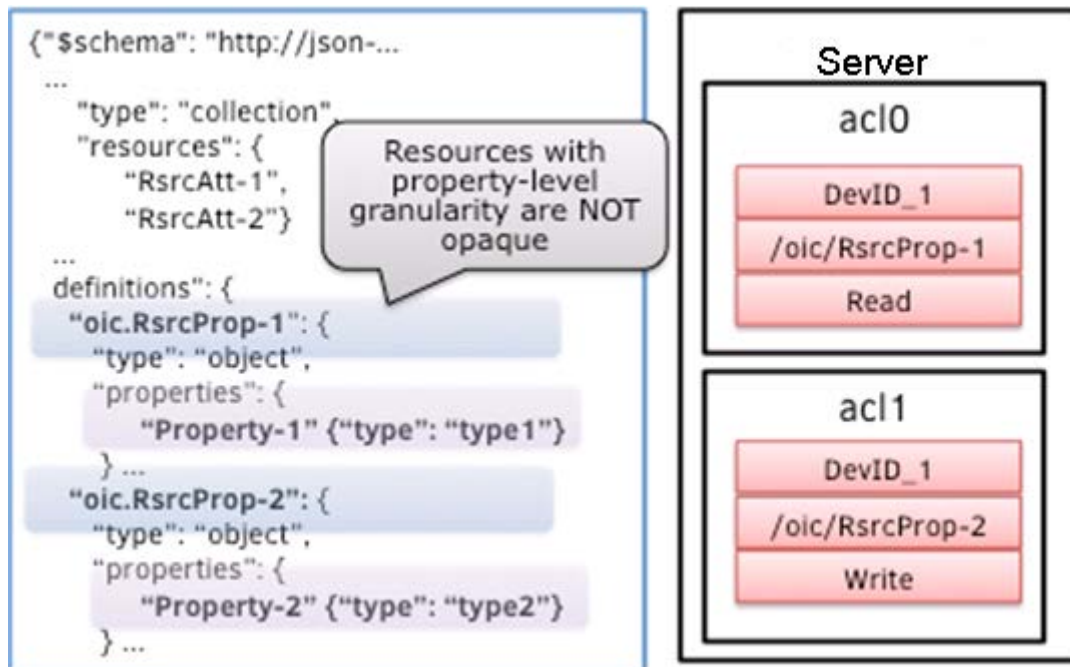


Figure 9 – Property Level Access Control

830

831

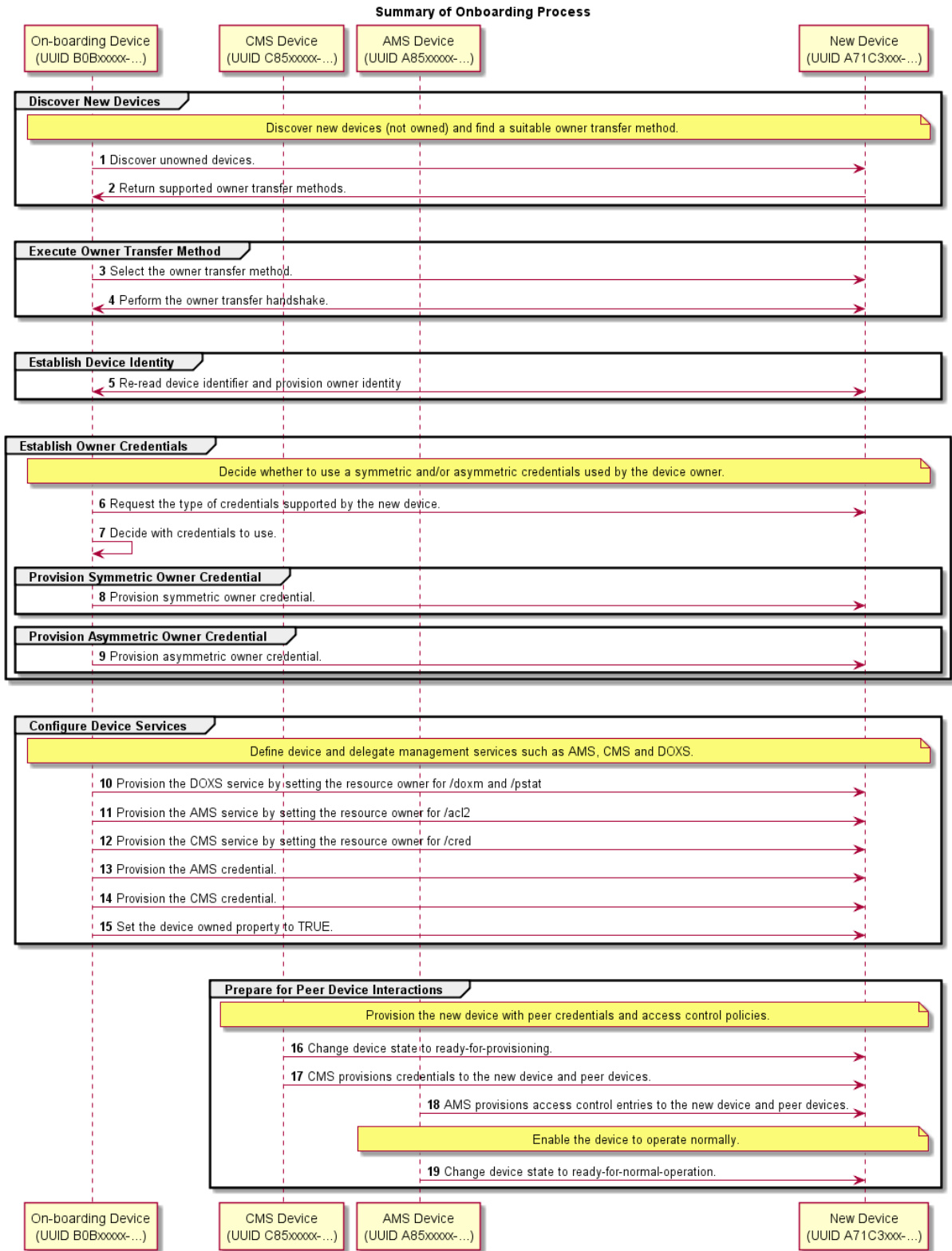
5.3 Onboarding Overview

832

5.3.1 General

833

834 Before a Device becomes operational in an OCF environment and is able to interact with other
 835 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to
 836 configure the ownership where the legitimate user that owns/purchases the Device uses an
 837 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTM) to
 838 establish ownership. Once ownership is established, the OBT becomes the mechanism through
 839 which the Device can then be provisioned, at the end of which the Device becomes operational
 840 and is able to interact with other Devices in an OCF environment.



841
842

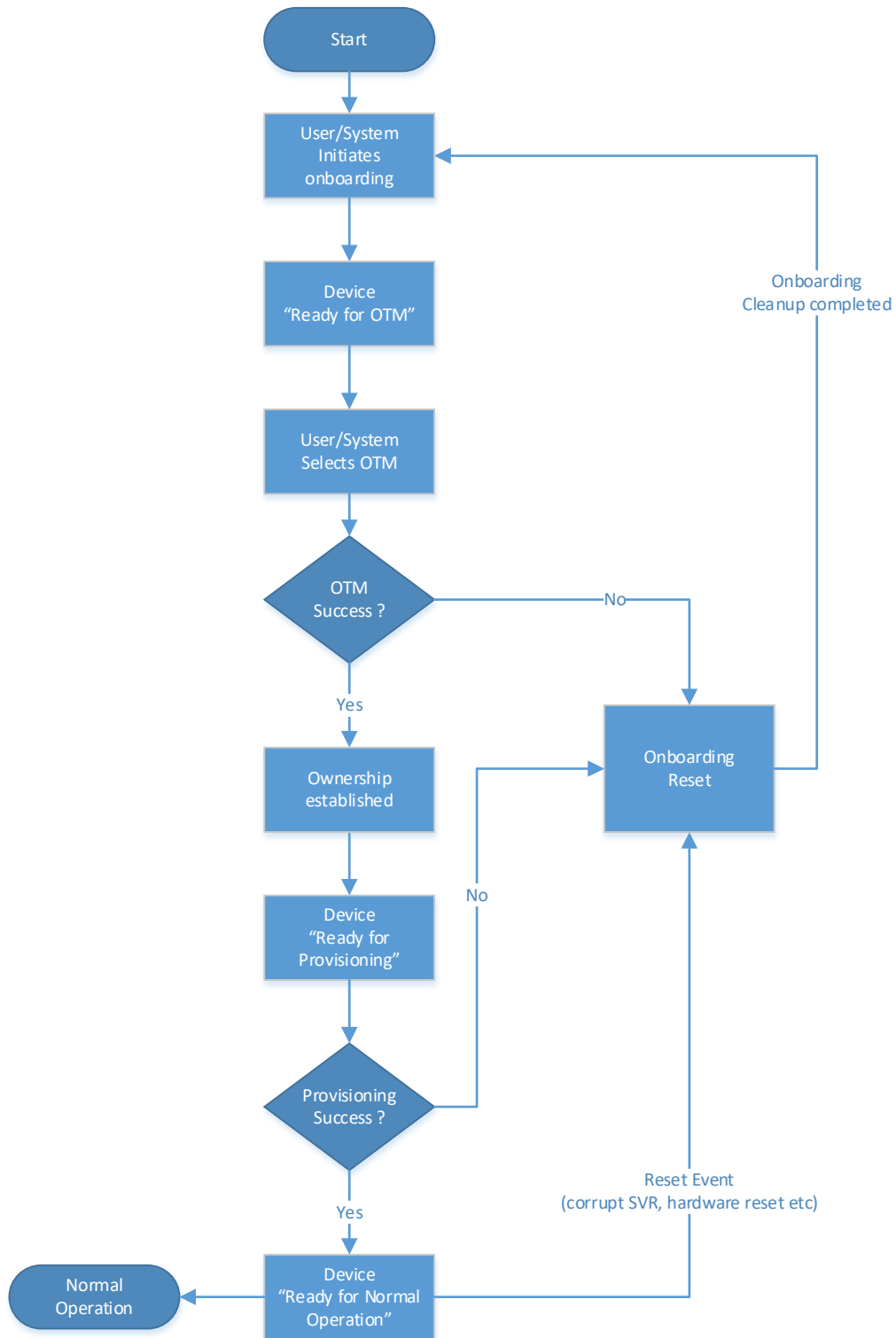
Figure 10 - Onboarding Overview

843 This section explains the onboarding and security provisioning process but leaves the provisioning
844 of non-security aspects to other OCF specifications. In the context of security, all Devices are
845 required to be provisioned with minimal security configuration that allows the Device to securely

846 interact/communicate with other Devices in an OCF environment. This minimal security
847 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified
848 in Section 8.

849 **5.3.2 OnBoarding Steps**

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



857

858

Figure 11 – OCF Onboarding Process

859 **5.3.3 Establishing a Device Owner**

860 **5.3.3.1 General**

861 The objective behind establishing Device ownership is to allow the legitimate user that
862 owns/purchased the Device to assert itself as the owner and manager of the Device. This is done
863 through the use of an OBT that includes the creation of an ownership context between the new
864 Device and the OBT tool and asserts operational control and management of the Device. The OBT
865 can be considered a logical entity hosted by tools/ Servers such as a network management console,
866 a device management tool, a network-authoring tool, a network provisioning tool, a home gateway
867 device, or a home automation controller. A physical device hosting the OBT will be subject to some
868 security hardening requirements, thus preserving integrity and confidentiality of any credentials
869 being stored. The tool/Server that establishes Device ownership is referred to as the OBT.

870 The OBT uses one of the Owner Transfer method specified in Section 7.3 to securely establish
871 Device ownership. The term owner transfer is used since it is assumed that even for a new Device,
872 the ownership is transferred from the manufacturer/provider of the Device to the buyer/legitimate
873 user of the new Device.

874 An owner transfer method establishes a new owner (the operator of OBT) that is authorized to
875 manage the Device. Owner transfer establishes the following

- 876 • An Owner Credential (OC) that is provisioned by the OBT in the /oic/sec/doxm Resource
877 of the Device. This OC allows the Device and OBT to mutually authenticate during
878 subsequent interactions. The OC asserts the user/system's ownership of the Device by
879 recording the credential of the OBT as the owner. The OBT also records the identity of
880 Device as part of ownership transfer.
- 881 • The Device owner establishes trust in the Device through the OTM.
- 882 • Preparing the Device for provisioning by providing credentials that may be needed.

883 **5.3.3.2 Preparing the Device for provisioning**

884 Once Device ownership is established, the Device needs to be prepared for provisioning. This
885 could be in the form of getting bootstrapping parameters (BP) that allow the Device to contact the
886 bootstrap server (BS) and establish a secure session with the BS. The typical bootstrap parameters
887 are as follows

- 888 • Bootstrap server (BS)/ tool metadata: This information needs to include addressing and
889 access mechanism/ protocol to be used to access the bootstrap server. Addressing
890 information may include Server URI or FQDN if HTTP or TCP/IP is being used to contact
891 the Server.
- 892 • Bootstrapping credential (BC): This is the credential that the Device needs to use to contact
893 the BS, authenticate to the BS, and establish a secure session with the BS to receive
894 provisioning parameters from the BS. The BC may be derived from the OC depending on
895 the type of OC.

896 If the OBT itself acts as the bootstrap server, the metadata for the bootstrap server may point to a
897 service hosted by the OBT itself. In such a scenario additional BC credentials may not be needed.

898 **5.3.4 Provisioning for Normal Operation**

899 Once the Device has the necessary information to initiate provisioning, the next step is to provision
900 additional security configuration that allows the Device to become operational. This can include
901 setting various parameters and may also involve multiple steps. For example if the Device is to
902 receive its configuration from a bootstrapping server, then the provisioning may involve connecting
903 to the bootstrap server and receive its configuration. Also provisioning of ACL's for the various

904 Resources hosted by the Server on the Device is done at this time. Note that the provisioning step
905 is not limited to this stage only. Device provisioning can happen at multiple stages in the Device's
906 operational lifecycle. However specific security related provisioning of Resource and Property
907 state would likely happen at this stage at the end of which, each Device reaches the Onboarded
908 Device "Ready for Normal Operation" State. The "Ready for Normal Operation" State is expected
909 to be consistent and well defined regardless of the specific OTM used or regardless of the
910 variability in what gets provisioned. However individual OTM mechanisms and provisioning steps
911 may specify additional configuration of Resources and Property states. The minimal mandatory
912 configuration required for a Device to be in "Ready for Normal Operation" state is specified in
913 Section 8.

914 **5.4 Provisioning**

915 **5.4.1 General**

916 Note that in general, provisioning may include processes during manufacturing and distribution of
917 the Device as well as processes after the Device has been brought into its intended environment
918 (parts of onboarding process). In this specification, security provisioning includes, processes after
919 ownership transfer (even though some activities during ownership transfer and onboarding may
920 lead to provisioning of some data in the Device) configuration of credentials for interacting with
921 bootstrapping and provisioning services, configuration of any security related Resources and
922 credentials for dealing with any services that the Device need to contact later on.

923 Once the ownership transfer is complete and bootstrap credentials are established, the Device
924 needs to engage with the bootstrap server to be provisioned with proper security credentials and
925 parameters. These parameters can include

- 926 • Security credentials through a credential management service (CMS), currently assumed
927 to be deployed in the same OBT.
- 928 • Access control policies and ACLs through an ACL provisioning service (APS), currently
929 assumed to be deployed in the same OBT, but may be part of AMS in future.

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

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

938 **5.4.2 Provisioning a bootstrap service**

939 The Device is discovered or programmed with the bootstrap parameters (BP), including the
940 metadata required to discover and interact with the Bootstrap server (BS). The Device is configured
941 with bootstrap credential (BC) required to communicate with BS securely.

942 In the Resource structure, the rowneruid Property in the /oic/sec/pstat Resource identifies the
943 bootstrap service (BSS).

944 When symmetric keys are used, the OC is used to derive the BC. However, when the Device is
945 capable of using asymmetric keys for ownership transfer and other provisioning processes, there
946 may not be a need for a cryptographic relationship between BC and OC.

947 Regardless of how the BC is created, the communication between Device and BS (and potentially
948 other servers) must be done securely. For instance when a pre-shared key is used for secure
949 connection with the Device, the oic.sec.bss service includes the oic.sec.cred Resource is
950 provisioned with the PSK.

951 **5.4.3 Provisioning other services**

952 To be able to support the use of potentially different device management service hosts, each
953 Device Secure Virtual Resource (SVR) has an associated Resource owner. The onboarding Device,
954 also known as DOXS, provisions rowneruuid Properties with the appropriate provider identity.

- 955 • Credential Management Service (CMS) : (/oic/sec/cred.rowneruuid)
- 956 • Access Manager Service (AMS) : (/oic/sec/acl.rowneruuid and /oic/sec/acl2.rowneruuid)

957 When these services are populated the Device may proactively request provisioning and verify
958 provisioning requests are authorized. Each of the services above must be performed securely and
959 thus require specific credentials to be provisioned. The DOXS service may initiate of any services
960 above by signaling the service provider Device(s) or by setting the appropriate vector in the 'tm'
961 Property of the Device's /oic/sec/pstat Resource. This will cause the Device to re-provision its
962 credential and or access Resources

963 **5.4.4 Credential provisioning**

964 Several types of credential may be configured in a /oic/sec/cred Resource. Currently, they include
965 at least the following credential types; pairwise symmetric keys, group symmetric keys, certificates,
966 asymmetric keys and signed asymmetric keys. Keys may be provisioned by a CMS (e.g.
967 "oic.sec.cms") or dynamically using a Diffie-Hellman key agreement protocol or through other
968 means.

969 The following describe an example on how a Device can update a PSK for a secure connection. A
970 Device may discover the need to update credentials, e.g. because a secure connection attempt
971 fails. The Device will then need to request credential update from a CMS. The Device may enter
972 credential-provisioning mode (e.g. /oic/sec/pstat.Cm=16) and may configure operational mode (e.g.
973 /oic/sec/pstat.Om="1") to request an update to its credential Resource. The CMS responds with a
974 new pairwise pre-shared key (PSK).

975 **5.4.5 Role assignment and provisioning**

976 The Servers, receiving requests for Resources they host, need to examine the role asserted by
977 the Client requesting the Resource and compare that role with the constraints described in their
978 ACLs corresponding to the services. Thus, a Client Device may need to be provisioned with one
979 or more role credentials.

980 Each Device holds the role information as a Property within the credential Resource. Thus, it is
981 possible that a Client, seeking a role provisioning, enters a mode where both its credentials and
982 ACLs can be provisioned (if they are provisioned by the same sever!). The provisioning
983 mode/status is typically indicated by the content of /oic/sec/pstat.

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

986 Alternatively, if the server has been provisioned with role information for a client, or the client has
987 previously asserted roles to the server, the client can assert a specific role with the CoAP payload:

988 e.g. GET /a/light?roleid={"role":"Role-A"}

989 The client has no way to know in advance what roles are provisioned on the server, and must
990 attempt an action and observe the server's response. If the response is permission denied, the
991 client learns that either the server is not provisioned with the role, or the ACLs are misconfigured.
992 If no specific role is specified in the CoAP payload, all provisioned roles are used in ACL
993 enforcement. When a server has multiple roles provisioned for a client, access to a Resource is
994 granted if it would be granted under any of the roles.

995 **5.4.6 ACL provisioning**

996 During ACL provisioning, the Device establishes a secure connection to an APS (or bootstrap
997 server, if it is hosting the APS). The APS will instantiate or update Device ACLs according to the
998 ACL policy.

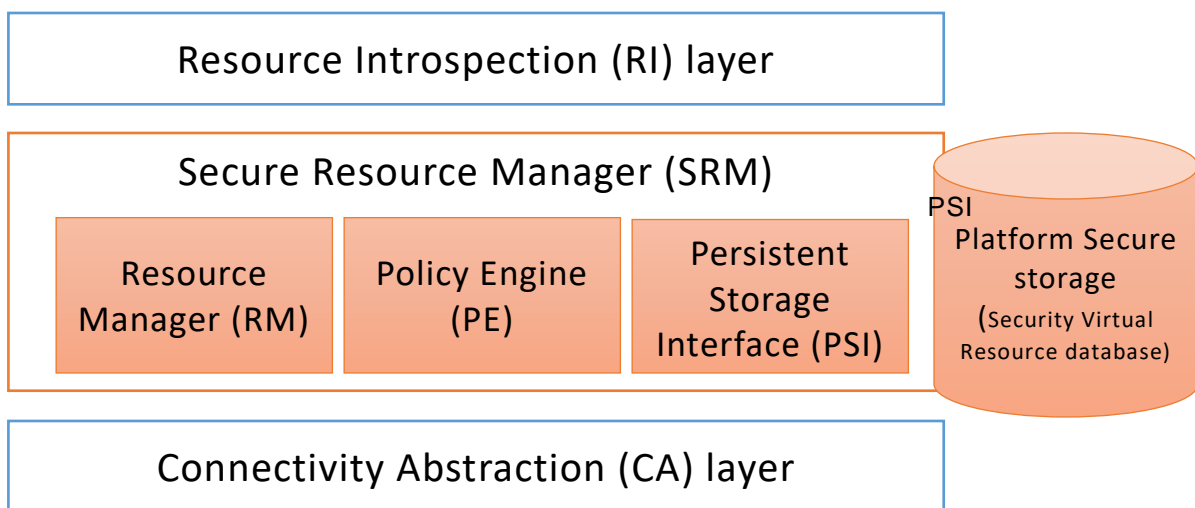
999 The Device and APS may establish an observer relationship such that when a change to the ACL
1000 policy is detected; the Device is notified triggering ACL provisioning.

1001 The APS (e.g. rt="oic.sec.aps") may digitally sign an ACL as part of issuing a /oic/sec/sacl
1002 Resource. The public key used by Servers to verify the signature may be provisioned as part of
1003 credential provisioning. A /oic/sec/cred Resource with an asymmetric key type or signed
1004 asymmetric key type is used. The PublicData Property contains the APS's public key.

1005 **5.5 Secure Resource Manager-(SRM)**

1006 SRM plays a key role in the overall security operation. In short, SRM performs both management
1007 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3
1008 main functional elements:

- 1009 • A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using
1010 PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3)
1011 Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a
1012 format that is consistent with device-specific data store format. However, the RM will use
1013 JSON format to marshal SVR data structures before be passed to PSI for storage, or travel
1014 off-device.
- 1015 • A Policy Engine (PE) that takes requests for access to SVRs and based on access control
1016 policies responds to the requests with either "ACCESS_GRANTED" or "ACCESS_DENIED".
1017 To make the access decisions, the PE consults the appropriate ACL and looks for best
1018 Access Control Entry (ACE) that can serve the request given the subject (Device or role)
1019 that was authenticated by DTLS.
- 1020 • Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate
1021 files in its own memory and storage. The SRM design is modular such that it may be
1022 implemented in the Platform's secure execution environment; if available.



1023 **Figure 12 – OCF's SRM Architecture**

1024 **5.6 Credential Overview**

1025 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional
1026 communication. Credentials can be symmetric or asymmetric. Each device stores secret and public
1027 parts of its own credentials where applicable, as well as credentials for other devices that have
1028 been provided by the OBT or a CMS. These credentials are then used in the establishment of
1029 secure communication sessions (e.g. using DTLS) to validate the identities of the participating
1030 parties. Role credentials are used once an authenticated session is established, to assert one or
1031 more roles for a device.

1032

1033 6 Security for the Discovery Process

1034 6.1 Introduction

1035 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,
1036 called links) for the Resources hosted by the Server, complemented by attributes about those
1037 Resources and possible further link relations. (in accordance to Section 10 in OCF Core
1038 Specification)

1039 6.2 Security Considerations for Discovery

1040 When defining discovery process, care must be taken that only a minimum set of Resources are
1041 exposed to the discovering entity without violating security of sensitive information or privacy
1042 requirements of the application at hand. This includes both data included in the Resources, as well
1043 as the corresponding metadata.

1044 To achieve extensibility and scalability, this specification does not provide a mandate on
1045 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on
1046 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any
1047 of the Resources.

1048 The `/oic/sec/acl2` Resource contains ACL entries governing access to the Server hosted
1049 Resources. (See Section 13.5)

1050 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery
1051 process itself needs to be secured. This specification sets the following requirements for the
1052 discovery process:

- 1053 1. Providing integrity protection for discovered Resources.
- 1054 2. Providing confidentiality protection for discovered Resources that are considered sensitive.

1055 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)
1056 on the known Resource `/oic/res`.

1057 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a
1058 Server cannot determine the identity of the requester. In such cases, a Server that wants to
1059 authenticate the Client before responding can list the secure discovery URI (e.g.
1060 `coaps://IP:PORT/oic/res`) in the unsecured `/oic/res` response. This means the secure discovery
1061 URI is by default discoverable by any Client. The Client will then be required to send a separate
1062 unicast request using DTLS to the secure discovery URI.

1063 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to
1064 `/oic/res` if and only if the Client has permissions to perform at least one of the CRUDN operations
1065 (i.e. the bitwise OR of the CRUDN flags must be true).

1066 For example, a Client with DeviceId "d1" makes a RETRIEVE request on the `/door` Resource
1067 hosted on a Server with DeviceId "d3" where d3 has the ACL2s below:

```
1068 {  
1069   "aclist2": [  
1070     {  
1071       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1072       "resources": [{"href": "/door"}],  
1073       "permission": 2, // RETRIEVE  
1074       "aceid": 1  
1075     }  
1076   ],  
1077   "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
```

```

1078 }
1079 {
1080   "aclist2": [
1081     {
1082       "subject": {"authority": "owner", "role": "owner"},
1083       "resources": [{"href": "/door"}],
1084       "permission": 2, // RETRIEVE
1085       "aceid": 2
1086     }
1087   ],
1088   "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1089 }
1090 {
1091   "aclist2": [
1092     {
1093       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1094       "resources": [{"href": "/door/lock"}],
1095       "permission": 4, // UPDATE
1096       "aceid": 3
1097     }
1098   ],
1099   "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1100 }
1101 {
1102   "aclist2": [
1103     {
1104       "subject": {"conntype": "anon-clear"},
1105       "resources": [{"href": "/light"}],
1106       "permission": 2, // RETRIEVE
1107       "aceid": 4
1108     }
1109   ],
1110   "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1111 }

```

1112 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when
1113 device "d1" does a discovery on the /oic/res Resource of the Server "d3", the response will include
1114 the URI of the "/door" Resource metadata. Client "d2" will have access to both the Resources.
1115 ACE2 will prevent "d4" from update.

1116 Discovery results delivered to d1 regarding d3's /oic/res Resource from the secure Interface:

```

1117 [
1118   {
1119     "href": "/door",
1120     "rt": ["oic.r.door"],
1121     "if": ["oic.if.b", "oic.ll"],
1122     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1123   }
1124 ]

```

1125 Discovery results delivered to d2 regarding d3's /oic/res Resource from the secure Interface:

```

1126 [
1127   {
1128     "href": "/door",
1129     "rt": ["oic.r.door"],
1130     "if": ["oic.if.b", "oic.ll"],

```

```
1131     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1132   },
1133   {
1134     "href": "/door/lock",
1135     "rt": ["oic.r.lock"],
1136     "if": ["oic.if.b"],
1137     "type": ["application/json", "application/exi+xml"]
1138   }
1139 ]
```

1140 Discovery results delivered to d4 regarding d3's /oic/res Resource from the secure Interface:

```
1141 [
1142   {
1143     "href": "/door/lock",
1144     "rt": ["oic.r.lock"],
1145     "if": ["oic.if.b"],
1146     "type": ["application/json", "application/exi+xml"],
1147     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1148   }
1149 ]
```

1150 Discovery results delivered to any device regarding d3's /oic/res Resource from the unsecure Interface:

```
1152 [
1153   {
1154     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1155     "href": "/light",
1156     "rt": ["oic.r.light"],
1157     "if": ["oic.if.s"]
1158   }
1159 ]
1160
```

1161 **7 Security Provisioning**

1162 **7.1 Device Identity**

1163 Each Device, which is a logical device, is identified with a device ID.

1164 Devices shall be identified by a Device ID value that is established as part of device onboarding.
1165 The `/oic/sec/doxm` Resource specifies the Device ID format (e.g. urn:uuid). Device IDs shall be
1166 unique within the scope of operation of the corresponding OCF network, and should be universally
1167 unique. Device ID uniqueness within the network shall be enforced at device onboarding. A Device
1168 OBT shall verify the chosen new device identifier does not conflict with other devices previously
1169 introduced into the network.

1170 Devices maintain an association of Device ID and cryptographic credential using a `/oic/sec/cred`
1171 Resource. Devices regard the `/oic/sec/cred` Resource as authoritative when verifying
1172 authentication credentials of a peer device.

1173 A Device maintains its device ID in the `/oic/sec/doxm` Resource. It maintains a list of credentials,
1174 both its own and other device credentials, in the `/oic/sec/cred` Resource. The device ID can be
1175 used to distinguish between a device's own credential, and credentials for other devices.
1176 Furthermore, the `/oic/sec/cred` Resource may contain multiple credentials for the device.

1177 Device ID shall be:

- 1178 • Unique
- 1179 • Immutable
- 1180 • Verifiable

1181 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the
1182 device as described later in this section.

1183 A physical device, referred to as a Platform in OCF specifications, may host multiple Devices. The
1184 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in the
1185 device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1186 Note: An OCF Platform may have a secure execution environment, which shall be used to secure
1187 unique identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to
1188 provide each Device with the appropriate and separate security.

1189 **7.1.1 Device Identity for Devices with UAID**

1190 **7.1.1.1 General**

1191 When a manufacturer certificate is used with certificates chaining to an OCF root CA (as specified
1192 in Section 7.1.1), the manufacturer shall include a Platform ID inside the certificate subject CN
1193 field. In such cases, the device ID may be created according to the Unique Authenticable Identifier
1194 (UAID) scheme defined in this section.

1195 For identifying and protecting Devices, the Platform Secure Execution Environment (SEE) may opt
1196 to generate new Dynamic Public Key Pair (DPKP) for each Device it is hosting, or it may opt to
1197 simply use the same public key credentials embedded by manufacturer; Embedded Platform
1198 Credential (EPC). In either case, the Platform SEE will use its Random Number Generator (RNG)
1199 to create a device identity called UAID for each Device. The UAID is generated using either EPC
1200 only or the combination of DPC and EPC if both are available. When both are available, the Platform
1201 shall use both key pairs to generate the UAID as described in this section.

1202 The Device ID is formed from the device's public keys and associated OCF Cipher Suite. The
1203 Device ID is formed by:

1204 1. Determining the OCF Cipher Suite of the Dynamic Public Key. The Cipher Suite curve must
1205 match the usage of the AlgorithmIdentifier used in SubjectPublicKeyInfo as intended for
1206 use with Device security mechanisms. Use the encoding of the CipherSuite as the 'csid'
1207 value in the following calculations. Note that if the OCF Cipher Suite for Dynamic Public
1208 key is different from the ciphersuite indicated in the Platform certificate (EPC), the OCF
1209 Cipher Suite shall be used below.

1210 2. From EPC extract the value of embedded public key. The value should correspond to the
1211 value of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate. In the
1212 following we refer to this as EPK. If the public key is extracted from a certificate, validate
1213 that the AlgorithmIdentifier matches the expected value for the CipherSuite within the
1214 certificate.

1215 3. From DPC Extract the value of the public key. The value should correspond to the value of
1216 subjectPublicKey defined in SubjectPublicKeyInfo. In the following we refer to this as DPK.

1217 4. Using the hash for the Cipher Suite calculate:
1218 $h = \text{hash}(\text{'uid'} \mid \text{csid} \mid \text{EPK} \mid \text{DPK} \mid \text{<other_info>})$

1219 Other_info could be 1) device type as indicated in /oic/d (could be read-only and set by
1220 manufacturer), 2) in case there are two sets of public key pairs (one embedded, and one
1221 dynamically generated), both public keys would be included.

1222 5. Truncate to 128 bits by taking the first 128 bits of h
1223 $\text{UAID} = h[0:16] \# \text{leftmost } 16 \text{ octets}$

1224 6. Convert the binary UAID to a ASCII string by
1225 $\text{USID} = \text{base27encode}(\text{UAID})$

```
1226 def base_N_encode(octets, alphabet):  
1227     long_int = string_to_int( octets )  
1228     text_out = ''  
1229     while long_int > 0:  
1230         long_int, remainder = divmod(long_int, len(alphabet))  
1231         text_out = alphabet[remainder] + text_out  
1232     return text_out
```

```
1233  
1234     b27chars = 'ABCDEFGHJKMNPQRTWXYZ2346789'  
1235     def b27encode(octet_string):  
1236         """Encode a octet string using 27 characters. """  
1237         return base_N_encode(octet_string, _b27chars )
```

1238 7. Append the string value of USID to 'urn:usid:' to form the final string
1239 value of the Device ID
1240 urn:usid:ABXW....

1241 Whenever the public key is encoded the format described in RFC 7250 for SubjectPublicKeyInfo
1242 shall be used.

1243 7.1.1.2 Validation of UAID

1244 To be able to use the newly generated Device ID (UAID) and public key pair (DPC), the device
1245 Platform shall use the embedded private key (corresponding to manufacturer embedded public key
1246 and certificate) to sign a token vouching for the fact that it (the Platform) has in fact generated the
1247 DPC and UAID and thus deferring the liability of the use of the DPC to the new device owner. This
1248 also allows the ecosystem to extend the trust from manufacturer certificate to a device issued

1249 certificate for use in the new DPC and UAID. The degree of trust is in dependent of the level of
1250 hardening of the device SEE.

1251 Dev_Token=Info, Signature(hash(info))

1252 Signature algorithm=ECDSA (can be same algorithm as that in EPC or that possible for DPC)

1253 Hash algorithm=SHA256

1254 Info=UAID| <Platform ID> | UAID_generation_data | validity

1255 UAID_generation_data=data passed to the hash algorithm used to generate UAID.

1256 Validity=validity period in days (how long the token will be valid)

1257 **7.2 Device Ownership**

1258 This is an informative section. Devices are logical entities that are security endpoints that have an
1259 identity that is authenticable using cryptographic credentials. A Device is 'un-owned' when it is first
1260 initialized. Establishing device ownership is a process by which the device asserts it's identity to
1261 an OBT and the OBT asserts its identity to the device. This exchange results in the device changing
1262 its ownership state, thereby preventing a different OBT from asserting administrative control over
1263 the device.

1264 The ownership transfer process starts with the OBT discovering a new device that is "un-owned"
1265 through examination of the "Owned" Property of the /oic/sec/doxm Resource of the new device. At
1266 the end of ownership transfer, the following is accomplished:

- 1267 1. Establish a secure session between new device and the OBT.
- 1268 2. Optionally asserts any of the following:
 - 1269 a. Proximity (using PIN) of the OBT to the Platform.
 - 1270 b. Manufacturer's certificate asserting Platform vendor, model and other Platform
1271 specific attributes.
- 1272 3. Determines the device identifier.
- 1273 4. Determines the device owner.
- 1274 5. Specifies the device owner (e.g. Device ID of the OBT).
- 1275 6. Provisions the device with owner's credentials.
- 1276 7. Sets the 'Owned" state of the new device to TRUE.

1277 **7.3 Device Ownership Transfer Methods**

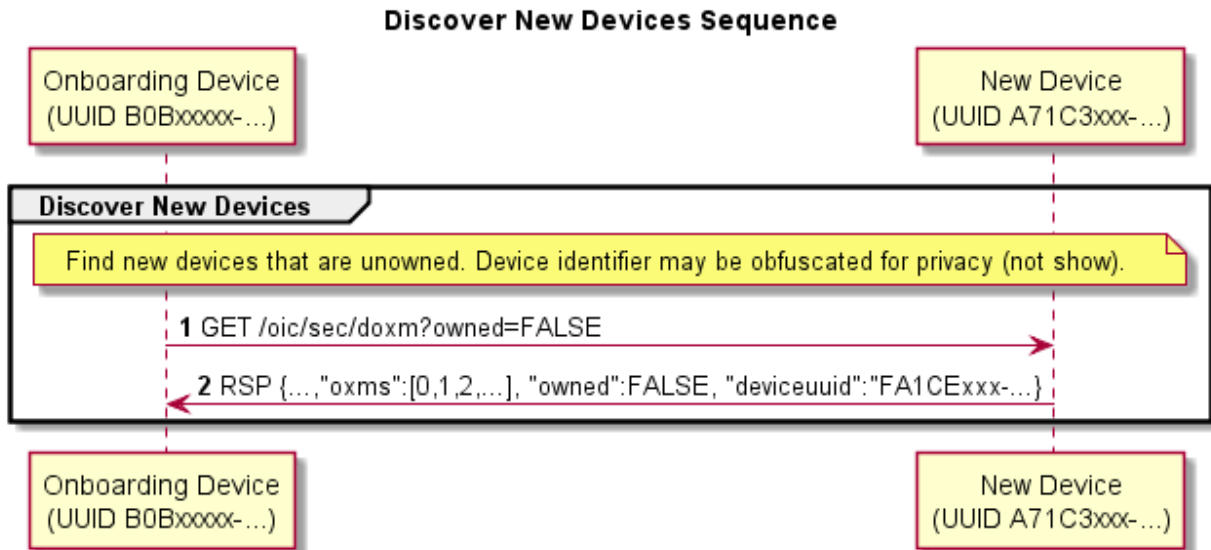
1278 **7.3.1 OTM implementation requirements**

1279 This document provides specifications for several methods for ownership transfer. Implementation
1280 of each individual ownership transfer method is considered optional. However, each device shall
1281 implement at least one of the ownership transfer methods not including vendor specific methods.

1282 All owner transfer methods (OTMs) included in this document are considered optional. Each vendor
1283 is required to choose and implement at least one of the OTMs specified in this specification. The
1284 OCF, does however, anticipate vendor-specific approaches will exist. Should the vendor wish to
1285 have interoperability between an vendor-specific owner transfer method and and OBTs from other
1286 vendors, the vendor must work directly with OBT vendors to ensure interoperability.
1287 Notwithstanding, standardization of OTMs is the preferred approach. In such cases, a set of

1288 guidelines is provided below to help vendors in designing vendor-specific OTMs. (See Section
 1289 7.3.6).

1290 The device owner transfer method (doxm) Resource is extensible to accommodate vendor-defined
 1291 methods. All OTM methods shall facilitate allowing the OBT to determine which OC is most
 1292 appropriate for a given new device within the constraints of the capabilities of the device. The OBT
 1293 will query the credential types that the new device supports and allow the OBT to select the
 1294 credential type from within device constraints.



1295
 1296
 1297

Figure 13 - Discover New Device Sequence

Step	Description
1	The OBT queries to see if the new device is not yet owned.
2	The new device returns the /oic/sec/doxm Resource containing ownership status and supported owner transfer methods. 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. Section 7.3.9 provides security considerations regarding selecting an owner transfer method.

1298

Table 2 - Discover New Device Details

1299 Vendor-specific device owner transfer methods shall adhere to the /oic/sec/doxm Resource
 1300 specification for OCs that results from vendor-specific device owner transfer. Vendor-specific
 1301 methods should include provisions for establishing trust in the new device by the OBT an optionally
 1302 establishing trust in the OBT by the new device.

1303 The end state of a vendor-specific owner transfer method shall allow the new device to authenticate
 1304 to the OBT and the OBT to authenticate to the new device.

1305 Additional provisioning steps may be applied subsequent to owner transfer success leveraging the
 1306 established session, but such provisioning steps are technically considered provisioning steps that
 1307 an OBT may not anticipate hence may be invalidated by OBT provisioning.

1308 **7.3.2 SharedKey Credential Calculation**

1309 The SharedKey credential is derived using a PRF that accepts the key_block value resulting from
1310 the DTLS handshake used for onboarding. The Server and Device OBT shall use the following
1311 calculation to ensure interoperability across vendor products:

1312 SharedKey = PRF(Secret, Message);

1313 Where:

- 1314 - PRF shall use TLS 1.2 PRF defined by RFC5246 section 5.
- 1315 - Secret is the key_block resulting from the DTLS handshake
 - 1316 ▪ See RFC5246 Section 6.3
 - 1317 ▪ The length of key_block depends on cipher suite.
 - 1318 • (e.g. 96 bytes for TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - 1319 40 bytes for TLS_PSK_WITH_AES_128_CCM_8)
- 1320 - Message is a concatenation of the following:
 - 1321 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
 - 1322 • See "Section 13.2.2 OCF defined owner transfer methods for specific DoxmTypes"
 - 1323 ▪ OwnerID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
 - 1324 • Use raw bytes as specified in RFC4122 section 4.1.2
 - 1325 ▪ Device ID is new device's UUID Device ID
 - 1326 • Use raw bytes as specified in RFC4122 section 4.1.2
- 1327 - SharedKey Length will be 32 octets.
 - 1328 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the leftmost 16 octets will be used.
 - 1329 DTLS sessions using 256 bit encryption cipher suites will use all 32 octets.

1330 **7.3.3 Certificate Credential Generation**

1331 The Certificate Credential will be used by Devices for secure bidirectional communication. The
1332 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used
1333 to mutually establish the authenticity of the Device. The onboarding details for certificate
1334 generation will be specified in a later version of this specification.

1335 **7.3.4 Just-Works Owner Transfer Method**

1336 **7.3.4.1 General**

1337 Just-works owner transfer method creates a symmetric key credential that is a pre-shared key
1338 used to establish a secure connection through which a device should be provisioned for use within
1339 the owner's network. Provisioning additional credentials and Resources is a typical step following
1340 ownership establishment. The pre-shared key is called SharedKey.

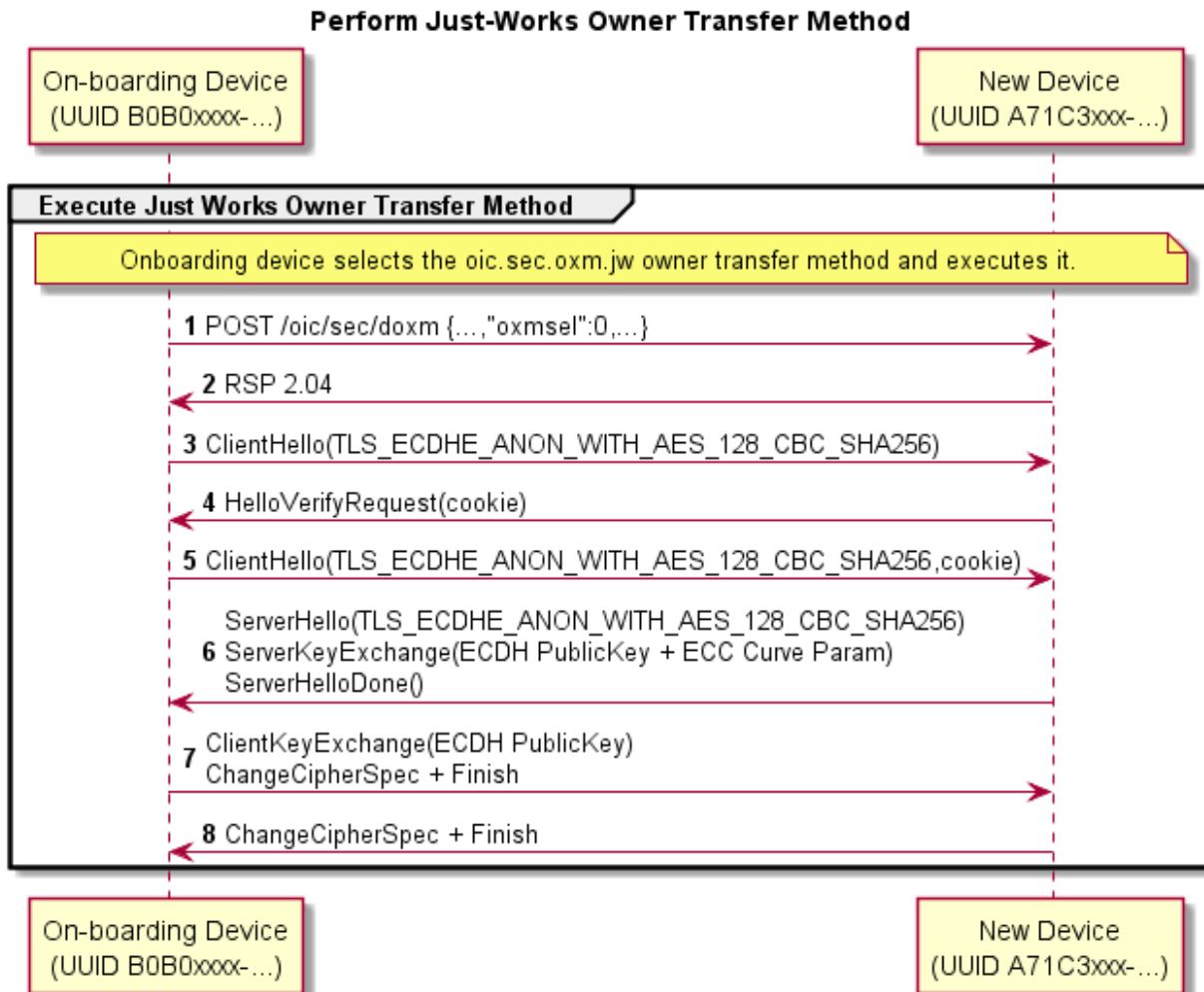
1341 The ownership transfer process starts with the OBT discovering a new device that is "un-owned"
1342 through examination of the "owned" Property of the /oic/sec/doxm Resource at the Device hosted
1343 by the new device.

1344 Once the OBT asserts that the device is un-owned, when performing the Just-works owner transfer
1345 method, the OBT relies on DTLS key exchange process where an anonymous Elliptic Curve Diffie-
1346 Hellman (ECDH) is used as a key agreement protocol.

1347 The following OCF-defined vendor-specific ciphersuites are used for the Just-works owner transfer
1348 method.

1349 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
1350 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

1351 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for
 1352 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.



1353
1354

Figure 14 – A Just Works Owner Transfer Method

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

1355

Table 3 – A Just Works Owner Transfer Method Details

1356 **7.3.4.2 Security Considerations**

1357 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this
 1358 method presumes that both the OBT and the new device perform the 'just-works' method assumes
 1359 onboarding happens in a relatively safe environment absent of an attack device.

1360 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to
 1361 the device.

