

1                   **OCF “Fargo” – Cloud API for Cloud Services – Core Technology WG**

2  
3  
4                   Legal Disclaimer

5  
6 THIS IS A DRAFT SPECIFICATION DOCUMENT ONLY AND HAS NOT BEEN ADOPTED BY THE  
7 OPEN CONNECTIVITY FOUNDATION. THIS DRAFT DOCUMENT MAY NOT BE RELIED UPON  
8 FOR ANY PURPOSE OTHER THAN REVIEW OF THE CURRENT STATE OF THE DEVELOPMENT  
9 OF THIS DRAFT DOCUMENT. THE OPEN CONNECTIVITY FOUNDATION AND ITS MEMBERS  
10 RESERVE THE RIGHT WITHOUT NOTICE TO YOU TO CHANGE ANY OR ALL PORTIONS  
11 HEREOF, DELETE PORTIONS HEREOF, MAKE ADDITIONS HERETO, DISCARD THIS DRAFT  
12 DOCUMENT IN ITS ENTIRETY OR OTHERWISE MODIFY THIS DRAFT DOCUMENT AT ANY  
13 TIME. YOU SHOULD NOT AND MAY NOT RELY UPON THIS DRAFT DOCUMENT IN ANY WAY,  
14 INCLUDING BUT NOT LIMITED TO THE DEVELOPMENT OF ANY PRODUCTS OR SERVICES.  
15 IMPLEMENTATION OF THIS DRAFT DOCUMENT IS DONE AT YOUR OWN RISK AMEND AND  
16 IT IS NOT SUBJECT TO ANY LICENSING GRANTS OR COMMITMENTS UNDER THE OPEN  
17 CONNECTIVITY FOUNDATION INTELLECTUAL PROPERTY RIGHTS POLICY OR OTHERWISE.  
18 IN CONSIDERATION OF THE OPEN CONNECTIVITY FOUNDATION GRANTING YOU ACCESS  
19 TO THIS DRAFT DOCUMENT, YOU DO HEREBY WAIVE ANY AND ALL CLAIMS ASSOCIATED  
20 HERewith INCLUDING BUT NOT LIMITED TO THOSE CLAIMS DISCUSSED BELOW, AS WELL  
21 AS CLAIMS OF DETRIMENTAL RELIANCE.

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

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

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

# CONTENTS

27  
28

29	1	Scope .....	1
30	2	Normative references .....	1
31	3	Terms, definitions, and abbreviated terms .....	2
32	3.1	Terms and definitions.....	2
33	3.2	Abbreviated terms.....	2
34	4	Document conventions and organization.....	3
35	4.1	Conventions.....	3
36	4.2	Notation.....	3
37	5	Overview .....	4
38	5.1	Introduction.....	4
39	5.2	General OCF Cloud API for Cloud Services Elements .....	4
40	5.3	Cloud to Cloud Operational Overview .....	5
41	5.3.1	Introduction .....	5
42	5.3.2	Conceptual Architecture .....	5
43	5.3.3	Authorizing Cloud Connectivity .....	6
44	5.3.4	Synchronization of Users set of Devices .....	6
45	5.3.5	Keeping Up-to-Date: Notifications of changes on other Clouds .....	6
46	5.3.6	Handling of Requests and Responses for Connected Devices .....	6
47	6	Authentication & Authorization.....	6
48	7	Account Linking API .....	7
49	7.1	General.....	7
50	7.2	OAuth 2.0 OCF Cloud API for Cloud Services Scopes.....	8
51	8	Devices API.....	9
52	8.1	Introduction.....	9
53	8.2	Parameters Supported .....	9
54	8.3	Retrieve All Devices.....	10
55	8.3.1	Summary .....	10
56	8.3.2	Request and Response Payload .....	11
57	8.3.3	Responses .....	11
58	8.4	Retrieve One Device .....	11
59	8.4.1	Summary .....	11
60	8.4.2	Request and Response Payload .....	12
61	8.4.3	Responses .....	12
62	8.5	Retrieve Specific Resource.....	12
63	8.5.1	Summary .....	12
64	8.5.2	Request and Response Payload .....	13
65	8.5.3	Responses .....	13
66	8.6	Update a Resource on a Device.....	14
67	8.6.1	Summary .....	14
68	8.6.2	Request and Response Payload .....	14

69	8.6.3	Responses .....	15
70	9	Events API .....	15
71	9.1	Introduction.....	15
72	9.2	Events Authentication .....	16
73	9.2.1	Create Event Signature .....	16
74	9.2.2	Verify the Event Signature .....	16
75	9.3	Parameters Supported .....	16
76	9.4	Events API subscription and notification request and response payload	
77		definitions .....	17
78	9.4.1	Subscription request.....	17
79	9.4.2	Subscription response .....	18
80	9.4.3	Notification request population.....	19
81	9.5	Subscribe and Unsubscribe to Devices level events, retrieval of the subscription	
82		details.....	20
83	9.5.1	Summary .....	20
84	9.5.2	Request and Response Payload .....	21
85	9.5.3	Responses .....	21
86	9.6	Subscribe and Unsubscribe to Device level events, retrieval of the subscription	
87		details.....	21
88	9.6.1	Summary .....	21
89	9.6.2	Request and Response Payload .....	22
90	9.6.3	Responses .....	22
91	9.7	Subscribe and Unsubscribe to Resource level events, retrieval of the	
92		subscription details .....	22
93	9.7.1	Summary .....	22
94	9.7.2	Request and Response Payload .....	23
95	9.7.3	Responses .....	23
96	9.8	Notification of /devices level events .....	24
97	9.8.1	Summary .....	24
98	9.8.2	Request and Response Payload .....	24
99	9.8.3	Responses .....	24
100	9.9	Notification of Device level events .....	24
101	9.9.1	Summary .....	24
102	9.9.2	Request and Response Payload .....	25
103	9.9.3	Responses .....	25
104	9.10	Notification of Resource level events .....	25
105	9.10.1	Summary .....	25
106	9.10.2	Request and Response Payload .....	25
107	9.10.3	Responses .....	25
108	A.1	Introduction.....	26
109	A.2	OAuth2.0 Application Registration.....	26
110	A.3	Account Linking .....	26
111	A.4	Retrieval of all Devices .....	27
112	A.4.1	Summary .....	27
113	A.4.2	Flow .....	27

114	A.4.3	Flow Description .....	28
115	A.5	Retrieval of a single Device .....	28
116	A.5.1	Summary .....	28
117	A.5.2	Flow .....	28
118	A.5.3	Flow Description .....	29
119	A.6	Retrieval of a single Resource .....	29
120	A.6.1	Summary .....	29
121	A.6.2	Flows.....	29
122	A.7	Update of a single Resource .....	31
123	A.7.1	Summary .....	31
124	A.7.2	Flows.....	31
125	A.8	Establishment of new subscription request.....	32
126	A.8.1	Summary .....	32
127	A.8.2	Flows.....	32
128	A.9	Event generated for a subscription.....	33
129	A.9.1	Summary .....	33
130	A.9.2	Flows.....	33
131	A.10	Addition of new registration.....	33
132	A.10.1	Summary .....	33
133	A.10.2	Flows.....	34
134	A.11	Removal of existing device registration .....	34
135	A.11.1	Summary .....	34
136	A.11.2	Flows.....	34
137	B.1	OpenAPI 2.0 definition .....	<b>Error! Bookmark not defined.</b>
138			
139			

140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160

## Figures

Figure 1 – OCF Cloud Overview .....	4
Figure 2 – Conceptual Architecture .....	5
Figure 3 – Subscription Request Example.....	18
Figure 4 – Subscription Response Example Payload.....	19
Figure A.1 – Establish Business Relationship Example Flow.....	26
Figure A.2 – Initial Association Example Flow .....	27
Figure A.3 – Retrieve All Devices Example Flow .....	28
Figure A.4 – Retrieve Single Device Example Flow .....	29
Figure A.5 – Retrieve Resource (Success) Example Flow .....	30
Figure A.6 – Retrieve Resource (Timeout) Example Flow .....	31
Figure A.7 – Update Resource (Success) Example Flow.....	31
Figure A.8 – Update Resource (Timeout) Example Flow .....	32
Figure A.9 – Observe Establishment Example Flow .....	33
Figure A.10 – "resource_contentchanged" Event Example Flow.....	33
Figure A.11 – Addition of new registered Device example flow.....	34
Figure A.12 – Removal of existing registration example flow.....	34

## Tables

161	
162	
163	Table 1 – OCF Cloud API for Cloud Services API OAuth Client Scopes .....8
164	Table 2 – Required scopes per API endpoint .....8
165	Table 3 – Parameters used in the Device API ..... 10
166	Table 4 – Retrieve All Devices API Summary ..... 10
167	Table 5 – Retrieve One Device API Summary ..... 11
168	Table 6 – Retrieve Specific Resource API Summary ..... 12
169	Table 7 – Update Resource API Summary ..... 14
170	Table 8 – Parameters used in the Events API ..... 17
171	Table 9 – Subscription Request Properties ..... 17
172	Table 10 – Subscription Response Properties ..... 18
173	Table 11 – Event HTTP Headers description ..... 19
174	Table 12 – Event type to content format map ..... 19
175	Table 13 – Subscription to /devices API Summary ..... 21
176	Table 14 – Subscription to Single Device API Summary ..... 22
177	Table 15 – Subscription to Resource API Summary ..... 23
178	Table 16 – Notification of /devices API Summary ..... 24
179	Table 17 – Notification of Single Device API Summary ..... 24
180	Table 18 – Notification of Resource API Summary ..... 25
181	Table A.1 – Retrieve all Devices Flow Summary ..... 28
182	Table A.2 – Retrieve single Device Flow Summary ..... 29
183	Table A.3 – Retrieve single Resource Flow Summary ..... 30
184	Table A.4 – Update single Resource Flow Summary ..... 31
185	

186 **1 Scope**

187 This document defines functional requirements for the OCF Cloud to Cloud Application  
188 Programming Interface (API).

189 **2 Normative references**

190 The following documents are referred to in the text in such a way that some or all of their content  
191 constitutes requirements of this document. For dated references, only the edition cited applies. For  
192 undated references, the latest edition of the referenced document (including any amendments)  
193 applies.

194 IETF RFC 2818, *HTTP over TLS*, May 2000  
195 <https://tools.ietf.org/html/rfc2818>

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

198 IETF RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*, October 2012  
199 <https://www.rfc-editor.org/info/rfc6750>

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

202 IETF RFC 8414, *OAuth 2.0 Authorization Server Metadata*, June 2018  
203 <https://tools.ietf.org/html/rfc8414>

204 IETF RFC 5785, *Defining Well-Known Uniform Resource Identifiers (URIs)*, April 2010  
205 <https://tools.ietf.org/html/rfc5785>

206 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)  
207 Specification -- Part 1: Core specification  
208 <https://www.iso.org/standard/53238.html>  
209 Latest version available at: [https://openconnectivity.org/specs/OCF\\_Core\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Core_Specification.pdf)

210 ISO/IEC 30118-2:2018 Information technology -- Open Connectivity Foundation (OCF)  
211 Specification -- Part 2: Security specification  
212 <https://www.iso.org/standard/74239.html>  
213 Latest version available at: [https://openconnectivity.org/specs/OCF\\_Security\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Security_Specification.pdf)

214 OCF Device to Cloud Services Specification, *Open Connectivity Foundation Device to Cloud  
215 Services Specification*,  
216 Latest version available at:  
217 [https://openconnectivity.org/specs/OCF\\_Cloud\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf)

218 OCF Cloud API for Cloud Services [https://github.com/openconnectivityfoundation/core-  
219 extensions/blob/ocfcloud-openapi/swagger2.0/oic.r.cloudopenapi.swagger.json](https://github.com/openconnectivityfoundation/core-extensions/blob/ocfcloud-openapi/swagger2.0/oic.r.cloudopenapi.swagger.json)

220 OpenAPI 2.0, *fka Swagger RESTful API Documentation Specification, Version 2.0*  
221 <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md>

222

223 **3 Terms, definitions, and abbreviated terms**

224 **3.1 Terms and definitions**

225 For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2018 and  
226 ISO/IEC 30118-2:2018 and the following apply.

227 ISO and IEC maintain terminological databases for use in standardization at the following  
228 addresses:

- 229 – ISO Online browsing platform: available at <https://www.iso.org/obp>
- 230 – IEC Electropedia: available at <http://www.electropedia.org/>

231

232 **3.2 Abbreviated terms**

233 **3.2.1**

234 **API**

235 Application Programming Interface

236 **3.2.2**

237 **JWT**

238 JSON Web Token

239



## 240 **4 Document conventions and organization**

### 241 **4.1 Conventions**

242 In this document a number of terms, conditions, mechanisms, sequences, parameters, events,  
243 states, or similar terms are printed with the first letter of each word in uppercase and the rest  
244 lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal  
245 technical English meaning.

### 246 **4.2 Notation**

247 In this document, features are described as required, recommended, allowed or DEPRECATED as  
248 follows:

249 Required (or shall or mandatory)(M).

- 250 – These basic features shall be implemented to comply with Core Architecture. The phrases "shall  
251 not", and "PROHIBITED" indicate behaviour that is prohibited, i.e. that if performed means the  
252 implementation is not in compliance.

253 Recommended (or should)(S).

- 254 – These features add functionality supported by Core Architecture and should be implemented.  
255 Recommended features take advantage of the capabilities Core Architecture, usually without  
256 imposing major increase of complexity. Notice that for compliance testing, if a recommended  
257 feature is implemented, it shall meet the specified requirements to be in compliance with these  
258 guidelines. Some recommended features could become requirements in the future. The phrase  
259 "should not" indicates behaviour that is permitted but not recommended.

260 Allowed (may or allowed)(O).

- 261 – These features are neither required nor recommended by Core Architecture, but if the feature  
262 is implemented, it shall meet the specified requirements to be in compliance with these  
263 guidelines.

264 DEPRECATED.

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

271 Conditionally allowed (CA)

- 272 – The definition or behaviour depends on a condition. If the specified condition is met, then the  
273 definition or behaviour is allowed, otherwise it is not allowed.

274 Conditionally required (CR)

- 275 – The definition or behaviour depends on a condition. If the specified condition is met, then the  
276 definition or behaviour is required. Otherwise the definition or behaviour is allowed as default  
277 unless specifically defined as not allowed.

278

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

280 Words that are emphasized are printed in italic.

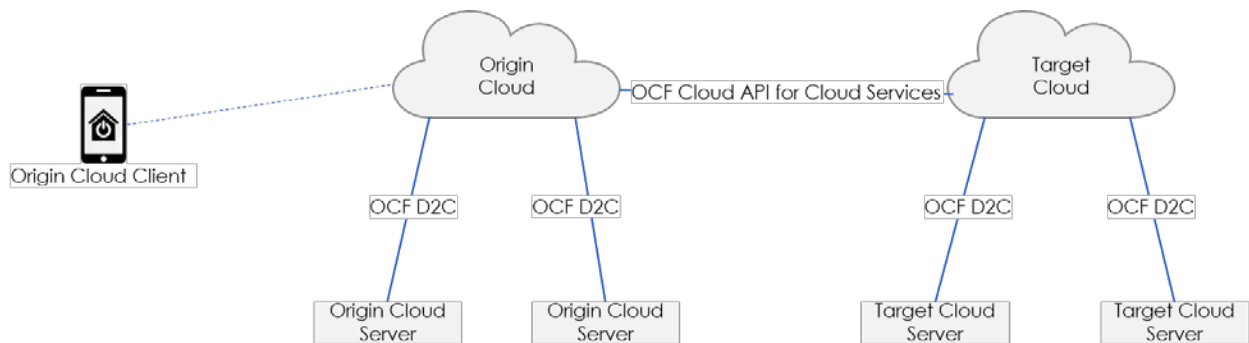
281 **5 Overview**

282 **5.1 Introduction**

283 This document defines the OCF Cloud API for Cloud Services. In this document "origin Cloud"  
284 refers to the OCF Cloud or the 3<sup>rd</sup> party Cloud through which the user works with his OCF Devices,  
285 "target Cloud" refers to the OCF Cloud to which OCF Servers (OCF Devices) are connected which  
286 the user wants to control via the "origin Cloud".

287 An OCF Device is a collection of Resources, each Resource being an OpenAPI 2.0 defined object  
288 that represents a physical property or characteristic of the Device (e.g. temperature sensed, light  
289 colour, power on switch). The Device itself has an associated Device Type that provides an  
290 indication of what the Device is, for example a Light is represented as a Device Type of "oic.d.light".

