OCF Core Specification

VERSION 2.2.8 | June 2025

CONTACT **admin@openconnectivity.org**

Legal Disclaimer

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

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

CONTENTS

Figures

298

299

Tables

Introduction

This document, and all the other parts associated with this document, were developed in response to worldwide demand for smart home focused Internet of Things (IoT) devices, such as appliances, door locks, security cameras, sensors, and actuators; these to be modelled and securely controlled, locally and remotely, over an IP network.

While some inter-device communication existed, no universal language had been developed for the IoT. Device makers instead had to choose between disparate frameworks, limiting their market share, or developing across multiple ecosystems, increasing their costs. The burden then falls on end users to determine whether the products they want are compatible with the ecosystem they bought into, or find ways to integrate their devices into their network, and try to solve interoperability issues on their own.

In addition to the smart home, IoT deployments in commercial environments are hampered by a lack of security. This issue can be avoided by having a secure IoT communication framework, which this standard solves.

The goal of these documents is then to connect the next 25 billion devices for the IoT, providing secure and reliable device discovery and connectivity across multiple OSs and platforms. There are multiple proposals and forums driving different approaches, but no single solution addresses the majority of key requirements. This document and the associated parts enable industry consolidation around a common, secure, interoperable approach.

The OCF specification suite is made up of nineteen discrete documents, the documents fall into logical groupings as described herein:

- Core framework
  - Core Specification
  - Security Specification
  - Onboarding Tool Specification
- Bridging framework and bridges
  - Bridging Specification
  - Resource to Alljoyn Interface Mapping Specification
  - OCF Resource to oneM2M Resource Mapping Specification
  - OCF Resource to BLE Mapping Specification
  - OCF Resource to EnOcean Mapping Specification
  - OCF Resource to LWM2M Mapping Specification
  - OCF Resource to UPlus Mapping Specification
  - OCF Resource to Zigbee Cluster Mapping Specification
  - OCF Resource to Z-Wave Mapping Specification
- Resource and Device models
  - Resource Type Specification
  - Device Specification
- Core framework extensions
  - Easy Setup Specification
  - Core Optional Specification
- OCF Cloud
  - Cloud API for Cloud Services Specification

426 – Device to Cloud Services Specification
427 – Cloud Security Specification

OCF Core Specification

1 Scope

The OCF Core specifications are divided into a set of documents:

– Core specification (this document): The Core specification document specifies the Framework, i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles implementation for Internet of Things (IoT) usages and ecosystems. This document is mandatory for all Devices to implement.

– Core optional specification: The Core optional specification document specifies the Framework, i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles implementation for Internet of Things (IoT) usages and ecosystems that can optionally be implemented by any Device.

– Core extension specification(s): The Core extension specification(s) document(s) specifies optional OCF Core functionality that are significant in scope (e.g., Wi-Fi easy setup, Cloud).

2 Normative references

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

ISO 8601, *Data elements and interchange formats – Information interchange –Representation of dates and times*, International Standards Organization, December 3, 2004

ISO/IEC DIS 20924, *Information Technology – Internet of Things – Vocabulary*, June 2018
https://www.iso.org/standard/69470.html

ISO/IEC 30118-2, *Information technology – Open Connectivity Foundation (OCF) Specification – Part 2: Security specification*
https://www.iso.org/standard/74239.html
Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

IETF RFC 768, *User Datagram Protocol*, August 1980
https://www.rfc-editor.org/info/rfc768

IETF RFC 3339, *Date and Time on the Internet: Timestamps*, July 2002
https://www.rfc-editor.org/info/rfc3339

IETF RFC 3986, *Uniform Resource Identifier (URI): General Syntax, January 2005*.
https://www.rfc-editor.org/info/rfc3986

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, July 2005
https://www.rfc-editor.org/info/rfc4122

IETF RFC 4287, *The Atom Syndication Format*, December 2005,
https://www.rfc-editor.org/info/rfc4287

IETF RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, September 2007
https://www.rfc-editor.org/info/rfc4941

IETF RFC 5646, *Tags for Identifying Languages*, September 2009
https://www.rfc-editor.org/info/rfc5646

IETF RFC 6347, *Datagram Transport Layer Security Version 1.2*, January 2012
https://www.rfc-editor.org/info/rfc6347

IETF RFC 6434, *IPv6 Node Requirements*, December 2011
https://www.rfc-editor.org/info/rfc6434

IETF RFC 6573, *The Item and Collection Link Relations*, April 2012
https://www.rfc-editor.org/info/rfc6573

IETF RFC 6690, *Constrained RESTful Environments (CoRE) Link Format*, August 2012
https://www.rfc-editor.org/info/rfc6690

IETF RFC 7049, *Concise Binary Object Representation (CBOR)*, October 2013
https://www.rfc-editor.org/info/rfc7049

IETF RFC 7084, *Basic Requirements for IPv6 Customer Edge Routers*, November 2013
https://www.rfc-editor.org/info/rfc7084

IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014
https://www.rfc-editor.org/info/rfc7159

IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014
https://www.rfc-editor.org/info/rfc7252

IETF RFC 7301, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*, July 2014
https://www.rfc-editor.org/info/rfc7301

IETF RFC 7346, *IPv6 Multicast Address Scopes*, August 2014
https://www.rfc-editor.org/info/rfc7346

IETF RFC 7595, *Guidelines and Registration Procedures for URI Schemes*, June 2015
https://www.rfc-editor.org/info/rfc7595

IETF RFC 7641, *Observing Resources in the Constrained Application Protocol (CoAP)*, September 2015
https://www.rfc-editor.org/info/rfc7641

IETF RFC 7721, *Security and Privacy Considerations for IPv6 Address Generation Mechanisms*, March 20016
https://www.rfc-editor.org/info/rfc7721

IETF RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*, August 2016
https://www.rfc-editor.org/info/rfc7959

IETF RFC 8075, *Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)*, February 2017
https://www.rfc-editor.org/info/rfc8075

IETF RFC 8085, *UDP Usage Guidelines*, March 2017
https://www.rfc-editor.org/info/rfc8085

IETF RFC 8288, *Web Linking*, October 2017
https://www.rfc-editor.org/info/rfc8288

IETF RFC 8323, *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*, February 2018
https://www.rfc-editor.org/info/rfc8323

IANA ifType-MIB Definitions
https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib

IANA IPv6 Multicast Address Space Registry
http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml

IANA Link Relations, October 2017
http://www.iana.org/assignments/link-relations/link-relations.xhtml

JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January 2013
http://json-schema.org/draft-04/json-schema-validation.html

MQTT Version 5.0 OASIS Standard 07 March 2019
https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.pdf

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

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in the following apply.

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

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

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

3.1.1
Atomic Measurement
design pattern that ensures that the *Client* (3.1.6) can only access the *Properties* (3.1.34) of linked *Resources* (3.1.32) atomically, that is as a single group

3.1.2
Bridged Client
logical entity that accesses data via a *Bridged Protocol* (3.1.4)

Note 1 to entry: For example, an AllJoyn Consumer application is a *Bridged Client* (3.1.2)

3.1.3
Bridged Device
Bridged Client (3.1.2) or *Bridged Server* (3.1.5)

3.1.4
Bridged Protocol
another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

3.1.5
Bridged Server
logical entity that provides data via a *Bridged Protocol* (3.1.4)

Note 1 to entry: For example an AllJoyn Producer is a *Bridged Server* (3.1.5).

Note 2 to entry: More than one *Bridged Server* (3.1.5) can exist on the same physical platform.

3.1.6
Client
logical entity that accesses a *Resource* (3.1.32) on a *Server* (3.1.37)

3.1.7
Collection
Resource (3.1.32) that contains zero or more *Links* (3.1.22)

3.1.8
Common Properties
Properties (3.1.34) specified for all *Resources* (3.1.32)

3.1.9
Composite Device
Device (3.1.13) that is modelled as multiple *Device Types* (3.1.14); with each component *Device Type* (3.1.14) being exposed as a *Collection* (3.1.7)

3.1.10
Configuration Source
cloud or service network or a local read-only file which contains and provides configuration related information to the *Devices* (3.1.13)

3.1.11
Core Resources
those *Resources* (3.1.32) that are defined in this document

3.1.12
Default OCF Interface
OCF Interface (3.1.19) used to generate the response when an *OCF Interface* (3.1.19) is omitted in a request

3.1.13
Device
logical entity that assumes one or more roles, e.g., *Client* (3.1.6), *Server* (3.1.37)

Note 1 to entry: More than one *Device* (3.1.13) can exist on a *Platform* (3.1.31).

3.1.14
Device Type
uniquely named definition indicating a minimum set of *Resource Types* (3.1.35) that a *Device* (3.1.13) supports

Note 1 to entry: A *Device Type* (3.1.14) provides a hint about what the *Device* (3.1.13) is, such as a light or a fan, for use during *Resource* (3.1.32) discovery.

3.1.15
Device UUID
stack instance identifier

3.1.16
Discoverable Resource
Resource (3.1.32) that is listed in "/oic/res"

3.1.17
OCF Endpoint
entity participating in the OCF protocol, further identified as the source or destination of a request and response messages for a given Transport Protocol Suite

Note 1 to entry: Example of a Transport Protocol Suite would be CoAP over UDP over IPv6.

3.1.18
Framework
set of related functionalities and interactions defined in this document, which enable interoperability across a wide range of networked devices, including IoT

3.1.19
OCF Interface
interface description extended by OCF that provides a view to and permissible responses from a *Resource* (3.1.32)

[SOURCE: IETF RFC 6690]

3.1.20
Introspection
mechanism to determine the capabilities of the hosted *Resources* (3.1.32) of a *Device* (3.1.13)

3.1.21
Introspection Device Data (IDD)
data that describes the payloads per implemented method of the *Resources* (3.1.32) that make up the *Device* (3.1.13)

Note 1 to entry: See 11.4 for all requirements and exceptions.

3.1.22
Links
extends typed web links

[SOURCE: IETF RFC 8288]

3.1.23
Non-Discoverable Resource
Resource (3.1.32) that is not listed in "/oic/res"

Note 1 to entry: The *Resource* (3.1.32) can be reached by a *Link* (3.1.22) which is conveyed by another *Resource* (3.1.32). For example a Resource (3.1.32) linked in a Collection (3.1.7) does not have to be listed in "/oic/res", since traversing the *Collection* (3.1.7) would discover the *Resource* (3.1.32) implemented on the *Device* (3.1.13).