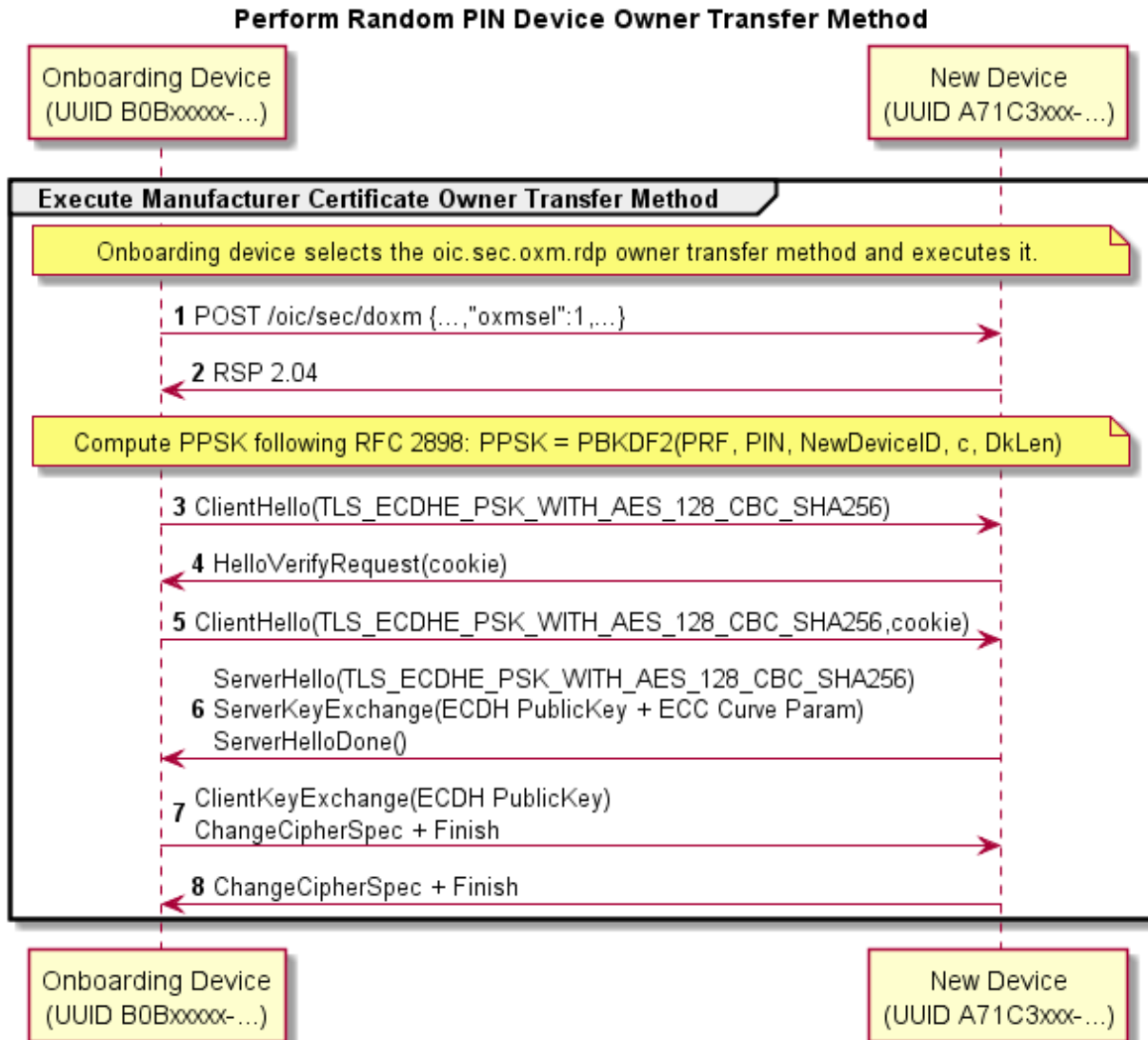
1362 The new device should use a temporal device ID prior to transitioning to an owned device while it
1363 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-
1364 temporal device ID that could differ from the temporal value during the secure session in which
1365 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not conflict
1366 with a Device ID already in use. If it is already in use the existing credentials are used to establish
1367 a secure session.

1368 An un-owned Device that also has established device credentials might be an indication of a
1369 corrupted or compromised device.

1370 **7.3.5 Random PIN Based Owner Transfer Method**

1371 **7.3.5.1 General**

1372 The Random PIN method establishes physical proximity between the new device and the OBT can
1373 prevent man-in-the-middle attacks. The Device generates a random number that is communicated
1374 to the OBT over an out-of-band channel. The definition of out-of-band communications channel is
1375 outside the scope of the definition of device owner transfer methods. The OBT and new Device
1376 use the PIN in a key exchange as evidence that someone authorized the transfer of ownership by
1377 having physical access to the new Device via the out-of-band-channel.



1379
1380

Figure 15 – Random PIN-based Owner Transfer Method

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

1381

Table 4 – Random PIN-based Owner Transfer Method Details

1382 The random PIN-based device owner transfer method uses a pseudo-random function (PBKDF2)
1383 defined by RFC2898 and a PIN exchanged via an out-of-band method to generate a pre-shared
1384 key. The PIN-authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a
1385 PSK.

1386 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)

- 1387 The PBKDF2 function has the following parameters:
- 1388 - PRF – Uses the TLS 1.2 PRF defined by RFC5246.
 - 1389 - PIN – obtain via out-of-band channel.
 - 1390 - Device ID – UUID of the new device.
 - 1391 • Use raw bytes as specified in RFC4122 section 4.1.2
 - 1392 - c – Iteration count initialized to 1000
 - 1393 - dkLen – Desired length of the derived PSK in octets.

1394 7.3.5.3 Security Considerations

1395 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with
1396 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials
1397 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,
1398 allows an attacker to masquerade as the onboarded device.

1399 It is recommended that the entropy of the PIN be enough to withstand an online brute-force attack,
1400 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-9a-z), or
1401 a 7 character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack (MITM) is
1402 when the attacker is active on the network and can intercept and modify messages between the
1403 OBT and device. In the MITM attack, the attacker must recover the PIN from the key exchange
1404 messages in "real time", i.e., before the peers time out and abort the connection attempt. Having
1405 recovered the PIN, he can complete the authentication step of key exchange. The guidance given
1406 here calls for a minimum of 40 bits of entropy, however, the assurance this provides depends on
1407 the resources available to the attacker. Given the parallelizable nature of a brute force guessing
1408 attack, the attack enjoys a linear speedup as more cores/threads are added. A more conservative
1409 amount of entropy would be 64 bits. Since the Random PIN OTM requires using a DTLS ciphersuite
1410 that includes an ECDHE key exchange, the security of the Random PIN OTM is always at least
1411 equivalent to the security of the JustWorks OTM.

1412 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN.
1413 The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess
1414 in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an effective
1415 way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify the
1416 reduction, since an X-fold increase in time spent by the honest peers does not directly translate to
1417 an X-fold increase in time by the attacker. This asymmetry is because the attacker may use
1418 specialized implementations and hardware not available to honest peers. For this reason, when
1419 deciding how much entropy to use for a PIN, it is recommended that implementers assume
1420 PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

1421 The Random PIN device owner transfer method security depends on an assumption that a secure
1422 out-of-band method for communicating a randomly generated PIN from the new device to the OBT
1423 exists. If the OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of
1424 the PIN, and the attacks described above apply. The out-of-band mechanism should be chosen
1425 such that it requires proximity between the OBT and the new device. The attacker is assumed to
1426 not have compromised the out-of-band-channel. As an example OOB channel, the device may
1427 display a PIN to be entered into the OBT software. Another example is for the device to encode
1428 the PIN as a 2D barcode and display it for a camera on the OBT device to capture and decode.

1429 7.3.6 Manufacturer Certificate Based Owner Transfer Method

1430 7.3.6.1 General

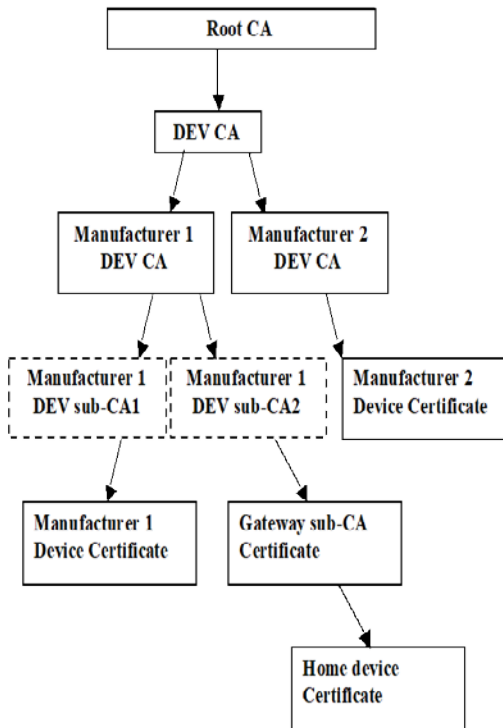
1431 The manufacturer certificate-based owner transfer method shall use a certificate embedded into
1432 the device by the manufacturer and may use a signed OBT, which determines the Trust Anchor
1433 between the device and the OBT.

1434 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with
1435 certificate data to authenticate their identities with the OBT in the process of bringing a new device
1436 into operation on a user's network. The onboarding process involves several discrete steps:

- 1437 1) Pre-on-board conditions
 - 1438 a. The credential element of the Device's credential Resource (/oic/sec/cred) containing the
1439 manufacturer certificate shall be identified by the following properties:
 - 1440 i. the subject Property shall refer to the Device
 - 1441 ii. the credusage Property shall contain the string "oic.sec.cred.mfgcert" to
1442 indicate that the credential contains a manufacturer certificate
 - 1443 b. The manufacturer certificate chain shall be contained in the identified credential
1444 element's publicdata Property with the optionaldata Property containing the Trust Anchor
 - 1445 c. The device shall contain a unique and immutable ECC asymmetric key pair.
 - 1446 d. If the device requires authentication of the OBT as part of ownership transfer, it is
1447 presumed that the OBT has been registered and has obtained a certificate for its unique
1448 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.
 - 1449 e. User has configured the OBT app with network access info and account info (if
1450 any).
- 1451 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally
1452 the Device may authenticate the OBT to verify the OBT signature.
- 1453 3) If authentication fails, the Device shall indicate the reason for failure and return to the
1454 Ready for OTM state. If authentication succeeds, the device and OBT shall establish an
1455 encrypted link in accordance with the negotiated cipher suite.

1456 7.3.6.2 Certificate Profiles

1457 Within the Device PKI, the following format shall be used for the `subject` within the certificates.
1458 It is anticipated that there may be N distinct roots for scalability and failover purposes. The vendor
1459 creating and operating root will be approved by the OCF based on due process described in
1460 Certificate Policy (CP) document and appropriate RFP documentation. Each root may issue one
1461 or more DEV CAs, which in turn issue Manufacturer DEV CAs to individual manufacturers. A
1462 manufacturer may decide to request for more than one Manufacturer CAs. Each Manufacturer CA
1463 issues one or more Device Sub-CAs (up to M) and issues one or more OCSP responders (up to
1464 O). For now we can assume that revocation checking for any CA certificates is handled by CRLs
1465 issued by the higher level CAs.



1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491

Figure 16 – Manufacturer Certificate Hierarchy

- Root CA: C=<country where the root was created>, O=<name of root CA vendor>, OU=OCF Root CA, CN=OCF (R) Device Root-CA<n>
- DEV CA: C=<country for the DEV CA>, O=<name of root CA vendor>, OU=OCF DEV CA, CN=<name of DEV CA defined by root CA vendor>
- Manufacturer DEV CA: C=<country where Manufacturer DEV CA is registered>, O=<name of root CA vendor>, OU=OCF Manufacturer DEV CA, CN=<name defined by manufacturer><m>
- Device Sub-CA: C=<country device sub-CA>, O=<name of root CA vendor>, OU=OCF Manufacturer Device sub-CA, OU=<defined by Manufacturer>, CN=<defined by manufacturer>
- For Device Sub-CA Level OCSP Responder: C=<country of device Sub-CA>, O=<name of root CA vendor>, OU=OCF Manufacturer OCSP Responder <o>, CN=<name defined by CA vendor >
- Device cert: C=<country>, O=<manufacturer>, OU=Device, CN=<device Type><single space (i.e., " ")><device model name>
 - The following optional naming elements MAY be included between the OU=OCF (R) Devices and CN= naming elements. They MAY appear in any order:
 OU=chipsetID: <chipsetID>, OU=<device type>, OU=<device model name>
 OU=<mac address> OU=<device security profile>
- Gateway Sub-CA: C=<country>, O=<manufacturer>, OU=<manufacture name> Gateway sub-CA, CN=<name defined by manufacturer>, <unique Gateway identifier generated with UAID method>
- Home Device Cert: C=<country>, O=<manufacturer>, OU=Non-Device cert, OU=<Gateway UAID>, CN=<device Tyle>

1492 Technical Note regarding Gateway Sub-CA: If a manufacturer decides to allow its Gateways to act
1493 as Gateway Sub-CA, it needs to accommodate this by setting the proper value on path-length-
1494 constraint value within the Device Sub-CA certificate, to allow the latter sub-CA to issue CA
1495 certificates to Gateway Sub-CAs. Given that the number of Gateway Sub-CAs can be very large a
1496 numbering scheme should be used for Gateway Sub-CA ID and given the Gateway does have
1497 public key pair, UAID algorithm shall be used to calculate the gateway identifier using a hash of
1498 gateway public key and inserted inside subject field of Gateway Sub-CA certificate.

1499 A separate Device Sub-CA shall be used to generate Gateway Sub-CA certificates. This Device
1500 Sub-CA shall not be used for issuance of non-Gateway device certificates.

1501 CRLs including Gateway Sub-CA certificates shall be issued on monthly basis, rather than quarterly
1502 basis to avoid potentially large liabilities related to Gateway Sub-CA compromise.

1503 Device certificates issued by Gateway Sub-CA shall include an OU=Non-Device cert, to indicate
1504 that they are not issued by an OCF governed CA.

1505 When the naming element is DirectoryString (i.e., O=, OU=) either PrintableString or UTF8String
1506 shall be used. The following determines which choice is used:

- 1507 • PrintableString only if it is limited to the following subset of US ASCII characters (as
1508 required by ASN.1):
1509 A, B, ..., Z
1510 a, b, ..., z
1511 0, 1, ...9,
1512 (space) ' () + , - . / : = ?
- 1513 • UTF8String for all other cases, e.g., subject name attributes with any other characters or
1514 for international character sets.

1515 A CVC CA is used by a trusted organization to issue CVC code signing certificates to software
1516 providers, system administrators, or other entities that will sign software images for the Devices.
1517 A CVC CA shall not sign and issue certificates for any specialization other than code signing. In
1518 other words, the CVC CA shall not sign and issue certificates that belong to any branches other
1519 than the CVC branch.

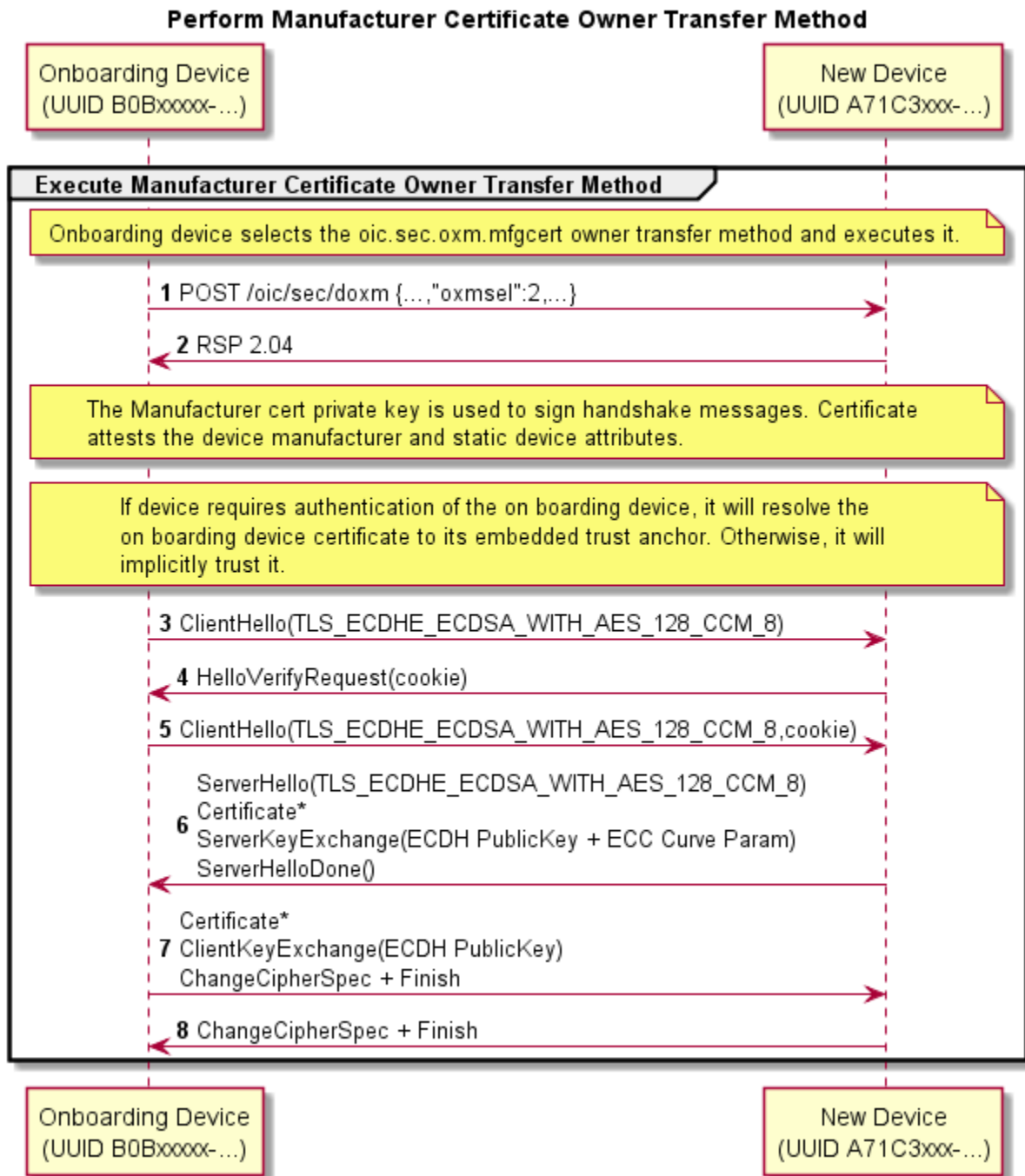
1520 7.3.6.3 Certificate Owner Transfer Sequence Security Considerations

1521 In order for full, mutual authentication to occur between the device and the OBT, both the device
1522 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This implies
1523 that OCF may need to obtain services from a Certificate Authority (e.g. Symantec, Verisign, etc.)
1524 to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors are derived.

1525 The OBT shall authenticate the device during onboarding. However, the device is not required to
1526 authenticate the OBT due to potential resource constraints on the device.

1527 In the case where the Device does NOT authenticate the OBT software, there is the possibility of
1528 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,
1529 which can compromise network access credentials and/or personal information.

7.3.6.4 Manufacturer Certificate Based Owner Transfer Method Sequence



1531
1532
1533

Figure 17 – Manufacturer Certificate Based Owner Transfer Method Sequence

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

Table 5 – Manufacturer Certificate Based Owner Transfer Method Details

1534

1535 **7.3.6.5 Security Considerations**

1536 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of
1537 assurance that the private key cannot be compromised.

1538 The Platform manufacturer issues the manufacturer certificate and attests the private key
1539 protection mechanism.

1540 The manufacturer certificate defines its uniqueness properties.

1541 There may be multiple Device instances hosted by a Platform containing a single manufacturer
1542 certificate

1543 **7.3.7 Vendor Specific Owner Transfer Methods**

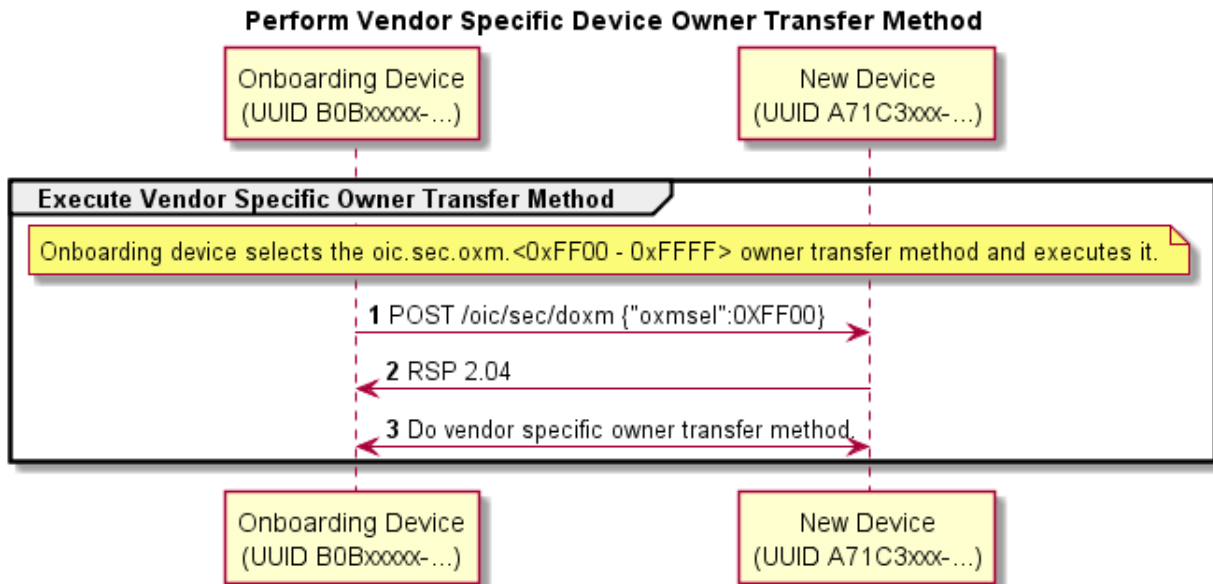
1544 **7.3.7.1 General**

1545 The OCF anticipates situations where a vendor will need to implement an owner transfer method
1546 that accommodates manufacturing or Device constraints. The Device owner transfer method
1547 resource is extensible for this purpose. Vendor-specific owner transfer methods must adhere to a
1548 set of conventions that all owner transfer methods follow.

- 1549 • The OBT must determine which credential types are supported by the Device. This is
1550 accomplished by querying the Device's /oic/sec/doxm Resource to identify supported
1551 credential types.
- 1552 • The OBT provisions the Device with OC(s).
- 1553 • The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1554 • The OBT will supply second carrier settings sufficient for accessing the owner's network
1555 subsequent to ownership establishment.
- 1556 • The OBT may perform additional provisioning steps but must not invalidate provisioning
1557 tasks to be performed by a bootstrap or security service.

1558 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1559



1560
1561

Figure 18 – Vendor-specific Owner Transfer Sequence

Step	Description
1, 2	The OBT selects a vendor-specific owner transfer method.
3	The vendor-specific owner transfer method is applied

1562

Table 6 – Vendor-specific Owner Transfer Details

1563 **7.3.7.3 Security Considerations**

1564 The vendor is responsible for considering security threats and mitigation strategies.

1565 **7.3.8 Establishing Owner Credentials**

1566 Once the OBT and the new Device have authenticated and established an encrypted connection
1567 using one of the defined OTM methods.

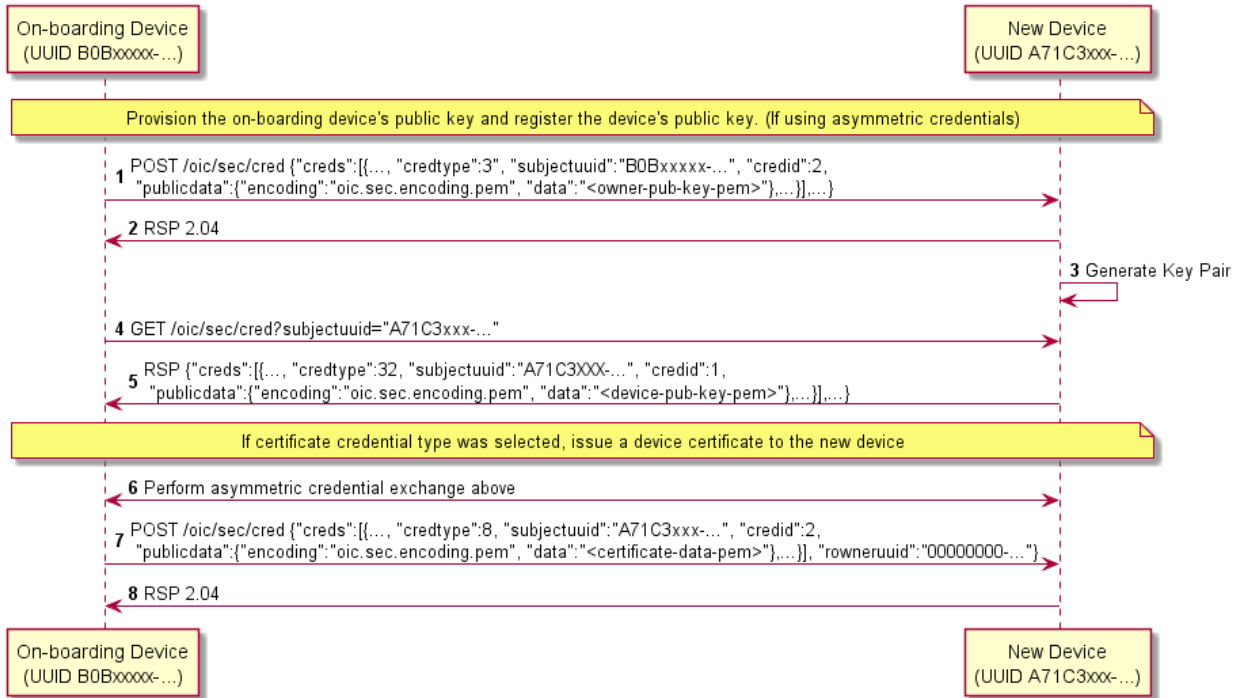
1568 Owner credentials may consist of certificates signed by the OBT or other authority, user network
1569 access information, provisioning functions, shared keys, or Kerberos tickets.

1570 The OBT might then provision the new Device with additional credentials for Device management
1571 and Device-to-Device communications. These credentials may consist of certificates with
1572 signatures, UAID based on the Device public key, PSK, etc.

1573 The steps for establishing Device's owner credentials (OC) are detailed below:

- 1574 a. The OBT shall establish the Device ID and Device owner uuid - Figure 19
- 1575 b. The OBT then establishes Device's owner credentials (OC) - Figure 20. This can be either:
 - 1576 i. Symmetric credential - Figure 21

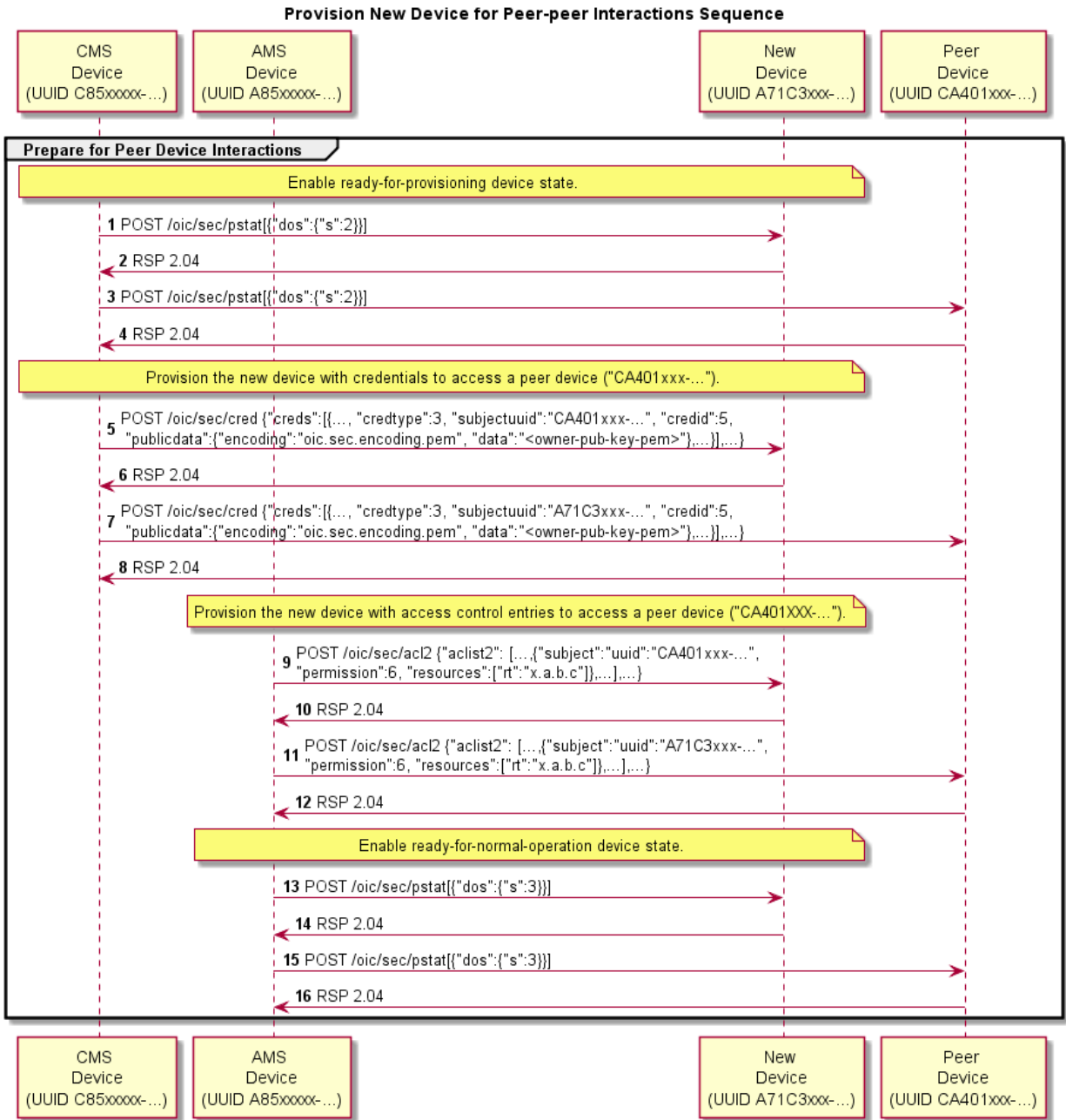
Asymmetric Owner Credential (OC) Assignment Sequence



1578

1579 ii. Figure 22

1580 c. Configure Device services. - Figure 23



1582
1583 d. Figure 24

1584 These credentials may consist of certificates signed by the OBT or other authority, user network
1585 access information, provisioning functions, shared keys, or Kerberos tickets.

1586 The OBT might then provision the new Device with additional credentials for Device management
1587 and Device-to-Device communications. These credentials may consist of certificates with
1588 signatures, UAID based on the Device public key, PSK, etc.

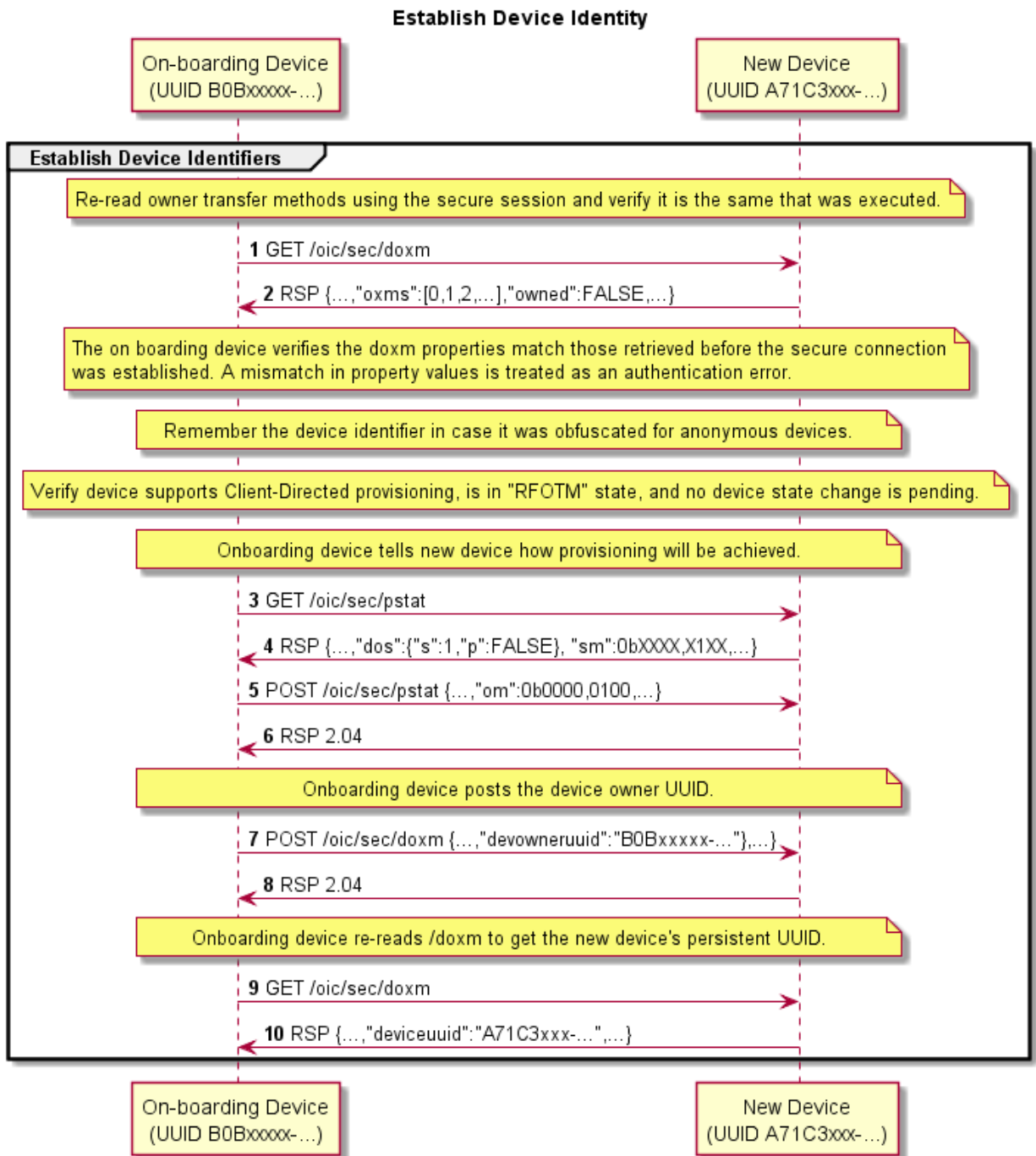


Figure 19 - Establish Device Identity Flow

1589
1590

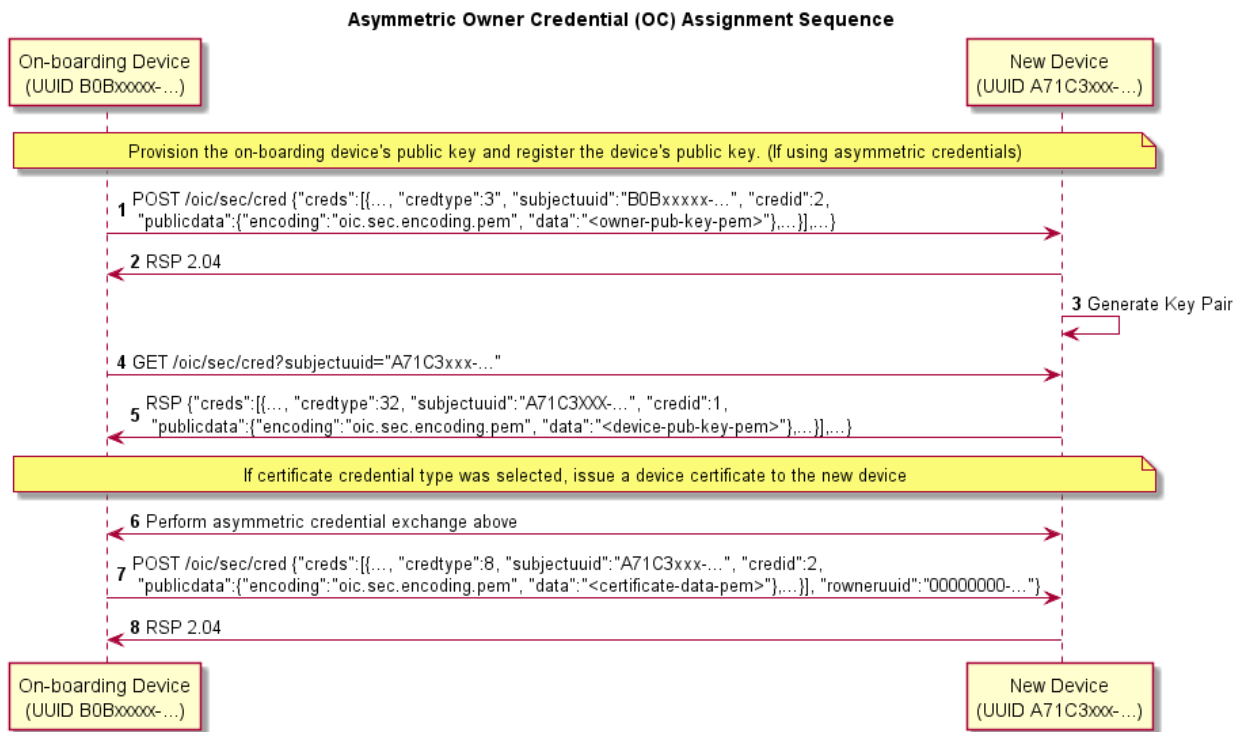
Step	Description
1, 2	The OBT obtains the doxm properties again, using the secure session. It verifies that these properties match those retrieved before the authenticated connection. A mismatch in parameters is treated as an authentication error.
3, 4	The OBT queries to determine if the Device is operationally 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.

1591

Table 7 - Establish Device Identity Details

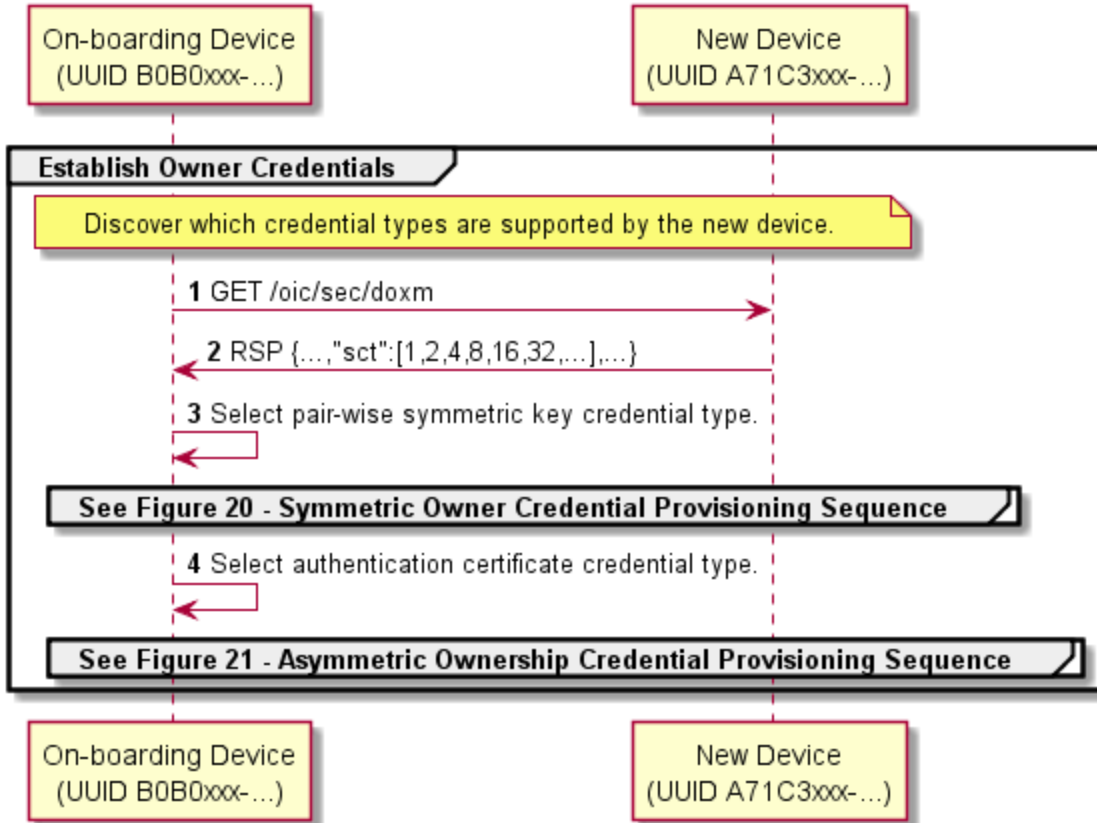
1592 group

See



1593

Establish Owner Credentials Sequence



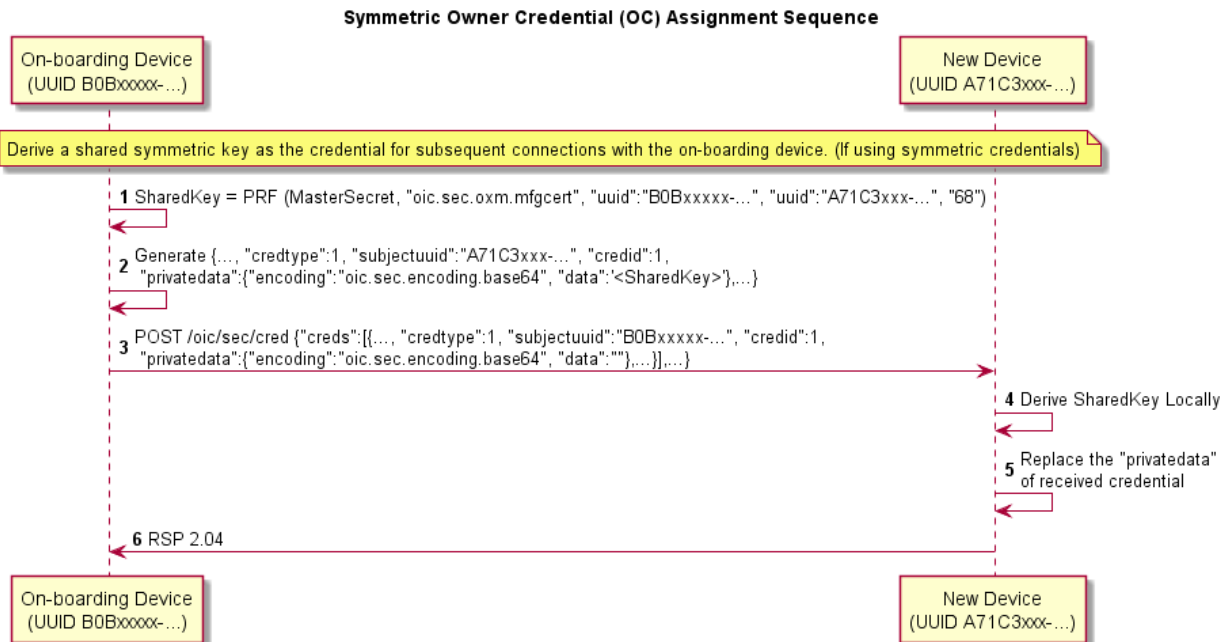
1594
1595

Figure 20 – Owner Credential Selection Provisioning Sequence

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.

1596

Table 8 - Owner Credential Selection Details



1597
1598

Figure 21 - Symmetric Owner Credential Provisioning Sequence

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.

1599

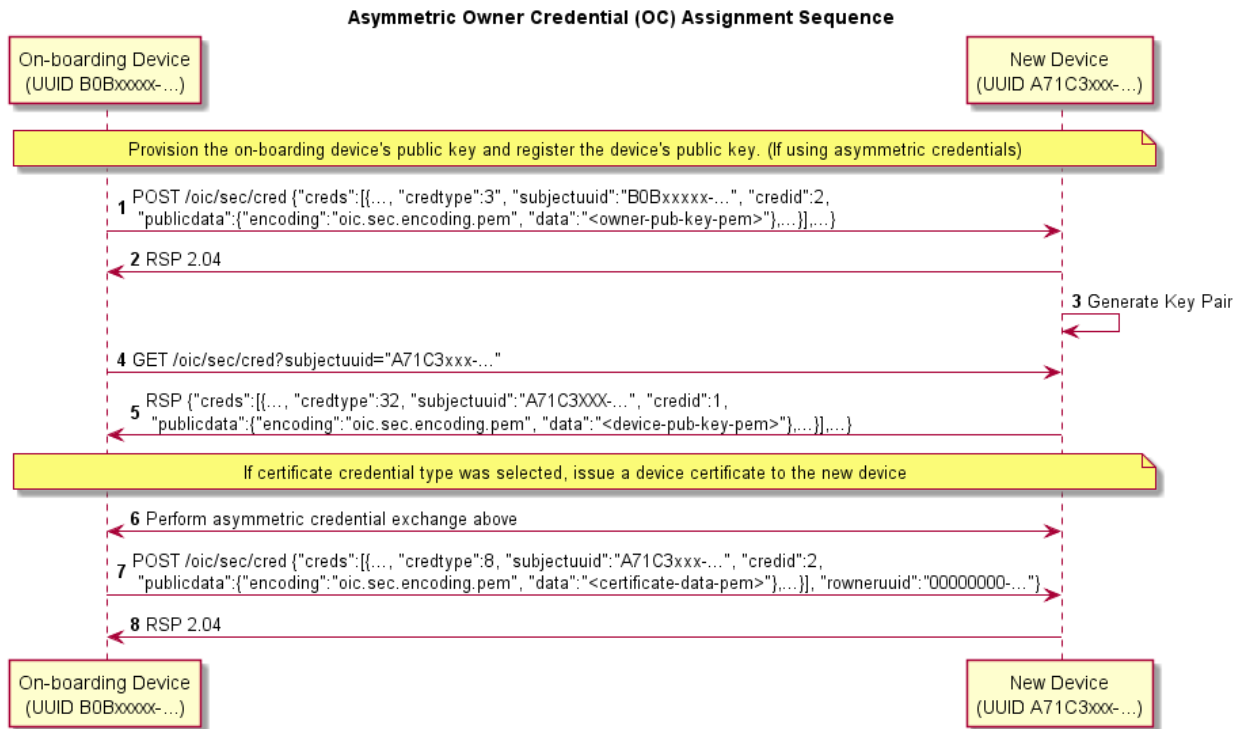
Table 9 - Symmetric Owner Credential Assignment Details

1600 In particular, if the OBT selects symmetric owner credentials:

- 1601 • The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
1602 described in Section 7.3.2.
- 1603 • The OBT shall send an empty key to the new Device's /oic/sec/cred Resource, identified
1604 as a symmetric pair-wise key.
- 1605 • Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
1606 generate the Shared Key using the SharedKey Credential Calculation method described in
1607 Section 7.3.2 and store it with the owner credential.