291 Please see Figure 1 for a representation of the target architecture.



292

293 **Figure 1 – OCF Cloud Overview**

294 The OCF Cloud API for Cloud Services supports the following cases:

- 295 – Account Linking API (clause 7)
  - 296 – Initial Account Linking
  - 297 – Removal of linked account
- 298 – Devices API (clause 8)
  - 299 – Retrieval of all Devices associated with a User (clause 8.3)
  - 300 – Retrieval of a single Device associated with a User (clause 8.4)
  - 301 – Retrieval of a single Resource (clause 8.5)
  - 302 – Update of a single Resource (clause 8.6)
- 303 – Events API (clause 9)
  - 304 – Subscription to an event: establishment of a subscription
  - 305 – Notification: event generated on an established subscription

306 **5.2 General OCF Cloud API for Cloud Services Elements**

307 The OCF Cloud API for Cloud Services is a RESTful API over HTTPS (IETF RFC 2818). The API  
308 is defined using OpenAPI 2.0.

309 The "origin Cloud" communicates with the "target Cloud" using the domain name or URI it has  
310 obtained from the initial OAuth 2.0 (IETF RFC 6749) Client Setup, covered in clause 7.  
311 Communication between OCF Devices and OCF Clouds is defined in the OCF Device to Cloud  
312 Services Specification.

313 All URIs presented as "href" Properties in a Link in response to a Client or Cloud discovery are in  
314 the form "/deviceid/resourceHref".

315 All requests include the bearer token for the user in question obtained via standard OAuth2.0 (IETF  
316 RFC 6749) mechanisms.

317 Any query parameters received by an "origin Cloud" in a request from an OCF Client are passed  
318 through clean (i.e. are part of the URI) in any request that is sent to a target Cloud.

319 Each request may contain an optional HTTP Correlation-ID header, which carries a unique identifier  
320 value that provides a reference to a particular transaction or event chain in the target cloud.

321 All requests include an HTTP Accept header. All requests or responses that carry content include  
322 an HTTP Content-Type header. At a minimum media-types "application/json" and  
323 "application/vnd.ocf+cbor" are supported. If the recipient of a request cannot provide a response  
324 that is encoded according to the content of the Accept header, then a HTTP 406 (not acceptable)  
325 response shall be sent. On reception of a 406 response the originator of the request may re-attempt  
326 the request using an alternative Content-Type if supported.

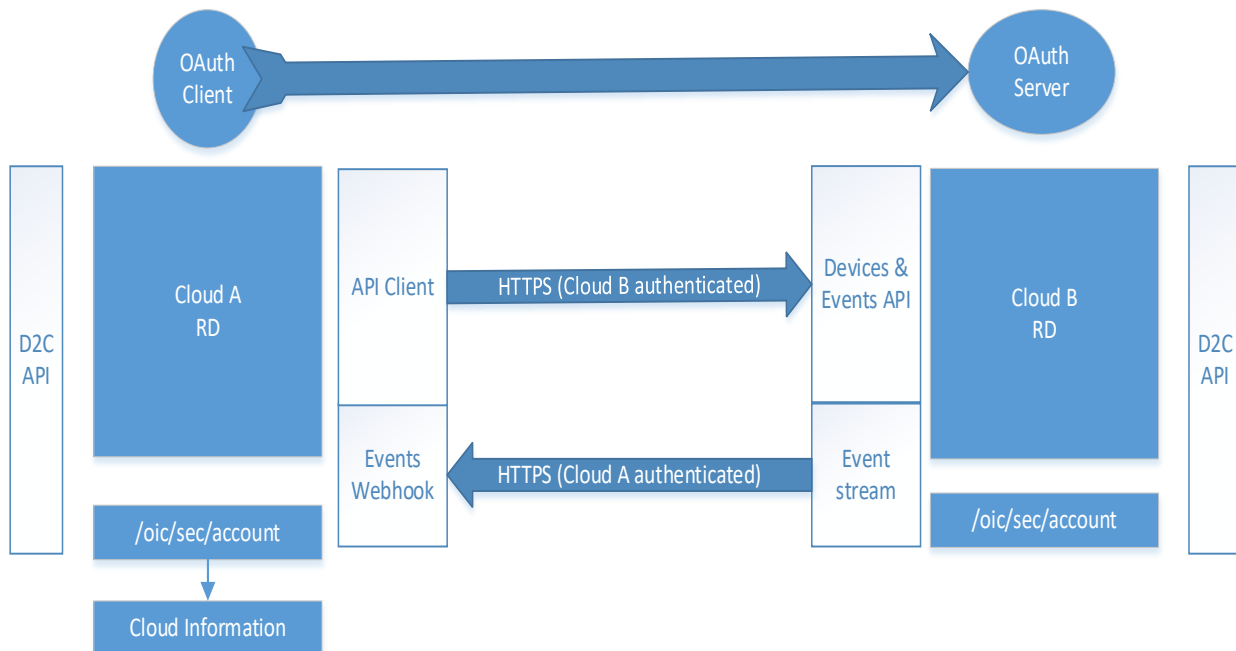
### 327 5.3 Cloud to Cloud Operational Overview

#### 328 5.3.1 Introduction

329 This clause provides an informative overview of the flows that are enabled by the detailed API  
330 defined in clauses 6, 7, 8, and 9. Clause 5.3 provides references to the applicable clauses within  
331 this document that define the API specifics.

#### 332 5.3.2 Conceptual Architecture

333 Figure 2 describes the overall conceptual architecture.



334

335

**Figure 2 – Conceptual Architecture**

### 336 **5.3.3 Authorizing Cloud Connectivity**

337 Consider a user who has accounts on two distinct, separately owned clouds, and devices  
338 associated with each of those accounts on those clouds. The user wants to have a unified view of  
339 all of their devices from a single client rather than having a client per cloud. The user via the client  
340 they want to use for all devices indicates to the directly connected cloud ("origin Cloud") that they  
341 want to link this account with an account on the other cloud ("target Cloud"). This initiates a  
342 standard OAuth2.0 Authorization Code Grant Type flow, see IETF RFC 6749, clause 1.3.1.  
343 Application of this flow is described in clause 7.

### 344 **5.3.4 Synchronization of Users set of Devices**

345 After completion of the Authorization Code Grant Type flow from clause 5.3.3 the "origin Cloud"  
346 (that is the cloud to which the user is connected) is authorized to use the Device API to obtain on  
347 behalf of the user the complete list of devices hosted on the "target Cloud" for which the user has  
348 access. The API is described in clause 8, and the flow is further illustrated in clause A.4.

349 The result of the invocation of the Device API is a complete set of device information that may then  
350 be provided in a response to a RETRIEVE on "/oic/res" from the "origin Cloud".

### 351 **5.3.5 Keeping Up-to-Date: Notifications of changes on other Clouds**

352 Once the set of devices has been obtained, the "origin Cloud" can subscribe to the events to which  
353 it is interested across the user's complete device set ("/devices"), or per device in that set  
354 ("/devices/{deviceid}"). See clause 9 for details of the API itself.

355 The subscription to "/devices" enables the "origin Cloud" to be notified whenever a new device is  
356 added or an existing device removed from the "target Cloud".

357 The subscription to "/devices/{deviceid}" enables the "origin Cloud" to be notified whenever there  
358 is a change in the state of a device (e.g. it has de-registered).

359 When a new Device registers on the "target Cloud", and a subscription exists, then a notification is  
360 sent to the "origin Cloud" with an event\_type of "devices\_registered" and a payload which contains  
361 the "/oic/d" of the newly onboarded device. The "origin Cloud" may then RETRIEVE the Links  
362 exposed by the newly added device using "/devices/{deviceid}" where "deviceid" was provided in  
363 the payload of the notification. See clause A.10 for a flow illustrating this interaction.

### 364 **5.3.6 Handling of Requests and Responses for Connected Devices**

365 From the perspective of the Client connected to the "origin Cloud" there is no distinction between  
366 devices and their resources hosted by the "origin Cloud" itself and devices and their resources that  
367 are hosted by an OCF Cloud API for Cloud Services connected cloud.

368 Thus all requests for a target resource are formed using the mechanisms described in the OCF  
369 Device to Cloud Services Specification.

370 The "origin Cloud" identifies the host Cloud for the requested Resource via the instance of  
371 "/oic/sec/account" for the "deviceid" that is in the request URI. The request is then effectively  
372 proxied to the "target Cloud" via the "/devices/{deviceid}/{resourcehref}" API exposed by the "target  
373 Cloud" (see clause 8.5 and 8.6). Any query parameters received over the device to cloud  
374 connection are included in the URI unaltered. The content-type of the payload in the request or  
375 response is honoured. See clauses A.6 and A.7 for illustrative flows of this mechanism for both  
376 RETRIEVE and UPDATE cases.

## 377 **6 Authentication & Authorization**

378 All requests to the "target Cloud" are transferred over an HTTPS connection using server-  
379 authenticated TLS1.2 connection and are protected by OAuth2.0 JWT Bearer Tokens, see IETF

380 RFC 7519. The "origin Cloud" uses the "Bearer" authentication scheme inside the "Authorization"  
381 request header field to transmit the access token, as per IETF RFC 6750 clause 2.1. For definition  
382 of the "Authorization" request header field, see IETF RFC 2818.

383 The OAuth2.0 server issuing JWT tokens must also serve the Authorization Server Metadata  
384 described in IETF RFC 8414 clause 2, published at the location that is ".well-known" according to  
385 IETF RFC 5785. Parsing of the OAuth2.0 JWT token identifies the user identity and the client  
386 delegating the request to the "target Cloud".

387 On the OCF Server side there is no distinction between requests forwarded from the "origin Cloud"  
388 and requests coming via the "target Cloud".

## 389 **7 Account Linking API**

### 390 **7.1 General**

391 The account linking API is the mechanism by which Devices hosted on behalf of a user by the  
392 "target Cloud" are linked with a user identity on the "origin Cloud". Account linking is established  
393 solely between the "origin Cloud" and the "target Cloud"; devices from the "target Cloud" cannot  
394 be proxied further through the OCF Cloud API for Cloud Services.

395 The OAuth Client of the "origin Cloud" has to be registered with the "target Cloud" as a prerequisite  
396 to initiating the Authorization Code Grant Type flow, which allows the user to link his "origin Cloud"  
397 account with the "target Cloud". This process is named OAuth Application registration and is  
398 beyond the scope of this specification. Successful registration of the OAuth "origin Cloud" Client in  
399 the "target Cloud" relies on the two entities establishing trust and obtaining the required client  
400 properties (e.g. client id, client secret, allowed redirect URIs). See IETF RFC 6749, clause 2.

401 The linking is then achieved via the use of an OAuth2.0 Authorization Code Grant Type. Part of the  
402 linking process is the end-user consent, which is very important in cross-domain identity federation,  
403 ensuring that a malicious OAuth 2.0 Client cannot obtain authorization without the awareness and  
404 explicit consent of the resource owner (that is the user) of the "target Cloud". The "target Cloud"  
405 should present to the user linking the account the precise scope of authorization information being  
406 requested by the Client. Details about scopes are available in clause 7.2. After the user's consent  
407 and subsequent authorization code exchange, the JWT bearer access and refresh tokens are  
408 obtained from the "target Cloud" by the "origin Cloud", following the format and Content Type in  
409 IETF RFC 6750 clause 4. The JWT bearer access token identifies a user identity on the "target  
410 Cloud".

411 Presence of the state query parameter, see IETF RFC 6749 clause 4.1.1 is required. State is an  
412 opaque value used by the "origin cloud" Client to maintain state between the request and the  
413 callback during the account linking process, see clause A.3.

414 Once such a JWT bearer token has been acquired, the "origin Cloud" links the OAuth2.0 access  
415 and refresh token with its known local userid. The user who linked his "target Cloud" account with  
416 the "origin Cloud" account is from this moment able to request all his devices through the "origin  
417 Cloud", because the "origin Cloud" can make requests to the "target Cloud" on behalf of the "target  
418 Cloud" user account. However, the "origin Cloud" can make requests only in the scope granted by  
419 the user during the consent screen.

420 On initial acquisition of the token the "origin Cloud" may use the Device API to retrieve the Device  
421 details for all Devices in scope of the bearer token.

422 If the "origin Cloud" supports the behaviour defined in the OCF Device to Cloud Services  
423 Specification, then once the "origin Cloud" has the set of Devices from the "target Cloud" it creates  
424 an instance of "/oic/sec/account" per Device. The optional Property "cloudid" in "/oic/sec/account"  
425 is set to the OCF Cloud UUID of the "target Cloud" available in the Common Name field of the End-  
426 Entity certificate used to secure the HTTPS OCF Cloud API for Cloud Services Endpoint. If the

427 Property is missing, empty, or contains the same value as OCF Cloud UUID of the "origin Cloud",  
 428 then the Device is local to the "origin Cloud".

429 The "origin Cloud" might use the Events API to establish an "observe" relationship with the Device(s)  
 430 on the "target Cloud"; such that addition or deletion of Devices on the "target Cloud" can be correctly  
 431 reflected in the "origin Cloud". When the Device is unregistered from the "target Cloud", that Device  
 432 is no longer accessible via the "origin Cloud". When the JWT bearer token obtained from the "target  
 433 Cloud" expires and the refresh token is still valid, the "origin Cloud" can ask for a new access token  
 434 through the OAuth2.0 token endpoint of the "target Cloud". Whenever the refresh token expires, is  
 435 not available, or the access token cannot be obtained, the "origin Cloud" removes all associations  
 436 with the Devices hosted by the "target Cloud".

437 It is recommended that the "origin Cloud" establishes an "event stream" for each Device that is  
 438 hosted on the "target Cloud" by using the subscription mechanism described in clause 9.6.

## 439 7.2 OAuth 2.0 OCF Cloud API for Cloud Services Scopes

440 The OCF Cloud API for Cloud Services defines core set of OAuth Access Token Scopes, see IETF  
 441 RFC 6749. List of scopes defined in Table 1 must be supported together with its description by  
 442 each implementation and presented to the user during the account linking process by the OAuth2.0  
 443 server of the "target Cloud". The "target Cloud" user shall see an appropriate description on the  
 444 consent screen and give an explicit consent that the "origin Cloud" requesting the token is  
 445 authorized to act on behalf of the user in the boundary of obtained scopes.

446

447 **Table 1 – OCF Cloud API for Cloud Services API OAuth Client Scopes**

Scope name	Scope description
	"The application will be able to:"
r:deviceinformation:*	Read basic device information
r:resources:*	Retrieve the data from Resources on your devices
w:resources:*	Update the data in Resources on your devices
w:subscriptions:*	Subscribe to events

448

449 Table 2 details the scopes that are applicable per API endpoint.

450

451 **Table 2 – Required scopes per API endpoint**

API Endpoint	HTTP Request Type	Required scopes
/api/v1/devices	GET	r:deviceinformation:*
/api/v1/devices?content=all	GET	r:deviceinformation:*
		r:resources:*

/api/v1/devices/{deviceid}	GET	r:deviceinformation:*
/api/v1/devices/{deviceid}?content=all	GET	r:deviceinformation: r:resources:*
/api/v1/devices/{deviceid}/{resourcehref}	GET	r:resources:*
	POST	w:resources:*
/api/v1/devices/subscriptions	POST	r:deviceinformation: w:subscriptions:*
	DELETE	r:deviceinformation: w:subscriptions:*
/api/v1/devices/{deviceid}/subscriptions	POST	r:deviceinformation: w:subscriptions:*
	DELETE	r:deviceinformation: w:subscriptions:*
/api/v1/devices/{deviceid}/{resourcehref}/subscriptions	POST	r:resources: w:subscriptions:*
	DELETE	r:resources: w:subscriptions:*

452

453 List of scopes is not finite; the "target Cloud" can derive from the defined list of scopes following  
454 the wildcard scheme. As an example, take the derived scope "w:resources:schedules:\*". If the  
455 "target Cloud" user give the consent to the "origin Cloud" to "w:resources:schedules:\*" - define new  
456 schedules for the device, "origin Cloud" can do it automatically on behalf of the user. But if the  
457 consent was given only for this scope, "origin Cloud" cannot for example update scenes -  
458 "w:resources:scenes:\*". In case the user gives consent to the "origin Cloud" to "w:resources:\*"  
459 which must be by default supported, consent applies to all derived scopes, so the user is able to  
460 do both, update scenes and define new schedules for the device.

461 The request to get the OAuth Access Token shall contain only scopes from Table 1. If needed, the  
462 "target Cloud" can extend the OAuth Access Token with pure vendor specific scopes, but they shall  
463 not be present in the OAuth Access Token request.

## 464 **8 Devices API**

### 465 **8.1 Introduction**

466 The Devices API supports the ability to retrieve and interact with the OCF Devices that are within  
467 the scope of the provided bearer token.