3.1.24
Notification
mechanism to make a *Client* (3.1.6) aware of state changes in a *Resource* (3.1.32)

3.1.25
Observe
act of monitoring a *Resource* (3.1.32) by sending a RETRIEVE operation which is cached by the *Server* (3.1.37) hosting the *Resource* (3.1.32) and reprocessed on every change to that *Resource* (3.1.32)

3.1.26
OpenAPI 2.0
Resource (3.1.32) and *Introspection Device Data* (3.1.21) definitions used in this document

[SOURCE: OpenAPI specification]

3.1.27
Parameter
element that provides metadata about a *Resource* (3.1.32) referenced by the target URI of a *Link* (3.1.22)

3.1.28
Partial UPDATE
UPDATE operation to a *Resource* (3.1.32) that includes a subset of the *Properties* (3.1.34) that are visible via the *OCF Interface* (3.1.19) being applied for the *Resource Type* (3.1.35)

3.1.29
Permanent Immutable ID
identity for a *Device* (3.1.13) that cannot be altered

3.1.30
Physical Device
physical thing on which a *Device(s)* (3.1.13) is exposed

3.1.31
Platform
Physical Device (3.1.30) containing one or more *Devices* (3.1.13)

3.1.32
Resource
represents an entity modelled and exposed by the *Framework* (3.1.18)

3.1.33
Resource Interface
qualification of the permitted requests on a *Resource* (3.1.32)

3.1.34
Property
significant aspect or *Parameter* (3.1.27) of a *Resource* (3.1.32), including metadata, that is exposed through the *Resource* (3.1.32)

3.1.35
Resource Type
uniquely named definition of a class of *Properties* (3.1.34) and the interactions that are supported by that class

Note 1 to entry: Each *Resource* (3.1.32) has a *Property* (3.1.34) "rt" whose value is the unique name of the *Resource Type* (3.1.35).

3.1.36
Secure OCF Endpoint
OCF Endpoint (3.1.17) with a secure connection (e.g., CoAPS)

3.1.37
Semantic Tag
meta-information that provides additional contextual information with regard to the *Resource* (3.1.32) that is the target of a *Link* (3.1.22)

3.1.38
Server
Device (3.1.13) with the role of providing *Resource* (3.1.32) state information and facilitating remote interaction with its *Resources* (3.1.32)

3.1.39
Sleepy Server
Server (3.1.38) that will have latency in responding to requests

3.1.40
Unsecure OCF Endpoint
OCF Endpoint (3.1.17) with an unsecure connection (e.g., CoAP)

3.1.41
Vertical Resource Type
Resource Type (3.1.35) in a vertical domain specification

Note 1 to entry: An example of a Vertical *Resource Type* (3.1.41) would be "oic.r.switch.binary".

3.2 Symbols and abbreviated terms

ACL	Access Control List
BLE	Bluetooth Low Energy
CBOR	Concise Binary Object Representation
CoAP	Constrained Application Protocol
CoAPs	Secure Constrained Application Protocol
DTLS	Datagram Transport Layer Security
IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
MTU	Maximum Transmission Unit
OCF	Open Connectivity Foundation
REST	Representational State Transfer
RESTful	REST-compliant Web services
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UUID	Universal Unique Identifier

4 Document conventions and organization

4.1 Conventions

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

In this document, to be consistent with the IETF usages for RESTful operations, the RESTful operation words CRUDN, CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY will have all letters capitalized. Any lowercase uses of these words have the normal technical English meaning.

The messaging payload examples in this document contain OCF Vertical Device Types and Resource Types, which are used for illustrative purposes only.

4.2 Notation

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

Required (or shall or mandatory)(M).

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

Recommended (or should)(S).

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

Allowed (may or allowed)(O).

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

DEPRECATED.

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

Conditionally allowed (CA).

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

Conditionally required (CR).

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

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

Words that are emphasized are printed in italic.

In all of the Property and Resource definition tables that are included throughout this document the "Mandatory" column indicates that the item detailed is mandatory to implement; the mandating of inclusion of the item in a Resource Payload associated with a CRUDN action is dependent on the applicable schema for that action.

4.3 Data types

Resources are defined using data types derived from JSON values as defined in IETF RFC 7159. However, a Resource can overload a JSON defined value to specify a particular subset of the JSON value, using validation keywords defined in JSON Schema Validation.

Among other validation keywords, clause 7 in JSON Schema Validation defines a "format" keyword with a number of format attributes such as "uri" and "date-time", and a "pattern" keyword with a regular expression that can be used to validate a string. This clause defines patterns that are available for use in describing OCF Resources. The pattern names can be used in document text where JSON format names can occur. The actual JSON schemas shall use the JSON type and pattern instead.

For all rows defined in Table 1, the JSON type is string.

Table 1 – Additional OCF Types

Pattern Name	Pattern	Description
"csv"	<none>	A comma separated list of values encoded within a string. The value type in the csv is described by the Property where the csv is used. For example, a csv of integers. NOTE csv is considered deprecated and an array of strings should be used instead for new Resources.
"date"	^([0-9]{4})-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$	The full-date format pattern according to IETF RFC 3339
"duration"	^(P(?!$)([0-9]+Y)?([0-9]+M)?([0-9]+W)?([0-9]+D)?((T(?=[0-9]+[HMS])([0-9]+H)?([0-9]+M)?([0-9]+S)?)?))$|^(P[0-9]+W)$\|^(P[0-9]{4})-(1[0-2]\|0[1-9])-(3[0-1]\|2[0-9]\|1[0-9]\|0[1-9])T(2[0-3]\|1[0-9]\|0[1-9]):([0-5][0-9]):([0-5][0-9])$|^(P[0-9]{4})(1[0-2]|0[1-9])(3[0-1]|2[0-9]|1[0-9]|0[1-9])T(2[0-3]|1[0-9]|0[1-9])([0-5][0-9])([0-5][0-9])$	A string representing duration formatted as defined in ISO 8601. Allowable formats are: P[n]Y[n]M[n]DT[n]H[n]M[n]S, P[n]W, P[n]Y[n]-M[n]-DT[0-23]H[0-59]:M[0-59]:S, and P[n]W, P[n]Y[n]M[n]DT[0-23]H[0-59]M[0-59]S. P is mandatory, all other elements are optional, time elements must follow a T.
"int64"	^0|(-?[1-9][0-9]{0,18})$	A string instance is valid against this attribute if it contains an integer in the range [-(2**63), (2**63)-1] NOTE IETF RFC 7159 clause 6 explains that JSON integers outside the range [-(2**53)+1, (2**53)-1] are not interoperable and so JSON numbers cannot be used for 64-bit numbers.
"language-tag"	^[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*$	An IETF language tag formatted according to IETF RFC 5646 clause 2.1.
"uint64"	^0|([1-9][0-9]{0,19})$	A string instance is valid against this attribute if it contains an integer in the range [0, (2**64)-1] Also see note for "int64"
"uuid"	^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$	A UUID string representation formatted according to IETF RFC 4122 clause 3.

Strings shall be encoded as UTF-8 unless otherwise specified.

In a JSON schema, "maxLength" for a string indicates the maximum number of characters not octets. However, "maxLength" shall also indicate the maximum number of octets. If no "maxLength" is defined for a string, then the maximum length shall be 64 octets.

4.4 Resource notation syntax

When it is desired to describe the Property of a Resource Type or the "anchor" Parameter value in an abbreviated notation, it can be described as follows:

– A value of the "rt" Property of the Resource Type or "anchor" Parameter value ":" Property name

– e.g., "oic.wk.d:di", which is the "di" Property of the Device Resource Type.

If Property name is a composite type (a type that is composed of several Properties), it can be described in recursive way. The following expression describes this as a regular expression format:

– A value of the "rt" Property of the Resource Type or "anchor" Parameter value (":" Property name)+

– e.g., "oic.r.pstat:dos:s", which is the "s" Property of the "dos" Property of the "pstat" Resource Type (see 13.8 of ISO/IEC 30118-2).

If there is a Resource URI (i.e., The Resource instance for a specific Resource Type), it can be used instead of using a value of "rt" Property of Resource Type or the “anchor" Parameter value as follows:

– A Resource URI (":" Property name)+

– e.g., "/oic/d:di", which is the "di" Property of the Device Resource Type instance.

– e.g. "/oic/sec/pstat:dos:s", which is the "s" Property of the "dos" Property of the "oic.r.pstat" Resource Type instance.

In the auto-generated Annex's Property definition tables for Resource Types, the Property names can be noted as belonging to the RETRIEVE schema or to the UPDATE schema by prefixing the Property name with "RETRIEVE" or "UPDATE" followed with the ":" separator. This is to avoid duplicate Property names appearing in the Property definition tables that are auto-generated. The following are examples using this notation with the "locn" Property of the "oic.wk.con" Resource Type:

– "RETRIEVE:locn"

– "UPDATE:locn"

5 Architecture

5.1 Overview

The architecture *Datagram* enables resource based interactions among IoT artefacts, i.e. physical devices or applications. The architecture leverages existing industry standards and technologies and provides solutions for establishing connections (either wireless or wired) and managing the flow of information among Devices, regardless of their form factors, operating systems or service providers.

Specifically, the architecture provides:

– A communication and interoperability framework for multiple market segments (Consumer, Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication, transports and use cases.

– A common and consistent model for describing the environment and enabling information and semantic interoperability.

– Common communication protocols for discovery and connectivity.

– Common security and identification mechanisms.

– Opportunity for innovation and product differentiation.

– A scalable solution addressing different Device capabilities, applicable to smart devices as well as the smallest connected things and wearable devices.

The architecture is based on the Resource Oriented Architecture design principles and described in the 5.2 through 5.4 respectively. 5.2 presents the guiding principles for OCF operations. 5.3 defines the functional block diagram and Framework.

5.2 Principle

In the architecture, Entities in the physical world (e.g., temperature sensor, an electric light or a home appliance) are represented as Resources. Interactions with an entity are achieved through its Resource representations (see 7.6.3.9) using operations that adhere to Representational State Transfer (REST) architectural style, i.e., RESTful interactions.

The architecture defines the overall structure of the Framework as an information system and the interrelationships of the Entities that make up OCF. Entities are exposed as Resources, with their unique identifiers (URIs) and support interfaces that enable RESTful operations on the Resources. Every RESTful operation has an initiator of the operation (the Client) and a responder to the operation (the Server). In the Framework, the notion of the Client and Server is realized through roles. Any Device can act as a Client and initiate a RESTful operation on any Device acting as a Server. Likewise, any Device that exposes Entities as Resources acts as a Server. Conformant to the REST architectural style, each RESTful operation contains all the information necessary to understand the context of the interaction and is driven using a small set of generic operations, i.e., CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY (CRUDN) defined in clause 7.11, which include representations of Resources.

Figure 1 depicts the architecture.