- The new Device shall use the Shared Key owner credential(s) stored via the /oic/sec/cred Resource to authenticate the owner during subsequent connections.

1610



1611
1612

Figure 22 - Asymmetric Ownership Credential Provisioning Sequence

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.

1613

Table 10 – Asymmetric Owner Credential Assignment Details

1614 If the OBT selects asymmetric owner credentials:

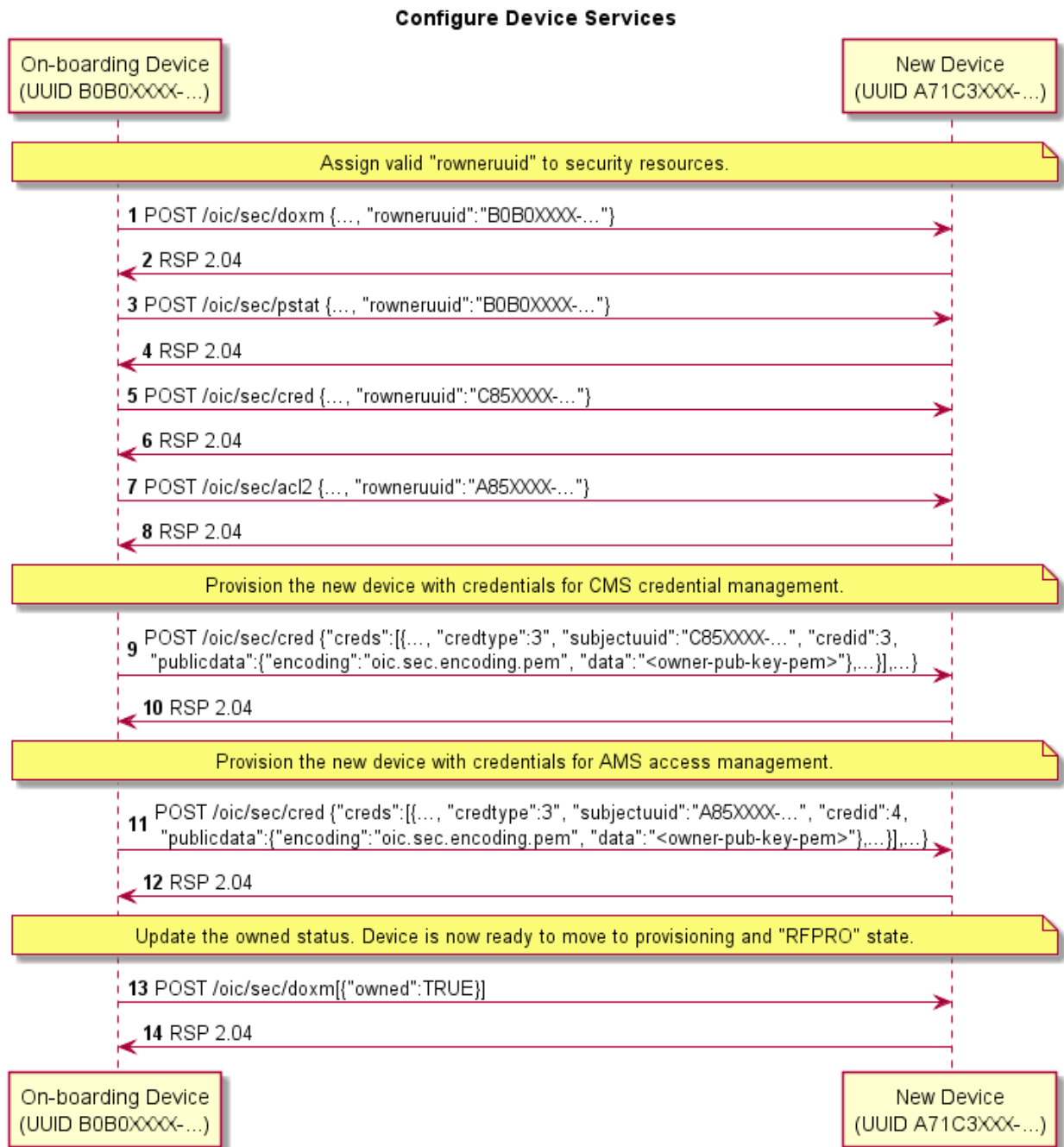
- The OBT shall add its public key to the new Device's /oic/sec/cred Resource, identified as an Asymmetric Encryption Key.
- The OBT shall query the /oic/sec/cred Resource from the new Device, supplying the new Device's UUID via the SubjectID query parameter. In response, the new Device shall return

1617
1618

1619 the public Asymmetric Encryption Key, which the OBT shall retain for future owner
1620 authentication of the new Device.

1621 If the OBT selects certificate owner credentials:

- 1622 • The OBT shall create a certificate or certificate chain with the leaf certificate containing the
1623 public key returned by the new Device, signed by a mutually-trusted CA, and complying
1624 with the Certificate Credential Generation requirements defined in Section 7.3.3.
- 1625 • The OBT shall add the newly-created certificate chain to the /oic/sec/cred Resource,
1626 identified as an Asymmetric Signing Key with Certificate.



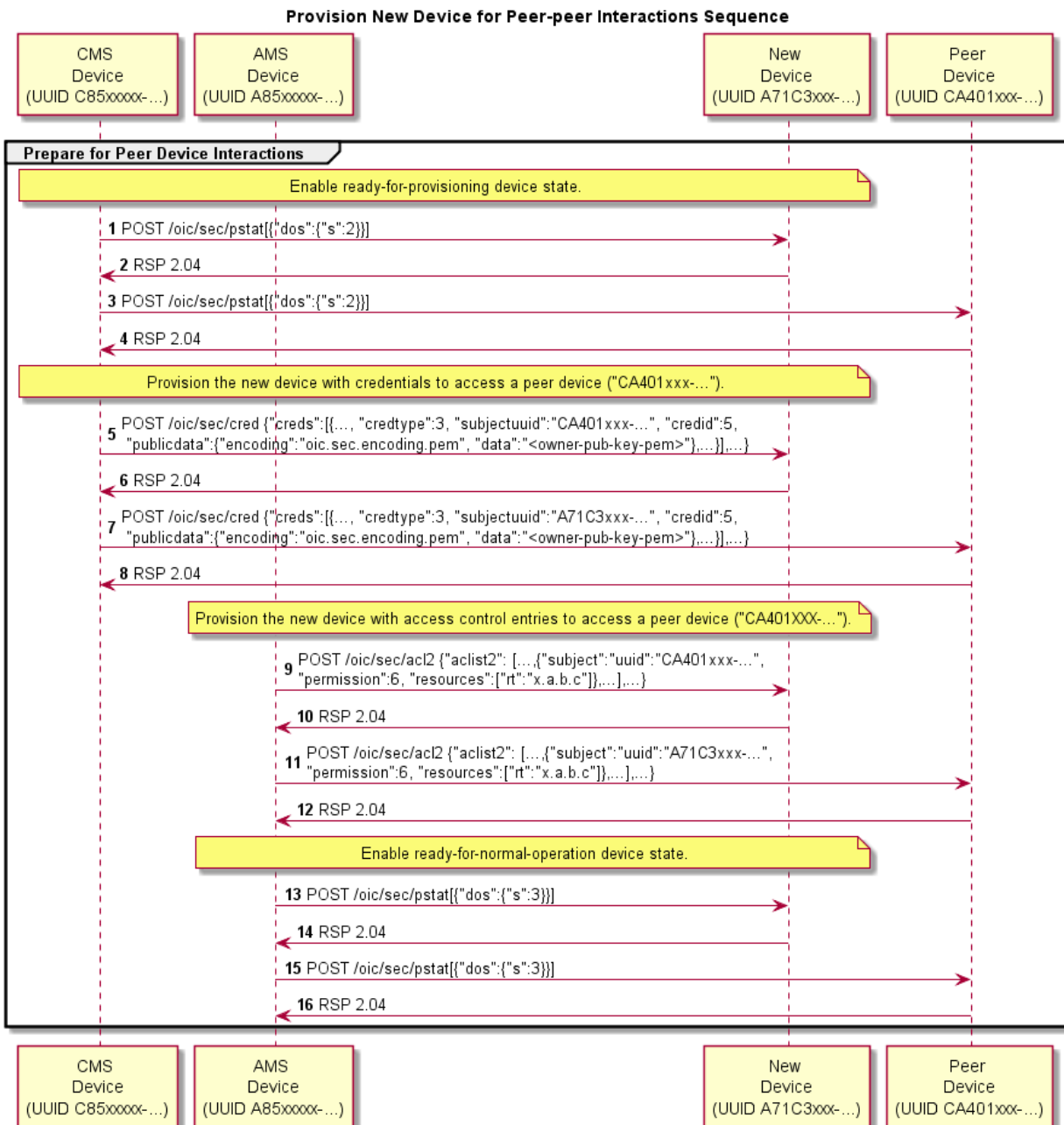
1627
1628

Figure 23 - Configure Device Services

Step	Description
1 - 8	The OBT assigns rowneruuid for different SVRs.
9 - 10	Provision the new Device with credentials for CMS
11 - 12	Provision the new Device with credentials for AMS
13 - 14	Update the oic.sec.doxm.owned to TRUE. Device is ready to move to provision and RFPRO state.

1629

Table 11 - Configure Device Services Detail



1630
1631

Figure 24 - Provision New Device for Peer to Peer Interaction Sequence

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.

Table 12 - Provision New Device for Peer to Peer Details

1632

1633 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1634 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are
1635 acceptable when transferring ownership of a new Device. Some of the factors to be considered
1636 when determining those requirements are:

- 1637 • The security considerations described above, for each of the OTMs
- 1638 • The probability that a man-in-the-middle attacker might be present in the environment used
1639 to perform the Ownership Transfer

1640 For example, the operator of an OBT might require that all of the Devices being onboarded support
1641 either the Random PIN or the Manufacturer Certificate OTM.

1642 When such a local OTM policy exists, the OBT should try to use just the OTMs that are acceptable
1643 according to that policy, regardless of the doxm contents obtained during step 1 from the sequence
1644 diagram above (GET /oic/sec/doxm). If step 1 is performed over an unauthenticated and/or
1645 unencrypted connection between the OBT and the Device, the contents of the response to the
1646 GET request might have been tampered by a man-in-the-middle attacker. For example, the list of
1647 OTMs supported by the new Device might have been altered by the attacker.

1648 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new
1649 Device to fail. In such cases, the OBT has no way of determining if the session failed because the
1650 new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle
1651 injected such a failure into the communication between the OBT and the new Device.

1652 The current version of this specification leaves the design and user experience related to the OTM
1653 policy mentioned above as OBT implementation details.

1654 **7.4 Provisioning**

1655 **7.4.1 Provisioning Flows**

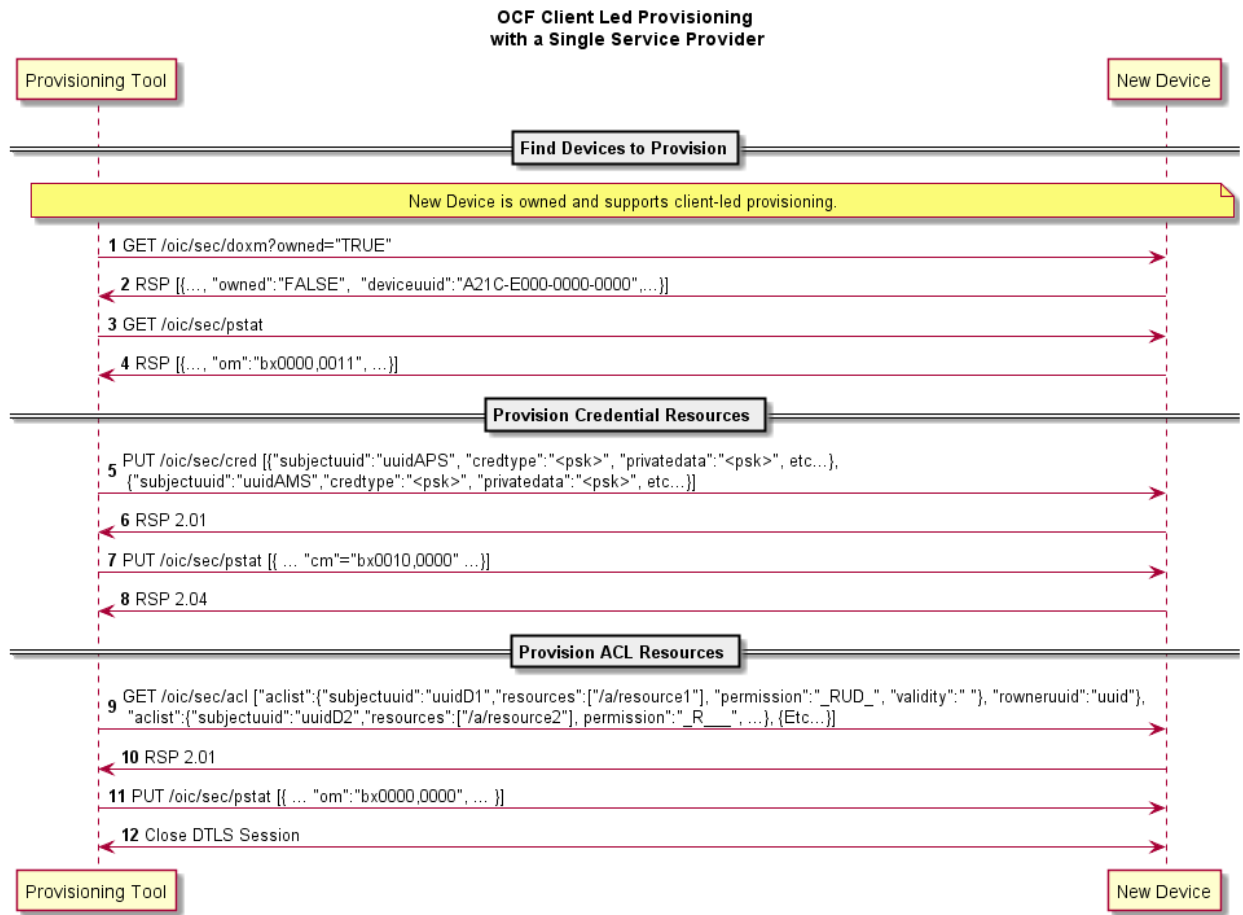
1656 **7.4.1.1 General**

1657 As part of onboarding a new Device a secure channel is formed between the new Device and the
1658 OBT. Subsequent to the Device ownership status being changed to 'owned', there is an opportunity
1659 to begin provisioning. The OBT decides how the new Device will be managed going forward and
1660 provisions the support services that should be subsequently used to complete Device provisioning
1661 and on-going Device management.

1662 The Device employs a Server-directed or Client-directed provisioning strategy. The /oic/sec/pstat
1663 Resource identifies the provisioning strategy and current provisioning status. The provisioning
1664 service should determine which provisioning strategy is most appropriate for the network. See
1665 Section 13.8 for additional detail.

1666 **7.4.1.2 Client-directed Provisioning**

1667 Client-directed provisioning relies on a provisioning service that identifies Servers in need of
 1668 provisioning then performs all necessary provisioning duties.



1669
1670

Figure 25 – Example of Client-directed provisioning

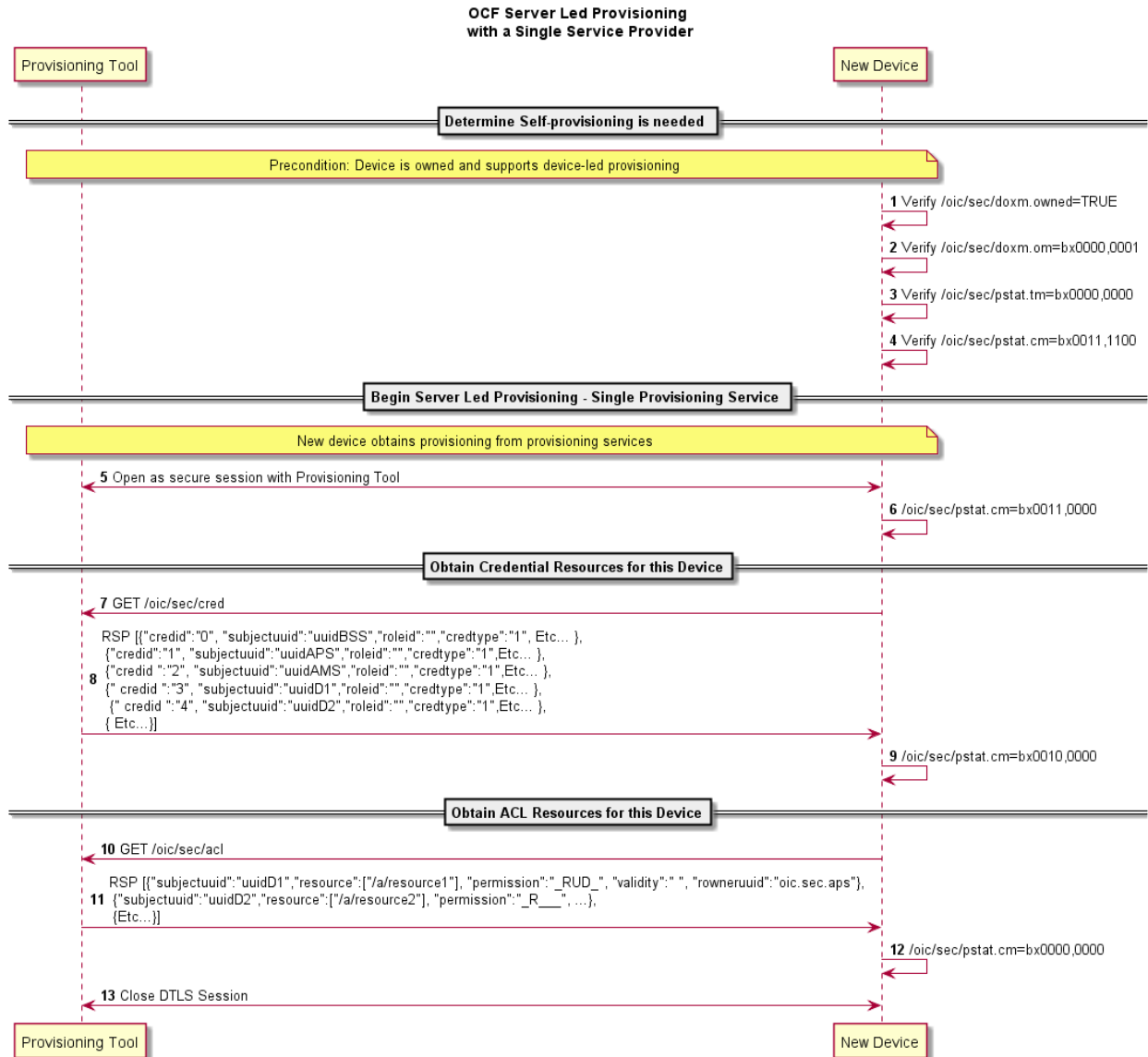
Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The /oic/sec/doxm Resource identifies the Device and it's owned status.
3	PT obtains the new Device's provisioning status found in /oic/sec/pstat Resource
4	The pstat Resource describes the types of provisioning modes supported and which is currently configured. A Device manufacturer should set a default current operational mode (om). If the Om isn't configured for Client-directed provisioning, its om value can be changed.
5 - 6	Change state to Ready-for-Provisioning. cm is set to provision credentials and ACLs.
7 - 8	PT instantiates the /oic/sec/cred Resource. It contains credentials for the provisioned services and other Devices
9 - 10	cm is set to provision ACLs.
11 - 12	PT instantiates /oic/sec/acl Resources.
13 -14	The new Device provisioning status mode is updated to reflect that ACLs have been configured. (Ready-for-Normal-Operation state)
15	The secure session is closed.

Table 13 – Steps describing Client -directed provisioning

1671

1672 **7.4.1.3 Server-directed Provisioning**

1673 Server-directed provisioning relies on the Server (i.e. New Device) for directing much of the
1674 provisioning work. As part of the onboarding process the support services used by the Server to seek
1675 additional provisioning are provisioned. The New Device uses a self-directed, state-driven approach
1676 to analyze current provisioning state, and tries to drive toward target state. This example assumes a
1677 single support service is used to provision the new Device.



1678
1679

Figure 26 – Example of Server-directed provisioning using a single provisioning service

Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a secure session with the provisioning tool using the /oic/sec/doxm.DevOwner value to open a TLS connection using SharedKey.
7	The new Device updates Cm to reflect provisioning of bootstrap and other services.
8 – 9	The new Devices gets the /oic/sec/cred Resources. It contains credentials for the provisioned services and other Devices.
10	The new Device updates Cm to reflect provisioning of credential Resources.
11 – 12	The new Device gets the /oic/sec/acl Resources.
13	The new Device updates Cm to reflect provisioning of ACL Resources.
14	The secure session is closed.

1680 **Table 14 – Steps for Server-directed provisioning using a single provisioning service**

1681 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1682 A Server-directed provisioning flow, involving multiple support services distributes the provisioning
1683 work across multiple support services. Employing multiple support services is an effective way to
1684 distribute provisioning workload or to deploy specialized support. The following example
1685 demonstrates using a provisioning tool to configure two support services, a credential management
1686 support service and an ACL provisioning support service.

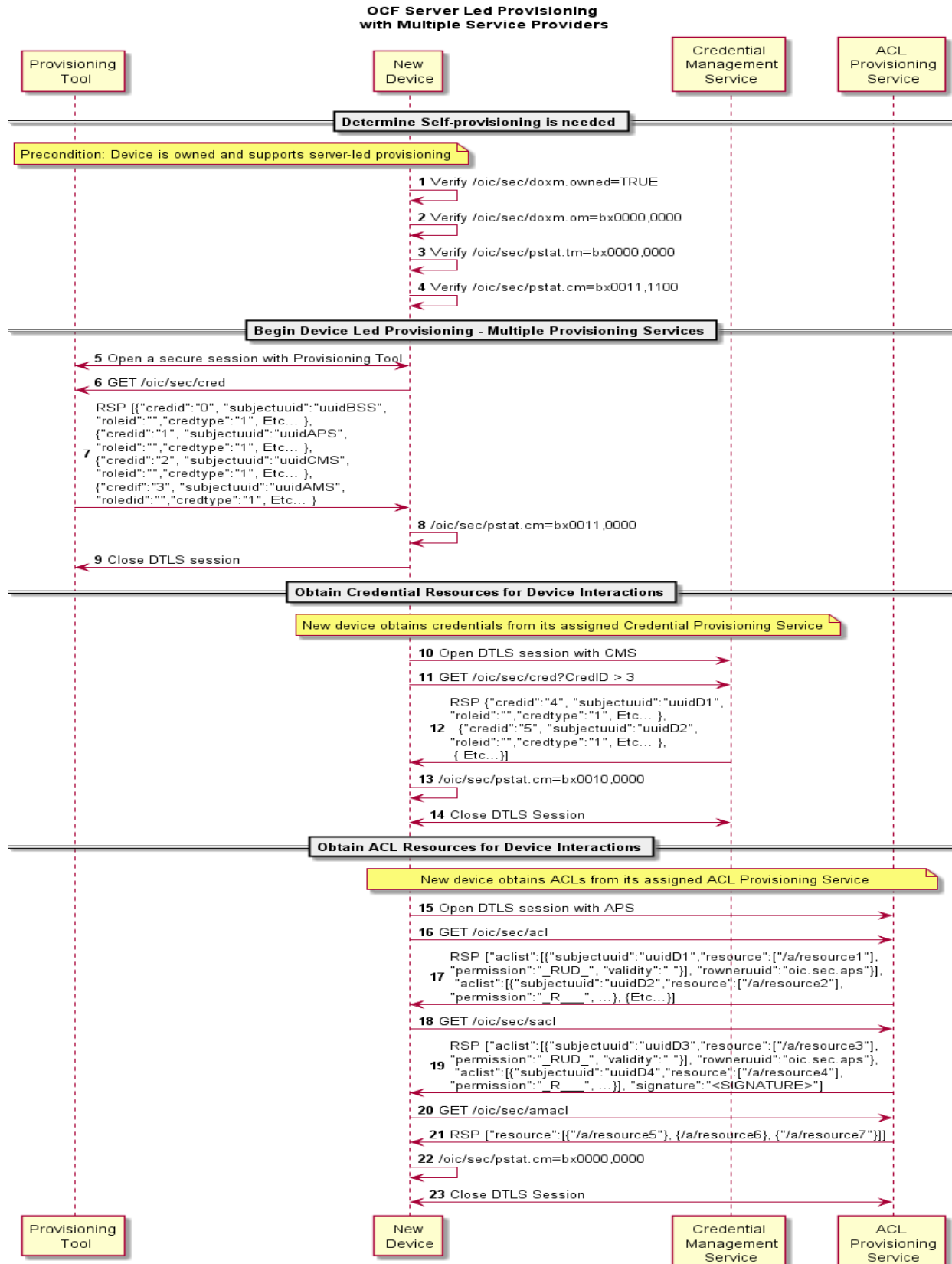


Figure 27 – Example of Server-directed provisioning involving multiple support services

Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a secure session with the provisioning tool using the /oic/sec/doxm.DevOwner value to open a TLS connection using SharedKey.
6	The new Device updates Cm to reflect provisioning of support services.
7	The new Device closes the DTLS session with the provisioning tool.
8	The new Device finds the CMS from the /oic/sec/cred Resource, rowneruuid property and opens a DTLS connection. The new device finds the credential to use from the /oic/sec/cred Resource.
9 – 10	The new Device requests additional credentials that are needed for interaction with other devices.
11	The new Device updates Cm to reflect provisioning of credential Resources.
12	The DTLS connection is closed.
13	The new Device finds the ACL provisioning and management service from the /oic/sec/acl2 Resource, rowneruuid property and opens a DTLS connection. The new device finds the credential to use from the /oic/sec/cred Resource.
14 – 15	The new Device gets ACL Resources that it will use to enforce access to local Resources.
16 – 18	The new Device should get SACL Resources immediately or in response to a subsequent Device Resource request.
19 – 20	The new Device should also get a list of Resources that should consult an Access Manager for making the access control decision.
21	The new Device updates Cm to reflect provisioning of ACL Resources.
22	The DTLS connection is closed.

1689 **Table 15 – Steps for Server-directed provisioning involving multiple support services**

1690 **7.5 Bootstrap Example**

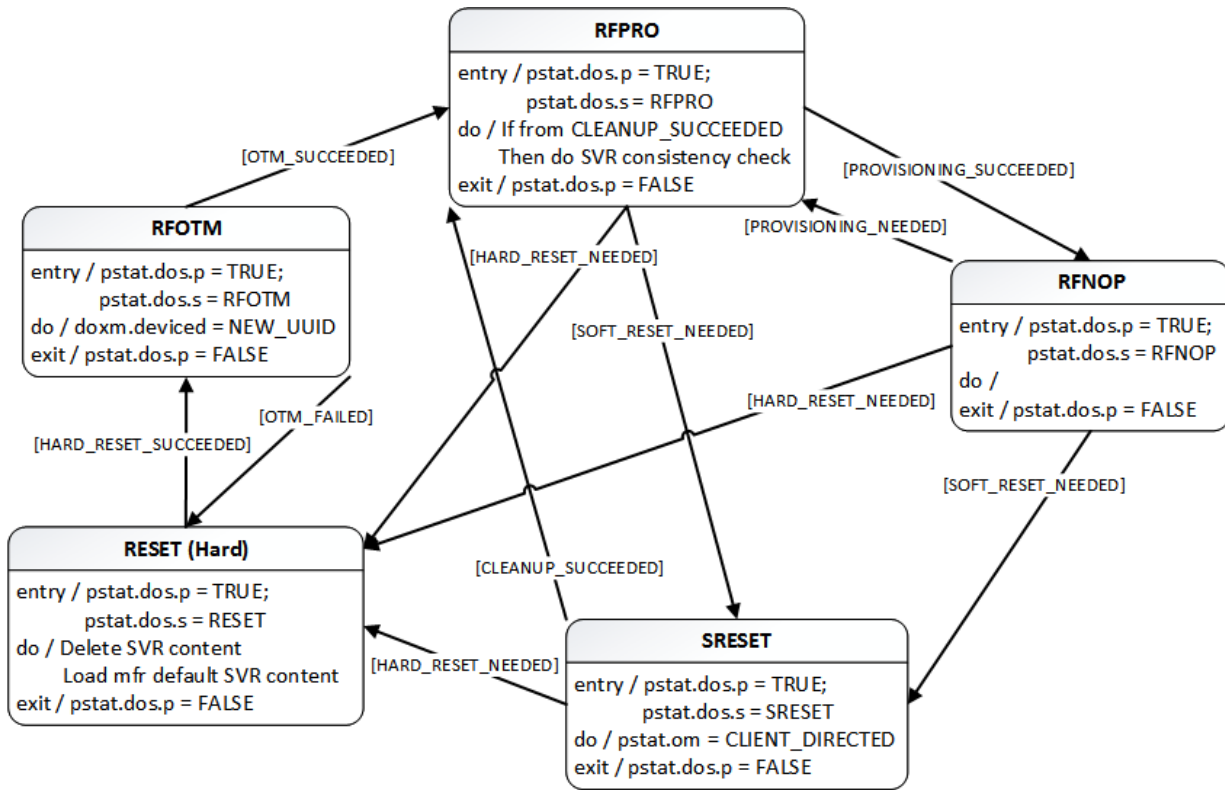
1691 This section is left intentionally blank.

1692 **8 Device Onboarding State Definitions**

1693 **8.1 Introduction**

1694 As explained in Section 5.3, the process of onboarding completes after the ownership of the Device
 1695 has been transferred and the Device has been provisioned with relevant configuration/services as
 1696 explained in Section 5.4. The diagram below shows the various states a Device can be in during
 1697 the Device lifecycle.

1698 The /pstat.dos.s property is RW by the /pstat resource owner (e.g. 'doxs' or 'bss' service) so that
 1699 the resource owner can remotely update the Device state. When the Device is in RFNOP or RFPRO,
 1700 ACLs can be used to allow remote control of Device state by other Devices. When the Device state
 1701 is SRESET the Device owner credential may be the only indication of authorization to access the
 1702 Device. The Device owner may perform low-level consistency checks and re-provisioning to get
 1703 the Device suitable for a transition to RFPRO.



1704
1705 **Figure 28 – Device state model**

1706 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the
 1707 "Ready for Normal Operation" state where it has all it needs in order to start interoperating with
 1708 other Devices. Section 8.2 specifies the minimum mandatory configuration that a Device shall hold
 1709 in order to be considered as "Ready for Normal Operation".

1710 In the event of power loss or Device failure, the Device should remain in the same state that it was
 1711 in prior to the power loss / failure

1712 If a Device or resource owner OBSERVEs /pstat.dos.s, then transitions to SRESET will give early
 1713 warning notification of Devices that may require SVR consistency checking.

1714 In order for onboarding to function, the Device shall have the following Resources installed:

1715 1. /oic/sec/doxm Resource

1716 2. /oic/sec/pstat Resource

1717 3. /oic/sec/cred Resource

1718 The values contained in these Resources are specified in the state definitions below.

1719 **8.2 Device Onboarding-Reset State Definition**

1720 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset
1721 also defines a state where the Device asset is ready to be transferred to another party.

1722 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces
1723 Platform reset. All Devices hosted on the same Platform transition their Device states to RESET
1724 when the Platform reset is asserted.

1725 The following Resources and their specific properties shall have the value as specified.

1726 1. The "owned" Property of the /oic/sec/doxm Resource shall transition to FALSE.

1727 2. The "devowneruuid" Property of the /oic/sec/doxm Resource shall be nil UUID.

1728 3. The "devowner" Property of the /oic/sec/doxm Resource shall be nil UUID, if this Property is
1729 implemented.

1730 4. The "deviceuuid" Property of the /oic/sec/doxm Resource shall be reset to the manufacturer's
1731 default value.

1732 5. The "deviceid" Property of the /oic/sec/doxm Resource shall be reset to the manufacturer's
1733 default value, if this Property is implemented.

1734 6. The "sct" Property of the /oic/sec/doxm Resource shall be reset to the manufacturer's default
1735 value.

1736 7. The "oxmsel" Property of the /oic/sec/doxm Resource shall be reset to the manufacturer's
1737 default value.

1738 8. The "isop" Property of the /oic/sec/pstat Resource shall be FALSE.

1739 9. The "dos" of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RESET" state
1740 and dos.p shall equal "FALSE".

1741 10. The current provisioning mode Property - "cm" of the /oic/sec/pstat Resource shall be
1742 "00000001".

1743 11. The target provisioning mode Property - "tm" of the /oic/sec/pstat Resource shall be
1744 "00000010".

1745 12. The operational modes Property - "om" of the /oic/sec/pstat Resource shall be set to the
1746 manufacturer default value.

1747 13. The supported operational modes Property - "sm" of the /oic/sec/pstat Resource shall be set
1748 to the manufacturer default value.

1749 14. The "rowneruuid" Property of /oic/sec/pstat, /oic/sec/doxm, /oic/sec/acl, /oic/sec/amacl,
1750 /oic/sec/sacl, and /oic/sec/cred Resources shall be nil UUID.

1751 .

1752 **8.3 Device Ready-for-OTM State Definition**

1753 The following Resources and their specific properties shall have the value as specified for an
1754 operational Device that is ready for ownership transfer

- 1755 1. The "owned" Property of the /oic/sec/doxm Resource shall be FALSE and will transition to
1756 TRUE.
- 1757 2. The "devowner" Property of the /oic/sec/doxm Resource shall be nil UUID, if this Property is
1758 implemented.
- 1759 3. The "devowneruuid" Property of the /oic/sec/doxm Resource shall be nil UUID.
- 1760 4. The "deviceid" Property of the /oic/sec/doxm Resource may be nil UUID, if this Property is
1761 implemented. The value of the "di" Property in /oic/d is undefined.
- 1762 5. The "deviceuuid" Property of the /oic/sec/doxm Resource may be nil UUID. The value of the
1763 "di" Property in /oic/d is undefined.
- 1764 6. The "isop" Property of the /oic/sec/pstat Resource shall be FALSE.
- 1765 7. The "dos" of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFOTM" state
1766 and dos.p shall equal "FALSE".
- 1767 8. The "cm" Property of the /oic/sec/pstat Resource shall be "00XXXX10".
- 1768 9. The "tm" Property of the /oic/sec/pstat shall be "00XXXX00".
- 1769 10. The /oic/sec/cred Resource should contain credential(s) if required by the selected OTM

1770 **8.4 Device Ready-for-Provisioning State Definition**

1771 The following Resources and their specific properties shall have the value as specified when the
1772 Device is ready for additional provisioning:

- 1773 1. The "owned" Property of the /oic/sec/doxm Resource shall be TRUE.
- 1774 2. The "devowneruuid" Property of the /oic/sec/doxm Resource shall not be nil UUID.
- 1775 3. The "deviceuuid" Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be
1776 set to the value that was determined during RFOTM processing. Also the value of the "di"
1777 Property in /oic/d Resource shall be the same as the deviceid Property in the /oic/sec/doxm
1778 Resource.
- 1779 4. The "oxmsel" Property of the /oic/sec/doxm Resource shall have the value of the actual OTM
1780 used during ownership transfer.
- 1781 5. The "isop" Property of the /oic/sec/pstat Resource shall be FALSE.
- 1782 6. The "dos" of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFPRO" state
1783 and dos.p shall equal "FALSE".
- 1784 7. The "cm" Property of the /oic/sec/pstat Resource shall be "00XXXX00".
- 1785 8. The "tm" Property of the /oic/sec/pstat shall be "00XXXX00".
- 1786 9. The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner
1787 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
1788 rowneruuid or rowner (at least one of the two) may result in an orphan Resource.

1789 10. The /oic/sec/cred Resource shall contain credentials for each entity referenced by an
1790 rowneruuid, amsuuid, devowneruuid.

1791 **8.5 Device Ready-for-Normal-Operation State Definition**

1792 The following Resources and their specific properties shall have the value as specified for an
1793 operational Device Final State

1794 1. The "owned" Property of the /oic/sec/doxm Resource shall be TRUE.

1795 2. The "devowneruuid" Property of the /oic/sec/doxm Resource shall not be nil UUID.

1796 3. The "deviceuuid" Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be
1797 set to the ID that was configured during OTM. Also the value of the "di" Property in /oic/d
1798 shall be the same as the deviceuuid.

1799 4. The "oxmsel" Property of the /oic/sec/doxm Resource shall have the value of the actual OTM
1800 used during ownership transfer.

1801 5. The "isop" Property of the /oic/sec/pstat Resource shall be TRUE.

1802 6. The "dos" of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFNOP" state
1803 and dos.p shall equal "FALSE".

1804 7. The "cm" Property of the /oic/sec/pstat Resource shall be "00XXXX00" (where "X" is
1805 interpreted as either 1 or 0).

1806 8. The "tm" Property of the /oic/sec/pstat shall be "00XXXX00".

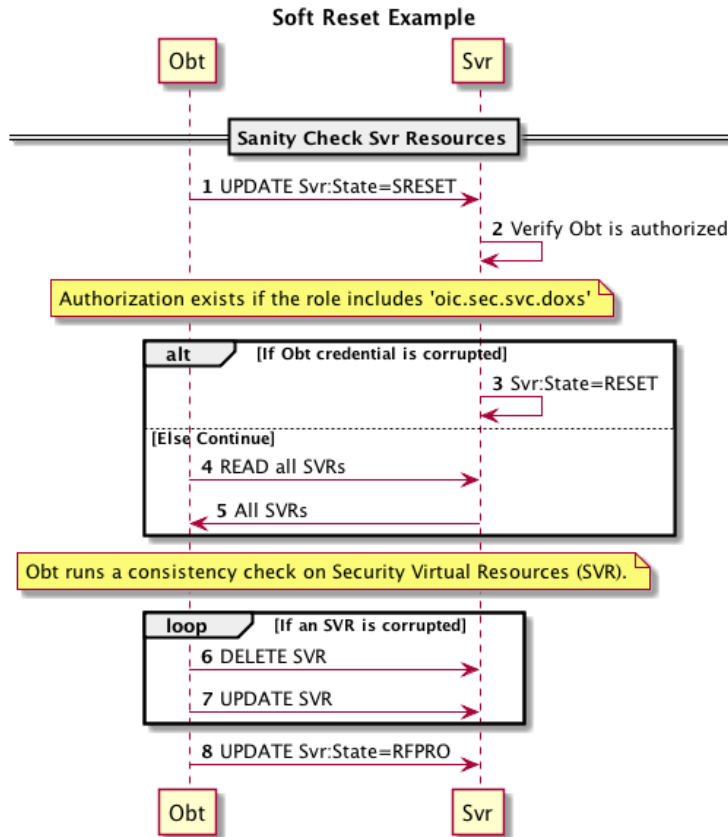
1807 9. The "rowneruuid" Property of every installed Resource shall be set to a valid resource owner
1808 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
1809 rowneruuid or rowner (at least one of the two) may result in an orphan Resource.

1810 10. The /oic/sec/cred Resource shall contain credentials for each service referenced by a
1811 rowneruuid, amsuuid, devowneruuid.

1812 **8.5 Device Soft Reset State Definition**

1813 The soft reset state is defined (e.g. /pstat.dos.s = SRESET) where entrance into this state means
1814 the Device is not operational but remains owned by the current owner. The Device may exit
1815 SRESET by authenticating to an OBT (e.g. "rt" = "oic.r.doxs") using the OC provided during original
1816 onboarding (but should not require use of an owner transfer method /doxm.oxms).

1817 The OBT should perform a consistency check of the SVR and if necessary, re-provision them
1818 sufficiently to allow the Device to transition to RFPRO.



1819

1820 **Figure 29 – OBT Sanity Check Sequence in SRESET**

1821 The OBT should perform a sanity check of SVRs before final transition to RFPRO Device state. If
 1822 the Device's OBT 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 OBT credential fails to validate
 1824 the OBT. This mitigates denial-of-service attacks that may be attempted by non-OBT Devices.

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

- 1827 1. The "owned" Property of the /oic/sec/doxm Resource shall be TRUE.
- 1828 2. The "devowneruuid" Property of the /oic/sec/doxm Resource shall remain non-null.
- 1829 3. The "devowner" Property of the /oic/sec/doxm Resource shall be non-null, if this Property is
 1830 implemented.
- 1831 4. The "deviceuuid" Property of the /oic/sec/doxm Resource shall remain non-null.
- 1832 5. The "deviceid" Property of the /oic/sec/doxm Resource shall remain non-null.
- 1833 6. The "sct" Property of the /oic/sec/doxm Resource shall retain its value.
- 1834 7. The "oxmsel" Property of the /oic/sec/doxm Resource shall retains its value.
- 1835 8. The "isop" Property of the /oic/sec/pstat Resource shall be FALSE.
- 1836 9. The /oic/sec/pstat.dos.s Property shall be SRESET.

- 1837 10. The current provisioning mode Property - "cm" of the /oic/sec/pstat Resource shall be
1838 "00000001".
- 1839 11. The target provisioning mode Property - "tm" of the /oic/sec/pstat Resource shall be
1840 "00XXXX00".
- 1841 12. The operational modes Property - "om" of the /oic/sec/pstat Resource shall be 'client-directed
1842 mode'.
- 1843 13. The supported operational modes Property (/pstat.sm) may be updated by the Device owner
1844 (aka DOXS).
- 1845 14. The "rowneruuid" Property of /oic/sec/pstat, /oic/sec/doxm, /oic/sec/acl, /oic/sec/acl2,
1846 /oic/sec/amacl, /oic/sec/sacl, and /oic/sec/cred Resources may be reset by the Device owner
1847 (aka DOXS) and re-provisioned.
- 1848

1849 **9 Security Credential Management**

1850 **9.1 Introduction**

1851 This section provides an overview of the credential types in OCF, along with details of credential
1852 use, provisioning and ongoing management.

1853 **9.2 Credential Lifecycle**

1854 **9.2.1 General**

1855 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)
1856 issuance and (5) revocation.

1857 **9.2.2 Creation**

1858 Devices may instantiate credential Resources directly using an ad-hoc key exchange method such
1859 as Diffie-Hellman. Alternatively, a CMS may be used to provision credential Resources to the
1860 Device.

1861 The credential Resource maintains a resource owner Property (`/oic/sec/cred.Rowner`) that
1862 identifies a CMS. If a credential was created ad-hoc, the peer Device involved in the Key Exchange
1863 is considered to be the CMS.

1864 Credential Resources created using a CMS may involve specialized credential issuance protocols
1865 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate
1866 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of
1867 a provisioning action by a provisioning, bootstrap or onboarding service.

1868 **9.2.3 Deletion**

1869 The CMS can delete credential Resources or the Device (e.g. the Device where the credential
1870 Resource is hosted) can directly delete credential Resources.

1871 An expired credential Resource may be deleted to manage memory and storage space.

1872 Deletion in OCF key management is equivalent to credential suspension.

1873 **9.2.4 Refresh**

1874 Credential refresh may be performed with the help of a CMS before it expires.

1875 The method used to obtain the credential initially should be used to refresh the credential.

1876 The `/oic/sec/cred` Resource supports expiry using the Period Property. Credential refresh may be
1877 applied when a credential is about to expire or is about to exceed a maximum threshold for bytes
1878 encrypted.

1879 A credential refresh method specifies the options available when performing key refresh. The
1880 Period Property informs when the credential should expire. The Device may proactively obtain a
1881 new credential using a credential refresh method using current unexpired credentials to refresh
1882 the existing credential. If the Device does not have an internal time source, the current time should
1883 be obtained from a CMS at regular intervals.

1884 Alternatively, a CMS can be used to refresh or re-issue an expired credential unless no trusted
1885 CMS can be found.

1886 If the CMS credential is allowed to expire, the BSS or onboarding service may be used to re-
1887 provision the CMS. If the onboarding established credentials are allowed to expire the Device will
1888 need to be re-onboarded and the device owner transfer steps re-applied.

1889 If credentials established through ad-hoc methods are allowed to expire the ad-hoc methods will
1890 need to be re-applied.

1891 All Devices shall support at least one credential refresh method.

1892 **9.2.5 Revocation**

1893 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where
1894 the revocation method involves provisioning of a revocation object that identifies a credential that
1895 is to be revoked prior to its normal expiration period, a credential Resource is created containing
1896 the revocation information that supersedes the originally issued credential. The revocation object
1897 expiration should match that of the revoked credential so that the revocation object is cleaned up
1898 upon expiry.

1899 It is conceptually reasonable to consider revocation applying to a credential or to a Device. Device
1900 revocation asserts all credentials associated with the revoked Device should be considered for
1901 revocation. Device revocation is necessary when a Device is lost, stolen or compromised. Deletion
1902 of credentials on a revoked Device might not be possible or reliable.

1903 **9.3 Credential Types**

1904 **9.3.1 General**

1905 The `/oic/sec/cred` Resource maintains a credential type Property that supports several
1906 cryptographic keys and other information used for authentication and data protection. The
1907 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric
1908 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.
1909 PIN/password).

1910 **9.3.2 Pair-wise Symmetric Key Credentials**

1911 Pair-wise symmetric key credentials have a symmetric key in common with exactly one other peer
1912 Device. A CMS might maintain an instance of the symmetric key. The CMS is trusted to issue or
1913 provision pair-wise keys and not misuse it to masquerade as one of the pair-wise peers.

1914 Pair-wise keys could be established through ad-hoc key agreement protocols.

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

1916 The `PublicData` Property may contain a token encrypted to the peer Device containing the pair-
1917 wise key.

1918 The `OptionalData` Property may contain revocation status.

1919 The Device implementer should apply hardened key storage techniques that ensure the
1920 `PrivateData` remains private.

1921 The Device implementer should apply appropriate integrity, confidentiality and access protection
1922 of the `/oic/sec/cred`, `/oic/sec/crl`, `/oic/sec/roles`, `/oic/sec/csr` Resources to prevent unauthorized
1923 modifications.