### 468 **8.2 Parameters Supported**

469 Table 3 lists the parameters that may be provided within the Device API.

**Table 3 – Parameters used in the Device API**

<b>Friendly Name</b>	<b>Parameter Name</b>	<b>Location</b>	<b>Description</b>
<b>Accept</b>	Accept	Header	An Accept request HTTP header advertises which content types, expressed as MIME types, the client is able to understand. The resource server then selects one of the proposal and informs the client of its choice with the Content-Type response header.
<b>Correlation ID</b>	Correlation-ID	Header	A Correlation ID, also known as a Transit ID, is a unique identifier value that is attached to requests and messages that allows reference to a particular transaction or event chain.
<b>All Content</b>	content=[base, all]	Query String Parameter	Indicates to the recipient that the response payload shall be the resolved (i.e. resource representation) Link and not the Link itself. Default is base.
<b>Request Payload</b>	payload	Body	Request payload as defined by OCF for the target Resource Type.

471

472 **8.3 Retrieve All Devices**473 **8.3.1 Summary**

474 This request is sent from the "origin Cloud" to the "target Cloud" in order to obtain information on  
475 all the Devices that are registered for the user that is in scope on the "target Cloud".

476 A request to this API may be triggered on the "origin Cloud" by the completion of account linking.  
477 Where the Cloud supports the behaviour defined in the OCF Device to Cloud Services Specification  
478 this may also be triggered by reception of a RETRIEVE to "/oic/res" of the Cloud Resource Directory  
479 from an OCF Client.

480 Table 4 provides a summary of the API.

481

**Table 4 – Retrieve All Devices API Summary**

<b>HTTP Request Type</b>	<b>URI</b>	<b>Parameters</b>	<b>Response Code</b>	<b>Response Payload</b>
<b>GET</b>	/api/v1/devices	content=[base, all], Correlation-ID, Accept	200	See clause B.1 OpenAPI 2.0 Definition - array of /definitions/Device
			400, 401, 403	Diagnostic payload containing detailed reason.



482 **8.3.2 Request and Response Payload**

483 There is no required payload in the request. The required response payload shall be an array of  
 484 objects; each object shall contain Properties from "/oic/d" that are relevant for Cloud operation, a  
 485 status Property ("status") that indicates whether the Device is online or offline, plus an array of  
 486 Links (as defined for "/oic/res") for the Resources exposed by the specific Device. The minimum  
 487 set of Resources that are exposed depends on the OCF Device Type of the Device; this shall be  
 488 in conformance with the requirements for Resource Publication in the OCF Device to Cloud  
 489 Services Specification.

490 If the request includes "content=all" (analogous to a batch retrieval of /oic/res in the proximal  
 491 network) then instead of an array of Links to the hosted Resources, the response payload shall be  
 492 an array of the representations of the Resources themselves that are exposed for each Device that  
 493 is available, which includes the Device Information sent in the case where this parameter is not  
 494 provided. This is illustrated in the examples provided for the Device API in Annex B. See also the  
 495 definition of a batch response in ISO/IEC 30118-1:2018.

496 **8.3.3 Responses**

497 A 200 response is provided in a success case. The payload contains information for all Devices  
 498 that are in the scope of the bearer token.

499 A 400 response indicates that the request was malformed or badly constructed.

500 A 401 response indicates that the request is unauthorized (likely an invalid bearer token)

501 A 403 response indicates that the requestor is known however the scope if the request is forbidden.

502 **8.4 Retrieve One Device**

503 **8.4.1 Summary**

504 This request is sent from the "origin Cloud" to the "target Cloud" in order to obtain information on  
 505 a specific Device that is registered for the user that is in scope on the "target Cloud".

506 A request to this API may be triggered on the "origin Cloud" by the reception of a notification that  
 507 a new Device has been added to a partner cloud, or as part of the flow following account linking.  
 508 Where the Cloud supports the OCF Device to Cloud Services Specification, a request to this API  
 509 may also be triggered by reception of a RETRIEVE to "/oic/res" of the Cloud Resource Directory  
 510 from an OCF Client with a query parameter that specifies a particular deviceid ("anchor").

511 Table 5 provides a summary of the API.

512 **Table 5 – Retrieve One Device API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
GET	/api/v1/devices/{deviceid}	content=[base, all], Correlation-ID, Accept	200	See B.1 OpenAPI 2.0 Definition - /definitions/Device
			400, 401, 403, 404	Diagnostic payload containing detailed reason.

513

514 **8.4.2 Request and Response Payload**

515 The deviceid in the URI of the request is the same as the "di" Property from /oic/d of the target  
516 OCF device.

517 The response payload shall be an object containing the mandatory Device information as defined  
518 in clause 8.3.2.

519 If the request includes "content=all" (analogous to a batch retrieval of /oic/res in the proximal  
520 network) then instead of an array of Links to the hosted Resources, the response payload shall be  
521 an array of the representations of the Resources themselves that are exposed by the Device, which  
522 includes the Device Information sent in the case where this parameter is not provided. This is  
523 illustrated in the examples provided for the Device API in Annex B. See also the definition of a  
524 batch response in ISO/IEC 30118-1:2018.

525 .

526 **8.4.3 Responses**

527 A 200 response is provided in a success case. The payload contains information for the requested  
528 Device.

529 A 400 response indicates that the request was malformed or badly constructed.

530 A 401 response indicates that the request is unauthorized (likely an invalid bearer token)

531 A 403 response indicates that the requestor is known however the scope of the request is forbidden.

532 A 404 response indicates that the indicated "deviceid" is no longer available on this cloud

533 **8.5 Retrieve Specific Resource**

534 **8.5.1 Summary**

535 This request is sent from the "origin Cloud" to the "target Cloud" in order to obtain information on  
536 a specific Resource that is exposed by a Device that is registered for the user that is in scope on  
537 the "target Cloud".

538 Where the Cloud supports the OCF Device to Cloud Services Specification this may be triggered  
539 by reception of a RETRIEVE to a URI exposed by a Link in the Cloud Resource Directory from an  
540 OCF Client.

541 Table 6 provides a summary of the API.

542 **Table 6 – Retrieve Specific Resource API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
GET	/api/v1/devices/{deviceid}/{resourcehref}	Correlation-ID, Accept	200	Response payload as defined by OCF for the target Resource Type

			400, 401, 403, 404	Diagnostic payload containing detailed reason.
			503	Diagnostic payload containing detailed reason.
			504	Retry-After header and diagnostic payload containing the detailed reason.

543

544 **8.5.2 Request and Response Payload**

545 The deviceid in the URI in the request is the same as the "di" Property from "/oic/d" of the target  
546 OCF device. The "resourcehref" in the URI is the same as the "href" Link Parameter for the target  
547 Resource instance.

548 The response payload shall be as defined by OCF for the Resource being received, or as defined  
549 by the vendor if the Resource is a 3<sup>rd</sup> party Resource.

550 The content-type of the response payload received from the target server is honoured; that is the  
551 content and payload as received by the "target Cloud" is proxied unaltered in the response. Thus  
552 for example in the case where the target server is an OCF Device the content type would be  
553 "application/vnd.ocf+cbor".

554 **8.5.3 Responses**

555 A 200 response is provided in a success case. The payload in the response is as defined in  
556 <http://oneiota.org> for the target Resource Type.

557 A 400 response indicates that the request was malformed or badly constructed.

558 A 401 response indicates that the request is unauthorized (likely an invalid bearer token)

559 A 403 response indicates that the requestor is known however the scope of the request is forbidden.

560 A 404 response indicates that the indicated "deviceid" is no longer available on this cloud

561 A 503 response indicates that the service on the "target Cloud" is unavailable for the reason  
562 indicated in the diagnostic payload.

563 A 504 response indicates that the target Device is registered at the "target Cloud", however the  
564 Device itself is unavailable, offline, or otherwise unreachable. The response should include a Retry-  
565 After header containing the time after which the request may be re-attempted. Additional  
566 information is indicated in the diagnostic payload.

567 **8.6 Update a Resource on a Device**

568 **8.6.1 Summary**

569 This request is sent from the "origin Cloud" to the "target Cloud" in order to update information  
570 contained within a specific Resource exposed by a Device that is registered for the user that is in  
571 scope on the "target Cloud".

572 Where the Cloud supports the OCF Device to Cloud Services Specification a request to this API  
573 may be triggered by reception of an UPDATE to a URI exposed by a Link in the Cloud Resource  
574 Directory from an OCF Client.

575 Table 7 provides a summary of the API.

576 **Table 7 – Update Resource API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
POST	/api/v1/devices/{deviceid}/{resourcehref}	payload, Correlation-ID, Accept	200	Optional resource representation
			400, 401, 403, 404, 415	Diagnostic payload containing detailed reason.
			503	Diagnostic payload containing detailed reason.
			504	Retry-After header and diagnostic payload containing the detailed reason..

577

578 **8.6.2 Request and Response Payload**

579 The deviceid in the URI in the request is the same as the "di" Property from /oic/d of the target  
580 OCF device. The resourcehref in the URI is the same as the "href" Link Parameter for the target  
581 Resource instance.

582 The response payload shall be as defined by OCF for the Resource being received, or as defined  
583 by the vendor if the Resource is a 3<sup>rd</sup> party Resource.

584 The Content-Type of the request is defined in an HTTP Content-Type header. In the case that the  
585 request was initiated by another OCF Device, CoAP content-format header value is mapped to the  
586 HTTP Content-Type header to the "target Cloud". If the value is not present, the "target Cloud" will  
587 forward the request as-is. Thus for example in the case where the origin client is an OCF Device  
588 the CoAP content-format option would be "application/vnd.ocf+cbor", which is passed to the "target  
589 Cloud" as an HTTP Content-Type header.

### 590 **8.6.3 Responses**

591 A 200 response is provided in a success case. The payload may optionally contain the  
592 representation of the Resource that was updated.

593 A 401 response indicates that the request is unauthorized (likely an invalid bearer token)

594 A 403 response indicates that the requestor is known however the scope if the request is forbidden.

595 A 415 response indicates unsupported media type specified in the Content-Type header.

596 A 403 response indicates an invalid bearer token.

597 A 404 response indicates that the indicated "deviceid" is no longer available on this cloud

598 A 503 response indicates that the service on the "target Cloud" is unavailable for the reason  
599 indicated in the diagnostic payload.

600 A 504 response indicates that the target Device is registered at the "target Cloud", however the  
601 Device itself is unavailable, offline, or otherwise unreachable. The response should include a Retry-  
602 After header containing the time after which the request may be re-attempted. Additional  
603 information is indicated in the diagnostic payload.

## 604 **9 Events API**

### 605 **9.1 Introduction**

606 The Events API supports the ability for an interested party to subscribe to events and subsequently  
607 receive notifications for those events. The events can be at the Resource level (like a CoAP  
608 observe) or at a more system level (such as for a change in the set of known Devices).

609 The Event's API make use of a webhook mechanism whereby the "target Cloud" can notify the  
610 "origin Cloud" when a new event has occurred on the "target Cloud" or any OCF Device linked with  
611 the "target Cloud". This event stream can be started by sending the initial subscription request to  
612 the "target Cloud", on behalf of the user, specifying "eventtypes", "eventsurl" - the endpoint to which  
613 events are sent and the "signingSecret" - to confidently verify whether requests from the "target  
614 Cloud" are authentic. The mechanism how the "signingSecret" shall be used is specification in the  
615 clause 9.2.

616 Subscription and events are done on behalf of the "target Cloud" user, who linked his account  
617 during the account linking process. As a response to the initial subscription request, a Subscription  
618 ID identifying this user's event stream is returned. The Subscription ID is the unique string of type  
619 UUID, which is created and persisted by the "target Cloud". The created ID is then part of each  
620 event sent to the configured "eventsUrl".

621 The Subscription ID is used also to DELETE this subscription.

622 After the subscription is successful, the state of the subscription is updated and the first event is  
623 sent with the actual state of the model. The subsequent events represent changes which have  
624 occurred since the time of the last state update of the model.

625 Following "origin Cloud's" successful subscription to events of the "target Cloud", the "target Cloud"  
626 can start sending notifications only after it establishes new server-authenticated TLS1.2 connection  
627 to a machine hosting "eventsUrl" as specified by "target Cloud".

628 The Event stream authorized by the "target Cloud" user sends only events related to devices and  
629 system changes he is authorized to see.

630 Whenever a "target Cloud" with active subscriptions has any device state errors, it shall recover  
631 from such errors by sending an event representing the current state of the model to its active  
632 subscribers. The subsequent events represent changes which have occurred since the time of the  
633 last state update of the model.

634 The "origin Cloud" is always informed about subscription cancellation through the Event stream as  
635 well as through the state available within the GET API Subscription endpoint response.  
636 Cancellation can be done by either, the "target Cloud" and the "origin Cloud".

## 637 **9.2 Events Authentication**

638 HMAC signatures are a way to sign the event data using the "signingSecret" that only the "origin  
639 Cloud" and "target Cloud" know. The "signingSecret" shall be created by the "originCloud" and  
640 send within the subscription request as defined in the clause 9.4.1. The "target Cloud" shall after  
641 successful subscription sign each event using HMAC-SHA256 hashing algorithm, following the  
642 formula from the clause 9.2.1. The calculated signature is afterwards attached as the "Event-  
643 Signature" header with each event request sent to the "origin Cloud".

644 The signature is then used by the "origin Cloud" to verify the legitimacy of the source and data  
645 itself. When the event is received by the "origin Cloud" it uses its stored secret and the event to  
646 generate its own HMAC-SHA256 signature using the formula from the clause 9.2.2 to compare with  
647 the value from the "Event-Signature" header.

648 When the secret and event are the same on both sides then the HMAC signature will match. This  
649 match proves the authenticity of the request and data.

650 Detailed flow overview can be visible on the figures A.8.2, A.9.2, A.10.2 and A.11.2.

### 651 **9.2.1 Create Event Signature**

- 652 1) Get the current timestamp in the Unix time format
- 653 2) Concatenate the "Content-Type", the "Event-Type", the "Subscription-ID", the  
654 "Sequence-Number", the "Event-Timestamp" header values and the event body together,  
655 using a colon as a delimiter.
- 656 3) Hash the resulting string, using the "signingSecret" as a key using the HMAC-SHA256  
657 hashing algorithm, and taking the hex digest of the hash.
- 658 4) Include the resulting signature to the "Event-Signature" header of the event and  
659 timestamp to the "Event-Timestamp" header

### 660 **9.2.2 Verify the Event Signature**

- 661 1) Extract the timestamp from the "Event-Timestamp" header and make sure that the  
662 request occurred recently
- 663 2) Concatenate the "Content-Type", the "Event-Type", the "Subscription-ID", the  
664 "Sequence-Number", the "Event-Timestamp" header values and the event body together,  
665 using a colon as a delimiter.
- 666 3) Hash the resulting string, using the "signingSecret" as a key using the HMAC-SHA256  
667 hashing algorithm and take the hex digest of the hash.
- 668 4) Compare the resulting signature to the "Event-Signature" header of the received event

## 669 **9.3 Parameters Supported**

670 Table 8 lists the parameters that may be provided within the Events API.

**Table 8 – Parameters used in the Events API**

Friendly Name	Parameter Name	Location	Description
<b>Accept</b>	Accept	Header	An Accept request HTTP header advertises which content types, expressed as MIME types, the client is able to understand. The resource server then selects one of the proposal and informs the client of its choice with the Content-Type response header. Each event sent to the defined "eventsUrl" is then using this Accepted content type.
<b>Correlation ID</b>	Correlation-ID	Header	A Correlation ID, also known as a Transit ID, is a unique identifier value that is attached to requests and messages that allows reference to a particular transaction or event chain.
<b>Content Type</b>	Content-Type	Header	The Content-Type header is used to indicate the media type of the resource. In responses, a Content-Type header tells the client what the content type of the returned content actually is.

672 **9.4 Events API subscription and notification request and response payload definitions**673 **9.4.1 Subscription request**

674 B.1 OpenAPI 2.0 Definition (/definitions/SubscribeRequest) provides a definition of the payload  
 675 contained within the subscription request. The Properties that are contained with the schema  
 676 definition are further defined in Table 9.