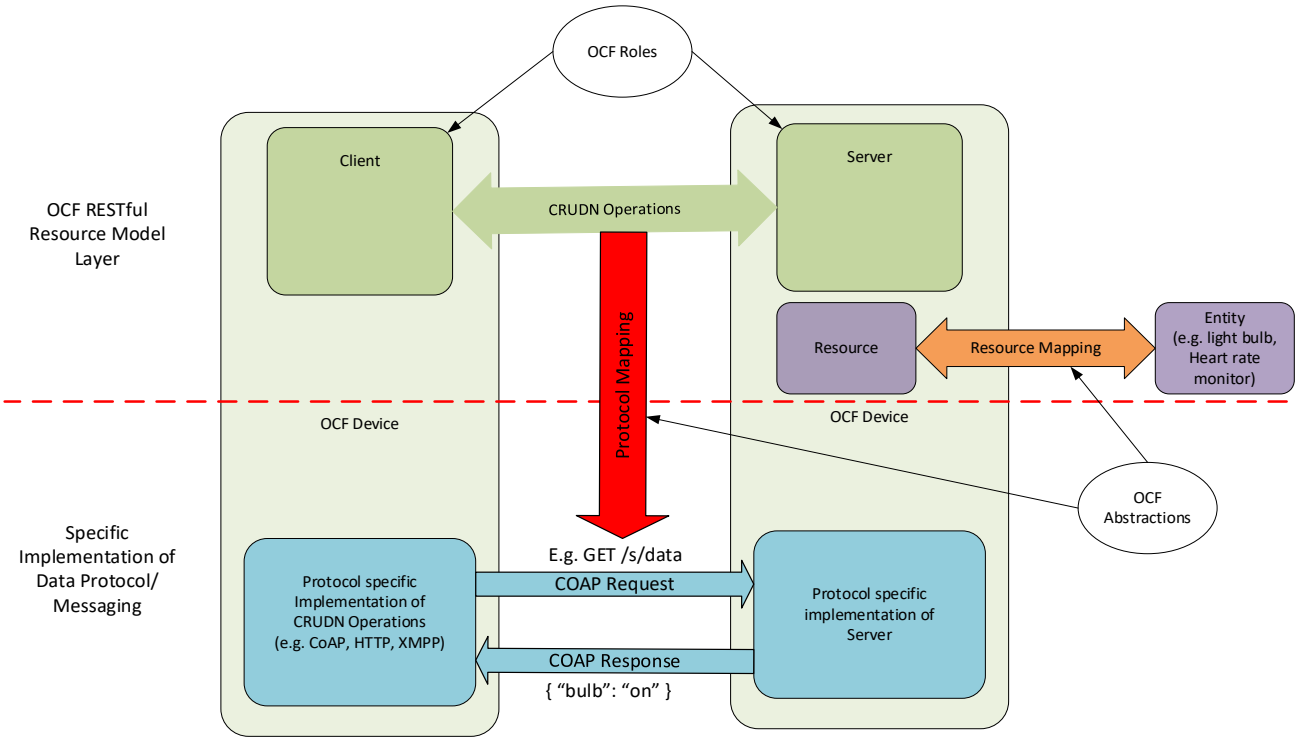


Figure 1 – Architecture - concepts

The architecture is organized conceptually into three major aspects that provide overall separation of concern: Resource model, RESTful operations and abstractions.

– Resource model: The Resource model provides the abstractions and concepts required to logically model, and logically operate on the application and its environment. The Core Resource model is common and agnostic to any specific application domain such as smart home, industrial or automotive. For example, the Resource model defines a Resource which abstracts an entity and the representation of a Resource maps the entity's state. Other Resource model concepts can be used to model other aspects, for example behaviour.

– RESTful operations: The generic CRUDN operations are defined using the RESTful paradigm to model the interactions with a Resource in a protocol and technology agnostic way. The specific communication or messaging protocols are part of the protocol abstraction and mapping of Resources to specific protocols is provided in 0.

– Abstraction: The abstractions in the Resource model and the RESTful operations are mapped to concrete elements using abstraction primitives. An entity handler is used to map an entity to a Resource and connectivity abstraction primitives are used to map logical RESTful operations to data connectivity protocols or technologies. Entity handlers may also be used to map Resources to Entities that are reached over protocols that are not natively supported by OCF.

5.3 Functional block diagram

The functional block diagram encompasses all the functionalities required for operation. These functionalities are categorized as L2 connectivity, networking, transport, Framework, and application profiles. The functional blocks are depicted in Figure 2.

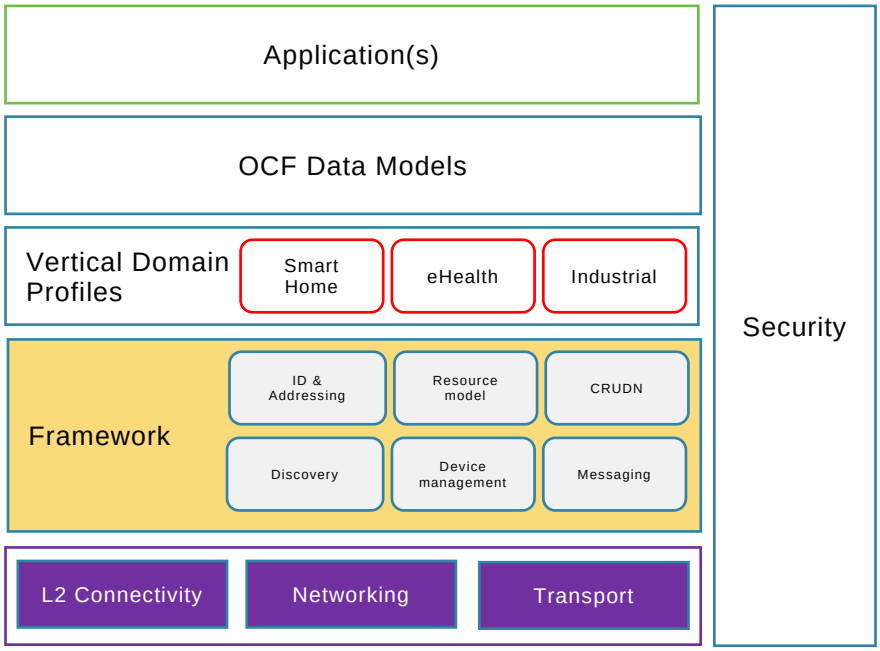


Figure 2 – Functional block diagram

– *L2 connectivity*: Provides the functionalities required for establishing physical and data link layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.

– *Networking*: Provides functionalities required for Devices to exchange data among themselves over the network (e.g., Internet).

– *Transport*: Provides end-to-end flow transport with specific QoS constraints. Examples of a transport protocol include TCP and UDP or new Transport protocols under development in the IETF, e.g., Delay Tolerant Networking (DTN).

– *Framework*: Provides the core functionalities as defined in this document. The functional block is the source of requests and responses that are the content of the communication between two Devices.

– *Vertical Domain profile*: Provides market segment specific functionalities, e.g., functions for the smart home market segment.

When two Devices communicate with each other, each functional block in a Device interacts with its counterpart in the peer Device as shown in Figure 3.

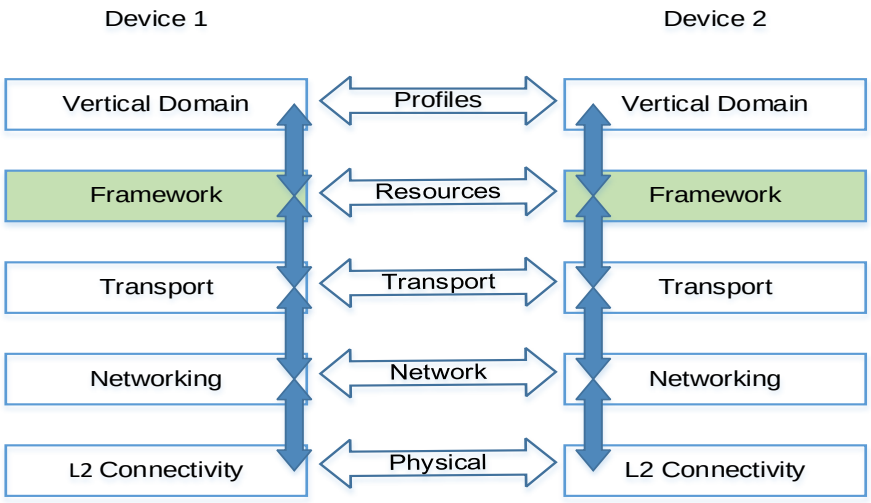


Figure 3 – Communication layering model

5.4 Framework

Framework consists of functions which provide core functionalities for operation.

– *Identification and addressing.* Defines the identifier and addressing capability. The Identification and addressing function is defined in clause 6.

– *Discovery*. Defines the process for discovering available.

  – Devices (OCF Endpoint Discovery in clause 10) and

  – Resources (Resource discovery in 11.2).

– *Resource model*. Specifies the capability for representation of entities in terms of Resources and defines mechanisms for manipulating the Resources. The Resource model function is defined in clause 7.

– *CRUDN*. Provides a generic scheme for the interactions between a Client and Server as defined in clause 7.11.

– *Messaging*. Provides specific message protocols for RESTful operation, i.e. CRUDN. For example, CoAP is a primary messaging protocol. The messaging function is defined in 11.5.

– *Security***.** Includes authentication, authorization, and access control mechanisms required for secure access to Entities. The security function is defined in clause 12.5.

6 Identification and addressing

6.1 Introduction

Facilitating proper and efficient interactions between elements in the Framework, requires a means to identify, name and address these elements.

The *identifier* unambiguously identifies an element in a context or domain. The context or domain may be determined by the use or the application. The identifier is expected to be immutable over the lifecycle of that element and is unambiguous within a context or domain.

The *address* is used to define a place, way or means of reaching or accessing the element in order to interact with it. An address may be mutable based on the context.

The *name* is a handle that distinguishes the element from other elements in the Framework. The name may be changed over the lifecycle of that element.

There may be methods or resolution schemes that allow determining any of these based on the knowledge of one or more of others (e.g., determine name from address or address from name).

Each of these aspects may be defined separately for multiple contexts (e.g., a context could be a layer in a stack). So an address may be a URL for addressing Resource and an IP address for addressing at the connectivity layer. In some situations, both these addresses would be required. For example, to do RETRIEVE (see 8.3) operation on a particular Resource representation, the Client needs to know the address of the target Resource and the address of the Server through which the Resource is exposed.

In a context or domain of use, a name or address could be used as identifier or vice versa. For example, a URL could be used as an identifier for a Resource and designated as a URI.

The remainder of this clause discusses the identifier, address and naming from the point of view of the Resource model and the interactions to be supported by the Resource model. Examples of interactions are the RESTful interactions, i.e. CRUDN operation (clause 7.11) on a Resource. Also the mapping of these to transport protocols, e.g., CoAP is described.

6.2 Identification

6.2.1 Device and Platform identification

This document defines three identifiers that are used for identification of the Device. All identifiers are exposed via Resources that are also defined within this document (see clause 11.2).

The Permanent Immutable ID ("piid" Property of "/oic/d") is the immutable identity of the Device, the persistent valid value of this property is typically only visible after the Device is on-boarded (when not on-boarded the Device typically exposes a temporary value). This value does not change across the life-cycle of the Device.

The Device UUID ("di" Property of "/oic/d") is a mutable identity. The value changes each time the Device is on-boarded. It reflects a specific on-boarded instance of the Device.

The Platform ID ("pi" Property of "/oic/p") is the immutable identity of the Platform on which the Device is resident. When multiple logical Devices are exposed on a single Platform (for example, on a Bridge) then the "pi" exposed by each Device should be the same.