1924 **9.3.3 Group Symmetric Key Credentials**

1925 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are
1926 used for efficient sharing of data among group participants.

1927 Group keys do not provide authentication of Devices but only establish membership in a group.

1928 Group keys are distributed with the aid of a CMS. The CMS is trusted to issue or provision group
1929 keys and not misuse them to manipulate protected data.

1930 The PrivateData Property in the /oic/sec/cred Resource contains the symmetric key.
1931 The PublicData Property may contain the group name.
1932 The OptionalData Property may contain revocation status.
1933 The Device implementer should apply hardened key storage techniques that ensure the
1934 PrivateData remains private.

1935 The Device implementer should apply appropriate integrity, confidentiality and access protection
1936 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
1937 modifications.

1938 **9.3.4 Asymmetric Authentication Key Credentials**

1939 **9.3.4.1 General**

1940 Asymmetric authentication key credentials contain either a public and private key pair or only a
1941 public key. The private key is used to sign Device authentication challenges. The public key is
1942 used to verify a device authentication challenge-response.

1943 The PrivateData Property in the /oic/sec/cred Resource contains the private key.

1944 The PublicData Property contains the public key.

1945 The OptionalData Property may contain revocation status.

1946 The Device implementer should apply hardened key storage techniques that ensure the
1947 PrivateData remains private.

1948 Devices should generate asymmetric authentication key pairs internally to ensure the private key
1949 is only known by the Device. See Section 9.3.4.2 for when it is necessary to transport private key
1950 material between Devices.

1951 The Device implementer should apply appropriate integrity, confidentiality and access protection
1952 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
1953 modifications.

1954 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

1955 Devices should employ industry-standard high-assurance techniques when allowing off-device key
1956 pair creation and provisioning. Use of such key pairs should be minimized, particularly if the key
1957 pair is immutable and cannot be changed or replaced after provisioning.

1958 When used as part of onboarding, these key pairs can be used to prove the Device possesses the
1959 manufacturer-asserted properties in a certificate to convince an OBT or a user to accept
1960 onboarding the Device. See Section 7.3.3 for the owner transfer method that uses such a certificate
1961 to authenticate the Device, and then provisions new network credentials for use.

1962 **9.3.5 Asymmetric Key Encryption Key Credentials**

1963 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when
1964 distributing or storing the key.

1965 The PrivateData Property in the /oic/sec/cred Resource contains the private key.

1966 The PublicData Property contains the public key.

1967 The OptionalData Property may contain revocation status.

1968 The Device implementer should apply hardened key storage techniques that ensure the
1969 PrivateData remains private.

1970 The Device implementer should apply appropriate integrity, confidentiality and access protection
1971 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
1972 modifications.

1973 **9.3.6 Certificate Credentials**

1974 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a CMS
1975 or an external certificate authority (CA).

1976 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

1977 The issued certificate is stored with the asymmetric key credential Resource.

1978 Other objects useful in managing certificate lifecycle such as certificate revocation status are
1979 associated with the credential Resource.

1980 Either an asymmetric key credential Resource or a self-signed certificate credential is used to
1981 terminate a path validation.

1982 The PrivateData Property in the /oic/sec/cred Resource contains the private key.

1983 The PublicData Property contains the issued certificate.

1984 The OptionalData Property may contain revocation status.

1985 The Device implementer should apply hardened key storage techniques that ensure the
1986 PrivateData remains private.

1987 The Device implementer should apply appropriate integrity, confidentiality and access protection
1988 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
1989 modifications.

1990 **9.3.7 Password Credentials**

1991 Shared secret credentials are used to maintain a PIN or password that authorizes Device access
1992 to a foreign system or Device that doesn't support any other OCF credential types.

1993 The PrivateData Property in the /oic/sec/cred Resource contains the PIN, password and other
1994 values useful for changing and verifying the password.

1995 The PublicData Property may contain the user or account name if applicable.

1996 The OptionalData Property may contain revocation status.

1997 The Device implementer should apply hardened key storage techniques that ensure the
1998 PrivateData remains private.

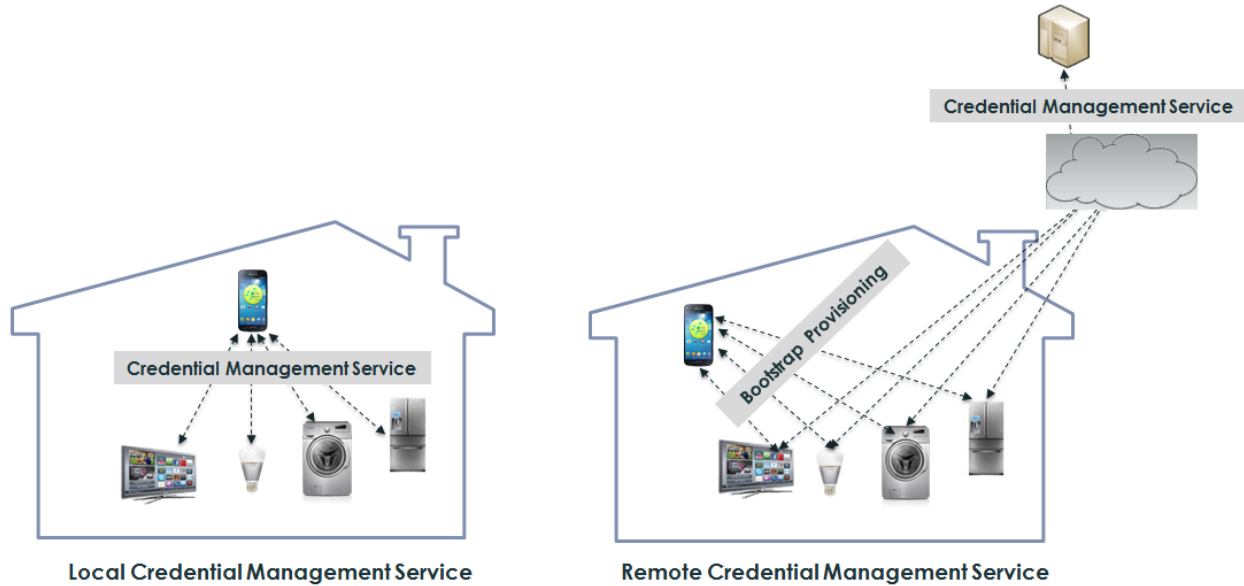
1999 The Device implementer should apply appropriate integrity, confidentiality and access protection
2000 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
2001 modifications.

2002 **9.4 Certificate Based Key Management**

2003 **9.4.1 Overview**

2004 To achieve authentication and transport security during communications in OCF network,
2005 certificates containing public keys of communicating parties and private keys can be used.

2006 The certificate and private key may be issued by a local or remote certificate authority (CA) when
 2007 a Device is deployed in the OCF network and credential provisioning is supported by a CMS
 2008 (Credential Management Service). For the local CA, a certificate revocation list (CRL) based on
 2009 X.509 is used to validate proof of identity. In the case of a remote CA, Online Certificate Status
 2010 Protocol (OCSP) can be used to validate proof of identity and validity.



2011
 2012 **Figure 30 – Certificate Management Architecture**

2013 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,
 2014 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in X.509
 2015 are not supported so that the format intends to meet the constrained Device's requirement.

2016 As for the certificate and CRL management in the Server, the process of storing, retrieving and
 2017 parsing Resources of the certificates and CRL will be performed at the security resource manager
 2018 layer; the relevant Interfaces may be exposed to the upper layer.

2019 A SRM is the security enforcement point in a Server as described in Section 5.5, so the data of
 2020 certificates and CRL will be stored and managed in SVR database.

2021 The request to issue a Device's certificate should be managed by a CMS when a Device is newly
 2022 onboarded or the certificate of the Device is revoked. When a certificate is considered invalid, it
 2023 must be revoked. A CRL is a data structure containing the list of revoked certificates and their
 2024 corresponding Devices that are not be trusted. The CRL is expected to be regularly updated (for
 2025 example; every 3 months) in real operations.

2026 **9.4.2 Certificate Format**

2027 **9.4.2.1 General**

2028 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in [RFC5280].

2029 **9.4.2.2 Certificate Profile and Fields**

2030 The OCF certificate shall support the following fields; `version`, `serialNumber`, `signature`,
 2031 `issuer`, `validity`, `subject`, `subjectPublicKeyInfo`, `extensions`,
 2032 `signatureAlgorithm` and `signatureValue`.

- 2033 • `version`: the version of the encoded certificate

- 2034 • serialNumber : certificate serial number
- 2035 • signature: the algorithm identifier for the algorithm used by the CA to sign this certificate
- 2036 • issuer: the entity that has signed and issued certificates
- 2037 • validity: the time interval during which CA warrants
- 2038 • subject: the entity associated with the subject public key field (deviceId)
- 2039 • subjectPublicKeyInfo: the public key and the algorithm with which key is used
- 2040 • extensions: certificate extensions as defined in section 9.4.2.3
- 2041 • signatureAlgorithm: the cryptographic algorithm used by the CA to sign this certificate
- 2042 • signatureValue: the digital signature computed upon the ASN.1 DER encoded
- 2043 OCFtbsCertificate (this signature value is encoded as a BIT STRING.)

2044 The OCF certificate syntax shall be defined as follows;

```

2045 OCFCertificate ::= SEQUENCE {
2046     OCFtbsCertificate      TBSertificate,
2047     signatureAlgorithm     AlgorithmIdentifier,
2048     signatureValue         BIT STRING
2049 }
```

2050 The OCFtbsCertificate field contains the names of a subject and an issuer, a public key
 2051 associated with the subject, a validity period, and other associated information. Per RFC5280,
 2052 version 3 certificates use the value 2 in the version field to encode the version number; the below
 2053 grammar does not allow version 2 certificates.

```

2054 OCFtbsCertificate ::= SEQUENCE {
2055     version                [0] 2 or above,
2056     serialNumber           CertificateSerialNumber,
2057     signature              AlgorithmIdentifier,
2058     issuer                 Name,
2059     validity               Validity,
2060     subject                Name,
2061     subjectPublicKeyInfo   SubjectPublicKeyInfo,
2062     extensions             [3] EXPLICIT Extensions
2063 }
2064 subjectPublicKeyInfo ::= SEQUENCE {
2065     algorithm              AlgorithmIdentifier,
2066     subjectPublicKey       BIT STRING
2067 }
2068 Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
2069
2070 Extension ::= SEQUENCE {
2071     extnID                 OBJECT IDENTIFIER,
2072     critical               BOOLEAN DEFAULT FALSE,
2073     extnValue              OCTET STRING
2074     -- contains the DER encoding of an ASN.1 value
2075     -- corresponding to the extension type identified
2076     -- by extnID
```

2077 }

Certificate Fields		Description	OCF	X.509
OCFtbsCertificate	version	2 or above	Mandatory	Mandatory
	serialNumber	CertificateSerialNumber	Mandatory	Mandatory
	signature	AlgorithmIdentifier	1.2.840.10045.4.3.2 (ECDSA algorithm with SHA256, Mandatory)	Specified in [RFC3279],[RFC4055], and [RFC4491]
	issuer	Name	Mandatory	Mandatory
	validity	Validity	Mandatory	Mandatory
	subject	Name	Mandatory	Mandatory
	subjectPublicKeyInfo	SubjectPublicKeyInfo	1.2.840.10045.2.1, 1.2.840.10045.3.1.7 (ECDSA algorithm with SHA256 based on secp256r1 curve, Mandatory)	Specified in [RFC3279],[RFC4055], and [RFC4491]
	issuerUniqueID	IMPLICIT UniqueIdentifier	Not supported	Optional
	subjectUniqueID	IMPLICIT UniqueIdentifier	Not supported	
extensions	EXPLICIT Extensions	Mandatory		
signatureAlgorithm	AlgorithmIdentifier	1.2.840.10045.4.3.2 (ECDSA algorithm with SHA256, Mandatory)	Specified in [RFC3279],[RFC4055], and [RFC4491]	
signatureValue	BIT STRING	Mandatory	Mandatory	

2078 **Table 16 – Comparison between OCF and X.509 certificate fields**

2079 **9.4.2.3 Supported Certificate Extensions**

2080 As these certificate extensions are a standard part of RFC 5280, this specification includes the
 2081 section number from that RFC to include it by reference. Each extension is summarized here, and
 2082 any modifications to the RFC definition are listed. Devices MUST implement and understand the
 2083 extensions listed here; other extensions from the RFC are not included in this specification and
 2084 therefore are not required. Section 10.4 describes what Devices must implement when validating
 2085 certificate chains, including processing of extensions, and actions to take when certain extensions
 2086 are absent.

- 2087 • Authority Key Identifier (4.2.1.1)

2088 The Authority Key Identifier (AKI) extension provides a means of identifying the public key
 2089 corresponding to the private key used to sign a certificate. This specification makes the
 2090 following modifications to the referenced definition of this extension:

2091 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier
 2092 sequence are not permitted; only keyIdentifier is allowed. This results in the following
 2093 grammar definition:

```
2094 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
```

```
2095 AuthorityKeyIdentifier ::= SEQUENCE {
2096     keyIdentifier [0] KeyIdentifier
2097 }
```

2098

2099 KeyIdentifier ::= OCTET STRING

2100 • Subject Key Identifier (4.2.1.2)

2101 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that
2102 contain a particular public key.

2103 This specification makes the following modification to the referenced definition of this
2104 extension:

2105 Subject Key Identifiers SHOULD be derived from the public key contained in the
2106 certificate's SubjectPublicKeyInfo field or a method that generates unique values. This
2107 specification RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING
2108 subjectPublicKey (excluding the tag, length, and number of unused bits). Devices verifying
2109 certificate chains must not assume any particular method of computing key identifiers,
2110 however, and must only base matching AKI's and SKI's in certification path constructions
2111 on key identifiers seen in certificates.

2112 • Subject Alternative Name

2113 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role
2114 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and
2115 interpreted as described below. When no EKU is present, or has another value, the
2116 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to
2117 encode one or more Role ID values in role certificates, binding the roles to the subject
2118 public key. The subjectAltName extension is defined in RFC 5280 (Section 4.2.1.6):

2119 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }

2120 SubjectAltName ::= GeneralNames

2121 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

2122 GeneralName ::= CHOICE {

2126 otherName	[0]	OtherName,
2127 rfc822Name	[1]	IA5String,
2128 dNSName	[2]	IA5String,
2129 x400Address	[3]	ORAddress,
2130 directoryName	[4]	Name,
2131 ediPartyName	[5]	EDIPartyName,
2132 uniformResourceIdentifier	[6]	IA5String,
2133 iPAddress	[7]	OCTET STRING,
2134 registeredID	[8]	OBJECT IDENTIFIER }

2135 EDIPartyName ::= SEQUENCE {

2137 nameAssigner	[0]	DirectoryString OPTIONAL,
2138 partyName	[1]	DirectoryString }

2139

2140 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a
2141 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name
2142 shall contain exactly one CN (Common Name) component, and zero or one OU
2143 (Organizational Unit) components. The OU component, if present, shall specify the
2144 authority that defined the semantics of the role. If the OU component is absent, the
2145 certificate issuer has defined the role. The CN component shall encode the role ID. Other
2146 GeneralName types in the SEQUENCE may be present, but shall not be interpreted as
2147 roles. Therefore, if the certificate issuer includes non-role names in the subjectAltName
2148 extension, the extension should not be marked critical.

2149 Note that the role, and authority need to be encoded as ASN.1 PrintableString type, the
2150 restricted character set [0-9a-z-A-z '()+, -./:=?].

2151 • Key Usage (4.2.1.3)

2152 The key usage extension defines the purpose (e.g., encipherment, signature, certificate
2153 signing) of the key contained in the certificate. The usage restriction might be employed
2154 when a key that could be used for more than one operation is to be restricted.

2155 This specification does not modify the referenced definition of this extension.

2156 • Basic Constraints (4.2.1.9)

2157 The basic constraints extension identifies whether the subject of the certificate is a CA and
2158 the maximum depth of valid certification paths that include this certificate. Without this
2159 extension, a certificate cannot be an issuer of other certificates.

2160 This specification does not modify the referenced definition of this extension.

2161 • Extended Key Usage (4.2.1.12)

2162 Extended Key Usage describes allowed purposes for which the certified public key may
2163 can be used. When a Device receives a certificate, it determines the purpose based on the
2164 context of the interaction in which the certificate is presented, and verifies the certificate
2165 can be used for that purpose.
2166

2167 This specification makes the following modifications to the referenced definition of this
2168 extension:

2169 CAs SHOULD mark this extension as critical.

2170 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).

2171 The list of OCF-specific purposes and the assigned OIDs to represent them are:
2172

- 2173 • Identity certificate 1.3.6.1.4.1.44924.1.6
- 2174 • Role certificate 1.3.6.1.4.1.44924.1.7

2175 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2176 All Devices support the certificate based key management shall support
2177 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite as defined in [RFC7251]. To
2178 establish a secure channel between two Devices the ECDHE_ECDSA (i.e. the signed version of
2179 Diffie-Hellman key agreement) key agreement protocol shall be used. During this protocol the two
2180 parties authenticate each other. The confidentiality of data transmission is provided by
2181 AES_128_CCM_8. The integrity of data transmission is provided by SHA256. Details are defined
2182 in [RFC7251] and referenced therein.

2183 To do lightweight certificate processing, the values of the following fields shall be chosen as follows:

- 2184 • signatureAlgorithm := ANSI X9.62 ECDSA algorithm with SHA256,
- 2185 • signature := ANSI X9.62 ECDSA algorithm with SHA256,
- 2186 • subjectPublicKeyInfo := ANSI X9.62 ECDSA algorithm with SHA256 based on
2187 secp256r1 curve.

2188 The certificate validity period is a period of time, the CA warrants that it will maintain
2189 information about the status of the certificate during the time; this information field is represented
2190 as a SEQUENCE of two dates:

- 2191 • the date on which the certificate validity period begins (notBefore)
- 2192 • the date on which the certificate validity period ends (notAfter).

2193 Both notBefore and notAfter should be encoded as UTCTime.

2194
2195 The field issuer and subject identify the entity that has signed and issued the certificate and the
2196 owner of the certificate. They shall be encoded as UTF8String and inserted in CN attribute.
2197

2198 9.4.2.5 Encoding of Certificate

2199 The ASN.1 distinguished encoding rules (DER) as defined in [ISO/IEC 8825-1] shall be used to
2200 encode certificates.

2201 9.4.3 CRL Format

2202 9.4.3.1 General

2203 An OCF CRL format is based on [RFC5280], but optional fields are not supported and signature-
2204 related fields are optional.

2205 9.4.3.2 CRL Profile and Fields

2206 The OCF CRL shall support the following fields; signature, issuer, this Update,
2207 revocationDate, signatureAlgorithm and signatureValue
2208

- 2209 • signature: the algorithm identifier for the algorithm used by the CA to sign this CRL
- 2210 • issuer: the entity that has signed or issued CRL.
- 2211 • this Update: the issue date of this CRL
- 2212 • userCertificate: certificate serial number
- 2213 • revocationDate: revocation date time
- 2214 • signatureAlgorithm: the cryptographic algorithm used by the CA to sign this CRL
- 2215 • signatureValue: the digital signature computed upon the ASN.1 DER encoded
2216 OCFtbsCertList (this signature value is encoded as a BIT STRING.)

2217 The signature-related fields such as signature, signatureAlgorithm, signatureValue
2218 are optional.

```
2219 CertificateList ::= SEQUENCE {
2220     OCFtbsCertList      TBSCertList,
2221     signatureAlgorithm  AlgorithmIdentifier,
2222     signatureValue      BIT STRING
2223 }
2224 OCFtbsCertList ::= SEQUENCE {
2225     signature           AlgorithmIdentifier OPTIONAL,
2226     issuer              Name,
2227     this Update        Time,
2228     revokedCertificates RevokedCertificates,
2229
```

```

2230     signatureAlgorithm AlgorithmIdentifier OPTIONAL,
2231     signatureValue      BIT STRING OPTIONAL
2232 }
2233 RevokedCertificates    SEQUENCE OF SEQUENCE {
2234     userCertificate     CertificateSerialNumber,
2235     revocationDate     Time
2236 }

```

CRL fields		Description	OCF	X.509	
OCFtbsCertificateList	version	Version v2	Not supported	Optional	
	signature	AlgorithmIdentifier	1.2.840.10045.4.3.2(ECDSA algorithm with SHA256,Optional)	Specified in [RFC3279], [RFC4055], and [RFC4491] list OIDs	
	issuer	Name	Mandatory	Mandatory	
	thisUpdate	Time	Mandatory	Mandatory	
	nextUpdate	Time	Not supported	Optional	
	revokedCertificates	userCertificate	Certificate Serial Number	Mandatory	Mandatory
		revocationDate	Time	Mandatory	Mandatory
		crlEntryExtensions	Time	Not supported	Optional
crlExtensions	Extensions	Not supported	Optional		
signatureAlgorithm	AlgorithmIdentifier	1.2.840.10045.4.3.2(ECDSA algorithm with SHA256,Optional)	Specified in [RFC3279], [RFC4055], and [RFC4491] list OIDs		
signatureValue	BIT STRING	Optional	Mandatory		

2237 **Table 17 – Comparison between OCF and X.509 CRL fields**

2238 **9.4.3.3 Encoding of CRL**

2239 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1] shall be used to encode CRL.

2241 **9.4.4 Resource Model**

2242 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated with a separate crl Resource that is defined for maintaining the revocation list.

2244 The cred Resource contains the certificate information pertaining to the Device. The PublicData Property holds the device certificate and CA certificate chain. PrivateData Property holds the Device private key paired to the certificate. (See Section 13.3 for additional detail regarding the /oic/sec/cred Resource).

2248 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained through the CMS. The Device must consider revoked certificates as part of certificate path verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain a full list, the Device must query the CMS for current revocation status. (See Section 13.4 for additional detail regarding the /oic/sec/crl Resource).

2253 **9.4.5 Certificate Provisioning**

2254 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have
 2255 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)
 2256 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate
 2257 shall have the format described in Section 9.4.2.

2258 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,
 2259 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer them
 2260 to the Device including its CA certificate chain. Optionally, the CMS may also transfer one or more
 2261 role certificates, which shall have the format described in Section 9.4.2. The subjectPublicKey of
 2262 each role certificate shall match the subjectPublicKey in the Device certificate.

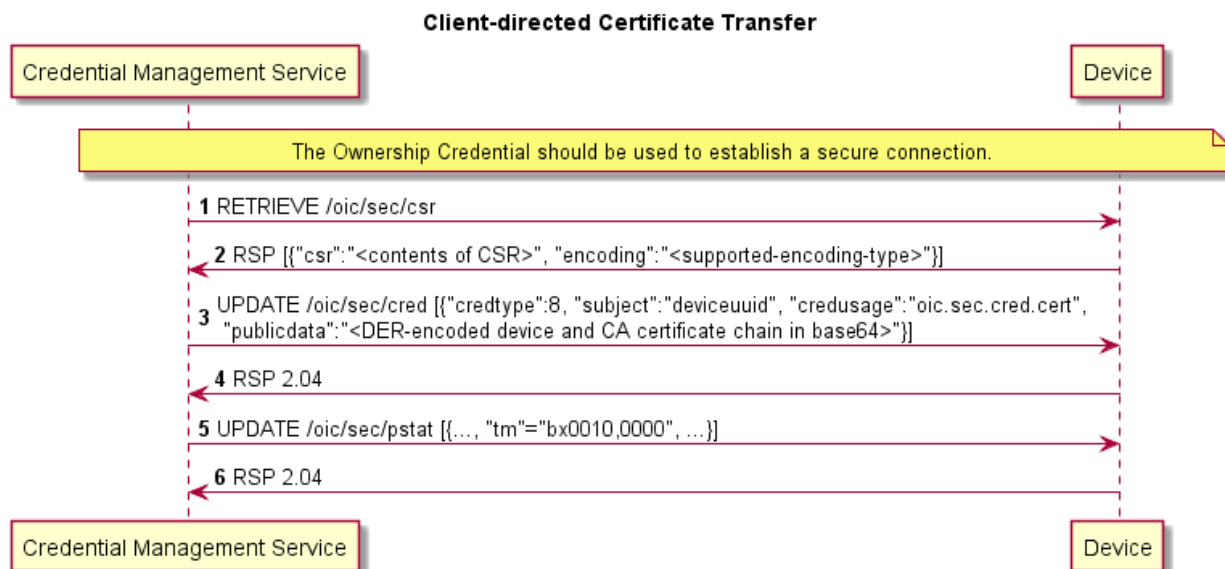
2263 In the below sequence, the Certificate Signing Request (CSR) is defined by PKCS#10 in RFC 2986,
 2264 and is included here by reference.

2265 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 31.

- 2266 1. The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the
 2267 Device shall place its requested UUID into the subject and its public key in the
 2268 SubjectPublicKeyInfo. The Device determines the public key to present; this may be an
 2269 already-provisioned key it has selected for use with authentication, or if none is present, it
 2270 may generate a new key pair internally and provide the public part. The key pair shall be
 2271 compatible with the allowed ciphersuites listed in Section 9.4.2.4 and 11.3.4, since the
 2272 certificate will be restricted for use in OCF authentication.

2273 If the Device does not have a pre-provisioned key pair and is unable to generate a key pair
 2274 on its own, then it is not capable of using certificates. The Device shall advertise this fact
 2275 both by setting the 0x8 bit position in the sct property of /oic/sec/doxm to 0, and return an
 2276 error that the /oic/sec/csr resource does not exist.

- 2277 2. The CMS shall transfer the issued certificate and CA chain to the designated Device using
 2278 the same credid, to maintain the association with the private key. The credential type
 2279 (oic.sec.cred) used to transfer certificates in Figure 31 is also used to transfer role
 2280 certificates, by including multiple credentials in the POST from CMS to Device. Identity
 2281 certificates shall be stored with the credusage property set to `oic.sec.cred.cert` and role
 2282 certificates shall be stored with the credusage property set to `oic.sec.cred.rolecert`.



2283

2284

Figure 31 – Client-directed Certificate Transfer

2285 9.4.6 CRL Provisioning

2286 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the function
2287 to register revocation certificates, to sign CRL and to transfer it to Devices.

2288 The CMS sends the CRL to the Device.

2289 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2290 • change of issuer name
- 2291 • change of association between Devices and CA
- 2292 • certificate compromise
- 2293 • suspected compromise of the corresponding private key

2294 CRL may be updated and delivered to all accessible Devices in the OCF network. In some special
2295 cases, Devices may request CRL to a given CMS.

2296 There are two options to update and deliver CRL;
2297

- 2298 • CMS pushes CRL to each Device
- 2299 • each Device periodically requests to update CRL

2300 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 32.

- 2301 1. The CMS may retrieve the CRL Resource Property.
- 2302 2. If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.

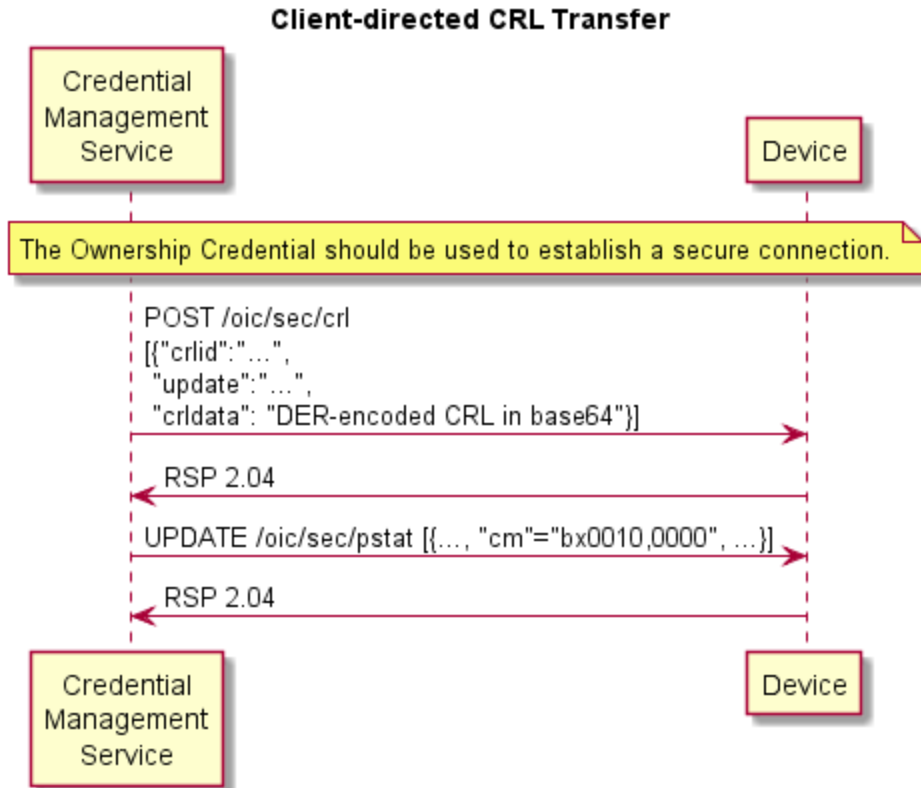


Figure 32 – Client-directed CRL Transfer

2303

2304 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 33.

2305

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

2306

2. If the CMS recognizes the updated CRL information after the designated update time, it may transfer its CRL to the Device.

2307

Server-directed CRL Transfer

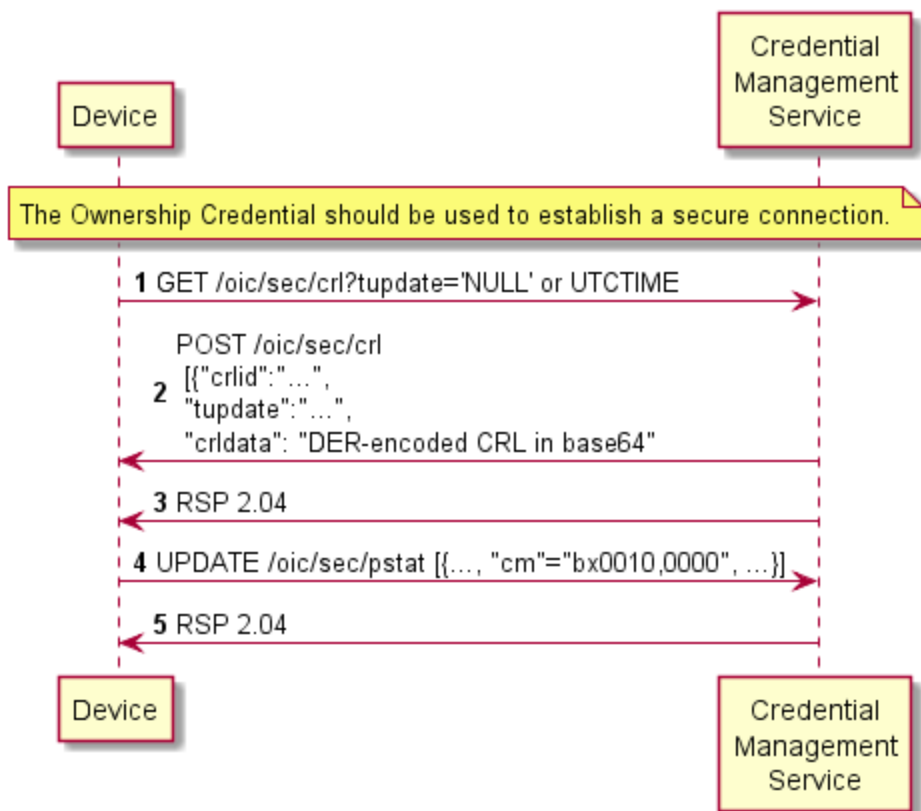


Figure 33 – Server-directed CRL Transfer

2308
2309

2310 **10 Device Authentication**

2311 **10.1 General**

2312 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the
2313 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or
2314 more roles that the server can use in access control decisions. Roles may be asserted when the
2315 Device authentication is done with certificates.

2316 **10.2 Device Authentication with Symmetric Key Credentials**

2317 When using symmetric keys to authenticate, the Server Device shall include the
2318 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client shall
2319 validate that it has a credential with the Subject ID set to the Server's Device ID, and a credential
2320 type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error or other
2321 suitable error.

2322 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that
2323 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a
2324 credential with the matching Subject ID and type. If it does not, the Server shall respond with an
2325 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS
2326 protocol, and both Client and Server shall compute the resulting premaster secret.

2327 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2328 When using raw asymmetric keys to authenticate, the Client and the Server shall include a suitable
2329 public key from a credential that is bound to their Device. Each Device shall verify that the provided
2330 public key matches the `PublicData` field of a credential they have, and use the corresponding
2331 Subject ID of the credential to identify the peer Device.

2332 **10.4 Device Authentication with Certificates**

2333 **10.4.1 General**

2334 When using certificates to authenticate, the Client and Server shall each include their certificate
2335 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.
2336 Each Device shall validate the certificate chain presented by the peer Device. Each certificate
2337 signature shall be verified until a public key is found within the `/oic/sec/cred` Resource with the
2338 `'oic.sec.cred.trustca'` credusage. Credential Resource found in `/oic/sec/cred` are used to terminate
2339 certificate path validation. Also validity period and revocation status should be checked for all
2340 above certificates.

2341 Devices must follow the certificate path validation algorithm in Section 6 of RFC 5280. In particular:

- 2342 • For all non-end-entity certificates, Devices shall verify that the basic constraints extension
2343 is present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate
2344 chain **MUST** be rejected. If the `pathLenConstraint` field is present, Devices will confirm the
2345 number of certificates between this certificate and the end-entity certificate is less than or
2346 equal to `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an end-entity
2347 certificate can be issued by this certificate. If the `pathLenConstraint` field is absent, there
2348 is no limit to the chain length.
- 2349 • For all non-end-entity certificates, Devices shall verify that the key usage extension is
2350 present, and that the `keyCertSign` bit is asserted.
- 2351 • Devices may use the Authority Key Identifier extension to quickly locate the issuing
2352 certificate. Devices **MUST NOT** reject a certificate for lacking this extension, and must
2353 instead attempt validation with the public keys of possible issuer certificates whose subject
2354 name equals the issuer name of this certificate.

2355 • The end-entity certificate of the chain shall be verified to contain an Extended Key Usage
2356 (EKU) suitable to the purpose for which it is being presented. An end-entity certificate which
2357 contains no EKU extension is not valid for any purpose and must be rejected. Any certificate
2358 which contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if
2359 other valid EKUs are also present.

2360 • Devices MUST verify "transitive EKU" for certificate chains. Issuer certificates (any
2361 certificate that is not an end-entity) in the chain MUST all be valid for the purpose for which
2362 the certificate chain is being presented. An issuer certificate is valid for a purpose if it
2363 contains an EKU extension and the EKU OID for that purpose is listed in the extension, OR
2364 it does not have an EKU extension. An issuer certificate SHOULD contain an EKU
2365 extension and a complete list of EKUs for the purposes for which it is authorized to issue
2366 certificates. An issuer certificate without an EKU extension is valid for all purposes; this
2367 differs from end-entity certificates without an EKU extension.

2368 The list of purposes and their associated OIDs are defined in Section 9.4.2.3.

2369 If the Device does not recognize an extension, it must examine the `critical` field. If the field is
2370 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the
2371 certificate as if the extension were absent and proceed accordingly. This applies to all certificates
2372 in a chain.

2373 Note: Certificate revocation mechanisms are currently out of scope of this version of the
2374 specification.

2375 **10.4.2 Role Assertion with Certificates**

2376 This section describes role assertion by a client to a server using a certificate role credential. If a
2377 server does not support the certificate credential type, clients should not attempt to assert roles
2378 with certificates.

2379 Following authentication with a certificate, a client may assert one or more roles by updating the
2380 server's roles resource with the role certificates it wants to use. The role credentials must be
2381 certificate credentials and shall include a certificate chain. The server shall validate each certificate
2382 chain as specified in Section 10.3. Additionally, the public key in the end-entity certificate used for
2383 Device authentication must be identical to the public key in all role (end-entity) certificates. Also,
2384 the subject distinguished name in the end-entity authentication and role certificates must match.
2385 The roles asserted are encoded in the `subjectAltName` extension in the certificate. Note that the
2386 `subjectAltName` field can have multiple values, allowing a single certificate to encode multiple roles
2387 that apply to the client. The server shall also check that the EKU extension of the role certificate(s)
2388 contains the value 1.3.6.1.4.1.44924.1.7 (see Section 9.4.2.2) indicating the certificate may be
2389 used to assert roles. Figure 34 describes how a client Device asserts roles to a server.

Asserting Certificate Role Credentials

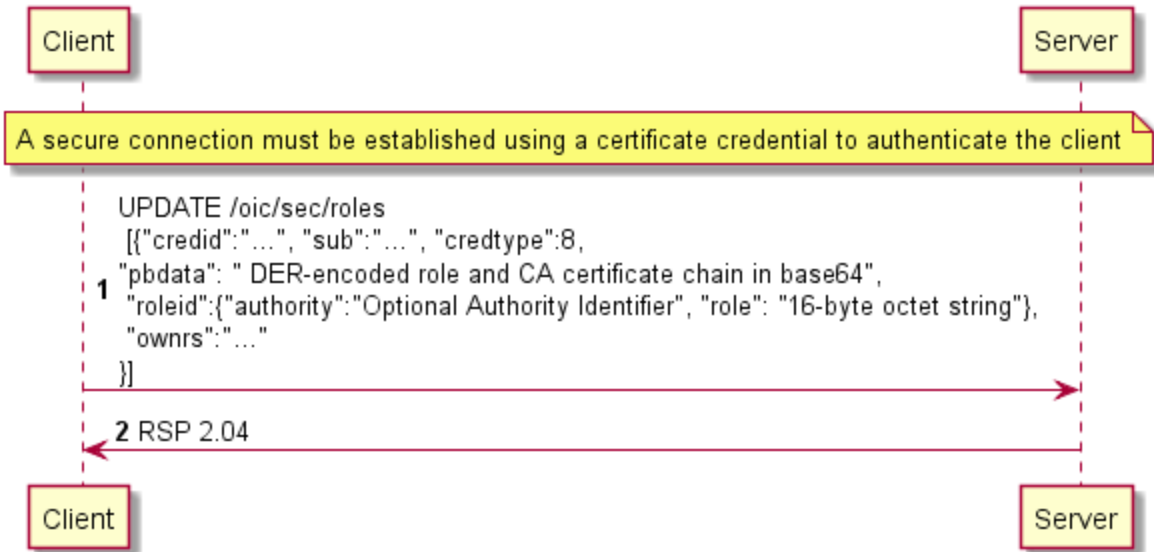


Figure 34 – Asserting a role with a certificate role credential.

2390
2391

2392 **Figure 34 Notes**

2393 1. The response shall contain "204 No Content" to indicate success or 4xx to indicate an error.
2394 If the server does not support certificate credentials, it should return "501 Not Implemented"

2395 2. Roles asserted by the client may be kept for a duration chosen by the server. The duration
2396 shall not exceed the validity period of the role certificate. When fresh CRL information is
2397 obtained, the certificates in /oic/sec/roles should be checked, and the role removed if the
2398 certificate is revoked or expired.

2399 Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a
2400 role by a client. It is recommended that servers use the validity period of the certificate as
2401 a duration, effectively allowing the CMS to decide the duration.

2402 3. The format of the data sent in the create call shall be a list of credentials (oic.sec.cred, see
2403 Table 23). They shall have credtype 8 (indicating certificates) and PrivateData field shall
2404 not be present. For fields that are duplicated in the oic.sec.cred object and the certificate,
2405 the value in the certificate shall be used for validation. For example, if the Period field is
2406 set in the credential, the server must treat the validity period in the certificate as
2407 authoritative. Similar for the roleid data (authority, role).

2408 4. Certificates shall be encoded as in Figure 31 (DER-encoded certificate chain in base64)

2409 5. Clients may GET the /oic/sec/roles resource to determine the roles that have been
2410 previously asserted. An array of credential objects must be returned, or "204 No Content"
2411 to indicate that no previously asserted roles are currently valid.

2412

2413

2414 **11 Message Integrity and Confidentiality**

2415 **11.1 General**

2416 Secured communications between Clients and Servers are protected against eavesdropping,
2417 tampering, or message replay, using security mechanisms that provide message confidentiality
2418 and integrity.

2419 **11.2 Session Protection with DTLS**

2420 **11.2.1 General**

2421 Devices shall support DTLS for secured communications as defined in [RFC 6347]. Devices using
2422 TCP shall support TLS v1.2 for secured communications as defined in [RFC 5246]. See Section
2423 11.3 for a list of required and optional cipher suites for message communication.

2424 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1
2425 or lower.

2426 Note: Multicast session semantics are not yet defined in this version of the security specification.

2427 **11.2.2 Unicast Session Semantics**

2428 For unicast messages between a Client and a Server, both Devices shall authenticate each other.
2429 See Section 10 for details on Device Authentication.

2430 Secured unicast messages between a Client and a Server shall employ a cipher suite from Section
2431 11.3. The sending Device shall encrypt and authenticate messages as defined by the selected
2432 cipher suite and the receiving Device shall verify and decrypt the messages before processing
2433 them.

2434 **11.3 Cipher Suites**

2435 **11.3.1 General**

2436 The cipher suites allowed for use can vary depending on the context. This section lists the cipher
2437 suites allowed during ownership transfer and normal operation. The following RFCs provide
2438 additional information about the cipher suites used in OCF.

2439 [RFC 4279]: Specifies use of pre-shared keys (PSK) in (D)TLS

2440 [RFC 4492]: Specifies use of elliptic curve cryptography in (D)TLS

2441 [RFC 5489]: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and
2442 PSKs

2443 [RFC 6655, 7251]: Specifies AES-CCM mode cipher suites, with ECDHE

2444

2445 **11.3.2 Cipher Suites for Device Ownership Transfer**

2446 **11.3.2.1 Just Works Method Cipher Suites**

2447 The Just Works owner transfer method may use the following (D)TLS cipher suites.

2448 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,

2449 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

2450

2451 All Devices supporting Just Works OTM shall implement:

2452 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (with the value 0xFF00)

2453

2454 All Devices supporting Just Works OTM should implement:

2455 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256 (with the value 0xFF01)

2456

2456 **11.3.2.2 Random PIN Method Cipher Suites**

2457 The Random PIN Based owner transfer method may use the following (D)TLS cipher suites.

2458 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
2459 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2460

2461 All Devices supporting Random Pin Based OTM shall implement:

2462 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

2463 **11.3.2.3 Certificate Method Cipher Suites**

2464 The Manufacturer Certificate Based owner transfer method may use the following (D)TLS cipher
2465 suites.

2466 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

2467 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2468 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2469 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2470 Using the following curve:

2471 secp256r1 (See [RFC4492])

2472 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2473 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2474 Devices supporting Manufacturer Certificate Based OTM should implement:

2475 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2476 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2477 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2478 **11.3.3 Cipher Suites for Symmetric Keys**

2479 The following cipher suites are defined for (D)TLS communication using PSKs:

2480 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2481 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2482 TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET Authentication tag *)

2483 TLS_PSK_WITH_AES_256_CCM_8,

2484 TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET Authentication tag *)

2485 TLS_PSK_WITH_AES_256_CCM,

2486 Note: All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2487

2488 All Devices shall implement the following:

2489 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2490

2491 Devices should implement the following:

2492 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2493 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

2494 TLS_PSK_WITH_AES_128_CCM_8,

2495 TLS_PSK_WITH_AES_256_CCM_8,

2496 TLS_PSK_WITH_AES_128_CCM,

2497 TLS_PSK_WITH_AES_256_CCM

2498 **11.3.4 Cipher Suites for Asymmetric Credentials**

2499 The following cipher suites are defined for (D)TLS communication with asymmetric keys or
2500 certificates:

2501 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

2502 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

2503 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

2504 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

2505 Using the following curve:

2506 secp256r1 (See [RFC4492])

2507

2508 All Devices supporting Asymmetric Credentials shall implement:

2509 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

2510

2511 All Devices supporting Asymmetric Credentials should implement:
2512 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
2513 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
2514 TLS_ECDHE_ECDSA_WITH_AES_256_CCM
2515

2516 **12 Access Control**

2517 **12.1 ACL Generation and Management**

2518 This section will be expanded in a future version of the specification.

2519 **12.2 ACL Evaluation and Enforcement**

2520 **12.2.1 General**

2521 The Server enforces access control over application Resources before exposing them to the
2522 requestor. The Security Resource Manager (SRM) in the Server authenticates the requestor when
2523 access is received via the secure port. Authenticated requestors, known as the "subject" can be
2524 used to match ACL entries that specify the requestor's identity, role or may match authenticated
2525 requestors using a subject wildcard.

2526 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a
2527 subject wildcard match of anonymous requestors.

2528 Access is denied if a requested resource is not matched by an ACL entry. (Note: There are
2529 documented exceptions pertaining to Device onboarding where access to security virtual resources
2530 may be permitted prior to provisioning of ACL resources.

2531 The second generation ACL (i.e. /oic/sec/acl2) contains an array of Access Control Entries (ACE2)
2532 that employ a resource matching algorithm that uses an array of resource references to match
2533 Resources to which the ACE2 access policy applies. Matching consists of comparing the values
2534 of the ACE2 "resources" property (see Section 13) to the requested Resource. Resources are
2535 matched in four ways; host reference (href), resource type (rt), resource interface (if) or resource
2536 wildcard.

2537 **12.2.2 Host Reference Matching**

2538 When present in an ACE2 matching element, the Host Reference (href) Property shall be used
2539 for resource matching.

- 2540 - The href Property shall be used to find an exact match of the Resource name.

2541 **12.2.3 Resource Type Matching**

2542 When present in an ACE2 matching element, The Resource Type (rt) Property shall be used for
2543 resource matching.

- 2544 - The rt Property shall be used to find an exact match of the Resource Type name.
- 2545 - An array of strings is used to match Resources that implement multiple Resource Type
2546 names (e.g. collection resources).

2547 **12.2.4 Interface Matching**

2548 When present in the ACE2 matching element, the Interface (if) property shall be used for
2549 resource matching.

- 2550 - The 'if' property shall be used to find an exact match of the Resource Interface string.
- 2551 - An array of strings is used when the Resource implements multiple Interfaces.

2552 **12.2.5 Multiple Criteria Matching**

2553 If multiple matching criteria are supplied in the same ACE2 Resources property (e.g. 'href' and 'rt'
2554 and 'if') then a logical AND of the criteria shall be applied. For example, if both 'href'="/a/light and
2555 'if'="oic.if.s" are in the Resources property, then a match exists only when both the 'href' and the
2556 'if' criterion are true for the candidate resources.