--

**Table 9 – Subscription Request Properties**

Property Name	Value type	Mandatory	Description
<b>eventsUrl</b>	URI	Y	URI to which events are to be sent
<b>eventTypes</b>	array of enum	Y	Event type(s) for which the subscription is targeted

<b>signingSecret</b>	String of length 32	Y	Secret used to create HMAC signature for each event
----------------------	---------------------	---	---

678

679 Figure 3 is an example of such a payload.

```
{
  "eventsurl": https://mynotificationuri,
  "eventtypes": ["resource-contentchanged"],
  "signingSecret": "DVDUEBe5nciVSXU85BPxAjSsHenTzWY"
}
```

680 **Figure 3 – Subscription Request Example**

681 **9.4.2 Subscription response**

682 The definition of the response response to a subscription request is in clause B.1 OpenAPI 2.0  
 683 Definition (/definitions/SubscribeResponse). The Properties that are contained with the schema  
 684 definition are further defined in Table 10.

685 **Table 10 – Subscription Response Properties**

Property Name	Value type	Mandatory	Description
<b>subscriptionId</b>	uuid	Y	Identity of the subscription. May be mapped from other protocols if a unique identifier exists. Note this cannot be mapped from a CoAP Token as the Token in CoAP is Client-local in scope (i.e. not guaranteed unique beyond the Client issuing the request).

686

687 Figure 4 is an example of such a payload.

```
{
  "subscriptionId": "1eeb465c-5e8d-4305-a366-bbf035fff671"
}
```



688

### Figure 4 – Subscription Response Example Payload

#### 9.4.3 Notification request population

690 All events sent to the "eventsUrl" defined during the subscription are populated with event metadata.  
691 Event metadata is transferred to the "origin Cloud" as HTTP Headers together with the event  
692 payload. All event metadata is required.

693

**Table 11 – Event HTTP Headers description**

Event metadata	Description
<b>Correlation-ID</b>	A Correlation ID, also known as a Transit ID, is a unique identifier value that is attached to requests and messages that allows reference to a particular transaction or event chain.
<b>Content-Type</b>	Indicates the media type of the event payload
<b>Event-Type</b>	Type of the event
<b>Subscription-ID</b>	Subscription identifier for which this event is being sent
<b>Sequence-Number</b>	Sequence number of the event; first event starting with number 0
<b>Event-Timestamp</b>	Time when the event occurred in standard Unix time format
<b>Event-Signature</b>	HMAC-SHA256 signature proving the authenticity of the request and data. See 9.2 Events Authentication

694

695 The format of the payload in a notification request depends on the event for which the subscription  
696 was created.

697 Table 12 lists the format of the payload per "eventType" that may be received.

698

**Table 12 – Event type to content format map**

Event Type	Event API Endpoint	Payload
<b>subscription_cancelled</b>	/api/v1/devices/subscriptions /api/v1/devices/{deviceid}/subscriptions	Not present

	/api/v1/devices/{deviceid}/{resourcehref}/subscriptions	
<b>devices_registered</b>	/api/v1/devices/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/DevicesRegisteredEvent
<b>devices_unregister ed</b>	/api/v1/devices/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/DevicesUnregisteredEvent
<b>devices_online</b>	/api/v1/devices/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/DevicesOnlineEvent
<b>devices_offline</b>	/api/v1/devices/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/DevicesOfflineEvent
<b>resource_contentc hanged</b>	/api/v1/devices/{deviceid}/{resourcehref}/subscriptions	Payload as received from the Device providing the representation
<b>resources_publish ed</b>	/api/v1/devices/{deviceid}/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/ResourcePublishedEvent
<b>resources_unpubli shed</b>	/api/v1/devices/{deviceid}/subscriptions	See B.1 OpenAPI 2.0 Definition - /definitions/ResourceUnpublishedEvent

699

700 **9.5 Subscribe and Unsubscribe to Devices level events, retrieval of the subscription**  
701 **details**

702 **9.5.1 Summary**

703 Note: this request is sent from the "origin Cloud" to the "target Cloud". This API is used when  
704 changes to the set of Devices that are exposed are required to be notified.

705 A POST request establishes the subscription; a GET provides details about the subscription; a  
706 DELETE request removes the subscription.

707 Table 13 provides a summary of the API.

**Table 13 – Subscription to /devices API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
<b>POST</b>	/api/v1/devices/subscriptions	Correlation-ID, Accept, Content-Type	201	See B.1 OpenAPI 2.0 Definition - /definitions /Subscribe Response
			400, 401, 403	
<b>DELETE</b>	/api/v1/devices/subscriptions/{subscriptionid}	Correlation-ID	202	
			400, 401, 403, 404	

709 **9.5.2 Request and Response Payload**

710 The "subscriptionid" in the URI for the DELETE case is the "subscriptionid" that is returned in the  
711 response of the subscription POST request.

712 The response payload for the subscription POST request shall be as defined in clause 9.4.2

713 **9.5.3 Responses**

714 A 201 response is provided in a success case.

715 A 202 response indicates that the subscription was marked for cancellation.

716 A 401 response indicates that the request is unauthorized (likely an invalid bearer token).

717 A 400 response indicates that the request was malformed or badly constructed.

718 A 403 response indicates an invalid bearer token.

719 A 404 response indicates the subscription was not found.

720 **9.6 Subscribe and Unsubscribe to Device level events, retrieval of the subscription  
721 details**

722 **9.6.1 Summary**

723 Note: this request is sent from the "origin Cloud" to the "target Cloud". This API is used when the  
724 "origin Cloud" wants to effectively receive an "event stream" of notifications from all observable  
725 Resources that exist for a specific Device on the "target Cloud".

726 A POST request establishes the subscription; a GET provides details about the subscription; a  
727 DELETE request removes the subscription

728 Table 14 provides a summary of the API.

**Table 14 – Subscription to Single Device API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
<b>POST</b>	/api/v1/devices/{ <b>deviceid</b> }/subscriptions	Correlation-ID, Accept, Content-Type	201	See B.1 OpenAPI 2.0 Definition - /definitions /Subscribe Response
			400, 401, 403, 404	
<b>DELETE</b>	/api/v1/devices/{ <b>deviceid</b> }/subscriptions/{ <b>subscriptionid</b> }	Correlation-ID	202	
			400, 401, 403, 404	

### 730 9.6.2 Request and Response Payload

731 The "deviceid" in the request URI is the same as the "di" Property from "/oic/d" of the target OCF  
732 device.

733 The "subscriptionid" in the URI in the DELETE case is the "subscriptionid" that is returned in the  
734 response of the subscription POST request.

735 The response payload for the subscription POST request shall be as defined in clause 9.4.2

### 736 9.6.3 Responses

737 A 201 response is provided in a success case.

738 A 202 response indicates that the subscription was marked for cancellation.

739 A 400 response indicates that the request was malformed or badly constructed.

740 A 401 response indicates that the request is unauthorized (likely an invalid bearer token).

741 A 403 response indicates an invalid bearer token.

742 A 404 response indicates the device / subscription was not found.

743

## 744 9.7 Subscribe and Unsubscribe to Resource level events, retrieval of the subscription 745 details

### 746 9.7.1 Summary

747 Note: this request is sent from the "origin Cloud" to the "target Cloud". This API is used when the  
748 "origin Cloud" wants to receive notifications from a specific Resource that exists on a specific  
749 Device on the "target Cloud".

750 A POST request establishes the subscription; a GET provides details about the subscription; a  
 751 DELETE request removes the subscription.

752 Table 15 provides a summary of the API.

753 **Table 15 – Subscription to Resource API Summary**

HTTP Request Type	URI	Parameters	Response Code	Response Payload
POST	/api/v1/devices/{deviceid}/{resourcehref}/subscriptions	Correlation-ID, Accept, Content-Type	201	See B.1 OpenAPI 2.0 Definition - /definitions /Subscribe Response
			400, 401, 403, 404	
DELETE	/api/v1/devices/{deviceid}/{resourcehref}/subscriptions/{subscriptionid}	Correlation-ID	202	
			400, 401, 403, 404	

754 **9.7.2 Request and Response Payload**

755 The "deviceid" in the URI in the request is the same as the "di" Property from /oic/d of the target  
 756 OCF device.

757 The "resourceHref" in the URI is the same as the "href" Link Parameter for the target Resource  
 758 instance.

759 The "subscriptionid" in the URI in the DELETE case is the "subscriptionid" that is returned in the  
 760 response of the subscription POST request.

761 The response payload for the subscription POST request shall be as defined in clause 9.4.2

762 **9.7.3 Responses**

763 A 201 response is provided in a success case.

764 A 202 response indicates that the subscription was marked for cancellation.

765 A 400 response indicates that the request was malformed or badly constructed.

766 A 401 response indicates that the request is unauthorized (likely an invalid bearer token).

767 A 403 response indicates an invalid bearer token.

768 A 404 response indicates the requested device / resource / subscription was not found.

769 **9.8 Notification of /devices level events**

770 **9.8.1 Summary**

771 Note: this request is sent from the "target Cloud" to the "origin Cloud".

772 Table 16 provides a summary of the API.

773 **Table 16 – Notification of /devices API Summary**

HTTP Request Type	Request	URI	Parameters	Response Code	Response Payload
POST		/{eventsUrl}	Correlation-ID, Content-Type Event-Type, Subscription-ID, Sequence- Number, Event-Signature	200	
				400, 410	

774 **9.8.2 Request and Response Payload**

775 The "eventsUrl" in the URI is the value of the "eventsUrl" Property that was provided in the  
776 subscription request.

777 The payload in the notification request shall be the content of an event in a specific type based on  
778 what event the "origin Cloud" subscribes to as defined in clause 9.4.3

779 **9.8.3 Responses**

780 A 200 response is provided in a success case.

781 A 400 response indicates that the request was malformed or badly constructed.

782 A 410 response indicates that the subscription identified by the Subscription-ID header is no more  
783 in demand and shall be canceled

784 **9.9 Notification of Device level events**

785 **9.9.1 Summary**

786 Note: this request is sent from the "target Cloud" to the "origin Cloud".

787 Table 17 provides a summary of the API.

788 **Table 17 – Notification of Single Device API Summary**

HTTP Request Type	Request	URI	Parameters	Response Code	Response Payload
POST		/{eventsUrl}	Correlation-ID, Content-Type Event-Type, Subscription-ID, Sequence- Number, Event-Signature	200	
				400, 410	

789 **9.9.2 Request and Response Payload**

790 The "eventsUrl" in the URI is the value of the "eventsUrl" Property that was provided in the  
791 subscription request.

792 The payload in the notification request shall be the content of an event in a specific type based on  
793 what event the "origin Cloud" subscribes to as defined in 9.4.3

794 **9.9.3 Responses**

795 A 200 response is provided in a success case.

796 A 400 response indicates that the request was malformed or badly constructed.

797 A 410 response indicates that the subscription identified by the Subscription-ID header is no more  
798 in demand and shall be canceled

799 **9.10 Notification of Resource level events**

800 **9.10.1 Summary**

801 Note: this request is sent from the "target Cloud" to the "origin Cloud".

802 Table 18 provides a summary of the API.

803 **Table 18 – Notification of Resource API Summary**

HTTP Type	Request	URI	Parameters	Response Code	Response Payload
POST		/{eventsUrl}	Correlation-ID, Content-Type Event-Type, Subscription-ID, Sequence- Number, Event-Signature	200	
				400, 410	

804 **9.10.2 Request and Response Payload**

805 The "eventsUrl" in the URI is the value of the "eventsUrl" Property that was provided in the  
806 subscription request.

807 The payload in the notification request shall be the content of an event in a specific type based on  
808 what event the "origin Cloud" subscribes to as defined in clause 9.4.3

809 **9.10.3 Responses**

810 A 200 response is provided in a success case.

811 A 400 response indicates that the request was malformed or badly constructed.

812 A 410 response indicates that the subscription identified by the Subscription-ID header is no more  
813 in demand and shall be canceled

814  
815

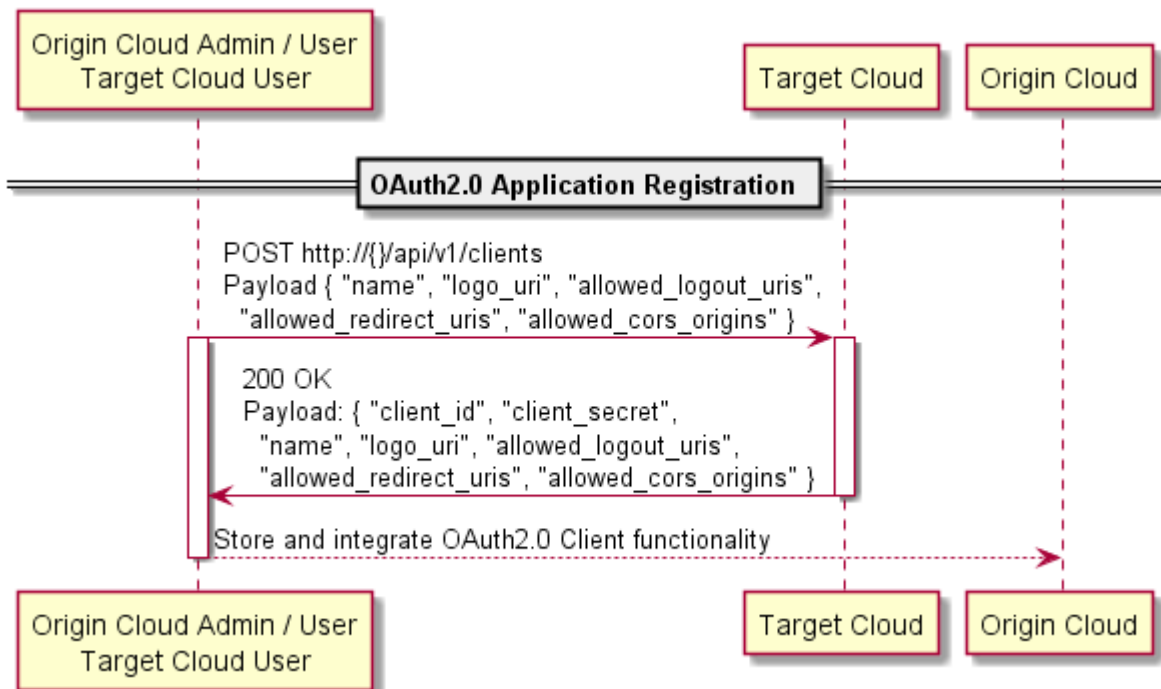
## Annex A Representative Flows

### 816 A.1 Introduction

817 The flows illustrate use of the OCF Cloud API for Cloud Services using OCF Devices as the  
818 endpoints and OCF Clouds as the two Clouds that are invoking/acting as endpoints. Note that this  
819 is for example use only and the API does not force this setup, which means non-OCF clouds with  
820 non-OCF devices can also use the API for interworking with other vendor's clouds.

### 821 A.2 OAuth2.0 Application Registration

822 Figure A.1 provides an example flow showing the registration of the OAuth Client / Application for  
823 the "origin Cloud" on the "target Cloud".