6.2.2 Resource identification and addressing

A Resource may be identified using a URI and addressed by the same URI if the URI is a URL. In some cases, a Resource may need an identifier that is different from a URI; in this case, the Resource may have a Property whose value is the identifier. When the URI is in the form of a URL, then the URI may be used to address the Resource.

An OCF URI is based on the general form of a URI as defined in IETF RFC 3986 as follows (note that the portion in square brackets is optional):

```
<scheme>://<authority>/<path>?<query>
```

Specifically, the OCF URI is specified in the following form:

```
ocf://<authority>/<path>?<query>
```

The following is a description of values that each component takes.

The "scheme" for the URI is "ocf". The "ocf" scheme represents the semantics, definitions and use as defined in this document. If a URI has the portion preceding the "//" (double slash) omitted, then the "ocf" scheme shall be assumed.

Each transport binding is responsible for specifying how an OCF URI is converted to a transport protocol URI before sending over the network by the requestor. Similarly on the receiver side, each transport binding is responsible for specifying how an OCF URI is converted from a transport protocol URI before handing over to the Resource model layer on the receiver.

The authority of an OCF URI shall be the Device UUID ("di") value, as defined in [OCF Security], of the Server.

The "path" is a string that unambiguously identifies or references a Resource within the context of the Server. In this version of the document, a path shall not include pct-encoded non-ASCII characters or NUL characters. A *path* shall be preceded by a "/" (slash). The *path* may have "/" (slash) separated segments for human readability reasons. In the OCF context, the "/" (slash) separated segments are treated as a single string that directly references the Resources (i.e. a flat structure) and not parsed as a hierarchy. On the Server, the path or some substring in the path may be shortened by using hashing or some other scheme provided the resulting reference is unique within the context of the host.

Once a path is generated, a Client accessing the Resource or recipient of the URI should use that path as an opaque string and should not parse to infer a structure, organization or semantic.

The "query" is a string that shall contain one or more "<name>=<value>" constructs (aka name-value pair). Where multiple such constructs are supported, each is separated by an "&" (ampersand); this is not a logical "and" operation, but purely a delimiter. Where the use of a query is supported, how the query is handled by the recipient thereof is explicitly defined by the relevant clause in this document or other specifications. The query string will be mapped to the appropriate syntax of the protocol used for messaging. (e.g., CoAP).

A URI may be either fully qualified or relative generation of URI.

A URI may be defined by the Client which is the creator of that Resource. Such a URI may be relative or absolute (fully qualified). A relative URI shall be relative to the Device on which it is hosted. Alternatively, a URI may be generated by the Server of that Resource automatically based on a pre-defined convention or organization of the Resources, based on an OCF Interface, based on some rules or with respect to different roots or bases.

The absolute path reference of a URI is to be treated as an opaque string and a Client should not infer any explicit or implied structure in the URI – the URI is simply an address. It is also recommended that Devices hosting a Resource treat the URI of each Resource as an opaque string that addresses only that Resource. (e.g., URI's "/a" and "/a/b" are considered as distinct addresses and Resource b cannot be construed as a child of Resource a).

6.3 Namespace:

The relative URI prefix "/oic/" is reserved as a namespace for URIs defined in OCF specifications and shall not be used for URIs that are not defined in OCF specifications. The prefix "oic." used for OCF Interfaces and Resource Types is reserved for OCF specification usage.

6.4 Network addressing

The following are the addresses used in this document:

IP address

– An IP address is used when the Device is using an IP configured interface.
– When a Device only has the identity information of its peer, a resolution mechanism is needed to map the identifier to the corresponding address.

7 Resource model

7.1 Introduction

The Resource model defines concepts and mechanisms that provide consistency and core interoperability between Devices in the OCF ecosystems. The Resource model concepts and mechanisms are then mapped to the transport protocols to enable communication between the Devices – each transport provides the communication protocol interoperability. The Resource model, therefore, allows for interoperability to be defined independent of the transports.

The primary concepts in the Resource model are: entity, Resources, Uniform Resource Identifiers (URI), Resource Types, Properties, Representations, OCF Interfaces, Collections and Links. In addition, the general mechanisms are CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY. These concepts and mechanisms may be composed in various ways to define the rich semantics and interoperability needed for a diverse set of use cases that the Framework is applied to.

In the OCF Resource model Framework, an entity needs to be visible, interacted with or manipulated, it is represented by an abstraction called a Resource. A Resource encapsulates and represents the state of an entity. A Resource is identified, addressed and named using URIs.

Properties are "key=value" pairs and represent state of the Resource. A snapshot of these Properties is the Representation of the Resource. A specific view of the Representation and the mechanisms applicable in that view are specified as OCF Interfaces. Interactions with a Resource are done as Requests and Responses containing Representations.

A Resource instance is derived from a Resource Type. The uni-directional relationship between one Resource and another Resource is defined as a Link. A Resource that has Properties and Links is a Collection.

A set of Properties can be used to define a state of a Resource. This state may be retrieved or updated using appropriate Representations respectively in the response from and request to that Resource.

A Resource (and Resource Type) could represent and be used to expose a capability. Interactions with that Resource can be used to exercise or use that capability. Such capabilities can be used to define processes like discovery, management, advertisement etc. For example: *discovery of Resources on a Device* can be defined as the retrieval of a representation of a specific Resource where a Property or Properties have values that describe or reference the Resources on the Device.

The information for Request or Response with the Representation may be communicated on the wire by serializing using a transfer protocol or encapsulated in the payload of the transport protocol – the specific method is determined by the normative mapping of the Request or Response to the transport protocol. See clause 12 for transport protocols supported.

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

7.2 Resource

A Resource shall be defined by one or more Resource Type(s) – see Annex A for Resource Type. A request to CREATE a Resource shall specify one or more Resource Types that define that Resource.

A Resource is hosted in a Device. A Resource shall have a URI as defined in clause 6. The URI may be assigned by the Authority at the creation of the Resource or may be pre-defined by the definition of the Resource Type. An example Resource representation is depicted in Figure 4.

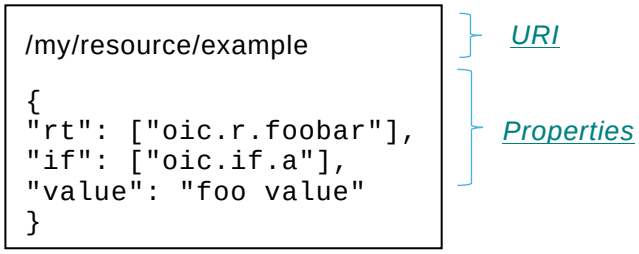


Figure 4 – Example Resource

Core Resources are the Resources defined in this document to enable functional interactions as defined in clause 10 (e.g., Discovery, Device management, etc.). Among the Core Resources, "/oic/res", "/oic/p", and "/oic/d" shall be supported on all Devices. Devices may support other Core Resources depending on the functional interactions they support.

7.3 Property

7.3.1 Introduction

A Property describes an aspect that is exposed through a Resource including meta-information related to that Resource.

A Property shall have a name i.e. Property Name and a value i.e. Property Value. The Property is expressed as a key-value pair where key is the Property Name and value the Property Value like <Property Name> = <Property Value>. For example, if the "temperature" Property has a Property Name "temp" and a Property Value "30F", then the Property is expressed as "temp=30F". The specific format of the Property depends on the encoding scheme. For example, in JSON, Property is represented as "key": value (e.g., "temp": 30).

In addition, the Property definition shall have a

– *Value Type* – the Value Type defines the values that a Property Value may take. The Value Type may be a simple data type (e.g. string, Boolean) as defined in 4.3 or may be a complex data type defined with a schema. The Value Type may define
  – Value Rules define the rules for the set of values that the Property Value may take. Such rules may define the range of values, the min-max, formulas, the set of enumerated values, patterns, conditional values, and even dependencies on values of other Properties. The rules may be used to validate the specific values in a Property Value and flag errors.
– *Mandatory* – specifies if the Property is mandatory or not for a given Resource Type.
– *Access modes* – specifies whether the Property may be read, written or both. Updates are equivalent to a write. "r" is used for read and "w" is used for write – both may be specified. Write does not automatically imply read.

The definition of a Property may include the following additional information – these items are informative:

– *Property Title* - a human-friendly name to designate the Property; usually not sent over the wire.
– *Description* – descriptive text defining the purpose and expected use of this Property.

In general, a Property is meaningful only within the Resource to which it is associated. However, a base set of Properties that may be supported by all Resources, known as Common Properties, keep their semantics intact across Resources i.e. their "key=value" pair means the same in any Resource. Detailed tables for all Common Properties are defined in 7.3.2.

7.3.2 Common Properties

7.3.2.1 Introduction

The mandatory Common Properties defined in clause 7.3.2 shall be exposed and the optional Common Properties may be exposed in all Resources. The following Properties are defined as Common Properties:

The Common Properties for all Resources are specified in 7.3.2.3 through 7.3.2.6 respectively and summarized as follows:

– *Resource Type* ("rt") – this mandatory Property is used to declare the Resource Type of that Resource. Since a Resource could be defined by more than one Resource Type the Property Value of the Resource Type Property may be used to declare more than one Resource Type (see clause 7.4.4). See 7.3.2.3 for details.

– *OCF Interface* ("if") – this mandatory Property declares the OCF Interfaces supported by the Resource. The Property Value of the OCF Interface Property may be multi-valued and lists all the OCF Interfaces supported. See 7.3.2.4 for details.

– *Name* ("n") – this optional Property declares human-readable name assigned to the Resource. See 7.3.2.5.

– *Resource Identity* ("id") – this optional Property Value shall be a unique (across the scope of the host Server) identifier for a specific instance of the Resource. The encoding of this identifier is Device and implementation dependent. See 7.3.2.6 for details.

An optional Common Property may be mandatory when explicitly specified in a particular Resource Type definition (e.g., the "n" Common Property for the "oic.wk.d" Resource Type).

The name of a Common Property is unique and is not used by other Properties. When defining a new Resource Type, its non-common Properties will not use the name of existing Common Properties (e.g., "rt", "if", "n", and "id").

The ability to UPDATE a Common Property (that supports write as an access mode) is restricted to the "oic.if.rw" (read-write) OCF Interface; thus a Common Property shall be updatable using the read-write OCF Interface if and only if the Property supports write access as defined by the Property definition and the associated schema for the read-write OCF Interface.

7.3.2.2 Property Name and Property Value definitions

The Property Name and Property Value as used in this document:

– *Property Name*– the key in "key=value" pair. Property Name is case sensitive and its data type is "string". Property names shall contain only letters A to Z, a to z, digits 0 to 9, hyphen, and dot, and shall not begin with a digit.

– *Property Value* – the value in "key=value" pair. Property Value is case sensitive when its data type is "string".

7.3.2.3 Resource Type

Resource Type Property is specified in 7.4.