2557 If the ACE2 "resources" property is an array of entries, then a logical OR is applied for each array
2558 element. For example, if a first array element of the Resources property contains 'href'="/a/light"

2559 and the second array element of the Resources property contains 'if'="oic.if.s", then Resources
 2560 that match either the 'href' criteria or the 'if' criteria are included in the set of matched Resources.

2561 **12.2.6 Resource Wildcard Matching**

2562 A wildcard expression may be used to match multiple Resources using a wildcard Property
 2563 contained in the oic.sec.ace2.resource-ref structure. The following wildcard matching strings are
 2564 defined:

String	Description
"+"	Shall match all discoverable resources.
"-"	Shall match all non-discoverable resources.
"*"	Shall match all resources.

2565 **Table 18 – ACE2 Wildcard Matching Strings Description**

2566 Note: Discoverable resources appear in the /oic/wk/res Resource, while non-discoverable
 2567 resources may appear in other collection resources but do not appear in the /res collection.

2568 Example JSON for Resource matching

```

2569 {
2570   [
2571     //Matches Resources named "/x/door1" or "/x/door2"
2572     {
2573       "href":"/x/door1"
2574     },
2575     {
2576       "href":"/x/door2"
2577     },
2578     //Matches Resources with Resource Type "oic.sec.crl" and "oic.sec.cred"
2579     {
2580       "rt":[" oic.sec.crl ", "oic.sec.cred "]
2581     },
2582     // Matches Resources that implement both "oic.if.baseline" and
2583     "oic.if.rw" Interfaces.
2584     "if":["oic.if.baseline", "oic.if.rw"]
2585   },
2586   //Matches Resources named "/x/light1" or "/x/light2" and have Resource
2587   Types "x.light.led", "x.light.flourescent" and "x.light.color".
2588   {
2589     "href":"/x/light1",
2590     "rt":["x.light.led", "x.light.flourescent", "x.light.color"]
2591   },
2592   {
2593     "href":"/x/light2",
2594     "rt":["x.light.led", "x.light.flourescent", "x.light.color"]
2595   },
2596   //Matches all Resources.
2597   {
2598     "wc":"*"
2599   }
2600 ]
2601 }
2602
  
```

2603 **12.2.7 Subject Matching using Wildcards**

2604 When the ACE subject is specified as the wildcard string "*" any requestor is matched. The OCF
2605 server may authenticate the OCF client, but is not required to.

2606 Examples: JSON for subject wildcard matching

```
2607 //matches all subjects that have authenticated and confidentiality  
2608 protections in place.
```

```
2609 "subject" : {  
2610     "conntype" : "auth-crypt"  
2611 }
```

```
2612 //matches all subjects that have NOT authenticated and have NO  
2613 confidentiality protections in place.
```

```
2614 "subject" : {  
2615     "conntype" : "anon-clear"  
2616 }
```

2617

2618 **12.2.8 Subject Matching using Roles**

2619 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 2620 1. The requestor authenticated with a symmetric key credential, and the role is present in the
2621 roleid property of the credential's entry in the credential resource, or
- 2622 2. The requestor authenticated with a certificate, and a valid role certificate is present in the
2623 roles resource with the requestor's certificate's public key at the time of evaluation.
2624 Validating role certificates is defined in section 10.3.1.

2625 **12.2.9 ACL Evaluation**

2626 The OCF Server shall apply an ACE2 matching algorithm that matches in the following
2627 sequence:

- 2628 1. If the /oic/sec/sacl Resource exists and if the signature verification is successful, these
2629 ACE2 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify
2630 the signature, at least once, following update of the /oic/sec/sacl Resource.
- 2631 2. The local /oic/sec/acl2 Resource contributes its ACE2 entries for matching.
- 2632 3. Access shall be granted when all these criteria are met:
 - 2633 a. The requestor is matched by the ACE2 "subject" Property.
 - 2634 b. The requested Resource is matched by the ACE2 "resources" Property and the
2635 requested Resource shall exist on the local Server.
 - 2636 c. The "period" Property constraint shall be satisfied.
 - 2637 d. The "permission" Property constraint shall be applied.

2638 Note: If multiple ACE2 entries match the Resource request, the union of permissions, for all
2639 matching ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN;
2640 Then UNION (Perm1, Perm2)=CRUDN.

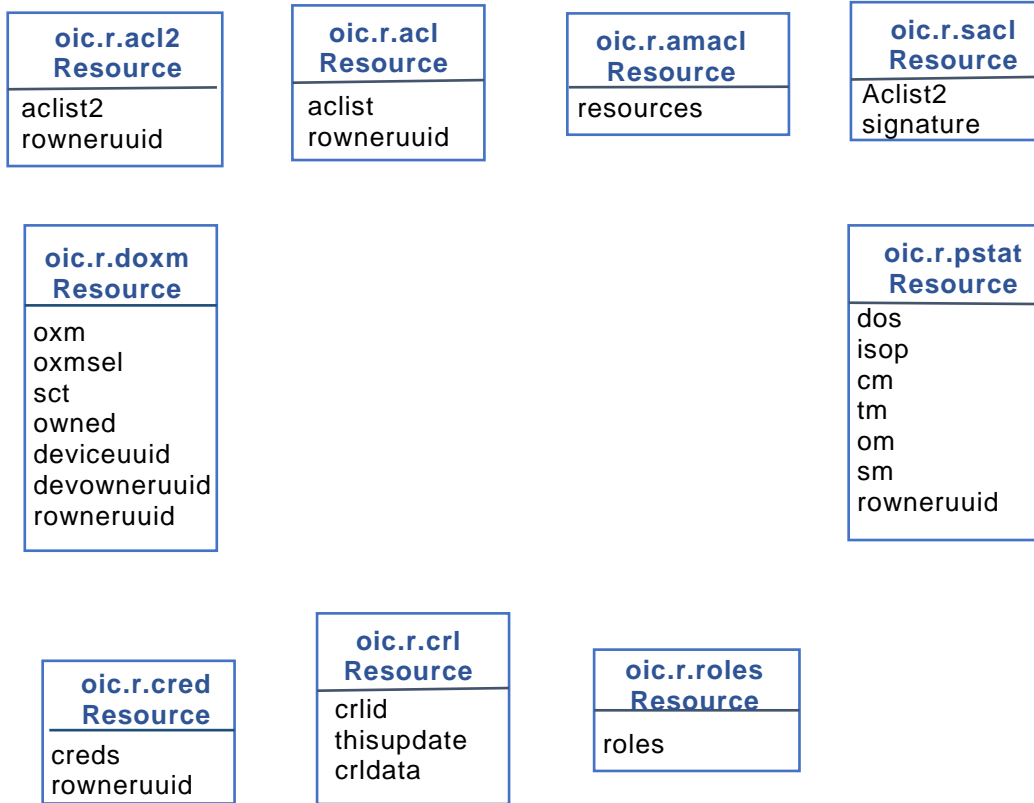
2641 The Server shall enforce access based on the effective permissions granted.

2642

2643

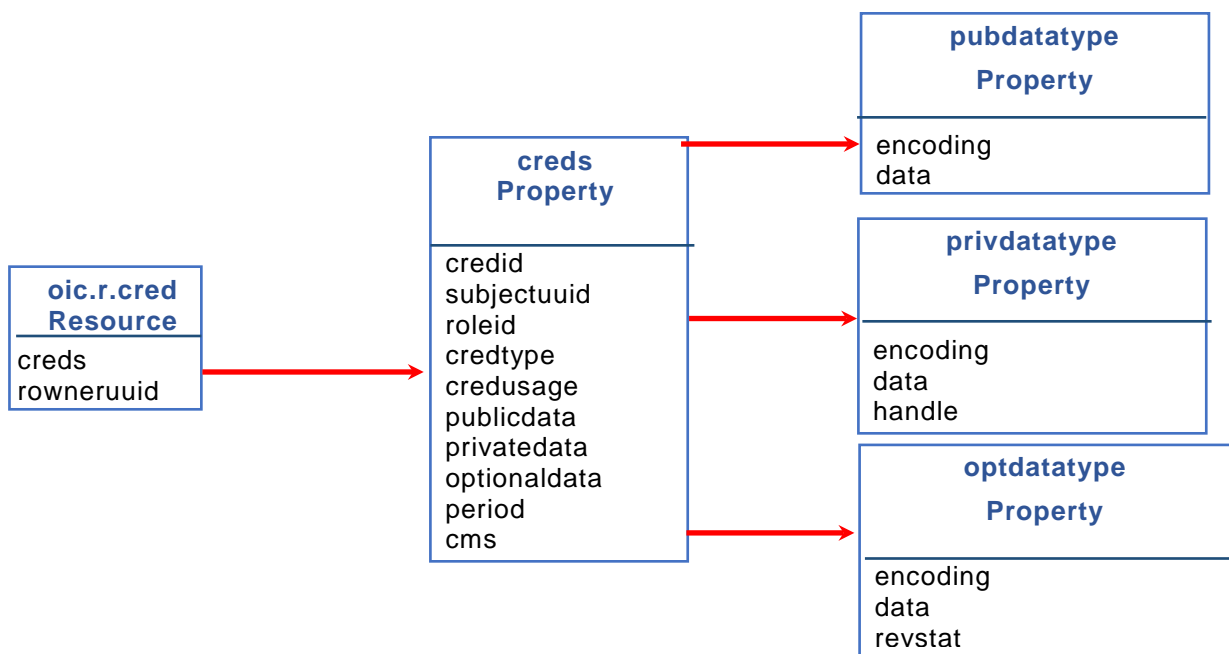
2644 13 Security Resources

2645 13.1 General



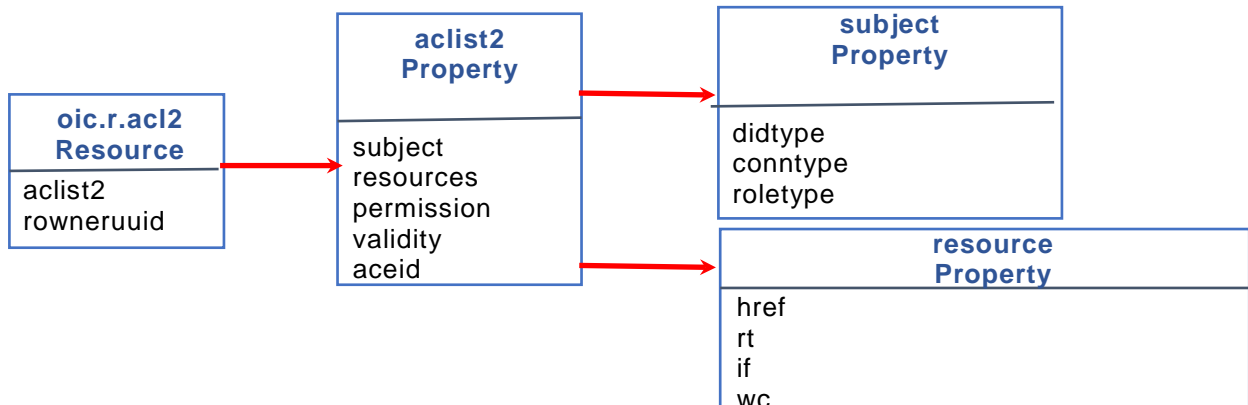
2646

Figure 35 – OCF Security Resources



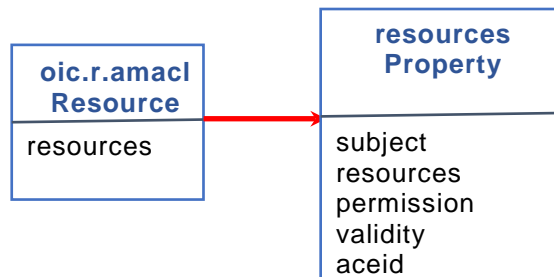
2647

Figure 36 – oic.r.cred Resource and Properties



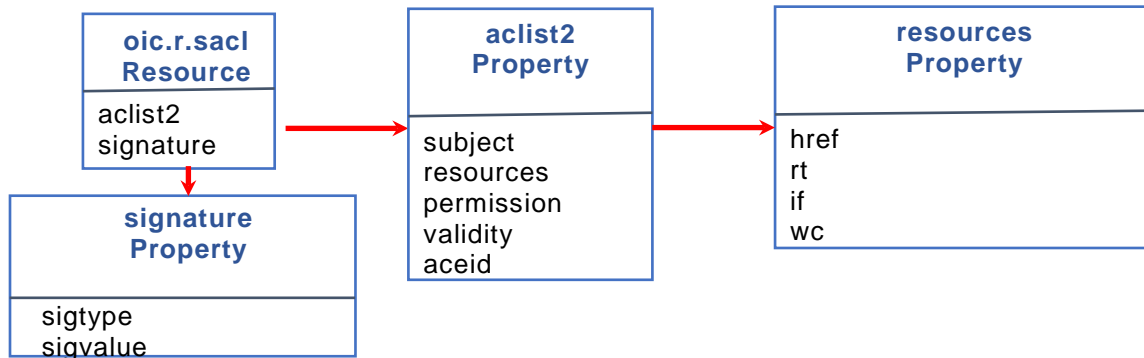
2648

Figure 37 – oic.r.acl2 Resource and Properties



2649

Figure 38 – oic.r.amacl Resource and Properties



2650

Figure 39 – oic.secr.sacl Resource and Properties

2651 13.2 Device Owner Transfer Resource

2652 13.2.1 Introduction

2653 The `/oic/sec/doxm` Resource contains the set of supported Device owner transfer methods.

2654 Resource discovery processing respects the CRUDN constraints supplied as part of the security
2655 Resource definitions contained in this specification.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfa ces	Description	Related Function al Interacti on
/oic/sec/doxm	Device Owner Transfer Methods	urn:oic.r.doxm	baseline	Resource for supporting Device owner transfer	Configurat ion

2656

Table 19 – Definition of the oic.r.doxm Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Owner Transfer Method	oxms	oic.sec.doxm type	array	Yes		R	Value identifying the owner-transfer-method and the organization that defined the method.
Oxm Selection	oxmsel	oic.sec.doxm type	UINT16	Yes	RESET	R	Server shall set to (4) "oic.sec.oxm.self"
					RFOTM	RW	The as yet unauthenticated DOXS shall set to its selected OTM and both parties execute the OTM. After secure owner transfer session is established, DOXS shall update the oxmsel again making it permanent. If the OTM 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.cred type	bitmask	Yes		R	Identifies the types of credentials the Device supports. The SRM sets this value at framework initialization after determining security capabilities.
Owned	owned	Boolean	T F	Yes	RESET	R	Server shall set to FALSE.
					RFOTM	RW	DOXS shall set to TRUE after secure owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a
Device UUID	deviceuuid	String	oic.sec.did type	Yes	RESET	R	Server shall construct a temporary random UUID that differs for each transition to RESET.
					RFOTM	RW	DOXS shall update to a value it has selected after secure owner transfer session is established. If update fails with error PROPERTY_NOT_FOUND the DOXS 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	devowneru uid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")

					RFOTM	RW	DOXS 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 DOXS should configure the rowneruuid property when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOXS (referenced via /doxm.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 DOXS the Server shall transition to RESET Device state.

2657

Table 20 – Properties of the oic.r.doxm Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Device ID	uuid	String	uuid	Yes	RW	-	A uuid value

2658

Table 21 - Properties of the oic.sec.didtype Property

2659 The owner transfer method (oxms) Property contains a list of owner transfer methods where the
 2660 entries appear in the order of preference. The Device manufacturer configures this Property with
 2661 the most desirable methods appearing before the lower priority methods. The network
 2662 management tool queries this list at the time of onboarding when the network management tool
 2663 selects the most appropriate method.

2664 Subsequent to an owner transfer method being chosen the agreed upon method shall be entered
 2665 into the /doxm Resource using the oxmsel Property.

2666 Owner transfer methods consist of two parts, a URN identifying the vendor or organization and the
 2667 specific method.

2668 **<OxmType> ::= "urn:" <NID> ":" <NSS>**

2669 **<NID> ::= <Vendor-Organization>**

2670 **<NSS> ::= <Method> | {<NameSpaceQualifier> "."} <Method>**

2671 **<NameSpaceQualifier> ::= String**

2672 **<Method> ::= String**

2673 **<Vendor-Organization> ::= String**

2674 When an owner transfer method successfully completes, the *owned* Property is set to '1' (TRUE).
2675 Consequently, subsequent attempts to take ownership of the Device will fail.

2676 The SRM generates a Device identifier (deviceuuid) that is stored in the /oic/sec/doxm Resource
2677 in response to successful ownership transfer.

2678 Owner transfer methods should communicate the deviceuuid to the service that is taking ownership.
2679 The service should associate the deviceuuid with the OC in a secured database.

2680 The Device vendor shall determine that the Device identifier (deviceuuid) is persistent (not
2681 updatable) or that it is non-persistent (updatable by the owner transfer service – a.k.a DOXS).

2682 If deviceuuid is persistent, the request to update shall fail with the error PROPERTY_NOT_FOUND.

2683 If it is non-persistent, the request to update shall succeed and the value supplied by DOXS shall
2684 be remembered until the Device is RESET. If the update fails for any other reason and Device
2685 state has not transitioned to RESET, the value of deviceuuid shall be the nil UUID (e.g. "00000000-
2686 0000-0000-0000-000000000000").

2687 Regardless of whether the Device has a persistent or non-persistent deviceuuid, a temporal
2688 random non-repeating UUID is found each time the Device enters RESET. The temporal
2689 deviceuuid is used while the Device state is in the RESET state and while in the RFOTM Device
2690 state until the DOXS establishes a secure OTM connection.

13.2.2 OCF defined owner transfer methods

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 OBT to assert ownership of the new Device. The first OBT 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 OBT and likewise authenticates the OBT to the Device. The Device allows the OBT to take ownership of the Device, after which a second attempt to take ownership by a different OBT will fail. Note: 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.
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a Device OBT. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the OBT signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec.doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.oxm.self	4	The manufacturer shall set the /doxm.oxmsel value to (4). The Server shall reset this value to (4) upon entering RESET Device state.
OCF Reserved	<Reserved>	5~0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	Reserved for vendor-specific OTM use

Table 22 – Properties of the oic.sec.doxmtype Property

2692

13.3 Credential Resource

2693

13.3.1 Introduction

2694

2695 The /oic/sec/cred Resource maintains credentials used to authenticate the Server to Clients and
2696 support services as well as credentials used to verify Clients and support services.

2697 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared
2698 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to
2699 distinguish the Clients and support services it recognizes by verifying an authentication
2700 challenge.

2701 In order to provide an interface which allows management of the "creds" Array Property, the
2702 RETRIEVE, UPDATE and DELETE operations on the oic.r.cred Resource shall behave as
2703 follows:

- 2704 1. A RETRIEVE shall return the full Resource representation, except that any write-only
2705 Properties shall be omitted (e.g. private key data).

- 2706 2. An UPDATE shall replace or add to the Properties included in the representation sent with
2707 the UPDATE request, as follows:
- 2708 a. If an UPDATE representation includes the "creds" array Property, then:
- 2709 i. Supplied creds with a "credid" that matches an existing "credid" shall
2710 replace completely the corresponding cred in the existing "creds" array.
- 2711 ii. Supplied creds without a "credid" shall be appended to the existing "creds"
2712 array, and a unique (to the cred Resource) "credid" shall be created and
2713 assigned to the new cred by the Server. The "credid" of a deleted cred
2714 should not be reused, to improve the determinism of the interface and
2715 reduce opportunity for race conditions.
- 2716 iii. Supplied creds with a "credid" that does not match an existing "credid" shall
2717 be appended to the existing "creds" array, using the supplied "credid".
- 2718 3. A DELETE without query parameters shall remove the entire "creds" array, but shall not
2719 remove the oic.r.cred Resource.
- 2720 4. A DELETE with one or more "credid" query parameters shall remove the cred(s) with the
2721 corresponding credid(s) from the "creds" array.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/cred	Credentials	urn:oic.r.cred	baseline	Resource containing credentials for Device authentication, verification and data protection	Security

2722

Table 23 – Definition of the oic.r.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 DOXS after successful OTM
					RFPRO	R	Set by the CMS (referenced via the /cred.owneruuid property) after successful authentication. Access to vertical resources is prohibited.
					RFNOP	R	Access to vertical resources is permitted after a matching ACE is found.
					SRESET	RW	The DOXS (referenced via /doxm.devowneruuid property) should evaluate the integrity of and may update creds entries when a secure session is established and the Server and DOXS 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 DOXS should configure the /cred.owneruuid property when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOXS (referenced via /doxm.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 DOXS the Server shall transition to RESET Device state.

2723

Table 24 – Properties of the oic.r.cred Resource

- 2724 All secure Device accesses shall have a /oic/sec/cred Resource that protects the end-to-end
2725 interaction.
- 2726 The /oic/sec/cred Resource can be created and modified by the services named in the 'owneruid'
2727 Property.
- 2728 ACLs naming /oic/sec/cred Resource should further restrict access beyond CRUDN access modes.

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
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	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: certificate
Private Data	privatedata	oic.sec.privdatatype	-	No			1:2: symmetric key 4: 8, 32, 64: Private asymmetric key 16: password hash, password value, security questions
					-	RESET	Server shall set to manufacturer default
					W	RFOTM	Set by DOXS after successful OTM
					W	RFPRO	Set by authenticated DOXS or CMS
					-	RFNOP	Not writable during normal operation.
					W	SRESET	DOXS may modify to enable transition to RFPRO.

Optional Data	optionaldata	oic.sec.optdata.type	-	No	RW		Credential revocation status information 1, 2, 4, 32: revocation status information 8: Revocation + CA certificate.
Period	period	String	-	No	RW		Period as defined by RFC5545. The credential should not be used if the current time is outside the Period window.
Credential Refresh Method	crms	oic.sec.crm.type	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.

2729

Table 25 – Properties of the oic.sec.cred Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	-	RW	No	A string specifying the encoding format of the data contained in the pubdata "oic.sec.encoding.jwt" - RFC7517 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - RFC 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	-	RW	No	The encoded value

2730

Table 26 – Properties of the oic.sec.pubdatatype Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	-	RW	Yes	A string specifying the encoding format of the data contained in the privdata "oic.sec.encoding.jwt" - RFC7517 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - RFC 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	-	RW	No	The encoded value This value shall never be readable.
Handle	handle	UINT16	-	RW	No	Handle to a key storage resource

Table 27 – Properties of the oic.sec.privdatatype 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	-	RW	No	A string specifying the encoding format of the data contained in the optdata "oic.sec.encoding.jwt" - RFC7517 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - RFC 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	-	RW	No	The encoded structure

Table 28 – Properties of the oic.sec.optdatatype Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Authority	authority	String	-	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	-	R	Yes	An identifier for the role. Must be expressible as an ASN.1 PrintableString.

Table 29 – Definition of the oic.sec.roletype Property.

2734 13.3.2 Properties of the Credential Resource

2735 13.3.2.1 Credential ID

2736 Credential ID (credid) is a local reference to a /oic/sec/cred instance. The SRM generates it. credid
2737 shall be used to disambiguate Resource instances that have the same Subject UUID.

2738 13.3.2.2 Subject UUID

2739 Subject UUID identifies the Device or service to which a credential Resource shall be used to
2740 establish a secure session, verify an authentication challenge-response or to authenticate an
2741 authentication challenge.

2742 A Subject UUID that matches the Server's own Device ID identifies credentials that authenticate
2743 this Device.

2744 Subject UUID shall be used to identify a group to which a group key is used to protect shared data.

2745 **13.3.2.3 Role ID**

2746 Role ID identifies the set of roles that have been granted to the Subject UUID. The asserted role
2747 or set of roles shall be a subset of the role values contained in the roleid Property.

2748 If a credential contains a set of roles, ACL matching succeeds if the asserted role is a member of
2749 the role set in the credential.

2750 **13.3.2.4 Credential Type**

2751 The Credential Type is used to interpret several of the other Property values whose contents can
2752 differ depending on the type of credential. These properties include publicdata, privatedata and
2753 optionaldata. The CredType value of '0' ("no security mode") is reserved for testing and debugging
2754 circumstances. Production deployments should not allow provisioning of credentials of type '0'.
2755 The SRM should introduce checking code that prevents its use in production deployments.

2756 **13.3.2.5 Public Data**

2757 Public Data contains information that provides additional context surrounding the issuance of the
2758 credential. For example, it might contain information included in a certificate or response data from
2759 a Key Management Service. It might contain wrapped data such as a SKDC issued ticket that has
2760 yet to be delivered.

2761 **13.3.2.6 Private Data**

2762 Private Data contains the secret information that is used to authenticate the Device, protect or
2763 unprotect data or verify an authentication challenge-response.

2764 Private Data shall not be disclosed outside of the SRM's trusted computing base. A secure element
2765 (SE) or trusted execution environment (TEE) should be used to implement the SRM's trusted
2766 computing base. In this situation, the Private Data contents should be a handle or reference to
2767 secure storage resources.

2768 **13.3.2.7 Optional Data**

2769 Optional Data contains information that is optionally supplied, but facilitates key management,
2770 scalability or performance optimization. For example, if the Credential Type identifies certificates,
2771 it contains a certificate revocation status value and the Certificate Authority (CA) certificate that
2772 will be used for mutual authentication.

2773 **13.3.2.8 Period**

2774 The Period Property identifies the validity period for the credential. If no validity period is specified
2775 the credential lifetime is undetermined. Constrained Devices that do not implement a date-time
2776 capability shall obtain current date-time information from its CMS.

2777 **13.3.2.9 Credential Refresh Method Type Definition**

2778 The oic.sec.crm defines the credential refresh methods that the CMS shall implement.

Value Type Name	Value Type URN	Applicable Credential Type	Description
Provisioning Service	oic.sec.crm.pro	All	A CMS initiates re-issuance of credentials nearing expiration. The Server should delete expired credentials to manage storage resources. The Resource Owner Property references the provisioning service. The Server uses its /oic/sec/cred.rowneruuid Resource to identify additional key management service that supports this credential refresh method.
Pre-shared Key	oic.sec.crm.psk	[1]	The Server performs ad-hoc key refresh by initiating a DTLS connection with the Device prior to credential expiration using a Diffie-Hellman based ciphersuite and the current PSK. The new DTLS MasterSecret value becomes the new PSK. The Server selects the new validity period. The new validity period value is sent to the Device who updates the validity period for the current credential. The Device acknowledges this update by returning a successful response or denies the update by returning a failure response. The Server uses its /oic/sec/cred.rowneruuid Resource to identify a key management service that supports this credential refresh method.
Random PIN	oic.sec.crm.rdp	[16]	The Server performs ad-hoc key refresh following the oic.sec.crm.psk approach, but in addition generates a random PIN value that is communicated out-of-band to the remote Device. The current PSK + PIN are hashed to form a new PSK' that is used with the DTLS ciphersuite. I.e. PSK' = SHA256(PSK, PIN). The Server uses its /oic/sec/cred.rowneruuid Resource to identify a key management service that supports this credential refresh method.
SKDC	oic.sec.crm.skdc	[1, 2, 4, 32]	The Server issues a request to obtain a ticket for the Device. The Server updates the credential using the information contained in the response to the ticket request. The Server uses its /oic/sec/cred.rowneruuid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruuid Resource to identify a key management service that supports this credential refresh method.
PKCS10	oic.sec.crm.pk10	[8]	The Server issues a PKCS#10 certificate request message to obtain a new certificate. The Server uses its /oic/sec/cred.rowneruuid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruuid Resource to identify a key management service that supports this credential refresh method.

2779 **Table 30 – Value Definition of the oic.sec.crmtype Property**

2780 **13.3.2.10 Credential Usage**

2781 Credential Usage indicates to the Device the circumstances in which a credential should be
2782 used. Five values are defined:

- 2783 • oic.sec.cred.trustca: This certificate is a trust anchor for the purposes of certificate chain
2784 validation, as defined in section 10.3.
- 2785 • oic.sec.cred.cert: This credusage is used for certificates for which the Device possesses
2786 the private key and uses it for identity authentication in a secure session, as defined in
2787 section 10.3.

- 2788 • oic.sec.cred.rolecert: This credusage is used for certificates for which the Device
2789 possesses the private key and uses to assert one or more roles, as defined in section
2790 10.3.1.
- 2791 • oic.sec.cred.mfgtrustca: This certificate is a trust anchor for the purposes of the
2792 Manufacturer Certificate Based Owner Transfer Method as defined in section 7.3.6.
- 2793 • oic.sec.cred.mfgcert: This certificate is used for certificates for which the Device possesses
2794 the private key and uses it for authentication in the Manufacturer Certificate Based Owner
2795 Transfer Method as defined in section 7.3.6.

2796 13.3.3 Key Formatting

2797 13.3.3.1 Symmetric Key Formatting

2798 Symmetric keys shall have the following format:

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.

2799 **Table 31 – 128-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.

2800 **Table 32 – 256-bit symmetric key**

2801 13.3.3.2 Asymmetric Keys

2802 Note: Asymmetric key formatting is not available in this revision of the specification.

2803 13.3.3.3 Asymmetric Keys with Certificate

2804 Key formatting is defined by certificate definition.

2805 13.3.3.4 Passwords

2806 Technical Note: Password formatting is not available in this revision of the specification.

2807 13.3.4 Credential Refresh Method Details

2808 13.3.4.1 Provisioning Service

2809 The resource owner identifies the provisioning service. If the Server determines a credential
2810 requires refresh and the other methods do not apply or fail, the Server will request re-provisioning
2811 of the credential before expiration. If the credential is allowed to expire, the Server should delete
2812 the Resource.

2813 13.3.4.2 Pre-Shared Key

2814 Using this mode, the current PSK is used to establish a Diffie-Hellman session key in DTLS. The
2815 TLS_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

2816 $PSK = TLS_PRF(MasterSecret, Message, length);$

- 2817 • MasterSecret – is the MasterSecret value resulting from the DTLS handshake
2818 using one of the above ciphersuites.
- 2819 • Message is the concatenation of the following values:
 - 2820 ○ RM - Refresh method – I.e. "oic.sec.crm.psk"
 - 2821 ○ Device ID_A is the string representation of the Device ID that supplied the
2822 DTLS ClientHello.
 - 2823 ○ Device ID_B is the Device responding to the DTLS ClientHello message
- 2824 • Length of Message in bytes.

2825 Both Server and Client use the PSK to update the /oic/sec/cred Resource's privatedata Property.
2826 If Server initiated the credential refresh, it selects the new validity period. The Server sends the
2827 chosen validity period to the Client over the newly established DTLS session so it can update it's
2828 corresponding credential Resource for the Server.

2829 **13.3.4.3 Random PIN**

2830 Using this mode, the current unexpired PIN is used to generate a PSK following RFC2898. The
2831 PSK is used during the Diffie-Hellman exchange to produce a new session key. The session key
2832 should be used to switch from PIN to PSK mode.

2833 The PIN is randomly generated by the Server and communicated to the Client through an out-of-
2834 band method. The OOB method used is out-of-scope.

2835 The pseudo-random function (PBKDF2) defined by RFC2898. PIN is a shared value used to
2836 generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a DTLS
2837 ciphersuite that accepts a PSK.

2838
$$\text{PPSK} = \text{PBKDF2}(\text{PRF}, \text{PIN}, \text{RM}, \text{Device ID}, c, \text{dkLen})$$

2839 The PBKDF2 function has the following parameters:

- 2840 - PRF – Uses the DTLS PRF.
- 2841 - PIN – Shared between Devices.
- 2842 - RM - Refresh method – I.e. "oic.sec.crm.rdp"
- 2843 - Device ID – UUID of the new Device.
- 2844 - c – Iteration count initialized to 1000, incremented upon each use.
- 2845 - dkLen – Desired length of the derived PSK in octets.

2846 Both Server and Client use the PPSK to update the /oic/sec/cred Resource's PrivateData Property.
2847 If Server initiated the credential refresh, it selects the new validity period. The Server sends the
2848 chosen validity period to the Client over the newly established DTLS session so it can update its
2849 corresponding credential Resource for the Server.

2850 **13.3.4.4 SKDC**

2851 A DTLS session is opened to the /oic/sec/cred.rownruuid with svctype="oic.sec.cms" that
2852 supports the oic.sec.crm.skdc credential refresh method. A ticket request message is delivered to
2853 the oic.sec.cms service and in response returns the ticket request. The Server updates or
2854 instantiates an /oic/sec/cred Resource guided by the ticket response contents.

2855 **13.3.4.5 PKCS10**

2856 A DTLS session is opened to the /oic/sec/cred.rownruuid with svctype="oic.sec.cms" that
2857 supports the oic.sec.crm.pk10 credential refresh method. A PKCS10 formatted message is
2858 delivered to the service. After the refreshed certificate is issued, the oic.sec.cms service pushes

2859 the certificate to the Server. The Server updates or instantiates an /oic/sec/cred Resource guided
 2860 by the certificate contents.

2861 **13.3.4.6 Resource Owner**

2862 The Resource Owner Property allows credential provisioning to occur soon after Device
 2863 onboarding before access to support services has been established. It identifies the entity
 2864 authorized to manage the /oic/sec/cred Resource in response to Device recovery situations.

2865 **13.4 Certificate Revocation List**

2866 **13.4.1 CRL Resource Definition**

2867 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated
 2868 with a separate crl Resource that is newly defined for maintaining the revocation list.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/crl	CRLs	urn:oic.r.crl	baseline	Resource containing CRLs for Device certificate revocation	Security

2869 **Table 33 – Definition of the oic.r.crl Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
CRL Id	crlid	UINT16	0 – 64K-1	RW	Yes	CRL ID for references from other Resource
This Update	thisupdate	String	-	RW	Yes	This indicates the time when this CRL has been updated.(UTC)
CRL Data	crldata	String	-	RW	Yes	CRL data based on CertificateList in CRL profile

2870 **Table 34 – Properties of the oic.r.crl Resource**

2871 **13.5 ACL Resources**

2872 **13.5.1 General**

2873 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be
 2874 expressed using three ACL Resource Types: /oic/sec/acl, /oic/sec/amacl and /oic/sec/sacl. The
 2875 subject (e.g. Device ID of the Client) requesting access to a Resource shall be authenticated prior
 2876 to applying the ACL check. Resources that are available to anyone can use a wildcard subject
 2877 reference. All Resource accessible via the unsecured communication channel shall be named
 2878 using the wildcard subject.

2879 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

2880 ACL structure in Backus-Naur Form (BNF) notation:

<ACL>	<ACE> {<ACE>}
<ACE>	<SubjectId> <ResourceRef> <Permission> {<Validity>}
<SubjectId>	<DeviceId> <Wildcard> <RoleId>
<DeviceId>	<UUID>
<RoleId>	[<Authority>] <RoleName> {<RoleName>}
<RoleName>	<URI>
<Authority>	<UUID>
<ResourceRef>	' (' <OIC_LINK> {',' {OIC_LINK}> } ')'
<Permission>	('C' '-')

<Validity>	<Period> {<Recurrence>}
<Wildcard>	'*'
<URI>	RFC3986 // OCF Core Specification defined
<UUID>	RFC4122 // OCF Core Specification defined
<Period>	RFC5545 Period
<Recurrence>	RFC5545 Recurrence
<OIC_LINK>	OCF Core Specification defined in JSON Schema

2881

Table 35 – BNF Definition of OCF ACL

2882
2883

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

2884
2885

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

2886
2887

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

2888
2889

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

2890

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

2891
2892

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

2893
2894
2895

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

2896

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

2897
2898
2899

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

2900

13.5.3 ACL Resource

2901
2902
2903

There are two types of ACLs, 'acl' is a list of type 'ace' and 'acl2' is a list of type 'ace2'. A Device shall not host the /acl Resource. Note: the /acl Resource is defined for backward compatibility and use by Provisioning Tools, etc.

2904
2905
2906

In order to provide an interface which allows management of "aces2" (for /oic/sec/acl2 Resource) Array Property, the RETRIEVE, UPDATE and DELETE operations on the oic.r.ace2 Resource shall behave as follows:

2907

1. A RETRIEVE shall return the full Resource representation.

2908
2909

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

2910

a. If an UPDATE representation includes the array Property, then:

2911
2912

i. Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace completely the corresponding ACE in the existing "aces2" array.

2913
2914
2915
2916
2917

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

2918 iii. Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be
2919 appended to the existing "aces2" array, using the supplied "aceid".

2920 3. A DELETE without query parameters shall remove the entire "aces2" array, but shall not
2921 remove the oic.r.ace2 Resource.

2922 4. A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the
2923 corresponding aceid(s) from the "aces2" array.

2924 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no
2925 entry can be found for the requested Resource for the requestor – e.g. /oic/sec/acl, /oic/sec/sacl
2926 and /oic/sec/amacl do not match the subject and the requested Resource.

2927 If an access manager ACL satisfies the request, the Server opens a secure connection to the AMS.
2928 If the primary AMS is unavailable, a secondary AMS should be tried. The Server queries the AMS
2929 supplying the subject and requested resource as filter criteria. The Server Device ID is taken from
2930 the secure connection context and included as filter criteria by the AMS. If the AMS policy satisfies
2931 the Permission Property is returned.

2932 If the requested Resource is still not matched, the Server returns an error. The requester should
2933 query the Server to discover the configured AMS services. The Client should contact the AMS to
2934 request a sacl (/oic/sec/sacl) Resource. Performing the following operations implement this type
2935 of request:

2936 1. Client: Open secure connection to AMS.

2937 2. Client: GET /oic/sec/acl?device="urn:uuid:XXX...",resource="URI"

2938 3. AMS: constructs a /oic/sec/sacl Resource that is signed by the AMS and returns it in
2939 response to the GET command.

2940 4. Client: POST /oic/sec/sacl [{ ...sacl... }]

2941 5. Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

2942 6. Client: retries original Resource access request. This time the new ACL is included in the
2943 local acl evaluation.

2944 The ACL contained in the /oic/sec/sacl Resource should grant longer term access that satisfies
2945 repeated Resource requests.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/acl	ACL	urn:oic.r.acl	baseline	Resource for managing access	Security

2946

Table 36 – Definition of the oic.r.acl Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
ACE List	acelist	oic.sec.ace	-	Yes		-	Access Control Entries in the ACL resource. This Property contains "aces", an array of oic.sec.ace1 resources and "aces2", an array of oic.sec.ace2 Resources
					R	RESET	Server shall set to manufacturer defaults.
					RW	RFOTM	The OBT shall configure select oic.sec.ace2 entries after a secure session is established.
					RW	RFPRO	The AMS (referenced via rowneruuid property) shall update the oic.sec.ace2 entries after mutually authenticated secure session is established. Access to vertical resources is prohibited.
					R	RFNOP	Access to vertical resources is permitted after a matching ACE is found.
					RW	SRESET	The OBT (referenced via devowneruuid property) should evaluate the integrity of and may update oic.sec.ace2 entries when a secure session is established and the Server and OBT are authenticated.
Resource Owner ID	rowneruuid	String	uuid	Yes	-	-	The resource owner property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
					R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOXS should configure the /acl.rowneruuid and /acl2.rowneruuid property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOXS (referenced via /doxm.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 DOXS the Server shall transition to RESET Device state..

Table 37 – Properties of the oic.r.acl Resource

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Resources	resources	oic.oic-link	array		Yes	The application's Resources to which a security policy applies
Permission	permission	oic.sec.crudntype	bitmask	RW	Yes	Bitmask encoding of CRUDN permission
Validity	validity	oic.sec.ace/definitions/time-interval	array	RW	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the RFC5545 Period, and a string array representing a recurrence rule using the RFC5545 Recurrence.
Subject ID	subjectuuid	String	uuid, ""	RW	Yes	A uuid that identifies the Device to which this ACE applies to or "" for anonymous access.

2948

Table 38 – Properties of the oic.r.ace Property

Value	Access Policy	Description	Notes
bx0000,0000 (0)	No permissions	No permissions	
bx0000,0001 (1)	C	CREATE	
bx0000,0010 (2)	R	RETRIEVE, OBSERVE, DISCOVER	Note that the "R" permission bit covers both the Read permission and the Observe permission.
bx0000,0100 (4)	U	WRITE, UPDATE	
bx0000,1000 (8)	D	DELETE	
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

2949

Table 39 – Value Definition of the oic.sec.crudntype Property

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/acl2	ACL2	oic.r.acl2	baseline	Resource for managing access	Security

2950

Table 40 – Definition of the oic.sec.acl2 Resource

Property Name	Value Type	Mandatory	Device State	Access Mode	Description
aclist2	array of oic.sec.ace2				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.
			RESET	R	Server shall set to manufacturer defaults.
			RFOTM	RW	The OBT shall configure select oic.sec.ace2 entries after a secure session is established.
			RFPRO	RW	The AMS (referenced via rowneruuid property) shall update the oic.sec.ace2 entries after mutually authenticated secure session is established. Access to vertical resources is prohibited.
			RFNOP	R	Access to vertical resources is permitted after a matching ACE is found.
			SRESET	RW	The OBT (referenced via devowneruuid property) should evaluate the integrity of and may update oic.sec.ace2 entries when a secure session is established and the Server and OBT are authenticated.
rowneruid	uuid	Yes			Same as rowneruid in oic.sec.acl

Table 41 – Properties of the oic.sec.acl2 Resource

2951

2952

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 RFC5545 Period, and a string array representing a recurrence rule using the RFC5545 Recurrence.
aceid	integer	Yes	An aceid is unique with respect to the 'aces' array.

2953

Table 42 – oic.sec.ace2 data type definition.

Property Name	Value Type	Mandatory	Description
href	uri	No	A URI referring to a resource to which the containing ACE applies
rt	array of strings	No	The resource types to which the containing ACE applies
if	array of strings	No	The interfaces to which the containing ACE applies
wc	string	No	A wildcard matching policy where: " + " – Matches all discoverable resources " - " – Matches all non-discoverable resources " * " – Matches all resources

2954

Table 43 – oic.sec.ace2.resource-ref data type definition.

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

2955

Table 44 – Value definition oic.sec.conntype Property

2956 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack
 2957 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's
 2958 request using policy in the ACL.

2959 Resources named in the ACL policy should be fully qualified or partially qualified. Fully qualified
 2960 Resource references should include the Device ID of a remote Device hosting the Resources.

2961 Partially qualified references means the local Resource Server is hosting the Resource. If a fully
2962 qualified resource reference is given, the Intermediary enforcing access shall have a secure
2963 channel to the Resource Server and the Resource Server shall verify the Intermediary is authorized
2964 to act on its behalf as a Resource access enforcement point.

2965 Resource Servers should include references to Device and ACL Resources where access
2966 enforcement is to be applied. However, access enforcement logic shall not depend on these
2967 references for access control processing as access to Server Resources will have already been
2968 granted.

2969 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and
2970 modify this Resource. This prevents non-terminating dependency on some other ACL Resource.
2971 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL
2972 Resource.

2973 An ACE or ACE2 entry is called *currently valid* if the validity period of the ACE or ACE2 entry
2974 includes the time of the request. Note that the validity period in the ACE or ACE2 may be a
2975 recurring time period (e.g., daily from 1:00-2:00). Matching the resource(s) specified in a request
2976 to the resource property of the ACE or ACE2 is defined in Section 12.2. For example, one way
2977 they can match is if the Resource URI in the request exactly matches one of the resource
2978 references in the ACE or ACE2 entries.

2979 A request will match an ACE if any of the following are true:

2980 1. The deviceuuid associated with the secure session matches the "subjectuuid" of the ACE; AND
2981 the resource of the request matches one of the "resources" of the ACE; AND the ACE is
2982 currently valid.

2983 2. The ACE "subjectuuid" contains the wildcard "*" character; AND the resource of the request
2984 matches one of the "resources" of the ACE; AND the ACE is currently valid.

2985 3. When authentication uses a symmetric key credential;

2986 AND the CoAP payload query string of the request specifies a role, which is associated with
2987 the symmetric key credential of the current secure session;

2988 AND the CoAP payload query string of the request specifies a role, which is contained in the
2989 oic.r.cred.creds.roleid property of the current secure session;

2990 AND the resource of the request matches one of the "resources" of the ACE;

2991 AND the ACE is currently valid.

2992 A request will match an ACE2 if any of the following are true:

2993 1. The ACE2 "subject" is of type oic.sec.didtype has a UUID value that matches the deviceuuid
2994 associated with the secure session;

2995 AND the resource of the request matches one of the "resources" of the ACE2
2996 oic.sec.ace2.resource-ref;

2997 AND the ACE2 is currently valid.

2998 2. The ACE2 "subject" is of type oic.sec.conntype and has the wildcard value that matches the
2999 currently established connection type;

3000 AND the resource of the request matches one of the "resources" of the ACE2
3001 oic.sec.ace2.resource-ref;

3002 AND the ACE2 is currently valid.

3003 3. When Client authentication uses a certificate credential;

3004 AND one of the roleid values contained in the role certificate matches the "roleid" of the
3005 ACE2 oic.sec.roletype;

3006 AND the role certificate public key matches the public key of the certificate used to establish
3007 the current secure session;

3008 AND the resource of the request matches one of the "resources" of the ACE2
3009 oic.sec.ace2.resource-ref;

3010 AND the ACE2 is currently valid.

3011 4. When Client authentication uses a certificate credential;

3012 AND the CoAP payload query string of the request specifies a role, which is member of the
3013 set of roles contained in the role certificate;

3014 AND the roleid values contained in the role certificate matches the "roleid" of the ACE2
3015 oic.sec.roletype;

3016 AND the role certificate public key matches the public key of the certificate used to establish
3017 the current secure session;

3018 AND the resource of the request matches one of the "resources" of the ACE2
3019 oic.sec.ace2.resource-ref;

3020 AND the ACE2 is currently valid.

3021 5. When Client authentication uses a symmetric key credential;

3022 AND one of the roleid values associated with the symmetric key credential used in the secure
3023 session, matches the "roleid" of the ACE2 oic.sec.roletype;

3024 AND the resource of the request matches one of the "resources" of the ACE2
3025 oic.sec.ace2.resource-ref;

3026 AND the ACE2 is currently valid.

3027 6. When Client authentication uses a symmetric key credential;

3028 AND the CoAP payload query string of the request specifies a role, which is contained in the
3029 oic.r.cred.creds.roleid property of the current secure session;

3030 AND CoAP payload query string of the request specifies a role that matches the "roleid" of
3031 the ACE2 oic.sec.roletype;

3032 AND the resource of the request matches one of the "resources" of the ACE2
3033 oic.sec.ace2.resource-ref;

3034 AND the ACE2 is currently valid.

3035 A request is granted if ANY of the 'matching' ACEs contains the permission to allow the
3036 request. Otherwise, the request is denied.