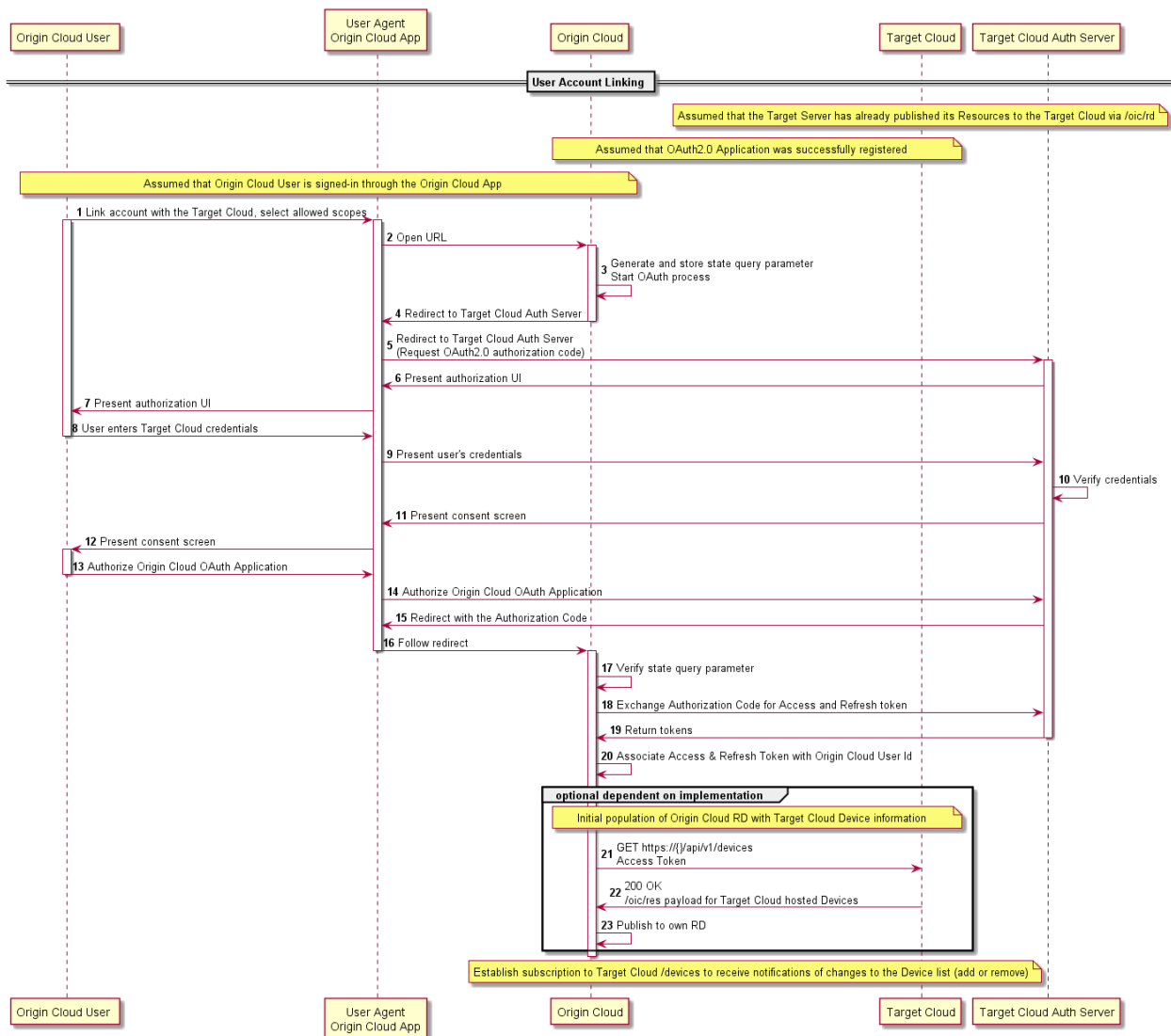
824  
825

826 **Figure A.1 – Establish Business Relationship Example Flow**

### 827 A.3 Account Linking

828 Figure A.2 provides an example flow of the account linking for a particular user.





829

830

**Figure A.2 – Initial Association Example Flow**

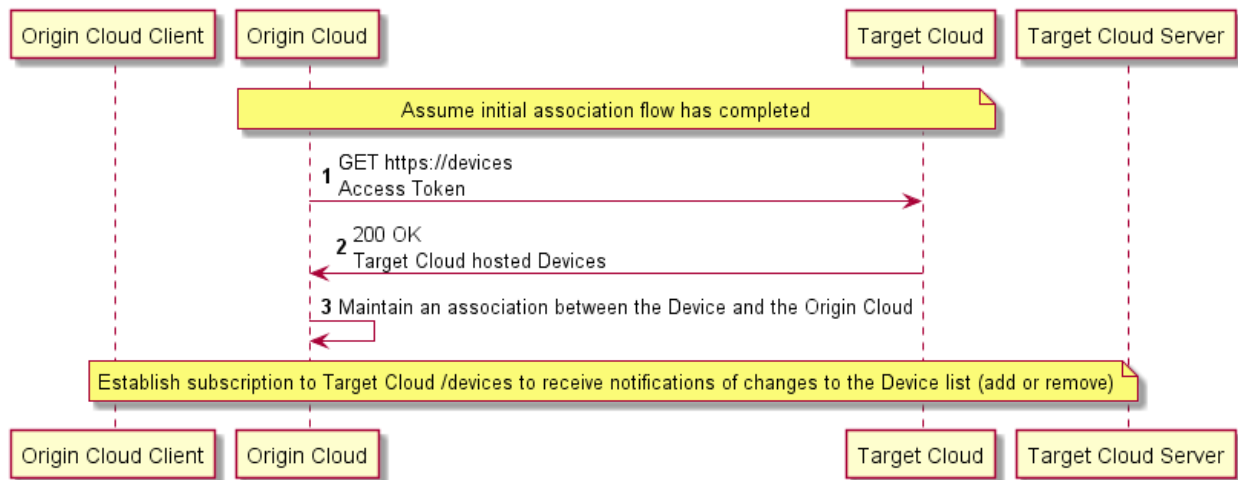
831 **A.4 Retrieval of all Devices**

832 **A.4.1 Summary**

833 The "origin Cloud" requests all Devices associated with a user (defined by the provided token).  
 834 This may be invoked following account linking in order to retrieve the set of Devices for the user.

835 **A.4.2 Flow**

836 Figure A.3 provides an example flow for the retrieval of all Devices.



837

838

**Figure A.3 – Retrieve All Devices Example Flow**

839 **A.4.3 Flow Description**

840 Table A.1 explains each element in the above sequence diagram

841 **Table A.1 – Retrieve all Devices Flow Summary**

Number	Description
1	Cloud requests all Devices given by the scope in the bearer token that was obtained via OAuth.
2	Response is an array of Device information ( Properties that are defined in /oic/d that are pertinent to Cloud functionality and Device status).
3	Cloud maintains an association between the Device and the host Cloud.

842

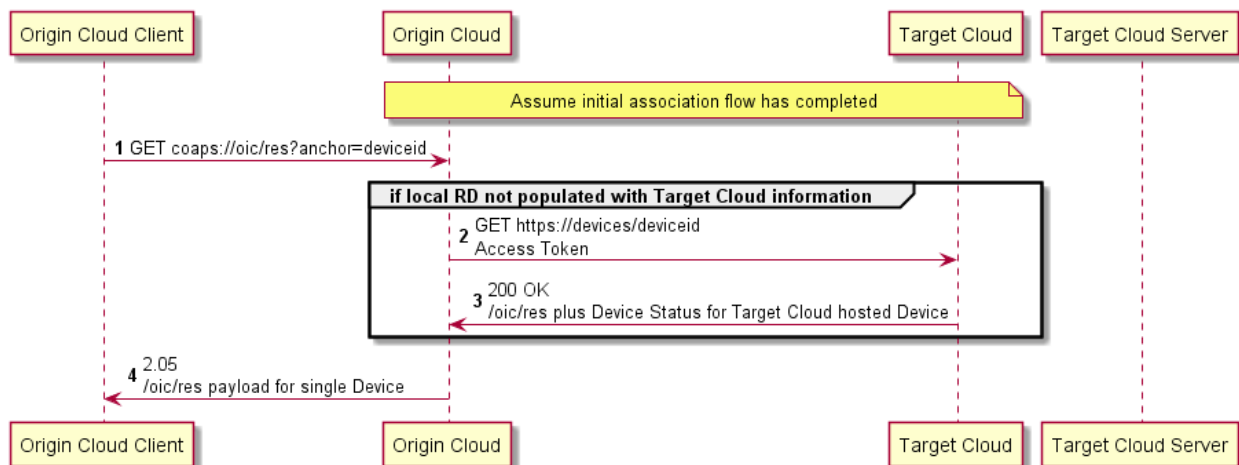
843 **A.5 Retrieval of a single Device**

844 **A.5.1 Summary**

845 The "origin Cloud" requests information for a single, specific Device associated with a user (defined  
 846 by the provided token). This may be invoked by the "origin Cloud" receiving a retrieve request from  
 847 a connected Client.

848 **A.5.2 Flow**

849 Figure A.4 provides an example flow for the retrieval of a single Device.



850  
851 **Figure A.4 – Retrieve Single Device Example Flow**

852 **A.5.3 Flow Description**

853 Table A.2 explains each element in the above sequence diagram

854 **Table A.2 – Retrieve single Device Flow Summary**

Number	Description
1	[OCF Device to Cloud] OCF Client role Device requests /oic/res from the Cloud for a specific anchor (device id).
2	[Assuming that the information hasn't been cached by the Cloud] For the instance of /oic/sec/account that exists for the Device the Cloud does a GET /devices/{deviceid} to the Cloud identified by the cloudid in /oic/sec/account. {deviceid} is also taken from /oic/sec/account.
3	Response is the Device information as well as an array of Links. The "href" in each Link will be of the form "/deviceid/resourcehref".
4	Response payload.

855 **A.6 Retrieval of a single Resource**

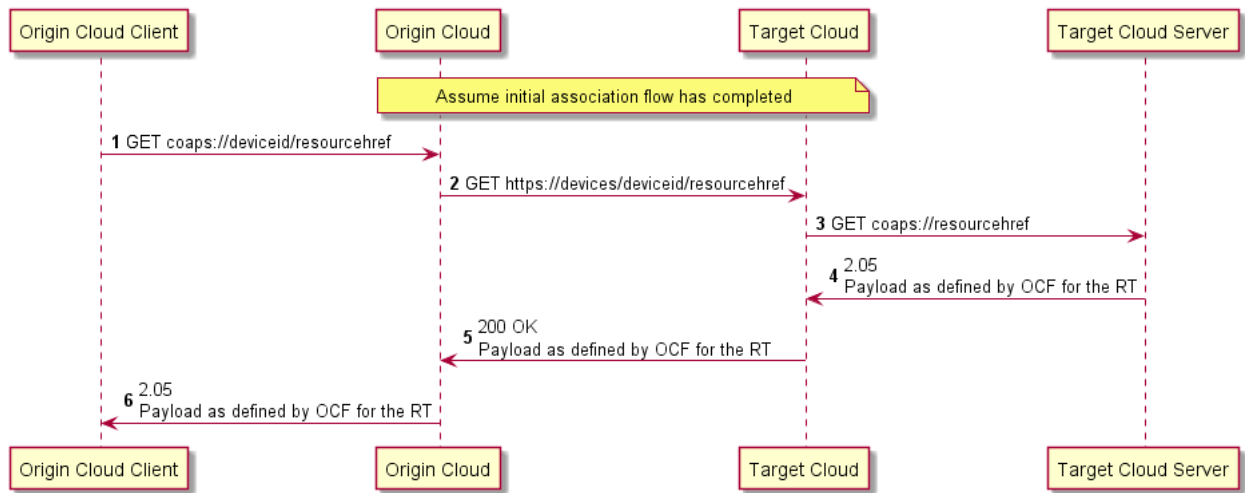
856 **A.6.1 Summary**

857 The "origin Cloud" requests information for a single, specific Resource exposed by a Device  
858 associated with a user (defined by the provided token). This may be invoked by the "origin Cloud"  
859 receiving a retrieve request from a connected Client.

860 **A.6.2 Flows**

861 **A.6.2.1 Success Path**

862 Figure A.5 provides an example flow for the retrieval of a single Resource.



863  
864

**Figure A.5 – Retrieve Resource (Success) Example Flow**

865 **A.6.2.2 Success Path Flow Description**

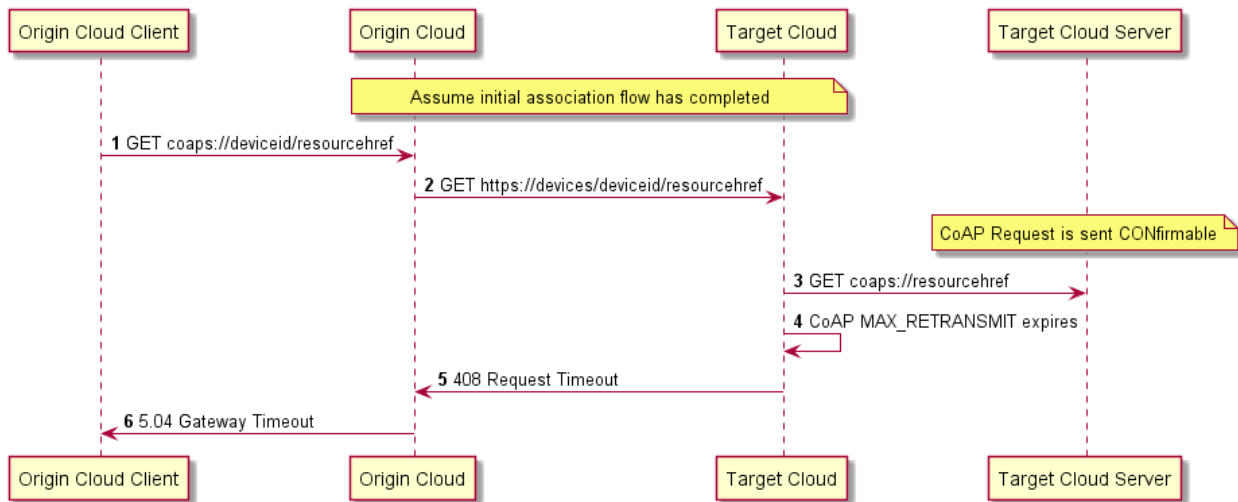
866 Table A.3 explains each element in the above sequence diagram

867 **Table A.3 – Retrieve single Resource Flow Summary**

Number	Description
1	[OCF Device to Cloud] OCF Client role Device requests a Resource from the Cloud using the "href" exposed in the /oic/res response. This will be of the form "/deviceid/resourcehref"
2	[Assuming that the resource representation hasn't been cached by the Cloud] Cloud identifies the host Cloud for the Resource via the instance of /oic/sec/account for the "deviceid". The request is then effectively proxied to the "target Cloud" via a GET /devices/{deviceid}/{resourcehref}. Any query parameters received over CoAP are included in the URI unaltered.
3	[OCF Device to Cloud] "target Cloud" identifies the TLS connection to the end Device via the {deviceid} and proxies the request.
4	Standard OCF response
5	Success path response including the response payload as received for the target Resource
6	Standard OCF response

868 **A.6.2.3 Device is Temporarily Unavailable**

869 Figure A.6 illustrates the case where the Device is temporarily unavailable.



870  
871 **Figure A.6 – Retrieve Resource (Timeout) Example Flow**

872 **A.7 Update of a single Resource**

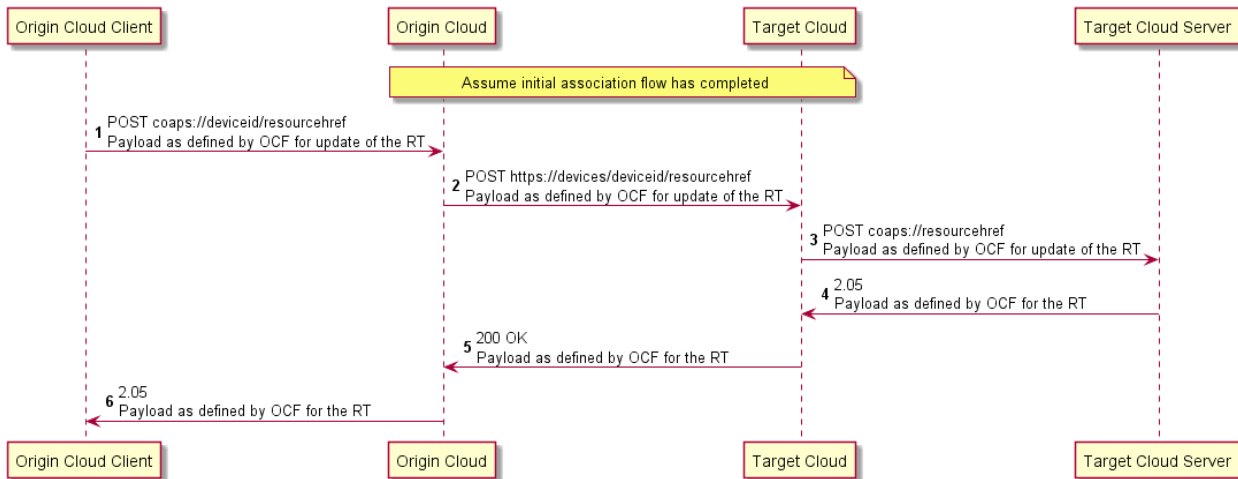
873 **A.7.1 Summary**

874 The "origin Cloud" updates information for a single, specific Device associated with a user (defined  
875 by the provided token). This may be invoked by the "origin Cloud" receiving an update request from  
876 a connected Client.