7.3.2.4 OCF Interface

OCF Interface Property is specified in 7.6.

7.3.2.5 Name

A human friendly name for the Resource, i.e. a specific resource instance name (e.g., MyLivingRoomLight), The Name Property is as defined in Table 2

Table 2 – Name Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	"n"	"string"	N/A	N/A	R, W	No	Human understandable name for the Resource.

Note: This Property may be mandatory when specifically defined for a Resource Type (e.g., "oic.wk.d").

The Name Property is read-write unless otherwise restricted by the Resource Type (i.e. the Resource Type does not support UPDATE or does not support UPDATE using the read-write OCF Interface ("oic.if.rw")).

7.3.2.6 Resource Identity

The Resource Identity Property shall be a unique (across the scope of the host Server) instance identifier for a specific instance of the Resource. The encoding of this identifier is Device and implementation dependent as long as the uniqueness constraint is met, noting that an implementation may use a uuid as defined in 4.3. The Resource Identity Property is as defined in Table 3.

Table 3 – Resource identity Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Identity	"id"	"string" or uuid	Implementation Dependent	N/A	R	No	Unique identifier of the Resource (over all Resources in the Device)

Note: This Property may be mandatory when specifically defined for a Resource Type.

7.4 Resource Type

7.4.1 Introduction

Resource Type is a class or category of Resources and a Resource is an instance of one or more Resource Types.

The Resource Types of a Resource is declared using the Resource Type Common Property as described in 7.3.2.3 or in a Link using the Resource Type Parameter.

A Resource Type may either be pre-defined by OCF or in custom definitions by manufacturers, end users, or developers of Devices (vendor-defined Resource Types). Resource Types and their definition details may be communicated out of band (i.e. in documentation) or be defined explicitly using a meta-language which may be downloaded and used by APIs or applications. OCF has adopted OpenAPI 2.0 as the specification method for OCF's RESTful interfaces and Resource definitions.

Every Resource Type shall be identified with a Resource Type ID which shall be represented using the requirements and ABNF governing the Resource Type attribute in IETF RFC 6690 (clause 2 for ABNF and clause 3.1 for requirements) with the caveat that segments are separated by a "." (period). The entire string represents the Resource Type ID. When defining the ID each segment may represent any semantics that are appropriate to the Resource Type. For example, each segment could represent a namespace. Once the ID has been defined, the ID should be used opaquely and implementations should not infer any information from the individual segments. The string "oic", when used as the first segment in the definition of the Resource Type ID, is reserved for OCF-defined Resource Types. All OCF defined Resource Types are to be registered with the IANA Core Parameters registry as described also in IETF RFC 6690.

7.4.2 Resource Type Property

A Resource when instantiated or created shall have one or more Resource Types that are the template for that Resource. The Resource Types that the Resource conforms to shall be declared using the "rt" Common Property for the Resource as defined in Table 4. The Property Value for the "rt" Common Property shall be the list of Resource Type IDs for the Resource Types used as templates (i.e., "rt"=<list of Resource Type IDs>).

Table 4 – Resource Type Common Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Type	"rt"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	Yes	The Property name rt is as described in IETF RFC 6690

Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and the host (i.e. Server) of the Resource.

7.4.3 Resource Type definition

Resource Type is specified as follows:

– *Pre-defined URI* (optional) – a pre-defined URI may be specified for a specific Resource Type in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that Resource Type shall use only the pre-defined URI. An instance of a different Resource Type shall not use the pre-defined URI.

– *Resource Type Title* (optional) – a human friendly name to designate the Resource Type**.**

– *Resource Type ID* – the value of "rt" Property which identifies the Resource Type, (e.g., "oic.wk.p").

– *Resource Interfaces* – list of the OCF Interfaces that may be supported by the Resource Type.

– *Properties* – definition of all the Properties that apply to the Resource Type. The Resource Type definition shall define whether a property is mandatory, conditional mandatory, or optional.

– *Related Resource Types* (optional) – the definition of other Resource Types that may be referenced as part of the Resource Type, applicable to Collections.

– *Mime Types* (optional) – mime types supported by the Resource including serializations (e.g., application/cbor, application/json, application/xml).

Table 5 and Table 6 provides an example description of an illustrative foobar Resource Type and its associated Properties.

Table 5 – Example foobar Resource Type

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction	M/CR/O
none	"foobar"	"oic.r.foobar"	"oic.if.a"	Example "foobar" Resource	Actuation	O

Table 6 – Example foobar Properties

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Type	"rt"	"array"	N/A	N/A	R	Yes	Resource Type
OCF Interface	"if"	"array"	N/A	N/A	R	Yes	OCF Interface
Foo value	value	"string"	N/A	N/A	R	Yes	Foo value

For example, an instance of the foobar Resource Type.

```
{
"rt": ["oic.r.foobar"],
"if": ["oic.if.a"],
"value": "foo value"
}
```

For example, a schema representation for the foobar Resource Type.

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "rt": {
      "type": "array",
      "items" : {
        "type" : "string",
        "maxLength": 64
      },
      "minItems" : 1,
      "readOnly": true,
      "description": "Resource Type of the Resource"
    },
    "if": {
      "type": "array",
      "items": {
        "type" : "string",
        "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
"oic.if.r", "oic.if.a", "oic.if.s"]
    },
    "value": {"type": "string"}
  },
  "required": ["rt", "if", "value"]
}
```

7.4.4 Multi-value "rt" Resource

Multi-value "rt" Resource means a Resource with multiple Resource Types where none of the included Resource Types denote a well-known Resource Type (i.e. "oic.wk.<thing>"). Such a Resource is associated with multiple Resource Types and so has an "rt" Property Value of multiple Resource Type IDs (e.g. "rt": ["oic.r.switch.binary", "oic.r.light.brightness"]). The order of the Resource Type IDs in the "rt" Property Value is meaningless. For example, "rt": ["oic.r.switch.binary", "oic.r.light.brightness"] and "rt": ["oic.r.light.brightness", "oic.r.switch.binary"] have the same meaning.

Resource Types for multi-value "rt" Resources shall satisfy the following conditions:

– Property Name – Property Names for each Resource Type shall be unique (within the scope of the multi-value "rt" Resource) with the exception of Common Properties, otherwise there will be conflicting Property semantics. If two Resource Types have a Property with the same Property "Name, a multi-value "rt" Resource shall not be composed of these Resource Types.

A multi-value "rt" Resource satisfies all the requirements for each Resource Type and conforms to the OpenAPI 2.0 definitions for each component Resource Type. Thus the mandatory Properties of a multi-value "rt" Resource shall be the union of all the mandatory Properties of each Resource Type. For example, mandatory Properties of a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"] are "value" and "brightness", where the former is mandatory for "oic.r.switch.binary" and the latter for "oic.r.light.brightness".

The multi-value "rt" Resource Interface set shall be the union of the sets of OCF Interfaces from the component Resource Types. The Resource Representation in response to a CRUDN action on an OCF Interface shall be the union of the schemas that are defined for that OCF Interface. The Default OCF Interface for a multi-value "rt" Resource shall be the baseline OCF Interface ("oic.if.baseline") as that is the only guaranteed common OCF Interface between the Resource Types.

For clarity if each Resource Type supports the same set of OCF Interfaces, then the resultant multi-value "rt" Resource has that same set of OCF Interfaces with a Default OCF Interface of baseline ("oic.if.baseline").

See 7.9.3 for the handling of query parameters as applied to a multi-value "rt" Resource.

7.5 Device Type

A Device Type is a class of Device. Each Device Type defined will include a list of minimum Resource Types that a Device shall implement for that Device Type. A Device may expose additional standard and vendor defined Resource Types beyond the minimum list. The Device Type is used in Resource discovery as specified in 11.2.3.

Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in a Link using the Resource Type Parameter.

A Device Type may either be pre-defined by an ecosystem that builds on this document, or in custom definitions by manufacturers, end users, or developers of Devices (vendor-defined Device Types). Device Types and their definition details may be communicated out of band (like in documentation).

Every Device Type shall be identified with a Resource Type ID using the same syntax constraints as a Resource Type.

7.6 OCF Interface

7.6.1 Introduction

An OCF Interface provides first a view into the Resource and then defines the requests and responses permissible on that view of the Resource. So this view provided by an OCF Interface defines the context for requests and responses on a Resource. Therefore, the same request to a Resource when targeted to different OCF Interfaces may result in different responses. Depending on the view requested (i.e., OCF Interface), the Resource representation may not include all mandatory Properties (e.g., the "rt" and "if" Common Properties). If Common Properties are desired in the view requested, use the "oic.if.baseline" OCF Interface (see clause 7.6.3.2) which every Resource Type shall implement.

An OCF Interface may be defined by either this document (a Core OCF Interface), manufacturers, end users or developers of Devices (a vendor-defined OCF Interface).

The OCF Interface Property lists all the OCF Interfaces the Resource support. All Resources shall have at least one OCF Interface. The Default OCF Interface shall be defined by the Resource Type definition. The Default OCF Interface associated with all OCF-defined Resource Types shall be the supported OCF Interface listed first within the *applicable enumeration* in the definition of the Resource Type (see Annex A for the OCF-defined Resource Types defined in this document). The *applicable enumeration* is in the "parameters" enumeration referenced from the first "get" method in the first "path" in the OpenAPI 2.0 file ("post" method if no "get" exists) for the Resource Type. All Default OCF Interfaces specified in an OCF specification shall be mandatory.

In addition to any defined OCF Interface in this document, all Resources shall support the baseline OCF Interface ("oic.if.baseline") as defined in 7.6.3.2.

See 7.9.4 for the use of queries to enable selection of a specific OCF Interface in a request.

An OCF Interface may accept more than one media type. An OCF Interface may respond with more than one media type. The accepted media types may be different from the response media types. The media types are specified with the appropriate header parameters in the transfer protocol. (NOTE: This feature has to be used judiciously and is allowed to optimize representations on the wire) Each OCF Interface shall have at least one media type.

7.6.2 OCF Interface Property

The OCF Interfaces supported by a Resource shall be declared using the OCF Interface Common Property (Table 7), e.g., ""if": ["oic.if.ll", "oic.if.baseline"]". The Property Value of an OCF Interface Property shall be a lower case string with segments separated by a "." (dot). The string "oic", when used as the first segment in the OCF Interface Property Value, is reserved for OCF-defined OCF Interfaces. The OCF Interface Property Value may also be a reference to an authority similar to IANA that may be used to find the definition of an OCF Interface. A Resource Type shall support one or more of the OCF Interfaces defined in 7.6.3.

Table 7 – Resource Interface Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
OCF **Interface**	"if"	"array"	Array of strings, conveying OCF Interfaces	N/A	R	Yes	Property to declare the OCF Interfaces supported by a Resource.

7.6.3 OCF Interface methods

7.6.3.1 Overview

OCF Interface methods shall not violate the defined OpenAPI 2.0 definitions for the Resources as defined in Annex A.

The defined OCF Interfaces are listed in Table 8:

Table 8 – Standard OCF Interfaces

OCF Interface	Name	Applicable Operations	Description
baseline	"oic.if.baseline"	RETRIEVE, NOTIFY, UPDATE[1]	The baseline OCF Interface defines a view into all Properties of a Resource including the Common Properties. This OCF Interface is used to operate on the full Representation of a Resource.

[1] The use of UPDATE with the baseline OCF Interface is not recommended, see clause 7.6.3.2.3.

links list	"oic.if.ll"	RETRIEVE, NOTIFY	The links list OCF Interface provides a view into Links in a Collection (Resource). Since Links represent relationships to other Resources, the links list OCF Interfaces may be used to discover Resources with respect to a context. The discovery is done by retrieving Links to these Resources. For example: the Core Resource "/oic/res" uses this OCF Interface to allow discovery of Resource hosted on a Device.
batch	"oic.if.b"	RETRIEVE, NOTIFY, UPDATE	The batch OCF Interface is used to interact with a Collection of Resources at the same time. This also removes the need for the Client to first discover the Resources it is manipulating – the Server forwards the requests and aggregates the responses
read-only	"oic.if.r"	RETRIEVE NOTIFY	The read-only OCF Interface exposes the Properties of a Resource that may be read. This OCF Interface does not provide methods to update Properties, so can only be used to read Property Values.
write-only	“oic.if.w”	UPDATE	The write-only OCF Interface writes the Property Values of a Resource that may be ‘written'. This OCF Interface does not provide methods to RETRIEVE Properties of a Resource and so may only be used to ‘write' Property Values. Note that the UPDATE may specify Properties to be returned in a response.
read-write	"oic.if.rw"	RETRIEVE, NOTIFY, UPDATE	The read-write OCF Interface exposes only those Properties that may be read from a Resource during a RETRIEVE operation and only those Properties that may be written to a Resource during and UPDATE operation.
actuator	"oic.if.a"	RETRIEVE, NOTIFY, UPDATE	The actuator OCF Interface is used to read or write the Properties of an actuator Resource.
sensor	"oic.if.s"	RETRIEVE, NOTIFY	The sensor OCF Interface is used to read the Properties of a sensor Resource.
create	"oic.if.create"	CREATE	The create OCF Interface is used to create new Resources in a Collection. Both the Resource and the Link pointing to it are created in a single atomic operation.
property startup	"oic.if.startup"	RETRIEVE, UPDATE	The property startup OCF Interface is used to set the default values of Properties on a Resource that will be applied on power-up of the Device hosting the Resource
Property revert	"oic.if.startup.revert"	RETRIEVE, UPDATE	The property startup revert OCF Interface is used to establish a revert state on a Resource such that on a power-up of the Device hosting the Resource the target Properties are populated with the values to which they were set prior to the power-down.

7.6.3.2 Baseline OCF Interface

7.6.3.2.1 Overview

The Representation that is visible using the baseline OCF Interface includes all the Properties of the Resource including the mandatory and implemented optional Common Properties. The baseline OCF Interface shall be defined for all Resource Types. All Resources shall support the baseline OCF Interface.

7.6.3.2.2 Use of RETRIEVE

The baseline OCF Interface is used when a Client wants to retrieve all Properties of a Resource; that is the Server shall respond with a Resource representation that includes all of the implemented Properties of the Resource. When the Server is unable to send back the whole Resource

representation, it shall reply with an error message. The Server shall not return a partial Resource representation.

An example response to a RETRIEVE request using the baseline OCF Interface:

```
{
"rt": ["oic.r.temperature"],
"if": ["oic.if.a","oic.if.baseline"],
"temperature": 20,
"units": "C",
"range": [0,100]
}
```

7.6.3.2.3 Use of UPDATE

Support for the UPDATE operation using the baseline OCF Interface should not be provided by a Resource Type. Where a Resource Type needs to support the ability to be UPDATED this should only be supported using one of the other OCF Interfaces defined in Table 8 that supports the UPDATE operation.

If a Resource Type is required to support UPDATE using the baseline OCF Interface, then all Properties of a Resource with the exception of Common Properties may be modified using an UPDATE operation only if the Resource Type defines support for UPDATE using baseline in the applicable OpenAPI 2.0 schema for the Resource Type. If the OCF Interfaces exposed by a Resource in addition to the baseline OCF Interface do not support the UPDATE operation, then UPDATE using the baseline OCF Interface shall not be supported.

7.6.3.3 Links list OCF Interface

7.6.3.3.1 Overview

The Links list OCF Interface is used to provide a view into a Collection, Atomic Measurement, or "/oic.res" Resource. This view shall be an array of all Links for those Resources subject to any applied filtering being applied. The Links list OCF Interface name is "oic.if.ll".

7.6.3.3.2 Use with RETRIEVE

The RETRIEVE operation is supported with the Links list OCF Interface. A successful RETRIEVE operation shall return a status code indicating success (i.e. "Content") with a payload with the Resource representation as an array of Links. If there are no Links present in a Resource representation, then an empty array list shall be returned in response to a RETRIEVE operation request.

An example of a RETRIEVE operation request using the Links list OCF Interface for a Collection is as illustrated:

```
RETRIEVE /scenes/scene1?if=oic.if.ll
```

The RETRIEVE operation response will be the array of Links to all Resources in the Collection as illustrated:

```
Response: Content
Payload:
[
  {
    "href": "/the/light/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/the/light/2",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
```



```
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/my/fan/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/his/fan/2",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  }
]
```

7.6.3.3.3 Use with NOTIFY

The NOTIFY operation is supported with the Links list OCF Interface. A successful NOTIFY operation shall return a status code indicating success (i.e. "Content") with a payload with the Resource representation as an array of Links. If there are no Links present in a Resource representation, then an empty array list shall be returned in response to a NOTIFY operation request. Future events that change the Resource representation (e.g. UPDATE operation) shall return a status code indicating success (i.e. "Content") with a payload with the newly updated Resource representation as an array of Links.

An example of a NOTIFY operation request using the Links list OCF Interface for a Collection is as illustrated:

```
NOTIFY /scenes/scene1?if=oic.if.ll
```

The NOTIFY operation response will be the array of Links to all Resources in the Collection as illustrated:

```
Response: Content
Payload:
[
  {
    "href": "/the/light/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/the/light/2",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/my/fan/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/his/fan/2",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  }
]
```


Later when the "/his/fan/2" Link is removed (e.g., UPDATE operation with the Link remove OCF Interface) the response to the NOTIFY operation request is as illustrated:

```
Response: Content
Payload:
[
  {
    "href": "/the/light/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/the/light/2",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/my/fan/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  }
]
```

If the result of removing a Link results in no Links being present, then an empty array list shall be sent in a notification. An example of a response with no Links being present is as illustrated:

```
Response: Content
Payload:
[
]
```

7.6.3.3.4Use with CREATE, UPDATE, and DELETE

The CREATE, UPDATE and DELETE operations are not allowed by the Links list OCF Interface. Attempts to perform CREATE, UPDATE or DELETE operations using the Links list OCF Interface shall return an appropriate error status code, for example "Method Not Allowed".

7.6.3.4 Batch OCF Interface

7.6.3.4.1 Overview

The batch OCF Interface is used to interact with a Collection of Resources using a single/same Request. The batch OCF Interface can be used to RETRIEVE or UPDATE the Properties of the linked Resources with a single request.

7.6.3.4.2 General requirements for realizations of the batch OCF Interface

All realisations of the batch OCF Interface adhere to the following:

– The batch OCF Interface name is "oic.if.b"

– A Collection Resource has linked Resources that are represented as URIs. In the "href" Property of the batch payload the URI shall be fully qualified for remote Resources and a relative reference for local Resources.

– The original request is modified to create new requests targeting each of the linked Resources in the Collection by substituting the URI in the original request with the URI of the linked Resource. The payload in the original request is replicated in the payload of the new requests.

– The requests shall be forwarded assuming use of the Default OCF Interface of the linked Resources.

– Requests shall only be forwarded to linked Resources that are identified by relation types "item" or "hosts" ("hosts" is the default relation type value should the "rel" Link Parameter not be present). Requests shall not be forwarded to linked Resources that do not contain the "item" or "hosts" relation type values.

– Properties of the Collection Resource itself may be included in payloads using "oic.if.b" OCF Interface by exposing a single Link with the link relation "self" along with "item" within the Collection, and ensuring that Link resolution cannot become an infinite loop due to recursive references. For example, if the Default OCF Interface of the Collection is "oic.if.b", then the Server might recursively include its batch representation within its batch representation, in an endless loop. See 7.6.3.4.5 for an example of use of a Link containing "rel": ["self","item"] to include Properties of the Collection Resource, along with linked Resources, in "oic.if.b" payloads.

– If the Default OCF Interface of a Collection Resource is exposed using the Link relation "self", and the Default OCF Interface contains Properties that expose any Links, those Properties shall not be included in a batch representation which includes the "self" Link.

– Any request forwarded to a linked Resource that is a Collection (including a "self" Link reference) shall have the Default OCF Interface of the linked Collection Resource applied.

– All the responses from the linked Resources shall be aggregated into a single Response to the Client. The Server may timeout the response to a time window, the Server may choose any appropriate window based on conditions.

– If a linked Resource cannot process the request, an empty response, i.e. a JSON object with no content ("{}") as the representation for the "rep" Property, or error response should the linked Resource Type provide an error schema or diagnostic payload, shall be returned by the linked Resource. These empty or error responses for all linked Resources that exhibit an error shall be included in the aggregated response to the original Client request. See the example in 7.6.3.4.5.

– If any of the linked Resources returns an error response, the aggregated response sent to the Client shall also indicate an error (e.g. 4.xx in CoAP). If all of the linked Resources return successful responses, the aggregated response shall include the success response code.

– The aggregated response shall be an array of objects representing the responses from each linked Resource. Each object in the response shall include at least two items: (1) the URI of the linked Resource (fully qualified for remote Resources, or a relative reference for local Resources) as "href": <URI> and (2) the individual response object or array of objects if the linked Resource is itself a Collection using "rep" as the key, e.g. "rep": { <representation of individual response> }.

– The Client may specify the Resource Type(s) of the linked Resources to which the request is forwarded by including one or more "rt" query parameters in the request, each separated by an "&" as a delimiter (e.g. "?if=oic.if.b&rt=oic.r.switch.binary"). The Server shall then process such additional query parameters in a request that includes "oic.if.b", as selectors for the Linked Resources that are to be processed by the request.

7.6.3.4.3 Observability of the batch OCF Interface

When a Collection supports the ability to be observed using the batch OCF Interface the following apply:

– If the Collection Resource is marked as Observable, linked Resources referenced in the Collection may be Observed using the batch OCF Interface. If the Collection Resource is not marked as Observable then the Collection cannot be Observed and Observe requests to the Collection shall be handled as defined for the case where request validation fails in clause 11.3.2.4. The Observe mechanism shall work as defined in 11.3.2 with the Observe request forwarded to each of the linked Resources. All responses to the request shall be aggregated into a single response to the Client using the same representations and status codes as for RETRIEVE operations using the batch OCF Interface.

– Should any one of the Observable linked Resources fail to honour the Observe request the response to the batch Observe request shall also indicate that the entire request was not honoured using the mechanism described in 11.3.2.4.

– If any of the Observable Resources in a request to a Collection using the batch OCF Interface replies with an error or Observe Cancel, the Observations of all other linked Resources shall be cancelled and the error or Observe Cancel status shall be returned to the Observing Client.

NOTE Behaviour may be different for Links that do network requests vs. local Resources.

– All notifications to the Client that initiated an Observe request using the batch OCF Interface shall use the batch representation for the Collection. This is the aggregation of any individual Observe notifications received by the Device hosting the Collection from the individual Observe requests that were forwarded to the linked Resources.

– Linked Resources which are not marked Observable in the Links of a Collection shall not trigger Notifications, but may be included in the response to, and subsequent Notifications resulting from, an Observe request to the batch OCF Interface of a Collection.

– Each notification shall contain the most current values for all of the Linked Resources that would be included if the original Observe request were processed again. The Server hosting the Collection may choose to RETRIEVE all of the linked Resources each time, or may choose to employ caching to avoid retrieving linked Resources on each Notification.

– If a Linked Resource is Observable and has responded with a successful Observe response, the most recently reported value of that Resource is considered to be the most current value and may be reported in all subsequent Notifications.

– Links in the Collection should be Observed by using the "oic.if.ll" OCF Interface. A notification shall be sent any time the contents of the "oic.if.ll" OCF Interface representation are changed; that is, if a Link is added, if a Link is removed, or if a Link is updated. Notifications on the "oic.if.ll" OCF Interface shall contain all of the Links in the "oic.if.ll" OCF Interface representation.

– Other Properties of the Collection Resource, if present, may be Observed by using the OCF Interfaces defined in the definition for the Resource Type, including using the "oic.if.baseline" OCF Interface.

7.6.3.4.4 UPDATE using the batch OCF Interface

When a Collection supports the ability for the linked Resources to be the subject of the UPDATE operation using the batch OCF Interface the following apply:

– A Client shall perform UPDATE operations using the batch OCF Interface by creating a payload that is similar to a RETRIEVE response payload from a batch OCF Interface request. The Server shall send a separate UPDATE request to each of the linked Resources according to each "href" Property and the corresponding value of the "rep" Property.

– Items shall always contain a link-specific "href".

– An UPDATE received by a Server with an empty "href" shall be rejected with a response indicating an appropriate error (e.g. bad request).

– Each linked Resource shall follow the requirements for an UPDATE request may not be supported by the linked Resource. In such cases, writable Properties in the UPDATE operation as defined in clause 8.4.

– The UPDATE response shall contain the updated values using the same payload schema as RETRIEVE operations if provided by the linked Resource, along with the appropriate status code. The aggregated response payload shall reflect the known state of the updated Properties after the batch update was completed. If no payload is provided by the updated Resource, then an empty response (i.e. "rep": {}) shall be provided for that Resource.

– A Collection shall not support the use of the UPDATE operation to add, modify, or remove Links in an existing Collection using the "oic.if.baseline", "oic.if.rw" or "oic.if.a" OCF Interfaces.

– A Collection shall not support the use of the UPDATE operation using the batch OCF Interface when the Collection contains Links that resolve to Resources that are not hosted on the Device that also hosts the Collection. If such a Collection receives an UPDATE operation, the operation shall be rejected with a response indicating an appropriate error (e.g. method not allowed). If the ability to UPDATE linked remote Resources is desired, the use of the optional scene feature (see clause 11.6 in [1]) to effect the UPDATE could be utilized.

7.6.3.4.5 Examples: Batch OCF Interface

Note that the examples provided in Table 9 are illustrative and do not include all mandatory schema elements in all cases. It is assumed that the Default OCF Interface for the Resource Type "x.org.example.rt.room" is specified in its Resource Type definition file as "oic.if.rw", which exposes the Properties "x.org.example.colour" and "x.org.example.size".

Table 9 – Batch OCF Interface example

Resources	/a/room/1
	{
	  "rt": "x.org.example.rt.room"],
	  "if": ["oic.if.rw","oic.if.baseline","oic.if.b","oic.if.ll"],
	  "x.org.example.colour": "blue",
	  "x.org.example.dimension": "15bx15wx10h",
	  "links": [
	  {"href": "/a/room/1", "rel": ["self", "item"], "rt":
	["x.org.example.rt.room"], "if":
	["oic.if.rw","oic.if.baseline","oic.if.b","oic.if.ll"],"p": {"bm": 2} },
	  {"href": "/the/light/1", "rel": ["item"], "rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a","oic.if.baseline"], "ins": "11111", "p": {"bm": 2} },
	  {"href": "/the/light/2", "rel": ["item"], "rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a" ,"oic.if.baseline"], "ins": "22222", "p": {"bm": 2} },
	  {"href": "/my/fan/1", "rel": ["item"], "rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a", "oic.if.baseline"], "ins": "33333", "p": {"bm": 2} },
	  {"href": "/his/fan/2", "rel": ["item"], "rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a", "oic.if.baseline"], "ins": "44444", "p": {"bm": 2} },
	  {"href": "/the/presence/1", "rel": ["item"], "rt":
	"oic.r.sensor.presence"], "if": ["oic.if.s", "oic.if.baseline"], "ins":
	"55555", "p": {"bm": 2} },
	  {"href": "/the/switches/1", "rel": ["item"], "rt": ["oic.wk.col"],
	"if":["oic.if.ll", "oic.if.b", "oic.if.baseline"], "ins": "55555", "p": {"bm":
	2} }
	  ]
	}
	
	/the/light/1
	{
	  "rt": ["oic.r.switch.binary"],
	  "if": ["oic.if.a", "oic.if.baseline"],
	  "value": false
	}
	
	/the/light/2
	{
	  "rt": ["oic.r.switch.binary"],
	  "if": ["oic.if.a", "oic.if.baseline"],
	  "value": true
	}
	
	/my/fan/1
	{
	  "rt": ["oic.r.switch.binary"],
	  "if": ["oic.if.a", "oic.if.baseline"],
	  "value": true
	}
	
	/his/fan/2
	{
	  "rt": ["oic.r.switch.binary"],
	  "if": ["oic.if.a", "oic.if.baseline"],
	  "value": false
	}
	
	/the/presence/1
	{
	  "rt": ["oic.r.sensor.presence"],
	  "if": ["oic.if.s","oic.if.baseline"],
	  "value": false
	}
	
	/the/switches/1
	{

	"rt": ["oic.wk.col"],
	"if":["oic.if.ll", "oic.if.b", "oic.if.baseline"],
	"links": [
	{
	"href": "/switch-1a",
	"rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a","oic.if.baseline"],
	"p": {"bm": 2}
	}
	{
	"href": "/switch-1b",
	"rt": ["oic.r.switch.binary"],
	"if": ["oic.if.a","oic.if.baseline"],
	"p": {"bm": 2 }
	}
	]
	}

Use of batch, successful response	Request: GET /a/room/1?if=oic.if.b Becomes the following individual request messages issued by the Device in the Client role GET /a/room/1 (NOTE: uses the Default OCF Interface as specified for the Collection Resource, in this example oic.if.rw) GET /the/light/1 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /the/light/2 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /my/fan/1 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /his/fan/2 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /the/presence/1 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /the/switches/1 (NOTE: Uses the Default OCF Interface for the Collection that is within the Collection) Response: [{ "href": "/a/room/1", "rep": {"x.org.example.colour": "blue","x.org.example.dimension": "15bx15wx10h"} }, { "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": true} }, { "href": "/my/fan/1", "rep": {"value": true} }, { "href": "/his/fan/2", "rep": {"value": false} }, { "href": "/the/presence/1", "rep": {"value": false} }, { "href": "/the/switches/1", "rep": [{ "href": "/switch-1a", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a","oic.if.baseline"], "p": {"bm": 2}, "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}] }, { "href": "/switch-1b", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a","oic.if.baseline"], "p": {"bm": 2 }, "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]

	`}  ]`
	`}`
	`]`

Use of batch, error response	Should any of the RETRIEVE requests in the previous example fail then the response includes an empty payload for that Resource instance and an error code is sent. The following example assumes errors from "/my/fan/1" and "/the/switches/1" Error Response: `[{ "href": "/a/room/1", "rep": {"x.org.example.colour": "blue","x.org.example.dimension": "15bx15wx10h"} }, { "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": true} }, { "href": "/my/fan/1", "rep": {} }, { "href": "/his/fan/2", "rep": {"value": false} }, { "href": "/the/presence/1", "rep": {"value": false} }, { "href": "/the/switches/1", "rep": {} }]`

Use of batch (UPDATE has POST semantics)	UPDATE /a/room/1?if=oic.if.b
	[
	  {
	  "href": "/the/light/1",
	  "rep": {
	  "value": false
	  }
	  },
	  {
	  "href": "/the/light/2",
	  "rep": {
	  "value": true
	  }
	  },
	  {
	  "href": "/a/room/1",
	  "rep": {
	  "x.org.example.colour": "red"
	  }
	  }
	]
	
	This turns /the/light/1 off, turns /the/light/2 on, and sets the colour of /a/room/1 to "red".
	
	The response will be same as response for GET /a/room/1?if=oic.if.b with the updated Property values as shown.
	[
	  {
	  "href": "/a/room/1",
	  "rep":{"x.org.example.colour": "red",
	  "x.org.example.dimension": "15bx15wx10h"}
	  },
	  {
	  "href": "/the/light/1",
	  "rep": {"value": false}
	  },
	  {
	  "href": "/the/light/2",
	  "rep": {"value": true}
	  }
	]
	Example use of additional query parameters to select items by matching Link Parameters.
	
	Retrieving all items that are Presence Sensors ("oic.r.sensor.presence"):
	RETRIEVE /a/room/1?if=oic.if.b&rt=oic.r.sensor.presence
	
	Response payload:
	[
	  {
	  "href": "/the/presence/1",
	  "rep": {
	  "value": false
	  }
	  }
	]

7.6.3.5 Actuator OCF Interface

The actuator OCF Interface is the OCF Interface for viewing Resources that may be actuated i.e. changes some value within or the state of the entity abstracted by the Resource:

– The actuator OCF Interface name shall be "oic.if.a"

– The actuator OCF Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable OpenAPI 2.0 schema; the actuator OCF Interface may also expose in the Resource Representation optional Properties as defined by the applicable OpenAPI 2.0 schema that are implemented by the target Device.

For example, a "Heater" Resource (for illustration only):

```
/a/act/heater
{
   "rt": ["x.com.acme.gas"],
   "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
   "x.com.acme.settemp": 10,
   "x.com.acme.currenttemp" : 7
}
```

The actuator OCF Interface with respect to "Heater" Resource (for illustration only):

a) Retrieving values of an actuator.