3037 Note that there is no way for an ACE to explicitly deny permission to a resource. Therefore, if one
3038 Device with a given role should have slightly different permissions than another Device with the
3039 same role, they must be provisioned with different roles.

3040 **13.6 Access Manager ACL Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/amacl	Managed ACL	urn:oic.r.amacl	baseline	Resource for managing access	Security

3041 **Table 45 – Definition of the oic.r.amacl Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Resources	resources	oic.sec.ace2	array	RW	Yes	Multiple links to this host's Resources

3042 **Table 46 – Properties of the oic.r.amacl Resource**

3043 **13.7 Signed ACL Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/sacl	Signed ACL	urn:oic.r.sacl	baseline	Resource for managing access	Security

3044 **Table 47 – Definition of the oic.r.sacl Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
ACE List	aclist2	oic.sec.ace2	array	RW	Yes	Access Control Entries in the ACL Resource
Signature	signature	oic.sec.sigtype	-	RW	Yes	The signature over the ACL Resource

3045 **Table 48 – Properties of the oic.r.sacl Resource**

Property Title	Property Name	Value Type	Value Rule	Unit	Access Mode	Mandatory	Description
Signature Type	sigtype	String	-	-	RW	Yes	The string specifying the predefined signature format. "oic.sec.sigtype.jws" – RFC7515 JSON web signature (JWS) object "oic.sec.sigtype.pk7" – RFC2315 base64-encoded object "oic.sec.sigtype.cws" – CBOR-encoded JWS object
Signature Value	sigvalue	String	-	-	RW	Yes	The encoded signature

3046 **Table 49 – Properties of the oic.sec.sigtype Property**

3047 **13.8 Provisioning Status Resource**

3048 The /oic/sec/pstat Resource maintains the Device provisioning status. Device provisioning should
3049 be Client-directed or Server-directed. Client-directed provisioning relies on a Client Device to

3050 determine what, how and when Server Resources should be instantiated and updated. Server-
3051 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-
3052 directed provisioning depends on configuration of the /oic/sec/cred.rownruuid and /oic/sec/cred
3053 Resources, at least minimally, to bootstrap the Server with settings necessary to open a secure
3054 connection with appropriate support services.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/pstat	Provisioning Status	urn:oic.r.pstat	baseline	Resource for managing Device provisioning status	Configuration

3055

Table 50 – Definition of the oic.r.pstat Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	-	Yes	RW		Device Onboarding State
Is Operational	isop	Boolean	T F	Yes	R	RESET	Device can function even when Cm is non-zero. Device will only service requests related to satisfying Tm when IsOp is FALSE. 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		Server shall set to 0000,0001
					R		Should be set by DOXS after successful OTM to 00xx,xx10.
					R		Set by CMS, AMS, DOXS after successful authentication
					R		Set by CMS, AMS, DOXS after successful authentication
					R		Server shall set to 0000,0001
Target Mode	tm	oic.sec.dpmttype	bitmask	No	R		Server shall set to 0000,0010
					RW		Set by DOXS after successful OTM
					RW		Set by CMS, AMS, DOXS after successful authentication
					RW		Set by CMS, AMS, DOXS after successful authentication
					RW		Set by DOXS as needed to recover from failures. Server shall set to XXXX,XX00 upon entry into SRESET.
Operational Mode	om	oic.sec.pomtype	bitmask	Yes	R		Server shall set to manufacturer default.
					RW		Set by DOXS after successful OTM
					RW		Set by CMS, AMS, DOXS after successful authentication
					RW		Set by CMS, AMS, DOXS after successful authentication
					RW		Set by DOXS.
Supported Mode	sm	oic.sec.pomtype	bitmask	Yes	R		Supported provisioning services operation modes

Device UUID	deviceuuid	String	uuid	Yes	RW		[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruuid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOXS should configure the /pstat.rowneruuid property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOXS (referenced via /doxm.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 DOXS the Server shall transition to RESET Device state.

3056

Table 51 – Properties of the oic.r.pstat Resource

3057 The provisioning status Resource /oic/sec/pstat is used to enable Devices to perform self-directed
3058 provisioning. Devices are aware of their current configuration status and a target configuration
3059 objective. When there is a difference between current and target status, the Device should consult
3060 the rowneruuid Property of /oic/sec/cred Resource to discover whether any suitable provisioning
3061 services exist. The Device should request provisioning if configured to do so. The om Property of
3062 /oic/sec/pstat Resource will specify expected Device behaviour under these circumstances.

3063 Self-directed provisioning enables Devices to function with greater autonomy to minimize
3064 dependence on a central provisioning authority that should be a single point of failure in the
3065 network.

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	The Device is in a Ready-For-Owner-Transfer-Method state. Set by DOXS after successful OTM to RFPRO.
					RW	RFPRO	The Device is in a Ready-For-PROvisioning state. Set by CMS, AMS, DOXS after successful authentication
					RW	RFNOP	The Device is in a Ready-For-Normal-Operation state. Set by CMS, AMS, DOXS after successful authentication
					RW	SRESET	The Device is in a Soft-Reset state. State transitions can be effected remotely by writing to /pstat.dos.s when the caller is authenticated and authorized to update the dos.s property. Set by CMS, AMS, DOXS after successful authentication
Pending state	p	Boolean	T F	Y	R		TRUE (1) – 's' state is pending until all necessary changes to Device resources are complete FALSE (0) – 's' state changes are complete

Table 52 – Properties of the oic.sec.dostype Property

3066

3067 In all states:

3068

- The /pstat.dos.p Property is read-only by all requestors.

3069

3070

3071

3072

3073

3074

3075

- An authenticated client can effect a Device state change by updating pstat.dos.s=<device_state>. Doing so instructs the Server to automatically perform all the changes required to transition to the designated Device state. There may be multiple steps, hence the dos.p value is set to TRUE as the first step and remains TRUE until all steps are complete. The final step sets dos.p to FALSE. The dos.s value is set to the designated Device state at the same time the dos.p value is set to FALSE. A client may observe /pstat.dos to be notified when a Device state change is completed.

3076

3077

- The Client is authenticated to write to /pstat.dos.s if it possesses an appropriate role (e.g. DOXS, CMS, AMS).

3078

3079

- Write requests to /pstat.dos.s where the requestor tries to transition to a state that isn't reachable will result in a DEVICE_STATE_NOT_PERMITTED error.

3080

3081

- Write requests to /pstat.dos.s when /pstat.dos.p is TRUE will result in a DEVICE_STATE_NOT_READY error.

3082

- /pstat.dos.p must be set to TRUE on entrance.

3083

- /pstat.dos.p property must be set to FALSE on exit.

3084 When Device state is RESET:

- 3085 • All SVR content is removed and reset to manufacturer default values.
- 3086 • The default manufacturer Device state is RESET.
- 3087 • Vertical resources are reset to manufacturer default values.
- 3088 • Vertical resources are inaccessible.
- 3089 • If client subscribers OBSERVE dos.p=FALSE, the notification is sent prior to the point
3090 where access control and credential resources (needed to deliver the message) are
3091 dismantled.
- 3092 • After successfully processing RESET the SRM transitions to RFOTM by setting /pstat.dos.s
3093 to RFOTM.

3094 When Device state is RFOTM:

- 3095 • Vertical Resources are inaccessible.
- 3096 • Before OTM is successful, the deviceid Property of /oic/sec/doxm Resource must be set to
3097 a randomized UUID value.
- 3098 • Before OTM is successful, the /pstat.dos.s Property is read-only by unauthenticated
3099 requestors
- 3100 • After the OTM is successful, the /pstat.dos.s Property is read-write by authorized
3101 requestors.
- 3102 • The negotiated Device owner credential is used to create an authenticated session over
3103 which the OBT directs the Device state to transition to RFPRO.
- 3104 • If an authenticated session cannot be established when the OTM completes after <td=60>
3105 seconds, the SRM asserts the OTM failed and transitions to RESET (/pstat.dos.s=RESET).
3106 (Note: The transfer of ownership is considered complete when /doxm.owned is set to
3107 TRUE. The Device state may continue in RFOTM to complete initial provisioning.)

3108 When Device state is RFPRO:

- 3109 • The /pstat.dos.s Property is read-only by unauthorized requestors and read-write by
3110 authorized requestors.
- 3111 • Vertical Resources are inaccessible.
- 3112 • The OCF Server may re-create vertical Resources.
- 3113 • An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
- 3114 • An authorized Client may perform consistency checks on SVRs to determine which shall
3115 be re-provisioned.
- 3116 • Failure to successfully provision SVRs may trigger a state change to RESET. For
3117 example, if the Device has already transitioned from SRESET but consistency checks
3118 continue to fail.
- 3119 • The authorized Client sets the /pstat.dos.s=RFNOP.

3120 When Device state is RFNOP:

- 3121 • The /pstat.dos.s Property is read-only by unauthorized requestors and read-write by
3122 authorized requestors.
- 3123 • Vertical resources, SVRs and core Resources are accessible following normal access
3124 processing.
- 3125 • An authorized may transition to RFPRO. Only the Device owner may transition to
3126 SRESET or RESET.

3127 When Device state is SRESET:

- 3128 • Vertical Resources are inaccessible. The integrity of vertical Resources may be suspect
3129 but the SRM doesn't attempt to access or reference them.
- 3130 • SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
3131 include /doxm.devowner, /cred[<devowner>] and /pstat.dos.
- 3132 • The certificates that identify and authorize the Device owner are sufficient to re-create
3133 minimalist /cred and /doxm resources enabling Device owner control of SRESET. If the
3134 SRM can't establish these Resources, then it will transition to RESET state.
- 3135 • An authorized Client performs SVR consistency checks. The caller may provision SVRs
3136 as needed to ensure they are available for continued provisioning in RFPRO or for
3137 normal functioning in RFNOP.
- 3138 • The authorized Device owner may avoid entering RESET state and RFOTM by writing
3139 RFPRO or RFNOP to /pstat.dos.s.
- 3140 • ACLs on SVR are presumed to be invalid. Access authorization is granted according to
3141 Device owner privileges.
- 3142 • The SRM asserts a Client-directed operational mode (e.g.
3143 /pstat.om=CLIENT_DIRECTED).

3144 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning
3145 modes. "{ProvisioningMode}" should be used in this document to refer to an instance of a
3146 provisioning mode without selecting any particular value.

Type Name	Type URN	Description
Device Provisioning Mode	urn:oc.sec.dpmttype	Device provisioning mode is a 16-bit bitmask describing various provisioning modes

3147

Table 53 – Definition of the oic.sec.dpmttype Property

Value	Device Mode	Description
bx0000,0001 (1)	Reset	Device reset mode enabling manufacturer reset operations
bx0000,0010 (2)	Take Owner	Device pairing mode enabling owner transfer operations
bx0000,0100 (4)	Bootstrap Service	Service provisioning mode enabling instantiation of a BSS. This allows authorized entities to install a BSS.
bx0000,1000 (8)	Security Management Services	Service provisioning mode enabling instantiation of Device security services and related credentials
bx0001,0000 (16)	Provision Credentials	Credential provisioning mode enabling instantiation of pairwise Device credentials using a management service of type urn:oc.sec.cms
bx0010,0000 (32)	Provision ACLs	ACL provisioning mode enabling instantiation of Device ACLs using a management service of type urn:oc.sec.ams
bx0100,0000 (64)	Initiate Software Version Validation	Software version validation requested/pending (1) Software version validation complete (0)
bx1000,0000 (128)	Initiate Secure Software Update	Secure software update requested/pending (1) Secure software update complete (0)

3148

Table 54 – Value Definition of the oic.sec.dpmttype Property (Low-Byte)

Value	Device Mode	Description
bx0000,0000 – bx1111,1111	<Reserved>	Reserved for later use

3149

Table 55 – Value Definition of the oic.sec.dpmttype Property (High-Byte)

3150 The *provisioning operation mode* type is a 8-bit mask enumerating the various provisioning
3151 operation modes.

Type Name	Type URN	Description
Device Provisioning OperationMode	urn:oc.sec.pomtype	Device provisioning operation mode is a 8-bit bitmask describing various provisioning operation modes

3152

Table 56 – Definition of the oc.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

3153

Table 57 – Value Definition of the oc.sec.pomtype Property

3154 **13.9 Certificate Signing Request Resource**

3155 The /oc/sec/csr Resource is used by a Device to provide its desired identity, public key to be
 3156 certified, and a proof of possession of the corresponding private key in the form of a RFC 2986
 3157 PKCS#10 Certification Request. If the Device supports certificates (the sct property of
 3158 /oc/sec/doxm has a 1 in the 0x8 bit position), the Device shall have a /oc/sec/csr resource.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/csr	Certificate Signing Request	urn:oic.r.csr	baseline	The CSR resource contains a Certificate Signing Request for the Device's public key.	Configuration

3159

Table 58 – Definition 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

3160

Table 59 – Properties of the oic.r.csr Resource

3161 The Device chooses which public key to use, and may optionally generate a new key pair for this
3162 purpose.

3163 In the CSR, the Common Name component of the Subject Name shall contain a string of the format
3164 "uuid:X" where X is the Device's requested UUID in the format defined by RFC 4122. The Common
3165 Name, and other components of the Subject Name, may contain other data. If the Device chooses
3166 to include additional information in the Common Name component, it shall delimit it from the UUID
3167 field by white space, a comma, or a semicolon.

3168 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to generate
3169 a new key pair, the Device may begin generation of a key pair as a result of a RETRIEVE of this
3170 resource. If the Device cannot immediately respond to the RETRIEVE request due to time required
3171 to generate a key pair, the Device shall return an "operation pending" error. This indicates to the
3172 Client that the Device is not yet ready to respond, but will be able at a later time. The Client should
3173 retry the request after a short delay.

3174 **13.10 Roles resource**

3175 The roles resource maintains roles that have been asserted with role certificates, as described in
3176 Section 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role
3177 certificate. Clients may only access roles associated with their public key of the certificate used
3178 to authenticate during (D)TLS session establishment. The roles resource should be viewed as an
3179 extension of the (D)TLS session state. See section 10.4.2 for how role certificates are validated.

3180 The roles resource shall be created by the server upon establishment of a secure (D)TLS session
3181 with a client, if is not already created. A server shall retain the roles resource at least as long as
3182 the (D)TLS session exists. A server shall retain each certificate in the roles resource at least until
3183 the certificate expires or the (D)TLS session ends, whichever is sooner. A server may retain the
3184 roles resource and its contents beyond the length of a (D)TLS session or a certificate's validity
3185 period, although the requirements of section 10.3 and 10.4.2 to validate a certificate's time validity
3186 at the point of use always apply. A server should regularly inspect the contents of the roles
3187 resource and purge contents based on a policy it determines based on its resource constraints.

3188 For example, expired certificates, and certificates from clients that have not been heard from for
 3189 some arbitrary period of time could be candidates for purging.

3190 As stated above, the resource is implicitly created by the server upon establishment of a (D)TLS
 3191 session. In more detail, the RETRIEVE, UPDATE and DELETE operations on the Roles Resource
 3192 should behave as follows. Unlisted operations are implementation specific and not reliable. Note
 3193 that this description is editorial, and the RAML provides the normative and formal behaviour
 3194 description.

3195 1. Retrieve shall return all previously asserted roles associated with the client’s public key. Note
 3196 that the public key is always available to the server as part of the secure channel information.
 3197 Retrieve with query parameters is not supported.

3198 2. Update includes the "roles" array property and distinct roles in this array are added to the
 3199 resource. This is also scoped to the client’s public key. Two roles are distinct if either of the
 3200 "role" or "authority" properties differs.

3201 3. Delete shall remove the entire "roles" array for the client’s public key.

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/sec/roles	Roles	urn:oic.r.roles	baseline	Resource containing roles that have previously been asserted to this server	Security

3202 **Table 60 – Definition of the oic.r.roles Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Roles	roles	oic.sec.cred	array	RW	Yes	List of roles previously asserted to this server

3203 **Table 61 – Properties of the oic.r.roles Resource**

3204 **13.11 Security Virtual Resources (SVRs) and Access Policy**

3205 The SVRs expose the security-related Properties of the Device.

3206 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to unauthenticated
 3207 (anonymous) Clients could create privacy or security concerns.

3208 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for
 3209 the oic.r.doxm Resource to anonymous requesters, so that the Device can be discovered and
 3210 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the oic.r.doxm
 3211 Resource to anonymous requesters, to preserve privacy.

3212 **13.12 SVRs, Discoverability and Endpoints**

3213 All implemented SVRs shall be “discoverable” (reference OCF Core Specification, Policy
 3214 Parameter section 7.8.2.1.2).

3215 All implemented discoverable SVRs shall expose a Secure Endpoint (e.g. CoAPS) (reference OCF
 3216 Core Specification, Endpoint chapter 10).

3217 The /doxm Resource shall expose an Unsecure Endpoint (e.g. CoAP) in RFOTM (reference OCF
 3218 Core Specification, Endpoint chapter 10).

3219 **13.13 Privacy Consideration for Core and SVRs**

3220 Unique identifiers are a privacy consideration due to their potential for being used as a tracking
3221 mechanism. These include the following Resources and Properties:

- 3222 • /d Resource containing the 'di' and 'piid' Properties.
- 3223 • /p Resource containing the 'pi' Property.
- 3224 • /doxm Resource containing the 'deviceuuid' Property.

3225 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous
3226 requestors. This implies any Client Device, including those with malicious intent, are able to
3227 reliably obtain identifiers useful for building a log of activity correlated with a specific Platform and
3228 Device.

3229 There are two strategies for privacy protection of Devices:

- 3230 1. Apply an ACL policy that restricts read access to Resources containing unique identifiers
- 3231 2. Limit identifier persistence to make it impractical for tracking use.

3232 Both techniques can be used effectively together to limit exposure to privacy attacks.

- 3233 1. A Platform / Device manufacturer should specify a default ACL policy that restricts
3234 anonymous requestors from accessing unique identifiers. A network administrator should
3235 modify the ACL policy to grant access to authenticated Devices who, presumably, do not
3236 present a privacy threat.
- 3237 2. Servers shall supply a temporary, non-repeating Device ID when the 'owned' Property in
3238 the /doxm Resource is FALSE and applies over multiple ownership transitions. The
3239 temporary identifiers are disjoint from and not correlated to the persistent identifiers shall
3240 be:
 - 3241 a. Disjoint from (i.e. not linked) the persistent identifiers
- 3242 3. Generated by a function that is pre-image resistant, second pre-image resistant and
3243 collision resistant

3244 A new Device seeking deployment needs to inform would-be OBTs of the identifier used to begin
3245 the onboarding process. However, attackers could obtain the value too and use it for Device
3246 tracking throughout the Device's lifetime. To address this privacy threat, Servers shall supply the
3247 temporary 'deviceuuid' to unauthenticate /oic/res requests when the 'deviceowneruuid' is the nil
3248 UUID. The Server shall generate a new pseudo-random temporary 'deviceuuid' value when the
3249 Device state transitions to RESET. This ensures the 'deviceuuid' value cannot be used to track
3250 across multiple owners.

3251 The 'deviceowneruuid' property is initialized to the nil UUID at RESET which is retained until being
3252 set during RFOTM Device state. The Device shall supply a non-persistent identifier to RETRIEVE
3253 requests on /oic/sec/doxm and /oic/res Resources while 'deviceowneruuid' is the nil UUID. The
3254 onboarding utility shall update the 'devowneruuid' to a non-nil UUID value that triggers the Server
3255 to allow the persistent 'deviceuuid' to be returned in RETRIEVE requests to the /doxm and /res
3256 resources.

3257 The onboarding utility may also provision an ACL policy that restricts access to the /oic/sec/doxm
3258 resource such that only authenticated Clients are able to obtain the persistent 'deviceuuid' value.
3259 Clients avoid making unauthenticated discovery requests by having been provisioned with a
3260 /oic/sec/cred resource entry that contains the Server's 'deviceuuid'.

3261 The 'di' property in the /oic/d Resource shall mirror that of the 'deviceuuid' Property. The
 3262 onboarding utility should provision an ACL policy that restricts access to the /oic/d Resource such
 3263 that only authenticated Clients are able to obtain the persistent 'di' value.

3264 The 'piid' Property in the /oic/d Resource similarly should present a temporary and changing value
 3265 when 'deviceowneruuid' has the nil UUID value. The server shall provide a persistent value (or
 3266 allows the onboarding utility to provision) subsequent to 'deviceowneruuid' being changed to a
 3267 non-nil UUID. An ACL policy on the /d resource protects the 'piid' from being disclosed to
 3268 anonymous requestors.

3269 The 'pi' Property in the /oic/p Resource shall have a temporary and changing value when
 3270 'deviceowneruuid' has the nil UUID value and shall change to a persistent value upon
 3271 'deviceowneruuid' being changed to a non-nil UUID. An ACL policy shall protect the 'pi' property
 3272 in the /p resource from being disclosed to anonymous requestors.

Resource Type	Property title	Property name	Value type	Access Mode		Behavior
oic.wk.p	Platform ID	pi	oic.types-schema.uuid	All States	R	Server shall construct a temporary random UUID. (does not override the persistent pi)
oic.wk.p	Protocol Independent Identifier	piid	oic.types-schema.uuid	RESET, SRESET, RFPRO, RFNOP	R	Server should construct a temporary random UUID when entering RESET state.
				RFOTM	RW	DOXS may set the persistent value after secure owner transfer session is established; otherwise Server sets value.
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.

3273 **Table 62 – Core Resource Properties state**

3274

3275 **14 Core Interaction Patterns Security**

3276 This section is left intentionally blank and will be defined in future version..

3277 **14.1 Observer**

3278 **14.2 Subscription/Notification**

3279 **14.3 Groups**

3280 **14.4 Publish-subscribe Patterns and Notification**

3281

3282 **15 Security Hardening Guidelines/ Execution Environment Security**

3283 **15.1 General**

3284 This is an informative section. Many TGs in OCF have security considerations for their protocols
3285 and environments. These security considerations are addressed through security mechanisms
3286 specified in the security specifications for OCF. However, effectiveness of these mechanisms
3287 depends on security robustness of the underlying hardware and software Platform. This section
3288 defines the components required for execution environment security.

3289 **15.2 Execution environment elements**

3290 **15.2.1 General**

3291 Execution environment within a computing Device has many components. To perform security
3292 functions in a robustness manner, each of these components has to be secured as a separate
3293 dimension. For instance, an execution environment performing AES cannot be considered secure
3294 if the input path entering keys into the execution engine is not secured, even though the partitions
3295 of the CPU, performing the AES encryption, operate in isolation from other processes. Different
3296 dimensions referred to as elements of the execution environment are listed below. To qualify as a
3297 secure execution environment (SEE), the corresponding SEE element must qualify as secure.

- 3298 • (Secure) Storage
- 3299 • (Secure) Execution engine
- 3300 • (Trusted) Input/output paths
- 3301 • (Secure) Time Source/clock
- 3302 • (Random) number generator
- 3303 • (Approved) cryptographic algorithms
- 3304 • Hardware Tamper (protection)

3305 Note that software security practices (such as those covered by OWASP) are outside scope of this
3306 specification, as development of secure code is a practice to be followed by the open source
3307 development community. This specification will however address the underlying Platform
3308 assistance required for executing software. Examples are secure boot and secure software
3309 upgrade.

3310 Each of the elements above are described in the following subsections.

3311 **15.2.2 Secure Storage**

3312 **15.2.2.1 General**

3313 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive
3314 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,
3315 certificate data, network access credentials, or personal user information. Sensitive Data requires
3316 that its integrity be maintained, whereas *Critical* Sensitive Data requires that both its integrity and
3317 confidentiality be maintained.

3318 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive
3319 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either
3320 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication
3321 and encryption, it must maintain its integrity against intentional or accidental alteration.

3322 A partial list of Sensitive Data is outlined below:

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

3323

Table 63 – Examples of Sensitive Data

3324 Exact method of protection for secure storage is implementation specific, but typically
3325 combinations of hardware and software methods are used.

3326 **15.2.2.2 Hardware secure storage**

3327 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric
3328 and asymmetric private keys, access credentials, and personal private data. Hardware secure
3329 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes
3330 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

3331 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides
3332 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or electronic
3333 attacks. It is not necessary to prevent the attacks themselves, but an attempted attack should not
3334 result in an unauthorized entity successfully retrieving Sensitive Data.

3335 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data
3336 from attacks that include but are not limited to:

- 3337 1) Physical decapping of chip packages to optically read NVRAM contents
- 3338 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 3339 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit
3340 patterns of Critical Sensitive Data
- 3341 4) Use of malicious software or firmware to read memory contents at rest or in transit within
3342 a microcontroller
- 3343 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive
3344 Data

3345 **15.2.2.3 Software Storage**

3346 It is generally NOT recommended to rely solely on software and unsecured memory to store
3347 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and encryption
3348 keys should be housed in hardware secure storage whenever possible.

3349 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable
3350 algorithms to prevent access by unauthorized parties through methods described in Section
3351 15.2.2.2.

3352 **15.2.2.4 Additional Security Guidelines and Best Practices**

3353 Below are some general practices that can help ensure that Sensitive Data is not compromised by
3354 various forms of security attacks:

3355 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG
3356 used for authentication challenges can substantially degrade security strength. For this
3357 reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise
3358 source be used for all authentication challenges.

3359 2) Secure download and boot – To prevent the loading and execution of malicious software,
3360 where it is practical, it is recommended that Secure Download and Secure Boot methods
3361 that authenticate a binary's source as well as its contents be used.

3362 3) Deprecated algorithms –Algorithms included but not limited to the list below are considered
3363 unsecure and shall not be used for any security-related function:

- 3364 a. SHA-1
- 3365 b. MD5
- 3366 c. RC4
- 3367 d. RSA 1024

3368 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is
3369 stored in Secure Storage, any use of that data that requires its transmission out of that
3370 Secure Storage should be encrypted to prevent eavesdropping by malicious software within
3371 an MCU/MPU.

3372 **15.2.3 Secure execution engine**

3373 Execution engine is the part of computing Platform that processes security functions, such as
3374 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine requires
3375 the following

- 3376 • Isolation of execution of sensitive processes from unauthorized parties/ processes. This
3377 includes isolation of CPU caches, and all of execution elements that needed to be
3378 considered as part of trusted (crypto) boundary.
- 3379 • Isolation of data paths into and out of execution engine. For instance both unencrypted but
3380 sensitive data prior to encryption or after decryption, or cryptographic keys used for
3381 cryptographic algorithms, such as decryption or signing. See trusted paths for more details.

3382 **15.2.4 Trusted input/output paths**

3383 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be protected.
3384 This includes paths into and out secure execution engine and secure memory.

3385 Path protection can be both hardware based (e.g. use of a privileged bus) or software based (using
3386 encryption over an untrusted bus).
3387

3388 **15.2.5 Secure clock**

3389 Many security functions depend on time-sensitive credentials. Examples are time stamped
3390 Kerberos tickets, OAuth tokens, X.509 certificates, OSCP response, software upgrades, etc. Lack
3391 of secure source of clock can mean an attacker can modify the system clock and fool the validation
3392 mechanism. Thus an SEE needs to provide a secure source of time that is protected from
3393 tampering. Note that trustworthiness from security robustness standpoint is not the same as
3394 accuracy. Protocols such as NTP can provide rather accurate time sources from the network, but
3395 are not immune to attacks. A secure time source on the other hand can be off by seconds or
3396 minutes depending on the time-sensitivity of the corresponding security mechanism. Note that
3397 secure time source can be external as long as it is signed by a trusted source and the signature
3398 validation in the local Device is a trusted process (e.g. backed by secure boot).

3399 **15.2.6 Approved algorithms**

3400 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and
3401 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by
3402 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only
3403 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic
3404 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are
3405 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms (even
3406 if they deemed stronger by some parties) must be considered non-approved.

3407 The set of algorithms to be considered for approval are algorithms for

- 3408 • Hash functions
- 3409 • Signature algorithms
- 3410 • Encryption algorithms
- 3411 • Key exchange algorithms
- 3412 • Pseudo Random functions (PRF) used for key derivation

3413 This list will be included in this or a separate security robustness rules specification and must be
3414 followed for all security specifications within OCF.

3415 **15.2.7 Hardware tamper protection**

3416 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not
3417 requirements) regarding tamper protection for cryptographic module

- 3418 • Production-grade (lowest level): this means components that include conformal sealing
3419 coating applied over the module's circuitry to protect against environmental or other
3420 physical damage. This does not however require zeroization of secret material during
3421 physical maintenance. This definition is borrowed from FIPS 140-2 security level 1.
- 3422 • Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,
3423 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from
3424 FIPS 140-2 security level 2.
- 3425 • Tamper resistance (highest level), this means there is a response to physical tampering
3426 that typically includes zeroization of sensitive material on the module. This definition is
3427 borrowed from FIPS 140-2 security level 3.

3428 It is difficult of specify quantitative certification test cases for accreditation of these levels. Content
3429 protection regimes usually talk about different tools (widely available, specialized and professional
3430 tools) used to circumvent the hardware protections put in place by manufacturing. If needed, OCF

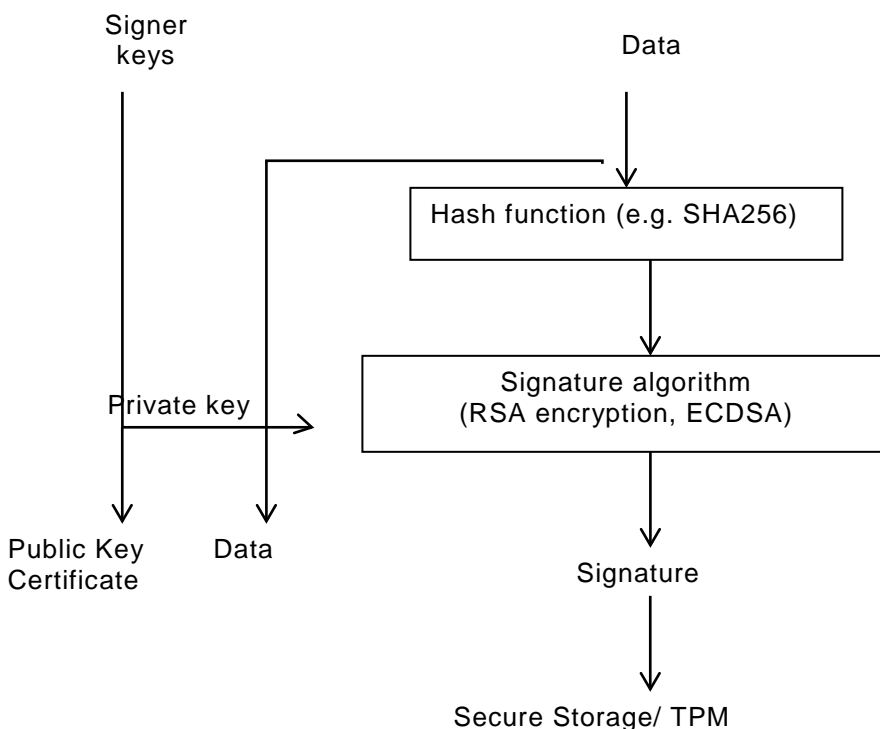
3431 can follow that model, if and when OCF engage in distributing sensitive key material (e.g. PKI) to
3432 its members.

3433 15.3 Secure Boot

3434 15.3.1 Concept of software module authentication

3435 In order to ensure that all components of a Device are operating properly and have not been
3436 tampered with, it is best to ensure that the Device is booted properly. There may be multiple stages
3437 of boot. The end result is an application running on top an operating system that takes advantage
3438 of memory, CPU and peripherals through drivers.

3439 The general concept is the each software module is invoked only after cryptographic integrity
3440 verification is complete. The integrity verification relies on the software module having been
3441 hashed (e.g. SHA_1, SHA_256) and then signed with a cryptographic signature algorithm with (e.g.
3442 RSA), with a key that only a signing authority has access to.



3443 **Figure 40 – Software Module Authentication**

3444 After the data is signed with the signer’s signing key (a private key), the verification key (the public
3445 key corresponding to the private signing key) is provided for later verification. For lower level
3446 software modules, such as bootloaders, the signatures and verification keys are inserted inside
3447 tamper proof memory, such as One time programmable memory or TPM. For higher level software
3448 modules, such as application software, the signing is typically performed according to the PKCS#7
3449 format (IETF CMS RFC), where the signedData format includes both indications for signature
3450 algorithm, hash algorithm as well as the signature verification key (or certificate). The secure boot
3451 specification however does not require use of PKCS#7 format.

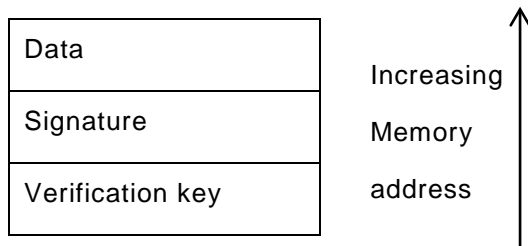


Figure 41 – Verification Software Module

3452
 3453 The verification module first decrypts the signature with the verification key (public key of the
 3454 signer). The verification module also calculates a hash of the data and then compares the
 3455 decrypted signature (the original) with the hash of data (actual) and if the two values match, the
 3456 software module is authentic.

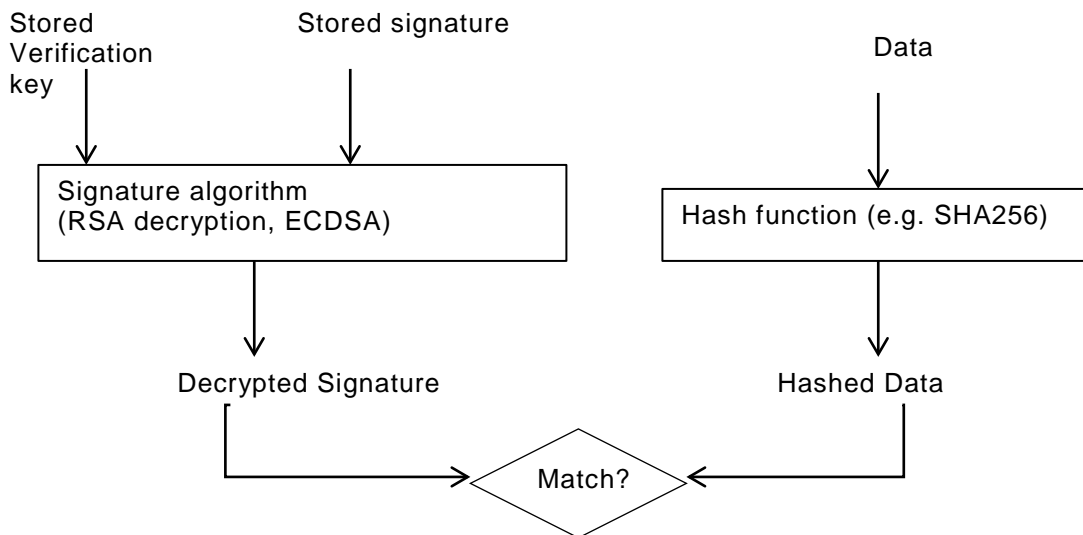


Figure 42 – Software Module Authenticity

3457
 3458 **15.3.2 Secure Boot process**

3459 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/
 3460 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to
 3461 find out where the boot code is and then run the boot code (second-stage boot loader). The second
 3462 stage bootloader is typically the process that loads the operating system (Kernel) and transfers
 3463 the execution to the where the Kernel code is. Once the Kernel starts, it may load external Kernel
 3464 modules and drivers.

3465 When performing a secure boot, it is required that the integrity of each boot loader is verified before
 3466 executing the boot loader stage. As mentioned, while the signature and verification key for the
 3467 lowest level bootloader is typically stored in tamper-proof memory, the signature and verification
 3468 key for higher levels should be embedded (but attached in an easily accessible manner) in the
 3469 data structures software.

3470 **15.3.3 Robustness requirements**

3471 **15.3.3.1 General**

3472 To qualify as high robustness secure boot process, the signature and hash algorithms shall be one
3473 of the approved algorithms, the signature values and the keys used for verification shall be stored
3474 in secure storage and the algorithms shall run inside a secure execution environment and the keys
3475 shall be provided the SEE over trusted path.

3476 **15.3.3.2 Next steps**

3477 Develop a list of approved algorithms and data formats

3478 **15.4 Software Update**

3479 **15.4.1 Overview:**

3480 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;
3481 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and
3482 end-of-life stages for the Device remain outstanding. It is possible for the Device to require update
3483 during any of these stages, although the most likely times are during onboarding, regular operation
3484 and maintenance. The aspects of the software include, but are not limited to, firmware, operating
3485 system, networking stack, application code, drivers, etc.

3486 **15.4.2 Recognition of Current Differences**

3487 Different manufacturers approach software update utilizing a collection of tools and strategies:
3488 over-the-air or wired USB connections, full or partial replacement of existing software, signed and
3489 verified code, attestation of the delivery package, verification of the source of the code, package
3490 structures for the software, etc.

3491 It is recommended that manufacturers review their processes and technologies for compliance with
3492 industry best-practices that a thorough security review of these takes place and that periodic
3493 review continue after the initial architecture has been established.

3494 This specification applies to software updates as recommended to be implemented by Devices; it
3495 does not have any bearing on the above-mentioned alternative proprietary software update
3496 mechanisms.

3497 **15.4.3 Software Version Validation**

3498 Setting the Initiate Software Version Validation bit in the /oic/sec/pstat.tm Property (see Table 51
3499 of Section 13.8) indicates a request to initiate the software version validation process, the process
3500 whereby the Device validates the software (including firmware, operating system, Device drivers,
3501 networking stack, etc.) against a trusted source to see if, at the conclusion of the check, the
3502 software update process will need to be triggered (see below). When the Initiate Software Version
3503 Validation bit of /oic/sec/pstat.tm is set to 1 (TRUE) by a sufficiently privileged Client, the Device
3504 sets the /oic/sec/pstat.cm Initiate Software Version Validation bit to 0 and initiates a software
3505 version check. Once the Device has determined if an update is available, it sets the Initiate
3506 Software Version Validation bit in the /oic/sec/pstat.cm property to 1 (TRUE) if an update is
3507 available or 0 (FALSE) if no update is available. To signal completion of the Software Version
3508 Validation process, the Device sets the Initiate Software Version Validation bit in the
3509 /oic/sec/pstat.tm Property back to 0 (FALSE). If the Initiate Software Version Validation bit of
3510 /oic/sec/pstat.tm is set to 0 (FALSE) by a Client, it has no effect on the validation process.

3511 **15.4.4 Software Update**

3512 Setting the Initiate Secure Software Update bit in the /oic/sec/pstat.tm property (see Table 51 of
3513 Section 13.8) indicates a request to initiate the software update process. When the Initiate Secure
3514 Software Update bit of /oic/sec/pstat.tm is set to 1 (TRUE) by a sufficiently privileged Client, the
3515 Device sets the /oic/sec/pstat.cm Initiate Software Version Validation bit to 0 and initiates a

3516 software update process. Once the Device has completed the software update process, it sets the
3517 Initiate Secure Software Update bit in the /oic/sec/pstat.cm property to 1 (TRUE) if/when the
3518 software was successfully updated or 0 (FALSE) if no update was performed. To signal completion
3519 of the Secure Software Update process, the Device sets the Initiate Secure Software Update bit in
3520 the /oic/sec/pstat.tm Property back to 0 (FALSE). If the Initiate Secure Software Update bit of
3521 /oic/sec/pstat.tm is set to 0 (FALSE) by a Client, it has no effect on the update process.

3522 **15.4.5 Recommended Usage**

3523 The Initiate Secure Software Update bit of /oic/sec/pstat.tm should only be set by a Client after the
3524 Initiate Software Version Validation check is complete.

3525 The process of updating Device software may involve state changes that affect the Device
3526 Operational State (/oic/sec/pstat.dos). Devices with an interest in the Device(s) being updated
3527 should monitor /oic/sec/pstat.dos and be prepared for pending software update(s) to affect Device
3528 state(s) prior to completion of the update.

3529 Note that the Device itself may indicate that it is autonomously initiating a software version
3530 check/update or that a check/update is complete by setting the pstat.tm and pstat.cm Initiate
3531 Software Version Validation and Secure Software Update bits when starting or completing the
3532 version check or update process. As is the case with a Client-initiated update, Clients can be
3533 notified that an autonomous version check or software update is pending and/or complete by
3534 observing pstat resource changes.

3535 **15.5 Security Levels**

3536 Security Levels are a way to differentiate Devices based on their security criteria. This need for
3537 differentiation is based on the requirements from different verticals such as industrial and health
3538 care and may extend into smart home. This differentiation is distinct from Device classification
3539 (e.g. RFC7228)

3540 These categories of security differentiation may include, but is not limited to:

- 3541 1. Security Hardening
- 3542 2. Identity Attestation
- 3543 3. Certificate/Trust
- 3544 4. Onboarding Technique
- 3545 5. Regulatory Compliance
 - 3546 a. Data at rest
 - 3547 b. Data in transit
- 3548 6. Cipher Suites – Crypto Algorithms & Curves
- 3549 7. Key Length
- 3550 8. Secure Boot/Update

3551
3552 In the future security levels can be used to define interoperability.

3553
3554 The following applies to Security Specification 1.1:

3555 The current specification does not define any other level beyond Security Level 0. All Devices will
3556 be designated as Level 0. Future versions may define additional levels.

3557
3558 Note the following points:

- 3559 • The definition of a given security level will remain unchanged between versions of the
3560 specification.
- 3561 • Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 3562 • Devices may be evaluated and re-classified at a higher level if it meets the requirements
3563 of the higher level (e.g. if a Device is manufactured under the 1.1 version of the
3564 specification, and a later spec version defines a security level 1, the Device could be
3565 evaluated and classified as level 1 if it meets level 1 requirements).

3566 • The security levels may need to be visible to the end user.

3567 16 Appendix A: Access Control Examples

3568 16.1 Example OCF ACL Resource

3569 The Server is required to verify that any hosted Resource has authorized access by the Client
3570 requesting access. The /oic/sec/acl2 Resource is co-located on the Resource host so that the
3571 Resource request processing should be applied securely and efficiently. This example shows how
3572 a /oic/sec/acl2 Resource could be configured to enforce an example access policy on the Server.

```
3573 {
3574     "aclist2": [
3575         {
3576             // Subject with ID ...01 should access two named Resources with access
3577             mode "CRUDN" (Create, Retrieve, Update, Delete and Notify)
3578             "subject": {"uuid": "XXXX-...-XX01"},
3579             "resources": [
3580                 {"href": "/oic/sh/light/1"},
3581                 {"href": "/oic/sh/temp/0"}
3582             ],
3583             "permission": 31, // 31 dec = 0b0001 1111 which maps to --N DURC
3584             "validity": [
3585                 // The period starting at 18:00:00 UTC, on January 1, 2015 and
3586                 // ending at 07:00:00 UTC on January 2, 2015
3587                 "period": ["20150101T180000Z/20150102T070000Z"],
3588                 // Repeats the {period} every week until the last day of Jan.
3589                 2015.
3590                 "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
3591             ],
3592             "aceid": 1
3593         }
3594     ],
3595     // An ACL provisioning and management service should be identified as
3596     // the resource owner
3597     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
3598 }
```

3599 16.2 Example Access Manager Service

3600 The AMS should be used to centralize management of access policy, but requires Servers to open
3601 a connection to the AMS whenever the named Resources are accessed. This example
3602 demonstrates how the /oic/sec/amacl Resource should be configured to achieve this objective.

```
3603 {
3604     "resources": [
3605         // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
3606         // supplied then use the {4} sacl validation credential to enforce access.
3607         {"href": "/oic/sh/light/1"},
3608         // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
3609         // supplied then use the {4} sacl validation credential to enforce access.
3610         {"href": "/oma/3"},
3611         // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
3612         // the {4} sacl validation credential to enforce access.
3613         {"wc": "*" }
3614     ]
3615 }
```


3616 **17 Appendix B: Execution Environment Security Profiles**

3617 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security
3618 robustness requirements meeting all IOT applications and services will not serve the needs of OCF,
3619 and security profiles of varying degree of robustness (trustworthiness), cost and complexity have
3620 to be defined. To address a large ecosystem of vendors, the profiles can only be defined as
3621 requirements and the exact solutions meeting those requirements are specific to the vendors' open
3622 or proprietary implementations, and thus in most part outside scope of this document.

3623 To align with the rest of OCF specifications, where Device classifications follow IETF RFC 7228
3624 (Terminology for constrained node networks) methodology, we limit the number of security profiles
3625 to a maximum of 3. However, our understanding is OCF capabilities criteria for each of 3 classes
3626 will be more fit to the current IoT chip market than that of IETF.

3627 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are
3628 either capable of no security functionality or easily breakable security that depend on
3629 environmental (e.g. availability of human) factors to perform security functions. This means the
3630 class 0 will not be equipped with an SEE.

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

3631 **Table 64 – OCF Security Profile**

3632 Technical Note: This analysis acknowledges that these Platform classifications do not take into
3633 consideration of possibility of security co-processor or other hardware security capability that
3634 augments classification criteria (namely CPU speed, memory, storage).

3635

3636 **18 Appendix C: RAML Definition**

3637 All the sub-clauses in Appendix C describe the Security Resources with a restful API definition
 3638 language. The Resources presented in Appendix C are formatted for readability, and so may
 3639 appear to have extra line breaks. The contents of the Resource Types without the extra line breaks
 3640 are available in OCF Security Resource Definitions.