877 **A.7.2 Flows**

878 **A.7.2.1 Success Path**

879 Figure A.7 provides an example flow for the updating of a single Resource.



880  
881 **Figure A.7 – Update Resource (Success) Example Flow**

882 **A.7.2.2 Success Path Flow Description**

883 Table A.4 explains each element in the above sequence diagram

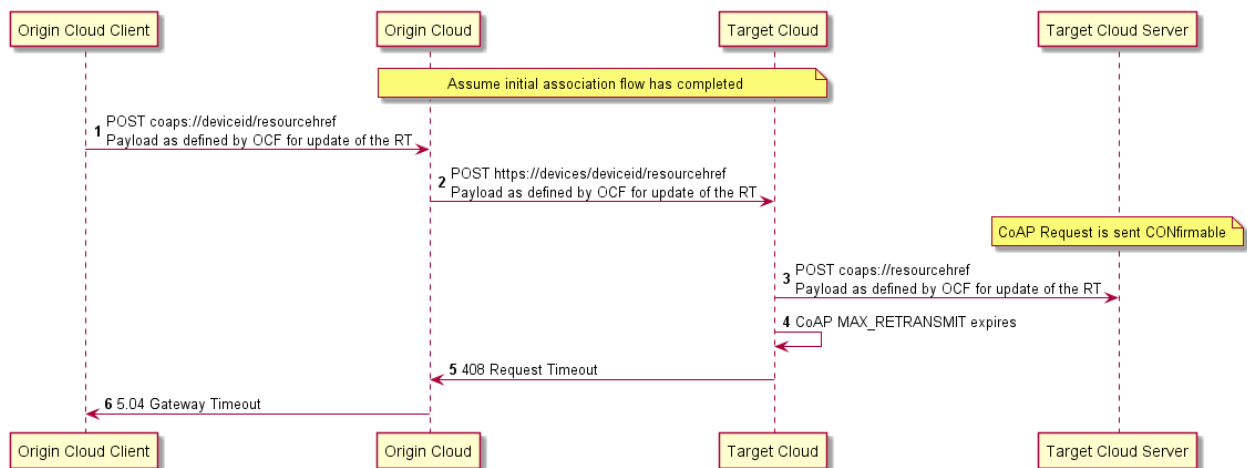
884 **Table A.4 – Update single Resource Flow Summary**

Number	Description
--------	-------------

1	[OCF Device to Cloud] OCF Client role Device requests a Resource from the Cloud using the "href" exposed in the /oic/res response. This will be of the form "/deviceid/resourcehref"
2	Cloud identifies the host Cloud for the Resource via the instance of /oic/sec/account for the "deviceid". The request is then effectively proxied to the "target Cloud" via a POST /devices/{deviceid}/{resourcehref} including the payload from the original request. Any query parameters received over CoAP are included in the URI unaltered.
3	[OCF Device to Cloud] "target Cloud" identifies the TLS connection to the end Device via the {deviceid} and proxies the request.
4	Standard OCF response
5	Success path response including the response payload as received for the target Resource
6	Standard OCF response

885 **A.7.2.3 Device is Temporarily Unavailable**

886 Figure A.8 illustrates the case where the Device is temporarily unavailable.



887  
888

889 **Figure A.8 – Update Resource (Timeout) Example Flow**

890 **A.8 Establishment of new subscription request**

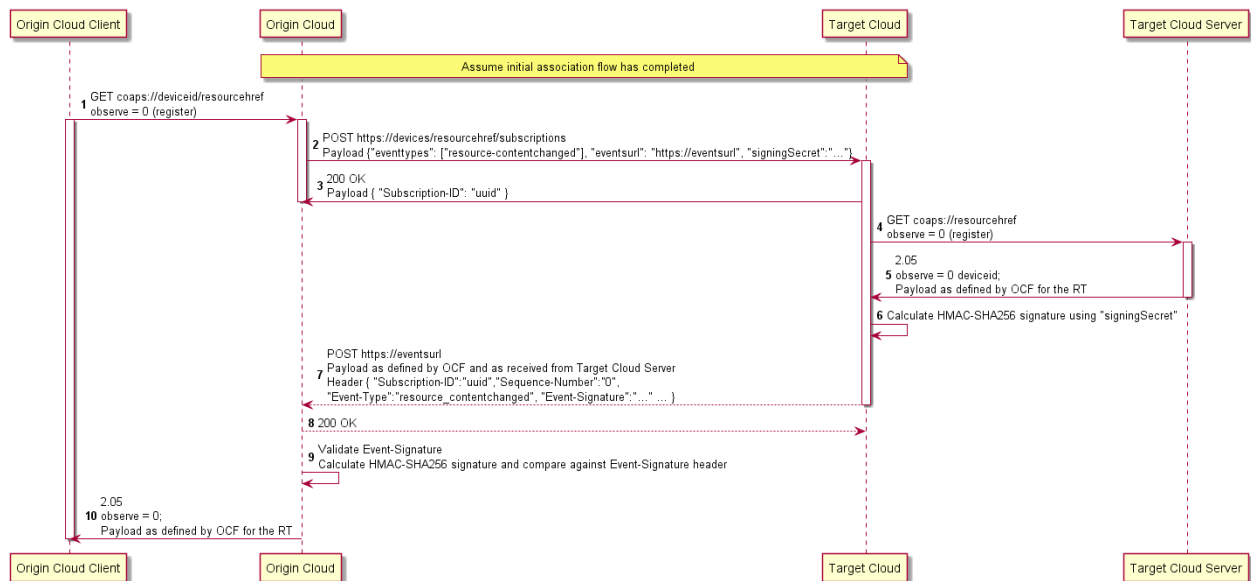
891 **A.8.1 Summary**

892 The "origin Cloud" requests the establishment of an observe relationship with a single, specific  
893 Resource on a Device associated with a user (defined by the provided token). This may be invoked  
894 by "origin Cloud" receiving a retrieve request containing an observe option from a connected Client.

895 **A.8.2 Flows**

896 Figure A.9 provides an example flow for the establishment of a subscription to the  
897 "resource\_contentchanged" event for a specific Resource.

898



899  
900 **Figure A.9 – Observe Establishment Example Flow**

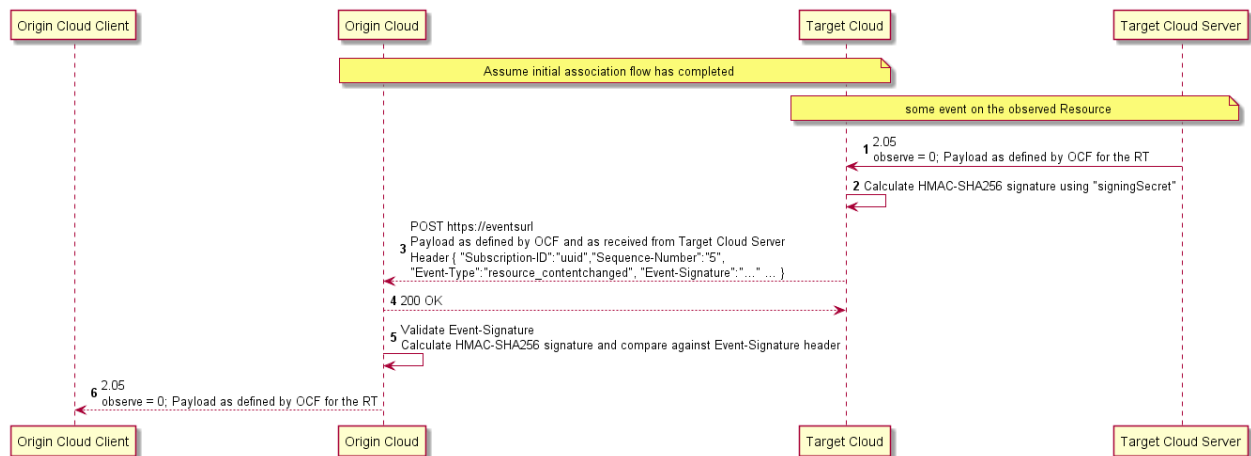
901 **A.9 Event generated for a subscription**

902 **A.9.1 Summary**

903 An event occurs for a Resource with which the "origin Cloud" has established a subscription/event relationship. This may be invoked by the target end Device being updated.

905 **A.9.2 Flows**

906 Figure A.10 provides an example flow for the handling of a generated "resource\_contentchanged" event.



908  
909 **Figure A.10 – "resource\_contentchanged" Event Example Flow**

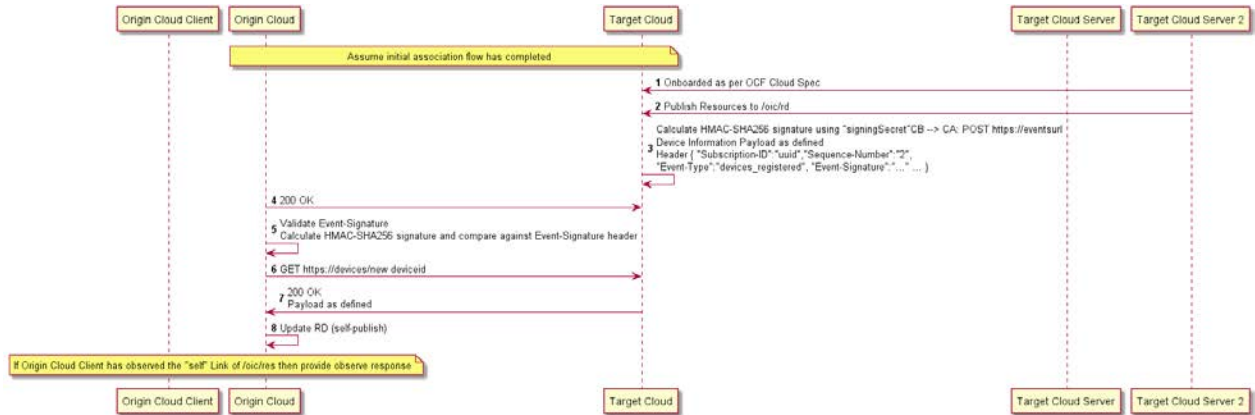
910 **A.10 Addition of new registration**

911 **A.10.1 Summary**

912 The "origin Cloud" has a priori established a subscription/event relationship with the set of Devices associated with a user exposed by "target Cloud". The user then registers a new Device with "target Cloud".

915 **A.10.2 Flows**

916 Figure A.11 provides an example flow for the generation of a notification (event) when a new Device  
 917 is registered.



918

919

**Figure A.11 – Addition of new registered Device example flow**

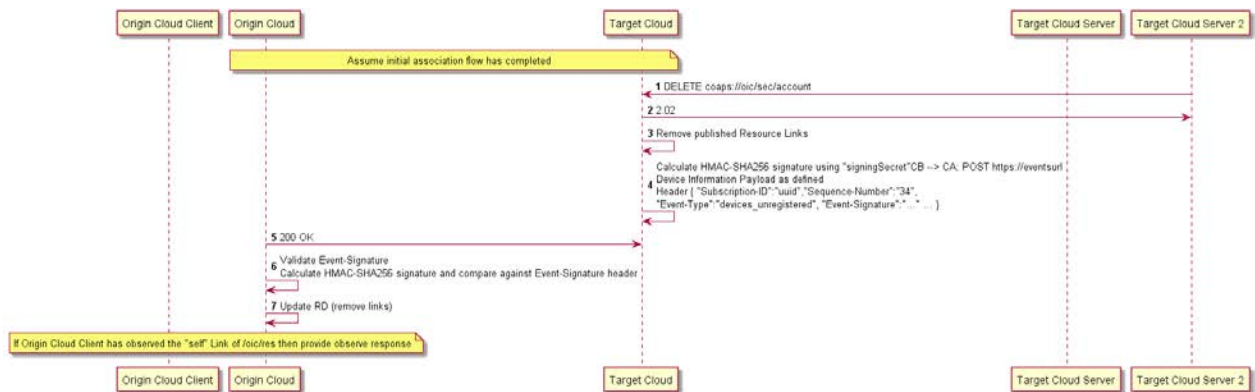
920 **A.11 Removal of existing device registration**

921 **A.11.1 Summary**

922 The "origin Cloud" has a priori established a subscription/event relationship with the set of Devices  
 923 associated with a user exposed by "target Cloud". The user then removes a Device from "target  
 924 Cloud".

925 **A.11.2 Flows**

926 Figure A.12 provides an example flow for the generation of a notification (event) when a Device is  
 927 removed.



928

929

**Figure A.12 – Removal of existing registration example flow**



## Annex B Open API Definition

930  
931  
932

### 933 B.1 OCF Cloud API for Cloud Services

#### 934 B.1.1 Supported APIs

##### 935 B.1.1.1 /api/v1/devices/subscriptions

936 Subscribe to devices events by providing `eventTypes` you're interested in and `eventsUrl`  
937 endpoint where events will be sent to as defined. Successful response contains `subscriptionId`  
938 which identifies registered subscription and is part of each event. First event for each registered  
939 event type is received immediately after subscription and contains actual state of the resource,  
940 followed by new events in case of any change.

941 **\*\*Supported events\*\*** and required scopes  
942 - `devices\_registered`: `r:deviceinformation:\*`  
943 - `devices\_unregistered`: `r:deviceinformation:\*`  
944 - `devices\_online`: `r:deviceinformation:\*`  
945 - `devices\_offline`: `r:deviceinformation:\*`  
946  
947

##### 948 B.1.1.2 /api/v1/devices

949 Get all devices which are signed up to the OCF Cloud - either `online` or `offline`. Devices which  
950 are `online` are signed in to the system and are accessible. Offline devices are signed up to the  
951 system, but currently disconnected.  
952

##### 953 B.1.1.3 /api/v1/devices/{deviceId}/subscriptions/{subscriptionId}

954 Cancel subscription identified by the id returned in a response for subscription.  
955

##### 956 B.1.1.4 /{eventsUrl}

957 Events endpoint provided during subscription where events specified in the subscription will be  
958 sent to as defined per event type. Confirmation of each event sent to the `eventsUrl` endpoint is  
959 required with `2xx` success code. Events you will receive based on event type you're subscribed  
960 to are:

961 - `subscription\_canceled`: `SubscriptionCanceledEvent`  
962 - `devices\_registered`: `DevicesRegisteredEvent`  
963 - `devices\_unregistered`: `DevicesUnregisteredEvent`  
964 - `resources\_published`: `ResourcesPublishedEvent`  
965 - `resources\_unpublished`: `ResourcesUnpublishedEvent`  
966 - `devices\_online`: `DevicesOnlineEvent`  
967 - `devices\_offline`: `DevicesOfflineEvent`  
968 - `resource\_contentchanged`: `ResourceContentChangedEvent`

##### 969 B.1.1.5 /api/v1/devices/{deviceId}/{resourceLinkHref}/subscriptions/{subscriptionId}

970 Cancel subscription identified by the id returned in a response for subscription.  
971

##### 972 B.1.1.6 /api/v1/devices/subscriptions/{subscriptionId}

973 Cancel subscription identified by the id returned in a response for subscription.  
974

975 **B.1.1.7 /api/v1/devices/{deviceId}/subscriptions**

976 Subscribe to device events by providing `eventTypes` you're interested in and `eventsUrl`  
977 endpoint where events will be sent to as defined. Successful response contains `subscriptionId`  
978 which identifies registered subscription and is part of each event. First event for each registered  
979 event type is received immediately after subscription and contains actual state of the resource,  
980 followed by new events in case of any change.

981  
982 **\*\*Supported events\*\*** and required scopes  
983 - `resources\_published`: `r:deviceinformation:\*`  
984 - `resources\_unpublished`: `r:deviceinformation:\*`  
985

986 **B.1.1.8 /api/v1/devices/{deviceId}/{resourceLinkHref}**

987 `#/responses/GatewayTimeout`

988 **B.1.1.9 /api/v1/devices/{deviceId}**

989 Device requested with content=all query parameter

990 **B.1.1.10 /api/v1/devices/{deviceId}/{resourceLinkHref}/subscriptions**

991 Subscribe to resource events by providing `eventTypes` you're interested in and `eventsUrl`  
992 endpoint where events will be sent to as defined. Successful response contains `subscriptionId`  
993 which identifies registered subscription and is part of each event. First event for each registered  
994 event type is received immediately after subscription and contains actual state of the resource,  
995 followed by new events in case of any change.