3641

Resource Name	Resource Type	Section
Access Control List	oic.r.acl	A.1
Access Control List 2	oic.r.acl2	A.2
Managed Access Control List	oic.r.amacl	A.3
Signed Access Control List	oic.r.sacl	A.4
Device Ownership Transfer	oic.r.doxm	A.5
Device Provisioning Status	oic.r.pstat	A.6
Credential	oic.r.cred	A.7
Certificate Signing Request	oic.r.csr	A.8
Roles	oic.r.roles	A.9
Certificate Revocation List	oic.r.crl	A.10

3642 **Table 65 – OCF SVR RAML**

3643 **A.1 OICSecurityAclResource**

3644 **A.1.1 Introduction**

3645 This resource specifies the local access control list.

3646 **A.1.2 Example URI**

3647 /oic/sec/acl

3648 **A.1.3 Resource Type**

3649 **A.1.4 RAML Definition**

```

3650 #%RAML 0.8
3651 title: OICSecurityAclResource
3652 version: v1.1-20161213
3653 traits:
3654   - interface :
3655     queryParameters:
3656       if:
3657         enum: ["oic.if.baseline"]
3658   - ace-filtered :
```

```

3659     queryParameters:
3660         subjectuuid:
3661
3662 /oic/sec/acl:
3663     description: |
3664         This resource specifies the local access control list.
3665
3666     is : ['interface']
3667
3668     get:
3669         description: |
3670             Retrieves the ACL entries.
3671             When used without query parameters, all the ACE entries are returned.
3672             When used with a subjectuuid, only the ACEs with the specified
3673             subjectuuid are returned
3674             If subjectuuid and resources are specified,
3675             only the ACEs with the specified subjectuuid and resource hrefs are
3676             returned.
3677
3678     is : ['ace-filtered']
3679
3680     responses :
3681         200:
3682             body:
3683                 application/json:
3684                     schema: /
3685                         {
3686                             "$schema": "http://json-schema.org/draft-04/schema#",
3687                             "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl.json#",
3688                             "title": "Access Control List information",
3689                             "definitions": {
3690                                 "oic.r.acl": {
3691                                     "type": "object",
3692                                     "properties": {
3693                                         "aclist": {
3694                                             "type": "object",
3695                                             "description": "Subject-based Access Control Entries in the ACL resource",
3696                                             "properties": {
3697                                                 "aces": {
3698                                                     "type": "array",
3699                                                     "items": {
3700                                                         "$ref": "oic.sec.ace.json#/definitions/oic.sec.ace"
3701                                                     }
3702                                                 }
3703                                             },
3704                                             "required": [ "aces" ]
3705                                         },
3706                                         "rowneruuid": {
3707                                             "description": "The value identifies the unique resource owner",
3708                                             "$ref": "../core/schemas/oic.types-schema.json#/definitions/uuid"
3709                                         }
3710                                     }
3711                                 }
3712                             },
3713                             "type": "object",
3714                             "allOf": [
3715                                 { "$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },
3716                                 { "$ref": "#/definitions/oic.r.acl" }
3717                             ],
3718                             "required": [ "aclist", "rowneruuid" ]
3719                         }
3720
3721     example: /

```

```

3720     {
3721         "aclist": {
3722             "aces": [
3723                 {
3724                     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
3725                     "resources": [
3726                         {
3727                             "href": "coaps://IP-ADDR/temp",
3728                             "rel": "some-rel",
3729                             "rt": ["oic.r.temperature"],
3730                             "if": ["oic.if.a"]
3731                         },
3732                         {
3733                             "href": "coaps://IP-ADDR/temp",
3734                             "rel": "some-rel",
3735                             "rt": ["oic.r.temperature"],
3736                             "if": ["oic.if.s"]
3737                         }
3738                     ],
3739                     "permission": 31,
3740                     "validity": [
3741                         {
3742                             "period": "20160101T180000Z/20170102T070000Z",
3743                             "recurrence": [ "DSTART:XXXXX",
3744 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
3745                         },
3746                         {
3747                             "period": "20160101T180000Z/PT5H30M",
3748                             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
3749                         }
3750                     ]
3751                 }
3752             ]
3753         },
3754         "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
3755     }
3756

```

3757 400:

```

3758     description: |
3759         The request is invalid.
3760

```

3761 post:

```

3762     description: |
3763         Updates the ACL resource with the provided values
3764         ACEs provided
3765         in the update not currently in the ACL are added
3766         ACEs that already
3767         exist in the ACL are ignored.
3768         Note that for the purposes of update, equivalency is determined
3769         by comparing the ACE subjectuuid, permission, string comparisons
3770         of all validity elements, and string comparisons of all resource
3771         hrefs.
3772

```

3773 body:

```

3774     application/json:
3775         schema: /
3776         {
3777             "$schema": "http://json-schema.org/draft-04/schema#",
3778             "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl.json#",
3779             "title": "Access Control List information",
3780             "definitions": {
3781                 "oic.r.acl": {
3782                     "type": "object",
3783                     "properties": {
3784                         "aclist": {
3785                             "type": "object",
3786                             "description": "Subject-based Access Control Entries in the ACL resource",

```

```

3787         "properties": {
3788             "aces": {
3789                 "type": "array",
3790                 "items": {
3791                     "$ref": "oic.sec.ace.json#/definitions/oic.sec.ace"
3792                 }
3793             },
3794         },
3795         "required": [ "aces" ]
3796     },
3797     "rowneruuid": {
3798         "description": "The value identifies the unique resource owner",
3799         "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
3800     }
3801 }
3802 },
3803 },
3804 "type": "object",
3805 "allOf": [
3806     { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
3807     { "$ref": "#/definitions/oic.r.acl" }
3808 ],
3809 "required": [ "aclist", "rowneruuid" ]
3810 }
3811

```

example: /

```

3812 {
3813     "aclist": {
3814         "aces": [
3815             {
3816                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
3817                 "resources": [
3818                     {
3819                         "href": "coaps://IP-ADDR/temp",
3820                         "rel": "some-rel",
3821                         "rt": [ "oic.r.temperature" ],
3822                         "if": [ "oic.if.a" ]
3823                     },
3824                     {
3825                         "href": "coaps://IP-ADDR/temp",
3826                         "rel": "some-rel",
3827                         "rt": [ "oic.r.temperature" ],
3828                         "if": [ "oic.if.s" ]
3829                     }
3830                 ]
3831             },
3832             "permission": 31,
3833             "validity": [
3834                 {
3835                     "period": "20160101T180000Z/20170102T070000Z",
3836                     "recurrence": [ "DSTART:XXXXX",
3837 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
3838                 },
3839                 {
3840                     "period": "20160101T180000Z/PT5H30M",
3841                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
3842                 }
3843             ]
3844         }
3845     ],
3846     "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
3847 }
3848

```

responses :

```

3850 400:
3851     description: |
3852         The request is invalid.
3853
3854

```

```

3855     201:
3856         description: |
3857             The ACL entry/entries is/are created.
3858
3859     204:
3860         description: |
3861             The ACL entry/entries is/are updated.
3862
3863     delete:
3864         description: |
3865             Deletes ACL entries.
3866             When DELETE is used without query parameters, all the ACE entries are deleted.
3867             When DELETE is used with a subjectuuid, only the ACEs with the specified
3868             subjectuuid are deleted
3869             If subjectuuid and resources are specified,
3870             only the ACEs with the specified subjectuuid and resource hrefs are
3871             deleted.
3872
3873     is : ['ace-filtered']
3874     responses :
3875         200:
3876             description: |
3877                 The matching ACEs or the entire ACL resource has been successfully deleted.
3878
3879         400:
3880             description: |
3881                 The request is invalid.
3882
3883

```

3883 A.1.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		The value identifies the unique resource owner
aclist	object: see schema	yes		Subject-based Access Control Entries in the ACL resource
aces (aclist)	array: see schema	yes		

3884 A.1.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/acl		get	post	delete	

3886 A.2 OICSecurityAcl2Resource

3887 A.2.1 Introduction

3888 This resource specifies the local access control list.

3889 A.2.2 Example URI

3890 /oic/sec/acl2

3891 **A.2.3 Resource Type**

3892 **A.2.4 RAML Definition**

```
3893 #%RAML 0.8
3894 title: OICSecurityAcl2Resource
3895 version: v1.0-20161214
3896 traits:
3897   - interface :
3898     queryParameters:
3899       if:
3900         enum: ["oic.if.baseline"]
3901   - ace-filtered :
3902     queryParameters:
3903       aceid:
3904
3905 /oic/sec/acl2:
3906   description: |
3907     This resource specifies the local access control list.
3908
3909   is : ['interface']
3910   get:
3911     description: |
3912       Retrieves the ACL data.
3913       When used without query parameters, all the ACE entries are returned.
3914       When used with a query parameter, only the ACEs matching the specified
3915       parameter are returned.
3916
3917   is : ['ace-filtered']
3918   responses :
3919     200:
3920       body:
3921         application/json:
3922           schema: /
3923             {
3924               "$schema": "http://json-schema.org/draft-04/schema#",
3925               "id": "https://www.openconnectivity.org/ocf-
3926 apis/security/schemas/oic.r.acl2.json#",
3927               "title": "Access Control List information",
3928               "definitions": {
3929                 "oic.r.acl2": {
3930                   "type": "object",
3931                   "properties": {
3932                     "aclist2": {
3933                       "type": "array",
3934                       "description": "Access Control Entries in the ACL resource",
3935                       "items": {
3936                         "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
3937                       }
3938                     },
3939                     "rowneruuid": {
3940                       "description": "The value identifies the unique resource owner",
3941                       "$ref": "../..../core/schemas/oic.types-schema.json#/definitions/uuid"
3942                     }
3943                   }
3944                 }
3945             },
3946             "type": "object",
3947             "allOf": [
3948               { "$ref": "../..../core/schemas/oic.core-schema.json#/definitions/oic.core" },
3949               { "$ref": "#/definitions/oic.r.acl2" }
3950             ],
3951
```

```

3951     "required": [ "aclist2", "rowneruuid" ]
3952   }
3953
3954   example: /
3955   {
3956     "aclist2": [
3957       {
3958         "aceid": 1,
3959         "subject": {
3960           "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
3961           "role": "SOME_STRING"
3962         },
3963         "resources": [
3964           {
3965             "href": "/light",
3966             "rt": ["oic.r.light"],
3967             "if": ["oic.if.baseline", "oic.if.a"]
3968           },
3969           {
3970             "href": "/door",
3971             "rt": ["oic.r.door"],
3972             "if": ["oic.if.baseline", "oic.if.a"]
3973           }
3974         ],
3975         "permission": 24
3976       },
3977       {
3978         "aceid": 2,
3979         "subject": {
3980           "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
3981         },
3982         "resources": [
3983           {
3984             "href": "/light",
3985             "rt": ["oic.r.light"],
3986             "if": ["oic.if.baseline", "oic.if.a"]
3987           },
3988           {
3989             "href": "/door",
3990             "rt": ["oic.r.door"],
3991             "if": ["oic.if.baseline", "oic.if.a"]
3992           }
3993         ],
3994         "permission": 24
3995       },
3996       {
3997         "aceid": 3,
3998         "subject": { "conntype": "anon-clear" },
3999         "resources": [
4000           {
4001             "href": "/light",
4002             "rt": ["oic.r.light"],
4003             "if": ["oic.if.baseline", "oic.if.a"]
4004           },
4005           {
4006             "href": "/door",
4007             "rt": ["oic.r.door"],
4008             "if": ["oic.if.baseline", "oic.if.a"]
4009           }
4010         ],
4011         "permission": 16,
4012         "validity": [
4013           {
4014             "period": "20160101T180000Z/20170102T070000Z",
4015             "recurrence": [ "DSTART:XXXXX",
4016 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4017           },
4018           {
4019             "period": "20160101T180000Z/PT5H30M",
4020             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]

```



```

4021         }
4022     ]
4023 }
4024 ],
4025 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
4026 }
4027
4028 400:
4029     description: |
4030         The request is invalid.
4031
4032 post:
4033     description: |
4034         Updates the ACL resource with the provided ACEs.
4035         ACEs provided in the update with aceids not currently in the ACL
4036         resource are added.
4037         ACEs provided in the update with aceid(s) already in the ACL completely
4038         replace the ACE(s) in the ACL resource.
4039         ACEs provided in the update without aceid properties are added and
4040         assigned unique aceids in the ACL resource.
4041
4042     body:
4043         application/json:
4044             schema: /
4045                 {
4046                     "$schema": "http://json-schema.org/draft-04/schema#",
4047                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl2.json#",
4048                     "title": "Access Control List information",
4049                     "definitions": {
4050                         "oic.r.acl2": {
4051                             "type": "object",
4052                             "properties": {
4053                                 "aclist2": {
4054                                     "type": "array",
4055                                     "description": "Access Control Entries in the ACL resource",
4056                                     "items": {
4057                                         "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
4058                                     }
4059                                 },
4060                                 "rowneruuid": {
4061                                     "description": "The value identifies the unique resource owner",
4062                                     "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
4063                                 }
4064                             }
4065                         }
4066                     },
4067                     "type": "object",
4068                     "allOf": [
4069                         { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
4070                         { "$ref": "#/definitions/oic.r.acl2" }
4071                     ],
4072                     "required": [ "aclist2", "rowneruuid" ]
4073                 }
4074
4075     example: /
4076         {
4077             "aclist2": [
4078                 {
4079                     "aceid": 1,
4080                     "subject": {
4081                         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4082                         "role": "SOME_STRING"
4083                     },
4084                     "resources": [
4085                         {
4086                             "href": "/light",

```

```

4087         "rt": ["oic.r.light"],
4088         "if": ["oic.if.baseline", "oic.if.a"]
4089     },
4090     {
4091         "href": "/door",
4092         "rt": ["oic.r.door"],
4093         "if": ["oic.if.baseline", "oic.if.a"]
4094     }
4095 ],
4096 "permission": 24
4097 },
4098 {
4099     "aceid": 3,
4100     "subject": {
4101         "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4102     },
4103     "resources": [
4104         {
4105             "href": "/light",
4106             "rt": ["oic.r.light"],
4107             "if": ["oic.if.baseline", "oic.if.a"]
4108         },
4109         {
4110             "href": "/door",
4111             "rt": ["oic.r.door"],
4112             "if": ["oic.if.baseline", "oic.if.a"]
4113         }
4114     ],
4115     "permission": 24
4116 }
4117 ],
4118 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
4119 }
4120
4121 responses :
4122 400:
4123     description: |
4124     The request is invalid.
4125
4126 201:
4127     description: |
4128     The ACL entry is created.
4129
4130 204:
4131     description: |
4132     The ACL entry is updated.
4133
4134 delete:
4135     description: |
4136     Deletes ACL entries.
4137     When DELETE is used without query parameters, all the ACE entries are deleted.
4138     When DELETE is used with a query parameter, only the ACEs matching the
4139     specified parameter are deleted.
4140
4141 is : ['ace-filtered']
4142 responses :
4143 200:
4144     description: |
4145     The matching ACEs or the entire ACL resource has been successfully deleted.
4146
4147 400:
4148     description: |

```

4149 The request is invalid.
4150

4151 **A.2.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		The value identifies the unique resource owner
aclist2	array: see schema	yes		Access Control Entries in the ACL resource

4152 **A.2.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/sec/acl2		get	post	delete	

4153

4154 **A.2.7 Referenced JSON schemas**

4155 **A.2.8 oic.sec.didtype.json**

4156 **A.2.9 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
uuid	multiple types: see schema	yes		A UUID Device ID

4157 **A.2.10 Schema Definition**

```
4158 {
4159   "$schema": "http://json-schema.org/draft-04/schema#",
4160   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.didtype.json#",
4161   "title": "Device Identifier Format type",
4162   "definitions": {
4163     "oic.sec.didtype": {
4164       "type": "object",
4165       "description": "Device identifier",
4166       "properties": {
4167         "uuid": {
4168           "description": "A UUID Device ID",
4169           "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
4170         }
4171       },
4172       "required": ["uuid"]
4173     }
4174   }
4175 }
4176
```

4177 **A.2.11 oic.sec.ace2.json**

4178 **A.2.12 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
aceid	integer			An identifier for the ACE that is unique within the ACL. In cases where it isn't supplied in an update, the Server will add the ACE and

				assign it a unique value.
subject	multiple types: see schema	yes		The subject to whom this ace applies, either a deviceId, role, or wildcard
validity	array: see schema			validity is an array of time-pattern objects
resources	array: see schema			References the application's resources to which a security policy applies
rt (resources)	multiple types: see schema			When present, the ACE only applies when the rt (resource type) matches
href (resources)	multiple types: see schema			When present, the ACE only applies when the href matches
wc (resources)	string			A wildcard matching policy
if (resources)	multiple types: see schema			When present, the ACE only applies when the if (interface) matches
permission	integer	yes		Bitmask encoding of CRUDN permission

4179 **A.2.13 Schema Definition**

```

4180 {
4181   "$schema": "http://json-schema.org/draft-04/schema#",
4182   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.ace2.json#",
4183   "title": "Subject-based Access Control Entry (ACE) object definition",
4184   "definitions": {
4185     "oic.sec.ace2": {
4186       "type": "object",
4187       "properties": {
4188         "aceid": {
4189           "type": "integer",
4190           "minimum": 1,
4191           "description": "An identifier for the ACE that is unique within the ACL. In cases where
4192 it isn't supplied in an update, the Server will add the ACE and assign it a unique value."
4193         },
4194         "resources": {
4195           "type": "array",
4196           "description": "References the application's resources to which a security policy
4197 applies",
4198           "items": {
4199             "type": "object",
4200             "description": "Each resource must have at least one of these properties set",
4201             "properties": {
4202               "href": {
4203                 "description": "When present, the ACE only applies when the href matches",

```

```

4204         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/href"
4205     },
4206     "rt": {
4207         "description": "When present, the ACE only applies when the rt (resource type)
4208 matches",
4209         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/rt"
4210     },
4211     "if": {
4212         "description": "When present, the ACE only applies when the if (interface)
4213 matches",
4214         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/if"
4215     },
4216     "wc": {
4217         "type": "string",
4218         "enum": [ "+", "-", "*" ],
4219         "description": "A wildcard matching policy",
4220         "detail-desc": [ "+ - Matches all discoverable resources",
4221                         "- - Matches all non-discoverable resources",
4222                         "* - Matches all resources"
4223     ]
4224     }
4225 }
4226 }
4227 },
4228 "permission": {
4229     "type": "integer",
4230     "description": "Bitmask encoding of CRUDN permission",
4231     "$ref": "oic.sec.crudntype.json#/definitions/oic.sec.crudntype/properties/bitmask"
4232 },
4233 "subject": {
4234     "description": "The subject to whom this ace applies, either a deviceId, role, or
4235 wildcard",
4236     "anyOf": [
4237         {
4238             "$ref": "oic.sec.didtype.json#/definitions/oic.sec.didtype"
4239         },
4240         {
4241             "$ref": "oic.sec.roletype.json#/definitions/oic.sec.roletype"
4242         },
4243         {
4244             "type": "object",
4245             "properties": {
4246                 "conntype": {
4247                     "type": "string",
4248                     "enum": [ "auth-crypt", "anon-clear" ],
4249                     "description": "This property allows an ACE to be matched based on the connection
4250 or message type",
4251                     "detail-desc": [ "auth-crypt - ACE applies if the Client is authenticated and
4252 the data channel or message is encrypted and integrity protected",
4253                                     "anon-clear - ACE applies if the Client is not authenticated
4254 and the data channel or message is not encrypted but may be integrity protected"
4255                     ]
4256                 }
4257             },
4258             "required": [ "conntype" ]
4259         }
4260     ]
4261 },
4262 "validity": {
4263     "type": "array",
4264     "description": "validity is an array of time-pattern objects",
4265     "items": {
4266         "$ref": "oic.sec.time-pattern.json#/definitions/time-pattern"
4267     }
4268 }
4269 },
4270 "required": [ "permission", "subject" ]
4271 }
4272 }
4273 }
4274

```

4275 **A.2.14 oic.sec.roletype.json**

4276 **A.2.15 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
role	string	yes		The ID of the role being identified.
authority	string			The Authority component of the entity being identified. A NULL <Authority> refers to the local entity or device.

4277 **A.2.16 Schema Definition**

```

4278 {
4279   "$schema": "http://json-schema.org/draft-04/schema#",
4280   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.roletype.json#",
4281   "title": "Security Role Types",
4282   "definitions": {
4283     "oic.sec.roletype": {
4284       "type": "object",
4285       "description": "Security role specified as an <Authority> & <Rolename>. A NULL <Authority>
4286 refers to the local entity or device.",
4287       "properties": {
4288         "authority": {
4289           "type": "string",
4290           "description": "The Authority component of the entity being identified. A NULL
4291 <Authority> refers to the local entity or device."
4292         },
4293         "role": {
4294           "type": "string",
4295           "description": "The ID of the role being identified."
4296         }
4297       },
4298       "required": ["role"]
4299     }
4300   }
4301 }
4302

```

4303 **A.2.17 oic.sec.time-pattern.json**

4304 **A.2.18 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
recurrence	array: see schema			String array represents a recurrence rule using the RFC5545 Recurrence
period	string	yes		String represents a period using the RFC5545 Period

4305 **A.2.19 Schema Definition**

```

4306 {
4307   "$schema": "http://json-schema.org/draft-04/schema#",
4308   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.time-pattern.json#",
4309   "title": "RFC5545 time period and recurrence rule(s)",
4310   "definitions": {

```

```

4311     "time-pattern": {
4312       "type": "object",
4313       "description": "The time-pattern contains a period and recurrence expressed in RFC5545
4314 syntax",
4315       "properties": {
4316         "period": {
4317           "type": "string",
4318           "description": "String represents a period using the RFC5545 Period"
4319         },
4320         "recurrence": {
4321           "type": "array",
4322           "description": "String array represents a recurrence rule using the RFC5545 Recurrence",
4323           "items": {
4324             "type": "string"
4325           }
4326         }
4327       },
4328       "required": [ "period" ]
4329     }
4330   }
4331 }
4332

```

4333 A.2.20 oic.sec.crudntype.json

4334 A.2.21 Property Definition

Property name	Value type	Mandatory	Access mode	Description
bitmask	integer	yes		The encoded bitmask indicating permissions

4335 A.2.22 Schema Definition

```

4336 {
4337   "$schema": "http://json-schema.org/draft-04/schema#",
4338   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.crudntype.json#",
4339   "title": "Permission BitMask",
4340   "definitions": {
4341     "oic.sec.crudntype": {
4342       "description": "OIC CRUDN types",
4343       "properties": {
4344         "bitmask": {
4345           "type": "integer",
4346           "minimum": 0,
4347           "maximum": 31,
4348           "description": "The encoded bitmask indicating permissions",
4349           "detail-desc": [ "0 - No permissions",
4350             "1 - Create permission is granted",
4351             "2 - Read, observe, discover permission is granted",
4352             "4 - Write, update permission is granted",
4353             "8 - Delete permission is granted",
4354             "16 - Notify permission is granted" ]
4355         }
4356       }
4357     }
4358   },
4359   "type": "object",
4360   "allOf": [
4361     { "$ref": "#/definitions/oic.sec.crudntype" }
4362   ],
4363   "required": ["bitmask"]
4364 }
4365
4366

```

```

4367 A.3 OICSecurityAmaclResource
4368 A.3.1 Introduction
4369 This resource specifies the host resources with access permission that is managed by an AMS.
4370 A.3.2 Example URI
4371 /oic/sec/amacl
4372 A.3.3 Resource Type
4373 A.3.4 RAML Definition
4374 ##RAML 0.8
4375 title: OICSecurityAmaclResource
4376 version: v1.0-20150819
4377 traits:
4378 - interface :
4379   queryParameters:
4380     if:
4381       enum: ["oic.if.baseline"]
4382
4383 /oic/sec/amacl:
4384   description: |
4385     This resource specifies the host resources with access permission that is managed by an AMS.
4386
4387   is : ['interface']
4388   get:
4389     description: |
4390       Retrieves the amacl data.
4391
4392   responses :
4393     200:
4394       body:
4395         application/json:
4396           schema: /
4397             {
4398               "$schema": "http://json-schema.org/draft-04/schema#",
4399               "id": "https://www.openconnectivity.org/ocf-
4400 apis/security/schemas/oic.r.amacl.json#",
4401               "title": "Managed Access Control information",
4402               "definitions": {
4403                 "oic.r.amacl": {
4404                   "type": "object",
4405                   "properties": {
4406                     "resources": {
4407                       "type": "array",
4408                       "description": "Multiple links to this host's resources",
4409                       "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
4410                     }
4411                   }
4412                 }
4413               },
4414               "type": "object",
4415               "allOf": [
4416                 { "$ref": "#/definitions/oic.r.amacl" }
4417               ],
4418               "required": [ "resources" ]
4419             }
4420
4421           example: /

```



```

4422     {
4423         "resources": [
4424             {
4425                 "href": "/temp",
4426                 "rt": ["oic.r.temperature"],
4427                 "if": ["oic.if.baseline", "oic.if.a"]
4428             },
4429             {
4430                 "href": "/temp",
4431                 "rt": ["oic.r.temperature"],
4432                 "if": ["oic.if.baseline", "oic.if.s"]
4433             }
4434         ]
4435     }
4436
4437 post:
4438     description: |
4439         Sets the new amacl data
4440
4441     body:
4442         application/json:
4443             schema: /
4444                 {
4445                     "$schema": "http://json-schema.org/draft-04/schema#",
4446                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.amacl.json#",
4447                     "title": "Managed Access Control information",
4448                     "definitions": {
4449                         "oic.r.amacl": {
4450                             "type": "object",
4451                             "properties": {
4452                                 "resources": {
4453                                     "type": "array",
4454                                     "description": "Multiple links to this host's resources",
4455                                     "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
4456                                 }
4457                             }
4458                         },
4459                     },
4460                     "type": "object",
4461                     "allOf": [
4462                         { "$ref": "#/definitions/oic.r.amacl" }
4463                     ],
4464                     "required": [ "resources" ]
4465                 }
4466
4467     example: /
4468         {
4469             "resources": [
4470                 {
4471                     "href": "/temp",
4472                     "rt": ["oic.r.temperature"],
4473                     "if": ["oic.if.baseline", "oic.if.a"]
4474                 },
4475                 {
4476                     "href": "/temp",
4477                     "rt": ["oic.r.temperature"],
4478                     "if": ["oic.if.baseline", "oic.if.s"]
4479                 }
4480             ]
4481         }
4482
4483     responses :
4484         400:
4485             description: |

```

```

4486         The request is invalid.
4487
4488     201:
4489         description: |
4490             The AMACL entry is created.
4491
4492     204:
4493         description: |
4494             The AMACL entry is updated.
4495
4496     put:
4497         description: |
4498             Creates the new acl data
4499
4500     body:
4501         application/json:
4502             schema: /
4503                 {
4504                     "$schema": "http://json-schema.org/draft-04/schema#",
4505                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.amacl.json#",
4506                     "title": "Managed Access Control information",
4507                     "definitions": {
4508                         "oic.r.amacl": {
4509                             "type": "object",
4510                             "properties": {
4511                                 "resources": {
4512                                     "type": "array",
4513                                     "description": "Multiple links to this host's resources",
4514                                     "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
4515                                 }
4516                             }
4517                         },
4518                     },
4519                     "type": "object",
4520                     "allOf": [
4521                         { "$ref": "#/definitions/oic.r.amacl" }
4522                     ],
4523                     "required": [ "resources" ]
4524                 }
4525
4526     example: /
4527         {
4528             "resources": [
4529                 {
4530                     "href": "/temp",
4531                     "rt": ["oic.r.temperature"],
4532                     "if": ["oic.if.baseline", "oic.if.a"]
4533                 },
4534                 {
4535                     "href": "/temp",
4536                     "rt": ["oic.r.temperature"],
4537                     "if": ["oic.if.baseline", "oic.if.s"]
4538                 }
4539             ]
4540         }
4541
4542     responses :
4543         400:
4544             description: |
4545                 The request is invalid.
4546
4547         201:

```

```

4548     description: |
4549         The AMACL entry is created.
4550
4551 delete:
4552     description: |
4553         Deletes the amacl data.
4554         When DELETE is used without query parameters, the entire collection is deleted.
4555         When DELETE uses the search parameter with "subject", only the matched entry is deleted.
4556
4557     queryParameters:
4558         subject:
4559             type: string
4560             description: Delete the ACE identified by the string matching the subject value.
4561
4562         required: false
4563         example: DELETE /myamacl?subject="de305d54-75b4-431b-adb2-eb6b9e546014"
4564
4565     responses :
4566         200:
4567             description: |
4568                 The ACE instance or the the entire AMACL resource has been successfully deleted.
4569
4570         400:
4571             description: |
4572                 The request is invalid.
4573
4574

```

A.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
resources	array: see schema	yes		Multiple links to this host's resources

A.3.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/amacl	put	get	post	delete	

A.4 OICSecuritySignedAclResource

A.4.1 Introduction

This resource specifies a signed ACL object.

A.4.2 Example URI

/oic/sec/sacl

A.4.3 Resource Type

A.4.4 RAML Definition

```

4584 #%RAML 0.8
4585 title: OICSecuritySignedAclResource
4586 version: v1.0-20150819
4587 traits:
4588     - interface :
4589         queryParameters:
4590             if:
4591                 enum: ["oic.if.baseline"]

```

```

4592
4593 /oic/sec/sacl:
4594   description: |
4595     This resource specifies a signed ACL object.
4596
4597   is : ['interface']
4598   get:
4599     description: |
4600       Retrieves the sacl resource data.
4601
4602   responses :
4603     200:
4604       body:
4605         application/json:
4606           schema: /
4607             {
4608               "$schema": "http://json-schema.org/draft-04/schema#",
4609               "id": "https://www.openconnectivity.org/ocf-
4610 apis/security/schemas/oic.r.sacl.json#",
4611               "title": "Signed Access Control List information",
4612               "definitions": {
4613                 "oic.r.sacl": {
4614                   "type": "object",
4615                   "properties": {
4616                     "aclist2": {
4617                       "type": "array",
4618                       "description": "Access Control Entries in the Acl resource",
4619                       "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
4620                     },
4621                     "signature": {
4622                       "type": "object",
4623                       "description": "The signature over the ACL resource",
4624                       "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
4625                     }
4626                   }
4627                 },
4628               "type": "object",
4629               "allOf": [
4630                 { "$ref": "#/definitions/oic.r.sacl" }
4631               ],
4632               "required": ["aclist2", "signature"],
4633               "additionalItems": false
4634             }
4635
4636   example: /
4637     {
4638       "aclist2": [
4639         {
4640           "aceid": 1,
4641           "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
4642           "resources": [
4643             {
4644               "href": "/temp",
4645               "rt": ["oic.r.temperature"],
4646               "if": ["oic.if.baseline", "oic.if.a"]
4647             },
4648             {
4649               "href": "/temp",
4650               "rt": ["oic.r.temperature"],
4651               "if": ["oic.if.baseline", "oic.if.s"]
4652             }
4653           ]
4654         },
4655         "permission": 31,

```

```

4656         "validity": [
4657             {
4658                 "period": "20160101T180000Z/20170102T070000Z",
4659                 "recurrence": [ "DSTART:XXXXX",
4660 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4661             },
4662             {
4663                 "period": "20160101T180000Z/PT5H30M",
4664                 "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4665             }
4666         ],
4667     },
4668     {
4669         "aceid": 2,
4670         "subject": {
4671             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4672             "role": "SOME_STRING"
4673         },
4674         "resources": [
4675             {
4676                 "href": "/light",
4677                 "rt": ["oic.r.light"],
4678                 "if": ["oic.if.a"]
4679             },
4680             {
4681                 "href": "/door",
4682                 "rt": ["oic.r.door"],
4683                 "if": ["oic.if.a"]
4684             }
4685         ],
4686         "permission": 15
4687     }
4688 ],
4689 "signature": {
4690     "sigtype": "oic.sec.sigtype.pk7",
4691     "sigvalue": "ENCODED-SIGNATURE-VALUE"
4692 }
4693 }
4694

```

post:

description: |
 Sets the sacl resource data

body:
 application/json:

```

4701     schema: /
4702     {
4703         "$schema": "http://json-schema.org/draft-04/schema#",
4704         "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.sacl.json#",
4705         "title": "Signed Access Control List information",
4706         "definitions": {
4707             "oic.r.sacl": {
4708                 "type": "object",
4709                 "properties": {
4710                     "aclist2": {
4711                         "type": "array",
4712                         "description": "Access Control Entries in the Acl resource",
4713                         "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
4714                     },
4715                     "signature": {
4716                         "type": "object",
4717                         "description": "The signature over the ACL resource",
4718                         "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
4719                     }
4720                 }
4721             }
4722         },
4723         "type": "object",

```

```

4724     "allOf": [
4725       { "$ref": "#/definitions/oic.r.sacl" }
4726     ],
4727     "required": ["aclist2", "signature"],
4728     "additionalItems": false
4729   }
4730
4731   example: /
4732   {
4733     "aclist2": [
4734       {
4735         "aceid": 1,
4736         "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
4737         "resources": [
4738           {
4739             "href": "/temp",
4740             "rt": ["oic.r.temperature"],
4741             "if": ["oic.if.baseline", "oic.if.a"]
4742           },
4743           {
4744             "href": "/temp",
4745             "rt": ["oic.r.temperature"],
4746             "if": ["oic.if.baseline", "oic.if.s"]
4747           }
4748         ],
4749         "permission": 31,
4750         "validity": [
4751           {
4752             "period": "20160101T180000Z/20170102T070000Z",
4753             "recurrence": [ "DSTART:XXXXX",
4754 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4755           },
4756           {
4757             "period": "20160101T180000Z/PT5H30M",
4758             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4759           }
4760         ]
4761       },
4762       {
4763         "aceid": 2,
4764         "subject": {
4765           "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4766           "role": "SOME_STRING"
4767         },
4768         "resources": [
4769           {
4770             "href": "/light",
4771             "rt": ["oic.r.light"],
4772             "if": ["oic.if.baseline", "oic.if.a"]
4773           },
4774           {
4775             "href": "/door",
4776             "rt": ["oic.r.door"],
4777             "if": ["oic.if.baseline", "oic.if.a"]
4778           }
4779         ],
4780         "permission": 15
4781       }
4782     ],
4783     "signature": {
4784       "sigtype": "oic.sec.sigtype.pk7",
4785       "sigvalue": "ENCODED-SIGNATURE-VALUE"
4786     }
4787   }
4788
4789   responses :
4790     400:
4791     description: |

```

```

4792         The request is invalid.
4793
4794     201:
4795         description: |
4796             The ACL entry is created.
4797
4798     204:
4799         description: |
4800             The ACL entry is updated.
4801
4802     put:
4803         description: |
4804             Sets the sacl resource data
4805
4806     body:
4807         application/json:
4808             schema: /
4809                 {
4810                     "$schema": "http://json-schema.org/draft-04/schema#",
4811                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.sacl.json#",
4812                     "title": "Signed Access Control List information",
4813                     "definitions": {
4814                         "oic.r.sacl": {
4815                             "type": "object",
4816                             "properties": {
4817                                 "aclist2": {
4818                                     "type": "array",
4819                                     "description": "Access Control Entries in the Acl resource",
4820                                     "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
4821                                 },
4822                                 "signature": {
4823                                     "type": "object",
4824                                     "description": "The signature over the ACL resource",
4825                                     "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
4826                                 }
4827                             }
4828                         }
4829                     },
4830                     "type": "object",
4831                     "allOf": [
4832                         { "$ref": "#/definitions/oic.r.sacl" }
4833                     ],
4834                     "required": ["aclist2", "signature"],
4835                     "additionalItems": false
4836                 }
4837
4838     example: /
4839         {
4840             "aclist2": [
4841                 {
4842                     "aceid": 1,
4843                     "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
4844                     "resources": [
4845                         {
4846                             "href": "/temp",
4847                             "rt": ["oic.r.temperature"],
4848                             "if": ["oic.if.baseline", "oic.if.a"]
4849                         },
4850                         {
4851                             "href": "/temp",
4852                             "rt": ["oic.r.temperature"],
4853                             "if": ["oic.if.baseline", "oic.if.s"]
4854                         }
4855                     ],
4856                     "permission": 31,

```

```

4857         "validity": [
4858             {
4859                 "period": "20160101T180000Z/20170102T070000Z",
4860                 "recurrence": [ "DSTART:XXXXX",
4861 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4862             },
4863             {
4864                 "period": "20160101T180000Z/PT5H30M",
4865                 "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
4866             }
4867         ]
4868     },
4869     {
4870         "aceid": 2,
4871         "subject": {
4872             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
4873             "role": "SOME_STRING"
4874         },
4875         "resources": [
4876             {
4877                 "href": "/light",
4878                 "rt": ["oic.r.light"],
4879                 "if": ["oic.if.baseline", "oic.if.a"]
4880             },
4881             {
4882                 "href": "/door",
4883                 "rt": ["oic.r.door"],
4884                 "if": ["oic.if.baseline", "oic.if.a"]
4885             }
4886         ],
4887         "permission": 15
4888     }
4889 ],
4890 "signature": {
4891     "sigtype": "oic.sec.sigtype.pk7",
4892     "sigvalue": "ENCODED-SIGNATURE-VALUE"
4893 }
4894 }
4895
4896 responses :
4897     400:
4898         description: |
4899             The request is invalid.
4900
4901     201:
4902         description: |
4903             The signed ACL entry is created.
4904
4905 delete:
4906     description: |
4907         Deletes the signed ACL data.
4908         When DELETE is used without query parameters, the entire collection is deleted.
4909         When DELETE is used with the query parameter where "acl2" is specified, only the matched
4910         entry is deleted.
4911
4912     queryParameters:
4913         subject:
4914             type: string
4915
4916         description: Delete the signed ACL identified by the string containing subject UUID.
4917
4918         required: false
4919         example: DELETE /mysacl?acl2="de305d54-75b4-431b-adb2-eb6b9e546014, ..."
4920
4921 responses :

```


4921 200:
 4922 description: |
 4923 The signed ACL instance or the the entire signed ACL resource has been successfully
 4924 deleted.
 4925
 4926 400:
 4927 description: |
 4928 The request is invalid.
 4929

4930 **A.4.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
aclist2	array: see schema	yes		Access Control Entries in the Acl resource
signature	object: see schema	yes		The signature over the ACL resource

4931 **A.4.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/sec/sacl	put	get	post	delete	

4932 **A.4.7 Referenced JSON schemas**

4933 **A.4.8 oic.sec.sigtype.json**

4934 **A.4.9 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
sigtype	string	yes		The string specifies the predefined signature format
sigvalue	string	yes		The encoded signature

4935 **A.4.10 Schema Definition**

```

4936 {
4937   "$schema": "http://json-schema.org/draft-04/schema#",
4938   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.sigtype.json#",
4939   "title": "Signature format for signed ACL resources",
4940   "definitions": {
4941     "oic.sec.sigtype": {
4942       "description": "Encoded signature data",
4943       "properties": {
4944         "sigtype": {
4945           "type": "string",
4946           "enum": ["oic.sec.sigtype.jws", "oic.sec.sigtype.pk7", "oic.sec.sigtype.cws"],
4947           "description": "The string specifies the predefined signature format",
4948           "detail-desc": [ "RFC7515 JSON web signature (JWS) object",
4949                         "RFC2315 base64 encoded object",
4950                         "CBOR encoded JWS object" ]
4951         },
4952         "sigvalue": {
4953           "type": "string",
4954           "description": "The encoded signature"
4955         }
4956       },
4957       "required": [ "sigtype", "sigvalue" ]
4958     }
4959   }
4960 }
4961
```

```

4962 A.5 OICSecurityDoxmResource
4963 A.5.1 Introduction
4964 This resource specifies properties needed to establish a device owner.
4965 A.5.2 Example URI
4966 /oic/sec/doxm
4967 A.5.3 Resource Type
4968 A.5.4 RAML Definition
4969 ##RAML 0.8
4970 title: OICSecurityDoxmResource
4971 version: v1.0-20150819
4972 traits:
4973   - interface :
4974     queryParameters:
4975       if:
4976         enum: ["oic.if.baseline"]
4977
4978 /oic/sec/doxm:
4979   description: |
4980     This resource specifies properties needed to establish a device owner.
4981
4982   is : ['interface']
4983   get:
4984     description: |
4985       Retrieves the DOXM resource data.
4986
4987   responses :
4988     200:
4989       body:
4990         application/json:
4991           schema: /
4992             {
4993               "$schema": "http://json-schema.org/draft-04/schema#",
4994               "id": "https://www.openconnectivity.org/ocf-
4995 apis/security/schemas/oic.r.doxm.json#",
4996               "title": "Device Owner Transfer Method information",
4997               "definitions": {
4998                 "oic.r.doxm": {
4999                   "type": "object",
5000                   "properties": {
5001                     "oxms": {
5002                       "type": "array",
5003                       "readOnly": true,
5004                       "description": "List of supported owner transfer methods",
5005                       "items": {
5006                         "$ref":
5007 "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5008                       }
5009                     },
5010                     "oxmsel": {
5011                       "description": "The selected owner transfer method used during on-
5012 boarding",
5013                       "$ref":
5014 "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5015                     },
5016                     "sct": {
5017                       "readOnly": true,
5018                       "description": "Bitmask encoding of supported credential types",

```

```

5019         "$ref":
5020 "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
5021     },
5022     "owned": {
5023         "type": "boolean",
5024         "description": "Ownership status flag"
5025     },
5026     "deviceuuid": {
5027         "description": "The uuid formatted identity of the device",
5028         "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5029     },
5030     "devowneruuid": {
5031         "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5032     },
5033     "rowneruuid": {
5034         "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5035     }
5036 }
5037 }
5038 },
5039 "type": "object",
5040 "allOf": [
5041     { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5042     { "$ref": "#/definitions/oic.r.doxm" }
5043 ],
5044 "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid",
5045 "rowneruuid" ]
5046 }
5047
5048     example: /
5049     {
5050         "oxms": [ 0, 2, 3 ],
5051         "oxmsel": 0,
5052         "sct": 16,
5053         "owned": true,
5054         "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
5055         "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5056         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5057     }
5058
5059     400:
5060     description: |
5061         The request is invalid.
5062
5063     post:
5064     description: |
5065         Updates the DOXM resource data
5066
5067     body:
5068     application/json:
5069     schema: /
5070     {
5071         "$schema": "http://json-schema.org/draft-04/schema#",
5072         "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.doxm.json#",
5073         "title": "Device Owner Transfer Method information",
5074         "definitions": {
5075             "oic.r.doxm": {
5076                 "type": "object",
5077                 "properties": {
5078                     "oxms": {
5079                         "type": "array",
5080                         "readOnly": true,
5081                         "description": "List of supported owner transfer methods",
5082                         "items": {
5083                             "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5084                         }

```

```

5085         },
5086         "oxmsel": {
5087             "description": "The selected owner transfer method used during on-boarding",
5088             "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5089         },
5090         "sct": {
5091             "readOnly": true,
5092             "description": "Bitmask encoding of supported credential types",
5093             "$ref":
5094 "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
5095         },
5096         "owned": {
5097             "type": "boolean",
5098             "description": "Ownership status flag"
5099         },
5100         "deviceuuid": {
5101             "description": "The uuid formatted identity of the device",
5102             "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5103         },
5104         "devowneruuid": {
5105             "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5106         },
5107         "rowneruuid": {
5108             "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5109         }
5110     }
5111 },
5112 "type": "object",
5113 "allOf": [
5114     { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5115     { "$ref": "#/definitions/oic.r.doxm" }
5116 ],
5117 "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid",
5118 "rowneruuid" ]
5119 }
5120 }
5121
5122 example: /
5123 {
5124     "oxms": [ 0, 2, 3 ],
5125     "oxmsel": 0,
5126     "sct": 16,
5127     "owned": true,
5128     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
5129     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5130     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5131 }
5132
5133 responses :
5134 400:
5135     description: |
5136     The request is invalid.
5137
5138 201:
5139     description: |
5140     The DOXM entry is created.
5141
5142 204:
5143     description: |
5144     The DOXM entry is updated.
5145
5146 put:
5147     description: |
5148     Creates the DOXM resource data
5149

```

```

5150     body:
5151         application/json:
5152             schema: /
5153                 {
5154                     "$schema": "http://json-schema.org/draft-04/schema#",
5155                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.doxm.json#",
5156                     "title": "Device Owner Transfer Method information",
5157                     "definitions": {
5158                         "oic.r.doxm": {
5159                             "type": "object",
5160                             "properties": {
5161                                 "oxms": {
5162                                     "type": "array",
5163                                     "readOnly": true,
5164                                     "description": "List of supported owner transfer methods",
5165                                     "items": {
5166                                         "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5167                                     }
5168                                 },
5169                                 "oxmsel": {
5170                                     "description": "The selected owner transfer method used during on-boarding",
5171                                     "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
5172                                 },
5173                                 "sct": {
5174                                     "readOnly": true,
5175                                     "description": "Bitmask encoding of supported credential types",
5176                                     "$ref":
5177 "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
5178                                 },
5179                                 "owned": {
5180                                     "type": "boolean",
5181                                     "description": "Ownership status flag"
5182                                 },
5183                                 "deviceuuid": {
5184                                     "description": "The uuid formatted identity of the device",
5185                                     "$ref": "../core/schemas/oic.types-schema.json#/definitions/uuid"
5186                                 },
5187                                 "devowneruuid": {
5188                                     "$ref": "../core/schemas/oic.types-schema.json#/definitions/uuid"
5189                                 },
5190                                 "rowneruuid": {
5191                                     "$ref": "../core/schemas/oic.types-schema.json#/definitions/uuid"
5192                                 }
5193                             }
5194                         },
5195                     "type": "object",
5196                     "allOf": [
5197                         { "$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5198                         { "$ref": "#/definitions/oic.r.doxm" }
5199                     ],
5200                     "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid",
5201 "rowneruuid" ]
5202                 }
5203             }
5204
5205         example: /
5206             {
5207                 "oxms": [ 0, 2, 3 ],
5208                 "oxmsel": 0,
5209                 "sct": 16,
5210                 "owned": true,
5211                 "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
5212                 "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5213                 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5214             }
5215
5216     responses :
5217         400:

```

```

5218     description: |
5219         The request is invalid.
5220
5221     201:
5222     description: |
5223         The DOXM entry is created.
5224
5225     delete:
5226     description: |
5227         Deletes the DOXM data.
5228         When DELETE is used without query parameters, the entire DOXM resource is deleted.
5229         When DELETE uses a query value naming the deviceid, only the matched entry is deleted.
5230
5231     queryParameters:
5232     subject:
5233         type: string
5234
5235         description: Delete the DOXM entry identified by the string containing device ID.
5236
5237         required: false
5238         example: DELETE /mydoxm?deviceuuid="de305d54-75b4-431b-adb2-eb6b9e546014"
5239
5240     responses :
5241     400:
5242     description: |
5243         The request is invalid.
5244
5245     204:
5246     description: |
5247         The CRL instance or the the entire CRL resource has been successfully deleted.

```

5248 A.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		
oxms	array: see schema	yes	Read Only	List of supported owner transfer methods
devowneruuid	multiple types: see schema	yes		
deviceuuid	multiple types: see schema	yes		The uuid formatted identity of the device
owned	boolean	yes		Ownership status flag
oxmsel	multiple types: see schema	yes		The selected owner transfer method used during on- boarding
sct	multiple types: see schema	yes	Read Only	Bitmask encoding of supported credential types

5249 **A.5.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/sec/doxm	put	get	post	delete	

5250 **A.5.7 Referenced JSON schemas**

5251 **A.5.8 oic.sec.doxmtype.json**

5252 **A.5.9 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
oxm	integer	yes		Each value indicates a specific Owner Transfer method

5253 **A.5.10 Schema Definition**

```

5254 {
5255   "$schema": "http://json-schema.org/draft-04/schema#",
5256   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.doxmtype.json#",
5257   "title": "Device Owner Transfer Method Types",
5258   "definitions": {
5259     "oic.sec.doxmtype": {
5260       "type": "object",
5261       "description": "The device owner transfer methods that may be selected at device on-
5262 boarding",
5263       "properties": {
5264         "oxm": {
5265           "type": "integer",
5266           "description": "Each value indicates a specific Owner Transfer method",
5267           "detail-desc": [ "0 - Numeric OTM identifier for the Just-Works method
5268 (oic.sec.doxm.jw)",
5269 "1 - Numeric OTM identifier for the random PIN method
5270 (oic.sec.doxm.rdp)",
5271 "2 - Numeric OTM identifier for the manufacturer certificate method
5272 (oic.sec.doxm.mfgcert)",
5273 "3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
5274 (deprecated) ]
5275         }
5276       }
5277     },
5278   },
5279   "type": "object",
5280   "allOf": [
5281     { "$ref": "#/definitions/oic.sec.doxmtype" }
5282   ],
5283   "required": [ "oxm" ]
5284 }
5285

```

5286 **A.5.11 oic.sec.credtype.json**

5287 **A.5.12 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
bitmask	integer	yes		Cred type encoded as a bitmask

5288 **A.5.13 Schema Definition**

```

5289 {
5290   "$schema": "http://json-schema.org/draft-04/schema#",
5291   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.credtype.json#",
5292   "title": "Credential Types",
5293   "definitions": {
5294     "oic.sec.credtype": {
5295       "type": "object",
5296       "description": "OIC credential types",

```

```

5297     "properties": {
5298       "bitmask": {
5299         "type": "integer",
5300         "minimum": 0,
5301         "maximum": 63,
5302         "description": "Cred type encoded as a bitmask ",
5303         "detail-desc": [ "0 - Empty credential used for testing",
5304                       "1 - Symmetric pair-wise key",
5305                       "2 - Symmetric group key",
5306                       "4 - Asymmetric signing key",
5307                       "8 - Asymmetric signing key with certificate",
5308                       "16 - PIN or password",
5309                       "32 - Asymmetric encryption key" ]
5310       }
5311     }
5312   },
5313 },
5314 "type": "object",
5315 "allOf": [
5316   { "$ref": "#/definitions/oic.sec.credtype" }
5317 ],
5318 "required": ["bitmask"]
5319 }
5320

```

5321 **A.6 OICSecurityPstatResource**

5322 **A.6.1 Introduction**

5323 This resource specifies device provisioning status.

5324 **A.6.2 Example URI**

5325 /oic/sec/pstat

5326 **A.6.3 Resource Type**

5327 **A.6.4 RAML Definition**

```

5328 ##RAML 0.8
5329 title: OICSecurityPstatResource
5330 version: v1.0-20150819
5331 traits:
5332   - interface :
5333     queryParameters:
5334       if:
5335         enum: ["oic.if.baseline"]
5336
5337 /oic/sec/pstat:
5338   description: |
5339     This resource specifies device provisioning status.
5340
5341   is : ['interface']
5342   get:
5343     description: |
5344       Retrieves device provisioning status data.
5345
5346   responses :
5347     200:
5348       body:
5349         application/json:
5350           schema: /

```



```

5351     {
5352         "$schema": "http://json-schema.org/draft-04/schema#",
5353         "id": "https://www.openconnectivity.org/ocf-
5354 apis/security/schemas/oic.r.pstat.json#",
5355         "title": "Device Provisioning Status information",
5356         "definitions": {
5357             "oic.r.pstat": {
5358                 "type": "object",
5359                 "properties": {
5360                     "dos": {
5361                         "type": "object",
5362                         "description": "Device on-boarding state",
5363                         "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
5364                     },
5365                     "isop": {
5366                         "type": "boolean",
5367                         "description": "true indicates device is operational"
5368                     },
5369                     "cm": {
5370                         "description": "Current device provisioning mode",
5371                         "$ref":
5372 "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
5373                     },
5374                     "tm": {
5375                         "description": "Target device provisioning mode",
5376                         "$ref":
5377 "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
5378                     },
5379                     "om": {
5380                         "description": "Current operational mode",
5381                         "$ref":
5382 "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
5383                     },
5384                     "sm": {
5385                         "readOnly": true,
5386                         "description": "Supported operational modes",
5387                         "$ref":
5388 "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
5389                     },
5390                     "rowneruuid": {
5391                         "description": "The UUID formatted identity of the Resource owner",
5392                         "$ref": "../..core/schemas/oic.types-schema.json#/definitions/uuid"
5393                     }
5394                 }
5395             }
5396         },
5397         "type": "object",
5398         "allOf": [
5399             { "$ref": "../..core/schemas/oic.core-schema.json#/definitions/oic.core" },
5400             { "$ref": "#/definitions/oic.r.pstat" }
5401         ],
5402         "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
5403     }
5404
5405     example: /
5406     {
5407         "dos": {"s": 3, "p": true},
5408         "isop": true,
5409         "cm": 8,
5410         "tm": 60,
5411         "om": 2,
5412         "sm": 7,
5413         "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5414     }
5415
5416     400:
5417     description: |
5418     The request is invalid.
5419

```

```

5420 post:
5421   description: |
5422     Sets or updates device provisioning status data.
5423
5424 body:
5425   application/json:
5426     schema: /
5427       {
5428         "$schema": "http://json-schema.org/draft-04/schema#",
5429         "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.pstat.json#",
5430         "title": "Device Provisioning Status information",
5431         "definitions": {
5432           "oic.r.pstat": {
5433             "type": "object",
5434             "properties": {
5435               "dos": {
5436                 "type": "object",
5437                 "description": "Device on-boarding state",
5438                 "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
5439               },
5440               "isop": {
5441                 "type": "boolean",
5442                 "description": "true indicates device is operational"
5443               },
5444               "cm": {
5445                 "description": "Current device provisioning mode",
5446                 "$ref": "oic.sec.dpmpmtype.json#/definitions/oic.sec.dpmpmtype/properties/bitmask"
5447               },
5448               "tm": {
5449                 "description": "Target device provisioning mode",
5450                 "$ref": "oic.sec.dpmpmtype.json#/definitions/oic.sec.dpmpmtype/properties/bitmask"
5451               },
5452               "om": {
5453                 "description": "Current operational mode",
5454                 "$ref": "oic.sec.pompmtype.json#/definitions/oic.sec.pompmtype/properties/bitmask"
5455               },
5456               "sm": {
5457                 "readOnly": true,
5458                 "description": "Supported operational modes",
5459                 "$ref": "oic.sec.pompmtype.json#/definitions/oic.sec.pompmtype/properties/bitmask"
5460               },
5461               "rowneruuid": {
5462                 "description": "The UUID formatted identity of the Resource owner",
5463                 "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5464               }
5465             }
5466           }
5467         },
5468         "type": "object",
5469         "allOf": [
5470           { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5471           { "$ref": "#/definitions/oic.r.pstat" }
5472         ],
5473         "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
5474       }
5475
5476   example: /
5477     {
5478       "dos": {"s": 3, "p": true},
5479       "isop": true,
5480       "cm": 8,
5481       "tm": 60,
5482       "om": 2,
5483       "sm": 7,
5484       "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5485     }
5486

```

```

5487     responses :
5488         400:
5489             description: |
5490                 The request is invalid.
5491
5492         201:
5493             description: |
5494                 The PSTAT entry is created.
5495
5496         204:
5497             description: |
5498                 The PSTAT entry is updated.
5499
5500     put:
5501         description: |
5502             Sets or updates device provisioning status data.
5503
5504     body:
5505         application/json:
5506             schema: /
5507                 {
5508                     "$schema": "http://json-schema.org/draft-04/schema#",
5509                     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.pstat.json#",
5510                     "title": "Device Provisioning Status information",
5511                     "definitions": {
5512                         "oic.r.pstat": {
5513                             "type": "object",
5514                             "properties": {
5515                                 "dos": {
5516                                     "type": "object",
5517                                     "description": "Device on-boarding state",
5518                                     "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
5519                                 },
5520                                 "isop": {
5521                                     "type": "boolean",
5522                                     "description": "true indicates device is operational"
5523                                 },
5524                                 "cm": {
5525                                     "description": "Current device provisioning mode",
5526                                     "$ref": "oic.sec.dpmltype.json#/definitions/oic.sec.dpmltype/properties/bitmask"
5527                                 },
5528                                 "tm": {
5529                                     "description": "Target device provisioning mode",
5530                                     "$ref": "oic.sec.dpmltype.json#/definitions/oic.sec.dpmltype/properties/bitmask"
5531                                 },
5532                                 "om": {
5533                                     "description": "Current operational mode",
5534                                     "$ref": "oic.sec.pomltype.json#/definitions/oic.sec.pomltype/properties/bitmask"
5535                                 },
5536                                 "sm": {
5537                                     "readOnly": true,
5538                                     "description": "Supported operational modes",
5539                                     "$ref": "oic.sec.pomltype.json#/definitions/oic.sec.pomltype/properties/bitmask"
5540                                 },
5541                                 "rowneruuid": {
5542                                     "description": "The UUID formatted identity of the Resource owner",
5543                                     "$ref": "../..../core/schemas/oic.types-schema.json#/definitions/uuid"
5544                                 }
5545                             }
5546                         }
5547                     },
5548                     "type": "object",
5549                     "allOf": [
5550                         { "$ref": "../..../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5551                         { "$ref": "#/definitions/oic.r.pstat" }
5552                     ]
5553                 }

```

```

5552     ],
5553     "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
5554   }
5555
5556   example: /
5557     {
5558       "dos": {"s": 3, "p": true},
5559       "isop": true,
5560       "cm": 8,
5561       "tm": 60,
5562       "om": 2,
5563       "sm": 7,
5564       "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5565     }
5566
5567   responses :
5568     400:
5569       description: |
5570         The request is invalid.
5571
5572     201:
5573       description: |
5574         The PSTAT entry is created.
5575
5576   delete:
5577     description: |
5578       Deletes the PSTAT data.
5579       When DELETE is used without query parameters, the entire collection is deleted.
5580       When DELETE is used with the query parameter where rowneruuid is specified, only the matched
5581   entry is deleted.
5582
5583   queryParameters:
5584     subject:
5585       type: string
5586       description: Delete the PSTAT identified by the string containing the UUID.
5587
5588     required: false
5589     example: DELETE /mypstat?rowneruuid="de305d54-75b4-431b-adb2-eb6b9e546014"
5590
5591   responses :
5592     200:
5593       description: |
5594         The PSTAT instance or the the entire resource has been successfully deleted.
5595
5596     400:
5597       description: |
5598         The request is invalid.
5599
5600

```

A.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		The UUID formatted identity of the Resource owner
om	multiple types: see schema	yes		Current operational mode

cm	multiple types: see schema	yes		Current device provisioning mode
isop	boolean	yes		true indicates device is operational
tm	multiple types: see schema			Target device provisioning mode
sm	multiple types: see schema	yes	Read Only	Supported operational modes
dos	object: see schema	yes		Device on-boarding state

5601 **A.6.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/sec/pstat	put	get	post	delete	

5602 **A.6.7 Referenced JSON schemas**

5603 **A.6.8 oic.sec.dostype.json**

5604 **A.6.9 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
p	boolean	yes		'p' is TRUE when the 's' state is pending until all necessary changes to device resources are complete.
s	integer	yes		The current or pending operational state

5605 **A.6.10 Schema Definition**

```

5606 {
5607   "$schema": "http://json-schema.org/draft-04/schema#",
5608   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.dostype.json#",
5609   "title": "Device On-boarding States",
5610   "definitions": {
5611     "oic.sec.dostype": {
5612       "description": "Device operation state machine",
5613       "properties": {
5614         "s": {
5615           "type": "integer",
5616           "minimum": 0,
5617           "maximum": 4,
5618           "description": "The current or pending operational state",
5619           "detail-desc": [ "0 - RESET - Device reset state",
5620                         "1 - RFOTM - Ready for device owner transfer method state",
5621                         "2 - RFPRO - Ready for device provisioning state",
5622                         "3 - RFNOP - Ready for device normal operation state",
5623                         "4 - SRESET - The device is in a soft reset state" ]
5624         },
5625         "p": {
5626           "type": "boolean",
5627           "default": true,
5628           "description": "'p' is TRUE when the 's' state is pending until all necessary changes to
5629 device resources are complete."
5630       }
5631     }
5632   }

```

```

5631     },
5632     "required": [ "s", "p" ]
5633   }
5634 }
5635 }
5636

```

5637 **A.6.11 oic.sec.dpmttype.json**

5638 **A.6.12 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
bitmask	integer	yes		The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value

5639 **A.6.13 Schema Definition**

```

5640 {
5641   "$schema": "http://json-schema.org/draft-04/schema#",
5642   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.dpmttype.json#",
5643   "title": "Device Provisioning Mode Types",
5644   "definitions": {
5645     "oic.sec.dpmttype": {
5646       "description": "Device provisioning mode maintains a bitmask of the possible provisioning
5647 states of a device",
5648       "properties": {
5649         "bitmask": {
5650           "type": "integer",
5651           "minimum": 0,
5652           "maximum": 255,
5653           "description": "The value can be either 8 or 16 character in length. If its only 8
5654 characters it represents the lower byte value",
5655           "detail-desc": [ "1 - Manufacturer reset state",
5656                         "2 - Device pairing and owner transfer state",
5657                         "4 - Provisioning of bootstrap services",
5658                         "8 - Provisioning of credential management services",
5659                         "16 - Provisioning of access management services",
5660                         "32 - Provisioning of local ACLs",
5661                         "64 - Initiate Software Version Validation",
5662                         "128 - Initiate Secure Software Update" ]
5663         }
5664       }
5665     }
5666   },
5667   "type": "object",
5668   "$ref": "#/definitions/oic.sec.dpmttype",
5669   "required": [ "bitmask" ]
5670 }
5671

```

5672 **A.6.14 oic.sec.pomttype.json**

5673 **A.6.15 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
bitmask	integer	yes		The value is a bitmask encoded as integer and indicates the provisioning operation modes

5674 **A.6.16 Schema Definition**

```
5675 {  
5676   "$schema": "http://json-schema.org/draft-04/schema#",  
5677   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.pomtype.json#",  
5678   "title": "Provisioning Operational Modes",  
5679   "definitions": {  
5680     "oic.sec.pomtype": {  
5681       "description": "Device provisioning operation may be server directed or client (aka  
5682 provisioning service) directed.",  
5683       "properties": {  
5684         "bitmask": {  
5685           "type": "integer",  
5686           "minimum": 1,  
5687           "maximum": 7,  
5688           "description": "The value is a bitmask encoded as integer and indicates the provisioning  
5689 operation modes",  
5690           "detail-desc": [ "1 - Server-directed utilizing multiple provisioning services",  
5691                           "2 - Server-directed utilizing a single provisioning service",  
5692                           "4 - Client-directed provisioning",  
5693                           "8 - Unused",  
5694                           "16 - Unused",  
5695                           "32 - Unused",  
5696                           "64 - Unused",  
5697                           "128 - Unused" ]  
5698         }  
5699       }  
5700     }  
5701   },  
5702   "type": "object",  
5703   "allOf": [  
5704     { "$ref": "#/definitions/oic.sec.pomtype" }  
5705   ],  
5706   "required": ["bitmask"]  
5707 }  
5708
```

5709 **A.6.17**

5710 **A.7 OICSecurityCredentialResource**

5711 **A.7.1 Introduction**

5712 This resource specifies credentials a device may use to establish secure communication.

5713 **A.7.2 Example URI**

5714 /oic/sec/cred

5715 **A.7.3 Resource Type**

5716 **A.7.4 RAML Definition**

```
5717 #%RAML 0.8  
5718 title: OICSecurityCredentialResource  
5719 version: v1.0-20150819  
5720 traits:  
5721   - interface :  
5722     queryParameters:  
5723       if:  
5724         enum: ["oic.if.baseline"]  
5725   - cred-filtered :  
5726     queryParameters:  
5727       credid:  
5728  
5729 /oic/sec/cred:  
5730   description: |
```

5731 This resource specifies credentials a device may use to establish secure communication.
5732

5733 is : ['interface']

5734 get:

5735 description: |

5736 Retrieves the credential data.

5737 When used without query parameters, all the credential entries are returned.

5738 When used with a query parameter, only the credentials matching the specified
5739 parameter are returned.

5740 Note that write-only credential data will not be returned.
5741

5742 is : ['cred-filtered']

5743 responses :

5744 200:

5745 body:

5746 application/json:

5747 schema: /

5748 {

5749 "\$schema": "http://json-schema.org/draft-04/schema#",

5750 "id": "https://www.openconnectivity.org/ocf-

5751 apis/security/schemas/oic.r.cred.json#",

5752 "title": "Device Credentials information",

5753 "definitions": {

5754 "oic.r.cred": {

5755 "type": "object",

5756 "properties": {

5757 "creds": {

5758 "type": "array",

5759 "description": "List of credentials available at this resource",

5760 "items": {

5761 "\$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"

5762 }
5763 },

5764 "rowneruuid": {

5765 "\$ref": "../core/schemas/oic.types-schema.json#/definitions/uuid"

5766 }
5767 }
5768 }

5769 },

5770 "type": "object",

5771 "allOf": [

5772 { "\$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },

5773 { "\$ref": "#/definitions/oic.r.cred" }
5774],

5775 "required": ["creds", "rowneruuid"]
5776 }
5777 }

5778 example: /

5779 {

5780 "creds": [

5781 {

5782 "credid": 55,

5783 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",

5784 "roleid": {

5785 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",

5786 "role": "SOME_STRING"

5787 }
5788 "credtype": 32,

5789 "publicdata": {

5790 "encoding": "oic.sec.encoding.base64",

5791 "data": "BASE-64-ENCODED-VALUE"

5792 }
5793 "privatedata": {

5794 "encoding": "oic.sec.encoding.base64",

5795 "data": "BASE-64-ENCODED-VALUE",
5796 }
5797 }


```

5796         "handle": 4
5797     },
5798     "optionaldata": {
5799         "revstat": false,
5800         "encoding": "oic.sec.encoding.base64",
5801         "data": "BASE-64-ENCODED-VALUE"
5802     },
5803     "period": "20160101T180000Z/20170102T070000Z",
5804     "crms": [ "oic.sec.crm.pk10" ]
5805 },
5806 {
5807     "credid": 56,
5808     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5809     "roleid": {
5810         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5811         "role": "SOME_STRING"
5812     },
5813     "credtype": 1,
5814     "publicdata": {
5815         "encoding": "oic.sec.encoding.base64",
5816         "data": "BASE-64-ENCODED-VALUE"
5817     },
5818     "privatedata": {
5819         "encoding": "oic.sec.encoding.base64",
5820         "data": "BASE-64-ENCODED-VALUE",
5821         "handle": 4
5822     },
5823     "optionaldata": {
5824         "revstat": false,
5825         "encoding": "oic.sec.encoding.base64",
5826         "data": "BASE-64-ENCODED-VALUE"
5827     },
5828     "period": "20160101T180000Z/20170102T070000Z",
5829     "crms": [ "oic.sec.crm.pk10" ]
5830 }
5831 ],
5832 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5833 }
5834

```

5835 400:

```

5836     description: |
5837         The request is invalid.
5838

```

5839 post:

```

5840     description: |
5841         Updates the credential resource with the provided credentials.
5842         Credentials provided in the update with credid(s) not currently in the
5843         credential resource are added.
5844         Credentials provided in the update with credid(s) already in the
5845         credential resource completely replace the creds in the credential
5846         resource.
5847         Credentials provided in the update without credid(s) properties are
5848         added and assigned unique credid(s) in the credential resource.
5849

```

5850 body:

```

5851     application/json:
5852         schema: /
5853         {
5854             "$schema": "http://json-schema.org/draft-04/schema#",
5855             "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.cred.json#",
5856             "title": "Device Credentials information",
5857             "definitions": {
5858                 "oic.r.cred": {
5859                     "type": "object",
5860                     "properties": {
5861                         "creds": {
5862                             "type": "array",

```

```

5863         "description": "List of credentials available at this resource",
5864         "items": {
5865             "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
5866         }
5867     },
5868     "rowneruuid": {
5869         "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
5870     }
5871 }
5872 }
5873 },
5874 "type": "object",
5875 "allOf": [
5876     { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
5877     { "$ref": "#/definitions/oic.r.cred" }
5878 ],
5879 "required": [ "creds", "rowneruuid" ]
5880 }
5881

```

example: /

```

5882
5883 {
5884     "creds": [
5885         {
5886             "credid": 55,
5887             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5888             "roleid": {
5889                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5890                 "role": "SOME_STRING"
5891             },
5892             "credtype": 32,
5893             "publicdata": {
5894                 "encoding": "oic.sec.encoding.base64",
5895                 "data": "BASE-64-ENCODED-VALUE"
5896             },
5897             "privatedata": {
5898                 "encoding": "oic.sec.encoding.base64",
5899                 "data": "BASE-64-ENCODED-VALUE",
5900                 "handle": 4
5901             },
5902             "optionaldata": {
5903                 "revstat": false,
5904                 "encoding": "oic.sec.encoding.base64",
5905                 "data": "BASE-64-ENCODED-VALUE"
5906             },
5907             "period": "20160101T180000Z/20170102T070000Z",
5908             "crms": [ "oic.sec.crm.pk10" ]
5909         },
5910         {
5911             "credid": 56,
5912             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5913             "roleid": {
5914                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5915                 "role": "SOME_STRING"
5916             }
5917         },
5918         "credtype": 1,
5919         "publicdata": {
5920             "encoding": "oic.sec.encoding.base64",
5921             "data": "BASE-64-ENCODED-VALUE"
5922         },
5923         "privatedata": {
5924             "encoding": "oic.sec.encoding.base64",
5925             "data": "BASE-64-ENCODED-VALUE",
5926             "handle": 4
5927         },
5928         "optionaldata": {
5929             "revstat": false,
5930             "encoding": "oic.sec.encoding.base64",
5931             "data": "BASE-64-ENCODED-VALUE"
5932         },

```

```

5933         "period": "20160101T180000Z/20170102T070000Z",
5934         "crms": [ "oic.sec.crm.pk10" ]
5935     }
5936 },
5937     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5938 }
5939
5940 responses :
5941     400:
5942         description: |
5943             The request is invalid.
5944
5945     201:
5946         description: |
5947             The credential entry is created.
5948
5949     204:
5950         description: |
5951             The credential entry is updated.
5952
5953 delete:
5954     description: |
5955         Deletes credential entries.
5956         When DELETE is used without query parameters, all the cred entries are deleted.
5957         When DELETE is used with a query parameter, only the entries matching
5958         the query parameter are deleted.
5959
5960     is : ['cred-filtered']
5961     responses :
5962         400:
5963             description: |
5964                 The request is invalid.
5965
5966         204:
5967             description: |
5968                 The specific credential(s) or the the entire credential resource has been successfully
5969         deleted.
5970

```

5971 A.7.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		
creds	array: see schema	yes		List of credentials available at this resource

5972 A.7.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/cred		get	post	delete	

5973 A.7.7 Referenced JSON schemas

5974 A.7.8 oic.sec.roletype.json

5975 A.7.9 Property Definition

Property name	Value type	Mandatory	Access mode	Description
---------------	------------	-----------	-------------	-------------

role	string	yes		The ID of the role being identified.
authority	string			The Authority component of the entity being identified. A NULL <Authority> refers to the local entity or device.

5976 **A.7.10 Schema Definition**

```

5977 {
5978   "$schema": "http://json-schema.org/draft-04/schema#",
5979   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.roletype.json#",
5980   "title": "Security Role Types",
5981   "definitions": {
5982     "oic.sec.roletype": {
5983       "type": "object",
5984       "description": "Security role specified as an <Authority> & <Rolename>. A NULL <Authority>
5985 refers to the local entity or device.",
5986       "properties": {
5987         "authority": {
5988           "type": "string",
5989           "description": "The Authority component of the entity being identified. A NULL
5990 <Authority> refers to the local entity or device."
5991         },
5992         "role": {
5993           "type": "string",
5994           "description": "The ID of the role being identified."
5995         }
5996       },
5997       "required": ["role"]
5998     }
5999   }
6000 }
6001

```

6002 **A.7.11 oic.sec.credtype.json**

6003 **A.7.12 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
bitmask	integer	yes		Cred type encoded as a bitmask

6004 **A.7.13 Schema Definition**

```

6005 {
6006   "$schema": "http://json-schema.org/draft-04/schema#",
6007   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.credtype.json#",
6008   "title": "Credential Types",
6009   "definitions": {
6010     "oic.sec.credtype": {
6011       "type": "object",
6012       "description": "OIC credential types",
6013       "properties": {
6014         "bitmask": {
6015           "type": "integer",
6016           "minimum": 0,
6017           "maximum": 63,
6018           "description": "Cred type encoded as a bitmask ",
6019           "detail-desc": [
6020             "0 - Empty credential used for testing",
6021             "1 - Symmetric pair-wise key",
6022             "2 - Symmetric group key",
6023             "4 - Asymmetric signing key",
6024             "8 - Asymmetric signing key with certificate",

```

```

6024         "16 - PIN or password",
6025         "32 - Asymmetric encryption key" ]
6026     }
6027 }
6028 },
6029 },
6030 "type": "object",
6031 "allOf": [
6032   { "$ref": "#/definitions/oic.sec.credtype" }
6033 ],
6034 "required": ["bitmask"]
6035 }
6036

```

6037 **A.7.14 oic.sec.pubdatatype.json**

6038 **A.7.15 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
data	string			The encoded value
encoding	string			A string specifying the encoding format of the data contained in the pubdata

6039 **A.7.16 Schema Definition**

```

6040 {
6041   "$schema": "http://json-schema.org/draft-04/schema#",
6042   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.pubdatatype.json#",
6043   "title": "Public Credential data",
6044   "definitions": {
6045     "oic.sec.pubdatatype": {
6046       "type": "object",
6047       "properties": {
6048         "encoding": {
6049           "type": "string",
6050           "enum": [ "oic.sec.encoding.jwt", "oic.sec.encoding.cwt", "oic.sec.encoding.base64",
6051 "oic.sec.encoding.uri", "oic.sec.encoding.pem", "oic.sec.encoding.der", "oic.sec.encoding.raw" ],
6052           "description": "A string specifying the encoding format of the data contained in the
6053 pubdata",
6054           "detail-desc": [ "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding",
6055 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding",
6056 "oic.sec.encoding.base64 - Base64 encoded object",
6057 "oic.sec.encoding.uri - URI reference",
6058 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
6059 "oic.sec.encoding.der - Encoding for DER encoded certificate",
6060 "oic.sec.encoding.raw - Raw hex encoded data" ]
6061         },
6062         "data": {
6063           "type": "string",
6064           "description": "The encoded value"
6065         }
6066       }
6067     }
6068   }
6069 }
6070

```

6071 **A.7.17 oic.sec.privdatatype.json**

6072 **A.7.18 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
handle	integer			Handle to a key storage resource

data	string			The encoded value
encoding	string	yes		A string specifying the encoding format of the data contained in the privdata

6073 **A.7.19 Schema Definition**

```

6074 {
6075   "$schema": "http://json-schema.org/draft-04/schema#",
6076   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.privdatatype.json#",
6077   "title": "Private Credential data",
6078   "definitions": {
6079     "oic.sec.privdatatype": {
6080       "type": "object",
6081       "description": "Credential resource non-public contents",
6082       "properties": {
6083         "encoding": {
6084           "type": "string",
6085           "enum": [ "oic.sec.encoding.jwt", "oic.sec.encoding.cwt", "oic.sec.encoding.base64",
6086 "oic.sec.encoding.uri", "oic.sec.encoding.handle", "oic.sec.encoding.raw" ],
6087           "description": "A string specifying the encoding format of the data contained in the
6088 privdata",
6089           "detail-desc": [ "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding",
6090 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding",
6091 "oic.sec.encoding.base64 - Base64 encoded object",
6092 "oic.sec.encoding.uri - URI reference",
6093 "oic.sec.encoding.handle - Data is contained in a storage sub-system
6094 referenced using a handle",
6095 "oic.sec.encoding.raw - Raw hex encoded data" ]
6096         },
6097         "data": {
6098           "type": "string",
6099           "description": "The encoded value"
6100         },
6101         "handle": {
6102           "type": "integer",
6103           "description": "Handle to a key storage resource"
6104         }
6105       },
6106       "required": [ "encoding" ]
6107     }
6108   }
6109 }
6110

```

6111 **A.7.20 oic.sec.optdatatype.json**

6112 **A.7.21 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
revstat	boolean	yes		Revocation status flag - true = revoked
data	string			The encoded structure
encoding	string			A string specifying the encoding format of the data contained in the optdata

6113 **A.7.22 Schema Definition**

```

6114 {
6115   "$schema": "http://json-schema.org/draft-04/schema#",
6116   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.optdatatype.json#",
6117   "title": "Optional Credential data",
6118   "definitions": {
6119     "oic.sec.optdatatype": {
6120       "description": "Optional credential contents describes revocation status for this
6121 credential",
6122       "properties": {
6123         "revstat": {
6124           "type": "boolean",
6125           "description": "Revocation status flag - true = revoked"
6126         },
6127         "encoding": {
6128           "type": "string",
6129           "enum": [ "oic.sec.encoding.jwt", "oic.sec.encoding.cwt", "oic.sec.encoding.base64",
6130 "oic.sec.encoding.pem",
6131 "oic.sec.encoding.der", "oic.sec.encoding.raw" ],
6132           "description": "A string specifying the encoding format of the data contained in the
6133 optdata",
6134           "detail-desc": [ "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding",
6135 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding",
6136 "oic.sec.encoding.base64 - Base64 encoded object",
6137 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
6138 "oic.sec.encoding.der - Encoding for DER encoded certificate",
6139 "oic.sec.encoding.raw - Raw hex encoded data" ]
6140         },
6141         "data": {
6142           "type": "string",
6143           "description": "The encoded structure"
6144         }
6145       },
6146       "required": [ "revstat" ]
6147     }
6148   }
6149 }
6150

```

6151 **A.7.23 oic.sec.crmtypes.json**

6152 **A.7.24 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
crm	string	yes		Each enum represents a method by which the credentials are refreshed.

6153 **A.7.25 Schema Definition**

```

6154 {
6155   "$schema": "http://json-schema.org/draft-04/schema#",
6156   "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.crmtypes.json#",
6157   "title": "Credential Refresh Methods",
6158   "definitions": {
6159     "oic.sec.crmtypes": {
6160       "properties": {
6161         "crm": {
6162           "type": "string",
6163           "enum": [ "oic.sec.crm.pro", "oic.sec.crm.psk", "oic.sec.crm.rdp", "oic.sec.crm.skdc",
6164 "oic.sec.crm.pk10" ],
6165           "description": "Each enum represents a method by which the credentials are refreshed.",
6166           "detail-desc": [ "oic.sec.crm.pro - Credentials refreshed by a provisioning service",
6167 "oic.sec.crm.rdp - Credentials refreshed by a key agreement protocol
6168 and random PIN",
6169 "oic.sec.crm.psk - Credentials refreshed by a key agreement protocol",
6170 "oic.sec.crm.skdc - Credentials refreshed by a key distribution
6171 service",

```

```

6172         "oic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA"
6173     ]
6174     }
6175 }
6176 },
6177 },
6178 "type": "object",
6179 "allOf": [
6180   { "$ref": "#/definitions/oic.sec.crmtypes" }
6181 ],
6182 "required": ["crm"]
6183 }
6184

```

6185

6186 **A.8 OICSecurityCsrResource**

6187 **A.8.1 Introduction**

6188 This resource specifies a Certificate Signing Request.

6189 **A.8.2 Example URI**

6190 /oic/sec/csr

6191 **A.8.3 Resource Type**

6192 **A.8.4 RAML Definition**

6193 *##RAML 0.8*

6194 *title: OICSecurityCsrResource*

6195 *version: v1.0-20150819*

6196 *traits:*

6197 *- interface :*

6198 *queryParameters:*

6199 *if:*

6200 *enum: ["oic.if.baseline"]*

6201

6202 */oic/sec/csr:*

6203 *description: |*

6204 *This resource specifies a Certificate Signing Request.*

6205

6206 *is : ['interface']*

6207 *get:*

6208 *description: |*

6209 *Retrieves CSR data.*

6210

6211 *responses :*

6212 *200:*

6213 *body:*

6214 *application/json:*

6215 *schema: |*

```

6216             {
6217               "$schema": "http://json-schema.org/draft-04/schema#",
6218               "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.csr.json#",
6219               "title": "Certificate Signing Request information",
6220               "definitions": {
6221                 "oic.r.csr": {
6222                   "type": "object",
6223                   "properties": {
6224                     "csr": {
6225                       "type": "string",
6226                       "description": "Signed CSR in ASN.1 in the encoding specified by the

```



```

6227 encoding property"
6228     },
6229     "encoding": {
6230         "type": "string",
6231         "enum": [ "oic.sec.encoding.pem", "oic.sec.encoding.der" ],
6232         "description": "A string specifying the encoding format of the data
6233 contained in csr",
6234         "detail-desc": [ "oic.sec.encoding.pem - Encoding for PEM encoded CSR",
6235                         "oic.sec.encoding.der - Encoding for DER encoded CSR" ]
6236     }
6237 }
6238 }
6239 },
6240 "type": "object",
6241 "allOf": [
6242     { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6243     { "$ref": "#/definitions/oic.r.csr" }
6244 ],
6245 "required": ["csr","encoding"]
6246 }
6247
6248 example: /
6249 {
6250     "rt": ["oic.r.csr"],
6251     "encoding" : "oic.sec.encoding.pem",
6252     "csr": "PEMENCODEDCSR"
6253 }
6254

```

```

6255 404:
6256     description: |
6257     The device does not support certificates and generating CSRs.
6258

```

```

6259 503:
6260     description: |
6261     The device is not yet ready to return a response
6262     Try again later.
6263

```

6264 A.8.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
encoding	string	yes		A string specifying the encoding format of the data contained in csr
csr	string	yes		Signed CSR in ASN.1 in the encoding specified by the encoding property

6265 A.8.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/csr		get			

6266 A.9 OICSecurityRolesResource

6267 A.9.1 Introduction

6268 This resource specifies roles that have been asserted.

6269 **A.9.2 Example URI**

6270 /oic/sec/roles

6271 **A.9.3 Resource Type**

6272 **A.9.4 RAML Definition**

6273 `##RAML 0.8`

6274 `title: OICSecurityRolesResource`

6275 `version: v1.0-20170323`

6276 `traits:`

6277 `- interface :`

6278 `queryParameters:`

6279 `if:`

6280 `enum: ["oic.if.baseline"]`

6281

6282 `/oic/sec/roles:`

6283 `description: |`

6284 `This resource specifies roles that have been asserted.`

6285

6286 `is : ['interface']`

6287 `get:`

6288 `description: |`

6289 `Retrieves the asserted roles data.`

6290

6291 `responses :`

6292 `200:`

6293 `body:`

6294 `application/json:`

6295 `schema: |`

```
6296     {
6297       "$schema": "http://json-schema.org/draft-04/schema#",
6298       "id": "https://www.openconnectivity.org/ocf-
6299 apis/security/schemas/oic.r.roles.json#",
6300       "title": "Asserted Role Certificates",
6301       "definitions": {
6302         "oic.r.cred": {
6303           "type": "object",
6304           "properties": {
6305             "roles": {
6306               "type": "array",
6307               "description": "List of role certificates",
6308               "items": {
6309                 "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
6310               }
6311             }
6312           }
6313         },
6314       },
6315       "type": "object",
6316       "allOf": [
6317         { "$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6318         { "$ref": "#/definitions/oic.r.cred" }
6319       ],
6320       "required": [ "roles" ]
6321     }
6322
```

6323 `example: |`

```
6324     {
6325       "roles" :[ #array of oic.sec.cred objects
6326         {
6327           "credid":1,
```

```

6328         "credtype":8, # Cred type 8 is SIGNED_ASYMMETRIC_KEY
6329         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6330         "pbData":      # Role cert, type oic.sec.pubdatatype
6331         {
6332             "encoding":"oic.sec.encoding.pem",
6333             "data":"PEMENCODEDROLECERT"
6334         },
6335         "optData":      # Optional issuer certificate, type oic.sec.pubdatatype
6336         {
6337             "encoding":"oic.sec.encoding.pem",
6338             "data":"PEMENCODEDISSUERCERT"
6339         }
6340     },
6341     {
6342         "credid":2,
6343         "credtype":8, # Cred type 8 is SIGNED_ASYMMETRIC_KEY
6344         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6345         "pbData":      # Role cert, type oic.sec.pubdatatype
6346         {
6347             "encoding":"oic.sec.encoding.pem",
6348             "data":"PEMENCODEDROLECERT"
6349         },
6350         "optData":      # Optional issuer certificate, type oic.sec.pubdatatype
6351         {
6352             "encoding":"oic.sec.encoding.pem",
6353             "data":"PEMENCODEDISSUERCERT"
6354         }
6355     }
6356 ],
6357 "rt":["oic.sec.cred"],
6358 "if":["oic.if.baseline"]
6359 }
6360 }
6361

```

6362 400:

```

6363     description: |
6364         The request is invalid.
6365

```

6366 post:

```

6367     description: |
6368         Update the roles resource, i.e., assert new roles to this server.
6369         New role certificates that match an existing certificate (i.e., pbData
6370         and optData are the same) are not added to the resource (and 204 is
6371         returned).
6372         The provided credid values are ignored, the resource assigns its own.
6373

```

6374 body:

```

6375     application/json:
6376         schema: /
6377         {
6378             "$schema": "http://json-schema.org/draft-04/schema#",
6379             "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.roles.json#",
6380             "title": "Asserted Role Certificates",
6381             "definitions": {
6382                 "oic.r.cred": {
6383                     "type": "object",
6384                     "properties": {
6385                         "roles": {
6386                             "type": "array",
6387                             "description": "List of role certificates",
6388                             "items": {
6389                                 "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
6390                             }
6391                         }
6392                     }
6393                 }
6394             }

```

```

6395     "type": "object",
6396     "allOf": [
6397       { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6398       { "$ref": "#/definitions/oic.r.cred" }
6399     ],
6400     "required": [ "roles" ]
6401   }
6402

```

6403 example: /

```

6404   {
6405     "roles" :[ #array of oic.sec.cred objects
6406       {
6407         "credid":1,    # Optional
6408         "credtype":8, # Cred type 8 is SIGNED_ASYMMETRIC_KEY
6409         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6410         "pbData":    # Role cert, type oic.sec.pubdatatype
6411           {
6412             "encoding":"oic.sec.encoding.pem",
6413             "data":"PEMENCODEDROLECERT"
6414           },
6415         "optData":    # Optional issuer certificate, type oic.sec.pubdatatype
6416           {
6417             "encoding":"oic.sec.encoding.pem",
6418             "data":"PEMENCODEDISSUERCERT"
6419           }
6420       },
6421       {
6422         "credid":2,    #Optional
6423         "credtype":8, # Cred type 8 is SIGNED_ASYMMETRIC_KEY
6424         "subjectuuid":"00000000-0000-0000-0000-000000000000",
6425         "pbData":    # Role cert, type oic.sec.pubdatatype
6426           {
6427             "encoding":"oic.sec.encoding.pem",
6428             "data":"PEMENCODEDROLECERT"
6429           },
6430         "optData":    # Optional issuer certificate, type oic.sec.pubdatatype
6431           {
6432             "encoding":"oic.sec.encoding.pem",
6433             "data":"PEMENCODEDISSUERCERT"
6434           }
6435       }
6436     ],
6437     "rt":["oic.sec.cred"],
6438     "if":["oic.if.baseline"]
6439   }
6440

```

6441 responses :

```

6442   400:
6443     description: |
6444       The request is invalid.
6445

```

```

6446   204:
6447     description: |
6448       The roles entry is updated.
6449

```

6450 delete:

```

6451   description: |
6452     Deletes roles resource entries.
6453     DELETE does not support query parameters, all the entries are deleted.
6454

```

6455 responses :

```

6456   200:
6457     description: |

```

6458 The roles resource has been successfully deleted.

6459

6460 400:

6461 description: |

6462 The request is invalid.

6463

6464 A.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	multiple types: see schema	yes		
creds	array: see schema	yes		List of credentials available at this resource

6465 A.9.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/roles		get	post	delete	

6466

6467 A.10 OICSecurityCrlResource

6468 A.10.1 Introduction

6469 This resource specifies certificate revocation lists as X.509 objects.

6470 A.10.2 Example URI

6471 /oic/sec/crl

6472 A.10.3 Resource Type

6473 A.10.4 RAML Definition

6474 #%RAML 0.8

6475 title: OICSecurityCrlResource

6476 version: v1.0-20150819

6477 traits:

6478 - interface :

6479 queryParameters:

6480 if:

6481 enum: ["oic.if.baseline"]

6482

6483 /oic/sec/crl:

6484 description: |

6485 This resource specifies certificate revocation lists as X.509 objects.

6486

6487 is : ['interface']

6488 get:

6489 description: |

6490 Retrieves crl data.

6491

6492 responses :

6493 200:

6494 body:

6495 application/json:

6496 schema: /

```

6497     {
6498     "$schema": "http://json-schema.org/draft-04/schema#",
6499     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
6500     "title": "Certificate Revocation List information",
6501     "definitions": {
6502       "oic.r.crl": {
6503         "type": "object",
6504         "properties": {
6505           "crlid": {
6506             "type": "integer",
6507             "description": "Local reference to a crl resource"
6508           },
6509           "thisupdate": {
6510             "type": "string",
6511             "description": "UTC time of last CRL update"
6512           },
6513           "crldata": {
6514             "type": "string",
6515             "description": "Base64 BER encoded crl data"
6516           }
6517         }
6518       }
6519     },
6520     "type": "object",
6521     "allOf": [
6522       { "$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6523       { "$ref": "#/definitions/oic.r.crl" }
6524     ],
6525     "required": ["crlid", "thisupdate", "crldata"]
6526   }
6527

```

```

6528   example: /
6529     {
6530       "rt": ["oic.r.crl"],
6531       "crlid": 1,
6532       "thisupdate": "2016-04-12T23:20:50.52Z",
6533       "crldata": "Base64ENCODEDCRL"
6534     }
6535

```

```

6536   post:
6537     description: |
6538       Updates the CRL data
6539

```

```

6540   body:
6541     application/json:
6542     schema: /
6543     {
6544     "$schema": "http://json-schema.org/draft-04/schema#",
6545     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
6546     "title": "Certificate Revocation List information",
6547     "definitions": {
6548       "oic.r.crl": {
6549         "type": "object",
6550         "properties": {
6551           "crlid": {
6552             "type": "integer",
6553             "description": "Local reference to a crl resource"
6554           },
6555           "thisupdate": {
6556             "type": "string",
6557             "description": "UTC time of last CRL update"
6558           },
6559           "crldata": {
6560             "type": "string",
6561             "description": "Base64 BER encoded crl data"
6562           }
6563         }
6564       }
6565     }

```

```

6564     }
6565     },
6566     "type": "object",
6567     "allOf": [
6568     { "$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6569     { "$ref": "#/definitions/oic.r.crl" }
6570     ],
6571     "required": ["crlid", "thisupdate", "crldata"]
6572 }
6573
6574 example: /
6575 {
6576   "rt": ["oic.r.crl"],
6577   "crlid": 1,
6578   "thisupdate": "2016-04-12T23:20:50.52Z",
6579   "crldata": "Base64ENCODEDCRL"
6580 }
6581
6582 responses :
6583 400:
6584   description: |
6585     The request is invalid.
6586
6587 201:
6588   description: |
6589     The CRL entry is created.
6590
6591 204:
6592   description: |
6593     The CRL entry is updated.
6594
6595 put:
6596   description: |
6597     Creates the CRL data
6598
6599 body:
6600 application/json:
6601   schema: /
6602   {
6603     "$schema": "http://json-schema.org/draft-04/schema#",
6604     "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
6605     "title": "Certificate Revocation List information",
6606     "definitions": {
6607       "oic.r.crl": {
6608         "type": "object",
6609         "properties": {
6610           "crlid": {
6611             "type": "integer",
6612             "description": "Local reference to a crl resource"
6613           },
6614           "thisupdate": {
6615             "type": "string",
6616             "description": "UTC time of last CRL update"
6617           },
6618           "crldata": {
6619             "type": "string",
6620             "description": "Base64 BER encoded crl data"
6621           }
6622         }
6623       }
6624     },
6625     "type": "object",
6626     "allOf": [

```

```

6627         { "$ref": "../../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
6628         { "$ref": "#/definitions/oic.r.crl" }
6629     ],
6630     "required": ["crlid", "thisupdate", "crldata"]
6631 }
6632
6633     example: /
6634     {
6635         "rt": ["oic.r.crl"],
6636         "crlid": 1,
6637         "thisupdate": "2016-04-12T23:20:50.52Z",
6638         "crldata": "Base64ENCODEDCRL"
6639     }
6640
6641     responses :
6642     400:
6643         description: |
6644             The request is invalid.
6645
6646     201:
6647         description: |
6648             The CRL entry is created.
6649
6650     delete:
6651         description: |
6652             Deletes the CRL data.
6653             When DELETE is used without query parameters, the entire CRL resource is deleted.
6654             When DELETE uses a query value naming the crlid, only the matched entry is deleted.
6655
6656     queryParameters:
6657     subject:
6658         type: string
6659
6660         description: Delete the CRL identified by the string containing CRL ID.
6661
6662         required: false
6663
6664         example: DELETE /mycrl?crlid=0
6665
6666     responses :
6667     400:
6668         description: |
6669             The request is invalid.
6670
6671     204:
6672         description: |
6673             The CRL instance or the the entire CRL resource has been successfully deleted.

```

6673 A.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
crldata	string	yes		Base64 BER encoded crl data
crlid	integer	yes		Local reference to a crl resource
thisupdate	string	yes		UTC time of last CRL update

6674

A.10.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/sec/crl	put	get	post	delete	

6675

6676