996  
997 **\*\*Supported events\*\*** and required scopes  
998 - `resource\_contentchanged`: `r:resources:\*`

999 **B.1.2 OpenAPI 2.0 definition**

```
1000 {
1001   "swagger": "2.0",
1002   "info": {
1003     "title": "OCF Cloud API for Cloud Services",
1004     "version": "0.0.3-20190828",
1005     "license": {
1006       "name": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved.",
1007       "x-description": "Redistribution and use in source and binary forms, with or without
1008 modification, are permitted provided that the following conditions are met:\n      1.
1009 Redistributions of source code must retain the above copyright notice, this list of conditions and
1010 the following disclaimer.\n      2. Redistributions in binary form must reproduce the above
1011 copyright notice, this list of conditions and the following disclaimer in the documentation and/or
1012 other materials provided with the distribution.\n\n      THIS SOFTWARE IS PROVIDED BY THE Open
1013 Connectivity Foundation, INC. \\"AS IS\\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
1014 LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR
1015 WARRANTIES OF NON-INFRINGEMENT, ARE DISCLAIMED.\n\n      IN NO EVENT SHALL THE Open Connectivity
1016 Foundation, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
1017 OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
1018 SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n\n      HOWEVER CAUSED AND ON
1019 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
1020 OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
1021 SUCH DAMAGE.\n"
1022     }
1023   },
1024   "host": "api.example.com",
1025   "schemes": [
1026     "https"
1027   ],
1028   "tags": [
1029     {
1030       "name": "Devices",
1031       "description": "Basic information about devices"
```

```

1032     },
1033     {
1034         "name": "Resources",
1035         "description": "Read or change the configuration of the device"
1036     },
1037     {
1038         "name": "Events",
1039         "description": "Be notified about changes occurring on the device"
1040     }
1041 ],
1042 "paths": {
1043     "/api/v1/devices": {
1044         "parameters": [
1045             {
1046                 "$ref": "#/parameters/CorrelationId"
1047             },
1048             {
1049                 "$ref": "#/parameters/Accept"
1050             },
1051             {
1052                 "$ref": "#/parameters/BatchFormat"
1053             }
1054         ],
1055         "get": {
1056             "tags": [
1057                 "Devices"
1058             ],
1059             "summary": "Get all devices",
1060             "description": "Get all devices which are signed up to the OCF Cloud - either `online` or
1061 `offline`. Devices which are `online` are signed in to the system and are accessible. Offline
1062 devices are signed up to the system, but currently disconnected.\n",
1063             "produces": [
1064                 "application/json"
1065             ],
1066             "responses": {
1067                 "200": {
1068                     "description": "An array of devices",
1069                     "schema": {
1070                         "type": "array",
1071                         "items": {
1072                             "$ref": "#/definitions/Device"
1073                         }
1074                     }
1075                 },
1076                 "400": {
1077                     "$ref": "#/responses/BadRequest"
1078                 },
1079                 "401": {
1080                     "$ref": "#/responses/Unauthorized"
1081                 },
1082                 "403": {
1083                     "$ref": "#/responses/Forbidden"
1084                 }
1085             },
1086             "security": [
1087                 {
1088                     "oauth2": [
1089                         "r:deviceinformation:*"
1090                     ]
1091                 }
1092             ]
1093         }
1094     },
1095     "/api/v1/devices/subscriptions": {
1096         "parameters": [
1097             {
1098                 "$ref": "#/parameters/CorrelationId"
1099             },
1100             {
1101                 "$ref": "#/parameters/Accept"
1102             },

```

```

1103     {
1104         "$ref": "#/parameters/ContentType"
1105     }
1106 ],
1107 "post": {
1108     "tags": [
1109         "Events"
1110     ],
1111     "summary": "Subscribe to events against the set of devices",
1112     "description": "Subscribe to devices events by providing `eventTypes` you're interested in
1113 and `eventsUrl` endpoint where events will be sent to as defined. Successful response contains
1114 `subscriptionId` which identifies registered subscription and is part of each event. First event for
1115 each registered event type is received immediately after subscription and contains actual state of
1116 the resource, followed by new events in case of any change.\n\n**Supported events** and required
1117 scopes\n- `devices_registered`: `r:deviceinformation:*`\n- `devices_unregistered`:
1118 `r:deviceinformation:*`\n- `devices_online`: `r:deviceinformation:*`\n- `devices_offline`:
1119 `r:deviceinformation:*`\n",
1120     "parameters": [
1121         {
1122             "$ref": "#/parameters/SubscribeRequest"
1123         }
1124     ],
1125     "consumes": [
1126         "application/json"
1127     ],
1128     "produces": [
1129         "application/json"
1130     ],
1131     "responses": {
1132         "201": {
1133             "$ref": "#/definitions/SubscribeResponse"
1134         },
1135         "400": {
1136             "$ref": "#/responses/BadRequest"
1137         },
1138         "401": {
1139             "$ref": "#/responses/Unauthorized"
1140         },
1141         "403": {
1142             "$ref": "#/responses/Forbidden"
1143         }
1144     },
1145     "security": [
1146         {
1147             "oauth2": [
1148                 "w:subscriptions:*",
1149                 "r:deviceinformation:*"
1150             ]
1151         }
1152     ]
1153 },
1154 },
1155 "/api/v1/devices/subscriptions/{subscriptionId}": {
1156     "parameters": [
1157         {
1158             "$ref": "#/parameters/CorrelationId"
1159         },
1160         {
1161             "$ref": "#/parameters/SubscriptionIdPath"
1162         }
1163     ],
1164     "delete": {
1165         "tags": [
1166             "Events"
1167         ],
1168         "summary": "Unsubscribe from events against the set of devices",
1169         "description": "Cancel subscription identified by the id returned in a response for
1170 subscription.\n",
1171         "responses": {
1172             "202": {
1173                 "description": "Subscription was marked for cancellation"

```

```

1174     },
1175     "400": {
1176       "$ref": "#/responses/BadRequest"
1177     },
1178     "401": {
1179       "$ref": "#/responses/Unauthorized"
1180     },
1181     "403": {
1182       "$ref": "#/responses/Forbidden"
1183     },
1184     "404": {
1185       "$ref": "#/responses/NotFound"
1186     }
1187   },
1188   "security": [
1189     {
1190       "oauth2": [
1191         "w:subscriptions:*",
1192         "r:deviceinformation:*"
1193       ]
1194     }
1195   ]
1196 },
1197 "/api/v1/devices/{deviceId}": {
1198   "parameters": [
1199     {
1200       "$ref": "#/parameters/CorrelationId"
1201     },
1202     {
1203       "$ref": "#/parameters/Accept"
1204     },
1205     {
1206       "$ref": "#/parameters/DeviceId"
1207     },
1208     {
1209       "$ref": "#/parameters/BatchFormat"
1210     }
1211   ],
1212   "get": {
1213     "tags": [
1214       "Devices"
1215     ],
1216     "summary": "Get the device by ID",
1217     "consumes": [
1218       "application/json"
1219     ],
1220     "produces": [
1221       "application/json"
1222     ],
1223     "responses": {
1224       "200": {
1225         "description": "Device requested with content=all query parameter",
1226         "schema": {
1227           "$ref": "#/definitions/DeviceContentAll"
1228         }
1229       },
1230       "400": {
1231         "$ref": "#/responses/BadRequest"
1232       },
1233       "401": {
1234         "$ref": "#/responses/Unauthorized"
1235       },
1236       "403": {
1237         "$ref": "#/responses/Forbidden"
1238       },
1239       "404": {
1240         "$ref": "#/responses/NotFound"
1241       }
1242     }
1243   },
1244   "security": [

```

```

1245     {
1246         "oauth2": [
1247             "r:deviceinformation:*"
1248         ]
1249     }
1250 ]
1251 }
1252 },
1253 "/api/v1/devices/{deviceId}/subscriptions": {
1254     "parameters": [
1255         {
1256             "$ref": "#/parameters/CorrelationId"
1257         },
1258         {
1259             "$ref": "#/parameters/DeviceId"
1260         },
1261         {
1262             "$ref": "#/parameters/Accept"
1263         },
1264         {
1265             "$ref": "#/parameters/ContentType"
1266         }
1267     ],
1268     "post": {
1269         "tags": [
1270             "Events"
1271         ],
1272         "summary": "Subscribe to events against a specific device",
1273         "description": "Subscribe to device events by providing `eventTypes` you're interested in
1274 and `eventsUrl` endpoint where events will be sent to as defined. Successful response contains
1275 `subscriptionId` which identifies registered subscription and is part of each event. First event for
1276 each registered event type is received immediately after subscription and contains actual state of
1277 the resource, followed by new events in case of any change.\n\n**Supported events** and required
1278 scopes\n- `resources_published`: `r:deviceinformation:*`\n- `resources_unpublished`:
1279 `r:deviceinformation:*`\n",
1280         "parameters": [
1281             {
1282                 "$ref": "#/parameters/SubscribeRequest"
1283             }
1284         ],
1285         "consumes": [
1286             "application/json"
1287         ],
1288         "produces": [
1289             "application/json"
1290         ],
1291         "responses": {
1292             "201": {
1293                 "$ref": "#/definitions/SubscribeResponse"
1294             },
1295             "400": {
1296                 "$ref": "#/responses/BadRequest"
1297             },
1298             "401": {
1299                 "$ref": "#/responses/Unauthorized"
1300             },
1301             "403": {
1302                 "$ref": "#/responses/Forbidden"
1303             },
1304             "404": {
1305                 "$ref": "#/responses/NotFound"
1306             }
1307         },
1308         "security": [
1309             {
1310                 "oauth2": [
1311                     "w:subscriptions:*",
1312                     "r:deviceinformation:*"
1313                 ]
1314             }
1315         ]

```

```

1316     }
1317   },
1318   "/api/v1/devices/{deviceId}/subscriptions/{subscriptionId}": {
1319     "parameters": [
1320       {
1321         "$ref": "#/parameters/CorrelationId"
1322       },
1323       {
1324         "$ref": "#/parameters/DeviceId"
1325       },
1326       {
1327         "$ref": "#/parameters/SubscriptionIdPath"
1328       }
1329     ],
1330     "delete": {
1331       "tags": [
1332         "Events"
1333       ],
1334       "summary": "Unsubscribe from events against a specific device",
1335       "description": "Cancel subscription identified by the id returned in a response for
1336 subscription.\n",
1337       "responses": {
1338         "202": {
1339           "description": "Subscription was marked for cancellation"
1340         },
1341         "400": {
1342           "$ref": "#/responses/BadRequest"
1343         },
1344         "401": {
1345           "$ref": "#/responses/Unauthorized"
1346         },
1347         "403": {
1348           "$ref": "#/responses/Forbidden"
1349         },
1350         "404": {
1351           "$ref": "#/responses/NotFound"
1352         }
1353       },
1354       "security": [
1355         {
1356           "oauth2": [
1357             "w:subscriptions:*",
1358             "r:deviceinformation:*"
1359           ]
1360         }
1361       ]
1362     }
1363   },
1364   "/api/v1/devices/{deviceId}/{resourceLinkHref}": {
1365     "parameters": [
1366       {
1367         "$ref": "#/parameters/CorrelationId"
1368       },
1369       {
1370         "$ref": "#/parameters/DeviceId"
1371       },
1372       {
1373         "$ref": "#/parameters/ResourceLinkHref"
1374       },
1375       {
1376         "$ref": "#/parameters/Accept"
1377       }
1378     ],
1379     "get": {
1380       "tags": [
1381         "Resources"
1382       ],
1383       "summary": "Retrieve resource values",
1384       "consumes": [
1385         "application/json",
1386         "application/vnd.ocf+cbor"

```

```

1387     ],
1388     "produces": [
1389         "application/json",
1390         "application/vnd.ocf+cbor"
1391     ],
1392     "responses": {
1393         "200": {
1394             "$ref": "#/definitions/ResourceRetrieveResponse"
1395         },
1396         "400": {
1397             "$ref": "#/responses/BadRequest"
1398         },
1399         "401": {
1400             "$ref": "#/responses/Unauthorized"
1401         },
1402         "403": {
1403             "$ref": "#/responses/Forbidden"
1404         },
1405         "404": {
1406             "$ref": "#/responses/NotFound"
1407         },
1408         "503": {
1409             "description": "#/responses/ServiceUnavailable"
1410         },
1411         "504": {
1412             "description": "#/responses/GatewayTimeout"
1413         }
1414     },
1415     "security": [
1416         {
1417             "oauth2": [
1418                 "r:resources:*"
1419             ]
1420         }
1421     ],
1422 },
1423 "post": {
1424     "tags": [
1425         "Resources"
1426     ],
1427     "summary": "Update resource values",
1428     "parameters": [
1429         {
1430             "$ref": "#/parameters/ResourceUpdateRequest"
1431         },
1432         {
1433             "$ref": "#/parameters/ContentType"
1434         }
1435     ],
1436     "consumes": [
1437         "application/json",
1438         "application/vnd.ocf+cbor"
1439     ],
1440     "produces": [
1441         "application/json",
1442         "application/vnd.ocf+cbor"
1443     ],
1444     "responses": {
1445         "200": {
1446             "$ref": "#/definitions/ResourceRetrieveResponse"
1447         },
1448         "400": {
1449             "$ref": "#/responses/BadRequest"
1450         },
1451         "401": {
1452             "$ref": "#/responses/Unauthorized"
1453         },
1454         "403": {
1455             "$ref": "#/responses/Forbidden"
1456         },
1457         "404": {

```



```

1458         "$ref": "#/responses/NotFound"
1459     },
1460     "415": {
1461         "description": "Unsupported media type specified in the Content-Type header"
1462     },
1463     "503": {
1464         "description": "#/responses/ServiceUnavailable"
1465     },
1466     "504": {
1467         "description": "#/responses/GatewayTimeout"
1468     }
1469 },
1470 "security": [
1471     {
1472         "oauth2": [
1473             "w:resources:*"
1474         ]
1475     }
1476 ]
1477 },
1478 },
1479 "/api/v1/devices/{deviceId}/{resourceLinkHref}/subscriptions": {
1480     "parameters": [
1481         {
1482             "$ref": "#/parameters/CorrelationId"
1483         },
1484         {
1485             "$ref": "#/parameters/DeviceId"
1486         },
1487         {
1488             "$ref": "#/parameters/ResourceLinkHref"
1489         },
1490         {
1491             "$ref": "#/parameters/Accept"
1492         },
1493         {
1494             "$ref": "#/parameters/ContentType"
1495         }
1496     ],
1497     "post": {
1498         "tags": [
1499             "Events"
1500         ],
1501         "summary": "Subscribe to events against a specific resource",
1502         "description": "Subscribe to resource events by providing `eventTypes` you're interested in
1503 and `eventsUrl` endpoint where events will be sent to as defined. Successful response contains
1504 `subscriptionId` which identifies registered subscription and is part of each event. First event for
1505 each registered event type is received immediately after subscription and contains actual state of
1506 the resource, followed by new events in case of any change.\n\n`**Supported events**` and required
1507 scopes\n- `resource_contentchanged`: `r:resources:*`,
1508         "parameters": [
1509             {
1510                 "$ref": "#/parameters/SubscribeRequest"
1511             }
1512         ],
1513         "consumes": [
1514             "application/json"
1515         ],
1516         "produces": [
1517             "application/json"
1518         ],
1519         "responses": {
1520             "201": {
1521                 "$ref": "#/definitions/SubscribeResponse"
1522             },
1523             "400": {
1524                 "$ref": "#/responses/BadRequest"
1525             },
1526             "401": {
1527                 "$ref": "#/responses/Unauthorized"
1528             }
1529         }
1530     }
1531 }

```

```

1529         "403": {
1530             "$ref": "#/responses/Forbidden"
1531         },
1532         "404": {
1533             "$ref": "#/responses/NotFound"
1534         }
1535     },
1536     "security": [
1537         {
1538             "oauth2": [
1539                 "w:subscriptions:*",
1540                 "r:resources:*"
1541             ]
1542         }
1543     ]
1544 },
1545 },
1546 "/api/v1/devices/{deviceId}/{resourceLinkHref}/subscriptions/{subscriptionId}": {
1547     "parameters": [
1548         {
1549             "$ref": "#/parameters/CorrelationId"
1550         },
1551         {
1552             "$ref": "#/parameters/DeviceId"
1553         },
1554         {
1555             "$ref": "#/parameters/ResourceLinkHref"
1556         },
1557         {
1558             "$ref": "#/parameters/SubscriptionIdPath"
1559         }
1560     ],
1561     "delete": {
1562         "tags": [
1563             "Events"
1564         ],
1565         "summary": "Unsubscribe from events against a specific resource",
1566         "description": "Cancel subscription identified by the id returned in a response for
1567 subscription.\n",
1568         "responses": {
1569             "202": {
1570                 "description": "Subscription was marked for cancellation"
1571             },
1572             "400": {
1573                 "$ref": "#/responses/BadRequest"
1574             },
1575             "401": {
1576                 "$ref": "#/responses/Unauthorized"
1577             },
1578             "403": {
1579                 "$ref": "#/responses/Forbidden"
1580             },
1581             "404": {
1582                 "$ref": "#/responses/NotFound"
1583             }
1584         },
1585         "security": [
1586             {
1587                 "oauth2": [
1588                     "w:subscriptions:*",
1589                     "r:resources:*"
1590                 ]
1591             }
1592         ]
1593     }
1594 },
1595 "/{eventsUrl}": {
1596     "post": {
1597         "tags": [
1598             "Events"
1599         ],

```

```

1600         "summary": "Events endpoint provided by the subscriber, where events are delivered",
1601         "description": "Events endpoint provided during subscription where events specified in the
1602 subscription will be sent to as defined per event type. Confirmation of each event sent to the
1603 `eventsUrl` endpoint is required with `2xx` success code. Events you will receive based on event
1604 type you're subscribed to are:\n - `subscription_canceled`: `SubscriptionCanceledEvent`\n -
1605 `devices_registered`: `DevicesRegisteredEvent`\n - `devices_unregistered`:
1606 `DevicesUnregisteredEvent`\n - `resources_published`: `ResourcesPublishedEvent`\n -
1607 `resources_unpublished`: `ResourcesUnpublishedEvent`\n - `devices_online`: `DevicesOnlineEvent`\n -
1608 `devices_offline`: `DevicesOfflineEvent`\n - `resource_contentchanged`:
1609 `ResourceContentChangedEvent`",
1610         "parameters": [
1611             {
1612                 "$ref": "#/parameters/CorrelationId"
1613             },
1614             {
1615                 "$ref": "#/parameters/ContentType"
1616             },
1617             {
1618                 "$ref": "#/parameters/EventType"
1619             },
1620             {
1621                 "$ref": "#/parameters/SubscriptionId"
1622             },
1623             {
1624                 "$ref": "#/parameters/SequenceNumber"
1625             },
1626             {
1627                 "$ref": "#/parameters/EventSignature"
1628             },
1629             {
1630                 "$ref": "#/parameters/EventTimestamp"
1631             },
1632             {
1633                 "$ref": "#/parameters/EventsUrl"
1634             },
1635             {
1636                 "$ref": "#/parameters/Event"
1637             }
1638         ],
1639         "consumes": [
1640             "application/json",
1641             "application/vnd.ocf+cbor"
1642         ],
1643         "responses": {
1644             "200": {
1645                 "description": "Event successfully recieved"
1646             },
1647             "400": {
1648                 "$ref": "#/responses/BadRequest"
1649             },
1650             "410": {
1651                 "description": "The subscription identified by the Subscription-ID header is no more in
1652 demand and shall be canceled"
1653             }
1654         }
1655     }
1656 }
1657 },
1658 "securityDefinitions": {
1659     "oauth2": {
1660         "type": "oauth2",
1661         "flow": "accessToken",
1662         "authorizationUrl": "https://example.com/api/oauth/dialog",
1663         "tokenUrl": "https://example.com/api/oauth/token",
1664         "scopes": {
1665             "r:deviceinformation:*": "Read basic device information",
1666             "r:resources:*": "Read content of published resource",
1667             "w:resources:*": "Update content of published resource",
1668             "w:subscriptions:*": "Create subscriptions"
1669         }
1670     }
1671 }

```

```

1671     },
1672     "parameters": {
1673         "CorrelationId": {
1674             "name": "Correlation-ID",
1675             "in": "header",
1676             "type": "string",
1677             "format": "uuid",
1678             "description": "A Correlation ID, also known as a Transit ID, is a unique identifier value
1679 that is attached to requests and messages that allow reference to a particular transaction or event
1680 chain.\n"
1681         },
1682         "ContentType": {
1683             "name": "Content-Type",
1684             "in": "header",
1685             "type": "string",
1686             "enum": [
1687                 "application/json",
1688                 "application/vnd.ocf+cbor"
1689             ],
1690             "required": true,
1691             "description": "The Content-Type header is used to indicate the media type of the resource. In
1692 responses, a Content-Type header tells the client what the content type of the returned content
1693 actually is. In requests, (such as POST), the client tells the server what type of data is actually
1694 sent.\n"
1695         },
1696         "Accept": {
1697             "name": "Accept",
1698             "in": "header",
1699             "type": "string",
1700             "enum": [
1701                 "application/json",
1702                 "application/vnd.ocf+cbor"
1703             ],
1704             "description": "The Accept request header can be used to specify certain media types which are
1705 acceptable for the response. Accept headers can be used to indicate that the request is specifically
1706 limited to a small set of desired types.\n"
1707         },
1708         "SubscriptionId": {
1709             "name": "Subscription-ID",
1710             "in": "header",
1711             "description": "Unique id of the subscription",
1712             "type": "string",
1713             "format": "uuid",
1714             "required": true
1715         },
1716         "SequenceNumber": {
1717             "name": "Sequence-Number",
1718             "in": "header",
1719             "description": "Sequence number of the event; first event starting with number 0",
1720             "type": "string",
1721             "required": true
1722         },
1723         "EventSignature": {
1724             "name": "Event-Signature",
1725             "in": "header",
1726             "description": "The signature created by combining the `signingSecret` from the subscription
1727 request, headers and the body of the request using a stanard HMAC-SHA256 keyed hash.",
1728             "type": "string",
1729             "required": true
1730         },
1731         "EventTimestamp": {
1732             "name": "Event-Timestamp",
1733             "in": "header",
1734             "description": "Time when the event occurred in standard Unix time format",
1735             "type": "string",
1736             "required": true
1737         },
1738         "EventType": {
1739             "name": "Event-Type",
1740             "in": "header",
1741             "type": "string",

```

```

1742     "enum": [
1743         "subscription_canceled",
1744         "devices_registered",
1745         "devices_unregistered",
1746         "resource_contentchanged",
1747         "resources_published",
1748         "resources_unpublished",
1749         "devices_online",
1750         "devices_offline"
1751     ],
1752     "required": true
1753 },
1754 "DeviceType": {
1755     "description": "Filter devices by device type",
1756     "name": "rt",
1757     "in": "query",
1758     "type": "array",
1759     "items": {
1760         "type": "string"
1761     }
1762 },
1763 "ResourceLinkHref": {
1764     "description": "Path to resource",
1765     "name": "resourceLinkHref",
1766     "in": "path",
1767     "type": "string",
1768     "required": true
1769 },
1770 "DeviceId": {
1771     "description": "Id of the device",
1772     "name": "deviceId",
1773     "in": "path",
1774     "type": "string",
1775     "format": "uuid",
1776     "required": true
1777 },
1778 "SubscriptionIdPath": {
1779     "name": "subscriptionId",
1780     "in": "path",
1781     "type": "string",
1782     "format": "uuid",
1783     "required": true
1784 },
1785 "BatchFormat": {
1786     "name": "content",
1787     "in": "query",
1788     "description": "Indicates to the recipient that the response payload shall be the resolved
1789 (i.e. resource representation) Link and not the Link itself. Default is `base`. When requesting
1790 `all`, additional scope `r:resources:*:*` is required",
1791     "type": "string",
1792     "enum": [
1793         "base",
1794         "all"
1795     ]
1796 },
1797 "EventsUrl": {
1798     "name": "eventsUrl",
1799     "type": "string",
1800     "in": "path",
1801     "required": true
1802 },
1803 "ResourceUpdateRequest": {
1804     "description": "Map of resource values to be updated",
1805     "name": "content",
1806     "in": "body",
1807     "schema": {
1808         "$ref": "#/definitions/ResourceUpdateRequest"
1809     },
1810     "required": true
1811 },
1812 "SubscribeRequest": {

```

```

1813     "name": "content",
1814     "in": "body",
1815     "schema": {
1816       "$ref": "#/definitions/SubscribeRequest"
1817     },
1818     "required": true
1819   },
1820   "Event": {
1821     "description": "Event of a specific type, based on what you are subscribed to",
1822     "name": "content",
1823     "in": "body",
1824     "schema": {
1825       "$ref": "#/definitions/ResourceContentChangedEvent"
1826     },
1827     "required": true
1828   }
1829 },
1830 "responses": {
1831   "Unauthorized": {
1832     "description": "Unauthorized"
1833   },
1834   "NotFound": {
1835     "description": "Not found"
1836   },
1837   "SubscriptionCancellationPending": {
1838     "description": "Subscription was marked for cancellation"
1839   },
1840   "Forbidden": {
1841     "description": "Insufficient permissions"
1842   },
1843   "BadRequest": {
1844     "description": "The request was malformed or badly constructed"
1845   },
1846   "ServiceUnavailable": {
1847     "description": "The service on the Target Cloud is unavailable for the reason indicated in the
1848 diagnostic payload"
1849   },
1850   "GatewayTimeout": {
1851     "description": "The target Device is registered at the target Cloud, however the Device itself
1852 is unavailable, offline, or otherwise unreachable. The response should include a Retry-After header
1853 containing the time after which the request may be re-attempted. Additional information is indicated
1854 in the diagnostic payload."
1855   }
1856 },
1857 "definitions": {
1858   "Device": {
1859     "type": "object",
1860     "properties": {
1861       "device": {
1862         "$ref": "https://raw.githubusercontent.com/ondrejtomcik/coreocf/Bugzilla-
1863 2709/swagger2.0/oic.wk.d.swagger.json#/definitions/Device"
1864       },
1865       "status": {
1866         "$ref": "#/definitions/DeviceStatus"
1867       },
1868       "links": {
1869         "type": "array",
1870         "items": {
1871           "$ref": "https://raw.githubusercontent.com/ondrejtomcik/coreocf/Bugzilla-
1872 2709/swagger2.0/oic.wk.res.swagger.json#/definitions/oic.oic-link"
1873         }
1874       }
1875     },
1876     "example": {
1877       "device": {
1878         "dmn": "Open Connectivity Foundation",
1879         "n": "Food safety sensor",
1880         "di": "53080a4f-5e3e-4291-802f-3436238232d2"
1881       },
1882       "status": "online",
1883       "links": [

```

```

1884         {
1885             "href": "/53080a4f-5e3e-4291-802f-3436238232d2/humidity",
1886             "rt": [
1887                 "oic.r.humidity"
1888             ]
1889         },
1890         {
1891             "href": "/53080a4f-5e3e-4291-802f-3436238232d2/temperature",
1892             "rt": [
1893                 "oic.r.temperature"
1894             ]
1895         }
1896     ]
1897 }
1898 },
1899 "DeviceContentAll": {
1900     "type": "object",
1901     "properties": {
1902         "device": {
1903             "$ref": "https://raw.githubusercontent.com/ondrejtomcik/coreocf/Bugzilla-
1904 2709/swagger2.0/oic.wk.d.swagger.json#/definitions/Device"
1905         },
1906         "status": {
1907             "$ref": "#/definitions/DeviceStatus"
1908         },
1909         "links": {
1910             "type": "array",
1911             "items": {
1912                 "type": "object",
1913                 "properties": {
1914                     "href": {
1915                         "type": "string"
1916                     },
1917                     "rep": {
1918                         "type": "object"
1919                     }
1920                 }
1921             }
1922         }
1923     },
1924     "example": {
1925         "device": {
1926             "dmn": "Open Connectivity Foundation",
1927             "n": "Food safety sensor",
1928             "di": "53080a4f-5e3e-4291-802f-3436238232d2"
1929         },
1930         "status": "online",
1931         "links": [
1932             {
1933                 "href": "/humidity",
1934                 "rep": {
1935                     "humidity": 62,
1936                     "desiredHumidity": 65
1937                 }
1938             },
1939             {
1940                 "href": "/temperature",
1941                 "rep": {
1942                     "temperature": 21,
1943                     "units": "C"
1944                 }
1945             }
1946         ]
1947     }
1948 },
1949 "DeviceStatus": {
1950     "description": "Device status available from the OCF Cloud, which tracks if the device has
1951 opened TCP connection and is signed in",
1952     "type": "string",
1953     "enum": [
1954         "online",

```

```

1955     "offline"
1956   ]
1957 },
1958 "ResourceUpdateRequest": {
1959   "type": "string",
1960   "description": "Desired content of the resource",
1961   "example": "o29kZXNpcmVkbHVtaWRpdHkYpGV0eXBlc4Fub2ljLnIuaHVtaWRpdHloaHVtaWRpdHkYKA=="
1962 },
1963 "ResourceRetrieveResponse": {
1964   "type": "string",
1965   "description": "Content of the resource returned from the device",
1966   "example": "o29kZXNpcmVkbHVtaWRpdHkYpGV0eXBlc4Fub2ljLnIuaHVtaWRpdHloaHVtaWRpdHkYKA=="
1967 },
1968 "EventType": {
1969   "type": "string",
1970   "enum": [
1971     "subscription_canceled",
1972     "devices_registered",
1973     "devices_unregistered",
1974     "resource_contentchanged",
1975     "resources_published",
1976     "resources_unpublished",
1977     "devices_online",
1978     "devices_offline"
1979   ]
1980 },
1981 "SubscriptionId": {
1982   "description": "Unique id of the subscription",
1983   "type": "string",
1984   "format": "uuid"
1985 },
1986 "SubscribeRequest": {
1987   "type": "object",
1988   "properties": {
1989     "eventsUrl": {
1990       "$ref": "#/definitions/EventsUrl"
1991     },
1992     "eventTypes": {
1993       "type": "array",
1994       "items": {
1995         "$ref": "#/definitions/EventType"
1996       }
1997     },
1998     "signingSecret": {
1999       "type": "string",
2000       "maxLength": 32,
2001       "minLength": 32
2002     }
2003   },
2004   "required": [
2005     "eventsUrl",
2006     "eventTypes",
2007     "signingSecret"
2008   ],
2009   "example": {
2010     "eventsUrl": "https://events.example.com/",
2011     "eventTypes": [
2012       "devices_registered",
2013       "devices_unregistered"
2014     ]
2015   }
2016 },
2017 "SubscribeResponse": {
2018   "description": "Subscription was registered, waiting for verification",
2019   "type": "object",
2020   "properties": {
2021     "subscriptionId": {
2022       "$ref": "#/definitions/SubscriptionId"
2023     }
2024   },
2025   "required": [

```



```

2026         "subscriptionId"
2027     ],
2028     "example": {
2029         "subscriptionId": "leeb465c-5e8d-4305-a366-bbf035fff671"
2030     }
2031 },
2032 "EventsUrl": {
2033     "type": "string",
2034     "format": "url",
2035     "example": "https://events.exaple.com/"
2036 },
2037 "SubscriptionCanceledEvent": {
2038     "type": "object",
2039     "description": "Subscription with provided id was canceled"
2040 },
2041 "DevicesRegisteredEvent": {
2042     "description": "Device was successfully signed up to the OCF Cloud, as defined in the
2043 `oic.sec.account`,
2044     "type": "object",
2045     "properties": {
2046         "content": {
2047             "type": "array",
2048             "items": {
2049                 "properties": {
2050                     "di": {
2051                         "type": "string",
2052                         "format": "uuid"
2053                     }
2054                 }
2055             }
2056         }
2057     }
2058 },
2059 "DevicesUnregisteredEvent": {
2060     "description": "Device was successfully signed off from the OCF Cloud, as defined in the
2061 `oic.sec.account`,
2062     "type": "object",
2063     "properties": {
2064         "content": {
2065             "type": "array",
2066             "items": {
2067                 "properties": {
2068                     "di": {
2069                         "type": "string",
2070                         "format": "uuid"
2071                     }
2072                 }
2073             }
2074         }
2075     }
2076 },
2077 "ResourcesPublishedEvent": {
2078     "type": "object",
2079     "properties": {
2080         "content": {
2081             "type": "array",
2082             "items": {
2083                 "$ref": "https://raw.githubusercontent.com/ondrejtomcik/coreocf/Bugzilla-
2084 2709/swagger2.0/oic.wk.res.swagger.json#/definitions/oic.oic-link"
2085             }
2086         }
2087     }
2088 },
2089 "ResourcesUnpublishedEvent": {
2090     "type": "object",
2091     "properties": {
2092         "content": {
2093             "type": "array",
2094             "items": {
2095                 "$ref": "https://raw.githubusercontent.com/ondrejtomcik/coreocf/Bugzilla-
2096 2709/swagger2.0/oic.wk.res.swagger.json#/definitions/oic.oic-link"

```

```

2097     }
2098   }
2099 },
2100 "DevicesOnlineEvent": {
2101   "type": "object",
2102   "properties": {
2103     "content": {
2104       "type": "array",
2105       "items": {
2106         "properties": {
2107           "di": {
2108             "type": "string",
2109             "format": "uuid"
2110           }
2111         }
2112       }
2113     }
2114   }
2115 },
2116 "DevicesOfflineEvent": {
2117   "type": "object",
2118   "properties": {
2119     "content": {
2120       "type": "array",
2121       "items": {
2122         "properties": {
2123           "di": {
2124             "type": "string",
2125             "format": "uuid"
2126           }
2127         }
2128       }
2129     }
2130   }
2131 },
2132 "ResourceContentChangedEvent": {
2133   "type": "string",
2134   "description": "New Content of the resource returned from the device",
2135   "example": "o29kzXNpcmVkSHVtaWRpdHkYPGV0eXB1c4Fub21jLnIuaHVtaWRpdHloaHVtaWRpdHkYKA=="
2136 }
2137 }
2138 }
2139 }
2140

```