```
Request: RETRIEVE /a/act/heater?if="oic.if.a"

Response: Content
Payload:
{
  "x.com.acme.settemp": 10,
  "x.com.acme.currenttemp" : 7
}
```

b) Correct use of actuator OCF Interface.

```
Request: UPDATE /a/act/heater?if="oic.if.a"
{
  "x.com.acme.settemp": 20
}
Response: Changed
Payload:
{
  "x.com.acme.settemp": 20
}
```

c) Incorrect use of actuator OCF Interface.

```
Request: UPDATE /a/act/heater?if="oic.if.a"
{
  "if": ["oic.if.s"]   ← this is visible through baseline OCF Interface
}
Response:Bad Request
Payload:
{
}
```

– A RETRIEVE request using this OCF Interface shall return the Representation for this Resource as defined by the applicable OpenAPI 2.0 schema, subject to any query parameters that may also be defined as part of the applicable OpenAPI 2.0 schema.

– An UPDATE request using this OCF Interface shall provide a payload or body that contains the Properties that will be updated on the target Resource.

7.6.3.6 Sensor OCF Interface

The sensor OCF Interface is the OCF Interface for retrieving measured, sensed or capability specific information from a Resource that senses:

– The sensor OCF Interface name shall be "oic.if.s".

– The sensor OCF Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable OpenAPI 2.0 schema; the sensor OCF Interface may also expose in the Resource Representation optional Properties as defined by the applicable OpenAPI 2.0 schema that are implemented by the target Device.

– A RETRIEVE request using this OCF Interface shall return this representation for the Resource as defined by the applicable OpenAPI 2.0 schema, subject to any query parameters that may also be defined as part of the applicable OpenAPI 2.0 schema.

NOTE: The example here is with respect to retrieving values of a sensor

```
Request: RETRIEVE /a/act/heater?if="oic.if.s"

Response: Content
Payload:
{
  "x.com.acme.currenttemp": 7
}
```

Incorrect use of the sensor.

```
Request: UPDATE /a/act/heater?if="oic.if.s" ← UPDATE is not allowed
{
  "x.com.acme.settemp": 20   ← this is possible through actuator OCF Interface
}
Response: Bad Request
Payload:
{
}
```

Another incorrect use of the sensor.

```
Request: UPDATE /a/act/heater?if="oic.if.s" ← UPDATE is not allowed
{
  "x.com.acme.currenttemp": 15   ← this is not possible to be updated
}
Response: Bad Request
Payload:
{
}
```

7.6.3.7 Read-only OCF Interface

The read-only OCF Interface exposes only the Properties that may be read. This includes Properties that may be read-only, read-write but not Properties that are write-only or set-only. The applicable operations that can be applied to a Resource are only RETRIEVE and NOTIFY. An attempt by a Client to apply a method other than RETRIEVE or NOTIFY to a Resource shall be rejected with an error response code.

The read-only OCF Interface with respect to "Heater" Resource (for illustration only):

```
Request: RETRIEVE /a/act/heater?if="oic.if.r"
Response: Content
Payload:
{
  "x.com.acme.settemp": 10,
```



```
  "x.com.acme.currenttemp" : 7
}
```

7.6.3.8 Read-write OCF Interface

The read-write OCF Interface is a generic OCF Interface to support reading and setting Properties in a Resource. The applicable methods that can be applied to a Resource are only RETRIEVE, NOTIFY, and UPDATE. For the RETRIEVE and NOTIFY operations, the behaviour is the same as for the "oic.if.r" OCF Interface defined in 7.6.3.7. For the UPDATE operation, read-only Properties (i.e. Properties tagged with "readOnly=true" in the OpenAPI 2.0 definition) shall not be in the UPDATE payload. An attempt by a Client to apply a method other than RETRIEVE, NOTIFY, or UPDATE to a Resource shall be rejected with an error response code.

For example, a "Grinder" Resource (for illustration only):

```
/a/mygrinder
{
  "rt": ["oic.r.grinder"],
  "if": ["oic.if.rw", "oic.if.baseline"],
  "coarseness": 10,
  "remaining": 50
}
```

The read-write OCF Interface with respect to “Grinder" Resource (for illustration only):

a) Retrieving the value with read-write OCF Interface

```
Request: RETRIEVE /a/mygrinder?if="oic.if.rw"

Response: Content
Payload:
{
  "coarseness": 10,
  "remaining": 50
}
```

b) Updating the value with read-write OCF Interface

```
Request: UPDATE /a/mygrinder?if="oic.if.rw"
{
  "coarseness": 20
}

Response: Changed
Payload:
{
  "coarseness": 20
}
```

7.6.3.9 Create OCF Interface

7.6.3.9.1 Overview

The create OCF Interface is used to create Resource instances in a Collection. An instance of a Resource and the Link pointing to the Resource are created together, atomically, according to a Client-supplied representation. The create OCF Interface name is "oic.if.create". A Collection which exposes the "oic.if.create" OCF Interface shall expose the "rts" Property (see clause 7.8.2.8) with all Resource Types that can be hosted with the Collection. If a Client attempts to create a Resource Type which is not supported by the Collection, the Server shall return an appropriate error status code, for example "Bad Request". Successful CREATE operations shall return a success code, i.e.

"Created". The IDD for all allowed Resource Types that may be created shall adhere to Introspection for dynamic Resources (see clause11.4).

7.6.3.9.2 Data format for CREATE

The data format for the create OCF Interface is similar to the data format for the batch OCF Interface. The create OCF Interface format consists of a set of Link Parameters and a "rep" Parameter which contains a representation for the created Resource.

The representation supplied for the Link pointing to the newly created Resource shall contain at least the "rt" and "if" Link Parameters.

The Link Parameter "p" should be included in representations supplied for all created Resources. If the "Discoverable" bit is set, then the supplied Link representation shall be exposed in "/oic/res" of the Device on which the Resource is being created. The Link Parameters representation in the "/oic/res" Resource does not have to mirror the Link Parameters in the Collection of the created Resource (e.g., "ins" Parameter).

Creating a discoverable Resource is the only way to add a Link to "/oic/res".

If the "p" Parameter is not included, the Server shall create the Resource using the default settings of not discoverable, and not observable.

The representation supplied for a created Resource in the value of the "rep" Parameter shall contain all mandatory Properties required by the Resource Type to be created excluding the Common Properties "rt" and "if" as they are already included in the create payload.

Note that the "rt" and "if" Property Values are created from the supplied Link Parameters of the Resource creation payload.

If the supplied representation does not contain all of the required Properties and Link Parameters, the Server shall return an appropriate error status code, for example "Bad Request".

An example of the create OCF Interface payload is as illustrated:

```
{
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a","oic.if.baseline"],
  "p": {"bm":3},
  "rep": {
    "temperature": 20
  }
}
```

The representation returned when a Resource is successfully created shall contain the "href", "if", and "rt" Link Parameters and all other Link Parameters that were included in the CREATE operation. In addition, the "rep" Link Parameter shall include all Resource Properties as well as the "rt" and "if" Link Parameters supplied in the CREATE operation. The Server may include additional Link Parameters and Properties in the created Resource as required by the application-specific Resource Type. The Server shall assign an "ins" value to each created Link and shall include the "ins" Parameter in the representation of each created Link as illustrated in the Collection that the Link of the created Resource was created within:

```
{
  "href": "/3755f3ac",
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a","oic.if.baseline"],
  "ins": 39724818,
  "p": {"bm":3},
  "rep": {
    "rt": ["oic.r.temperature"],
```



```
    "if": ["oic.if.a","oic.if.baseline"],
    "temperature": 20
  }
}
```

The Link Parameters representation in the "/oic/res" Resource, if the created Resource is discoverable, may not mirror exactly all the Link Parameters added in the Collection; except it shall expose at a minimum the mandatory Properties of the Link (i.e., "rt", "if", and "href") of the created Resource.

7.6.3.9.3 Use with CREATE

The CREATE operation shall be sent to the URI of the Collection in which the Resource is to be created. The query string "?if=oic.if.create" shall be included in all CREATE operations.

The Server shall generate a URI for the created Resource and include the URI in the "href" Parameter of the created Link.

When a Server successfully completes a CREATE operation using the "oic.if.create" OCF Interface addressing a Collection, the Server shall automatically modify the ACL Resource to provide initial authorizations for accessing for the newly created Resource according to ISO/IEC 30118-2.

An example performing a CREATE operation is as illustrated:

```
CREATE /scenes/scene1?if=oic.if.create
{
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a","oic.if.baseline"],
  "p": {"bm":3},
  "rep": {
    "temperature": 20
  }
}
Response: Created
Payload:
{
  "href": "/3755f3ac",
  "ins": 39724818,
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a","oic.if.baseline"],
  "p": {"bm":3},
  "rep": {
    "rt": ["oic.r.temperature"],
    "if": ["oic.if.a","oic.if.baseline"],
    "temperature": 20
  }
}
```

7.6.3.9.4 Use with UPDATE and DELETE

The UPDATE and DELETE operations are not allowed by the create OCF Interface. Attempts to perform UPDATE or DELETE operations using the create OCF Interface shall return an appropriate error status code, for example "Method Not Allowed", unless the UPDATE and CREATE operations map to the same transport binding method (e.g., CoAP with the POST method). In that situation where the UPDATE and CREATE operations map to the same transport binding method, this shall be processed as a CREATE operation according to clause 7.6.3.9.3.

7.6.3.10 Write-Only OCF Interface

The write-only OCF Interface is the OCF Interface for updating Property Values of a Resource that can’t be read back later i.e. changes some value within, or the state of the entity abstracted by the

Resource, but is not reflected as an OCF RESTful Property that can be read back with a RETRIEVE operation:

– The write-only OCF Interface name shall be "oic.if.w".
– The write-only OCF Interface shall use in the Resource Representation all mandatory Properties as defined by the OpenAPI 2.0 schema in the Introspection Device Data; the write-only OCF Interface may also use in the Resource Representation optional Properties as defined by the applicable OpenAPI 2.0 schema that are implemented by the target Device.
– A RETRIEVE request using the "oic.if.w" OCF Interface shall be rejected with an error response code.
– An UPDATE request using the "oic.if.w" OCF Interface may provide a Resource Representation that contains the Properties that are to be UPDATED on the target Resource. A Resource Representation is included as a payload if the Resource Type defines the use thereof for UPDATE using "oic.if.w" in the applicable OpenAPI 2.0 schema for the Resource Type. Depending on the actual function of the Resource the request and/or response payload of the UPDATE request may have no properties defined.

Use of a RETRIEVE using the baseline OCF Interface on a Resource that exposes "oic.if.w" shall only return the Common Properties see Clause 7.3.2. A minimal implementation of a RETRIEVE response using the baseline OCF Interface on a Resource that supports "oic.if.w" shall only contain the Properties "if" and "rt".

Resource Types that have the write-only OCF Interface defined shall not be part of a Multi-value "rt" Resource. See [7.4.4].

For example (for illustration only): setting a binary value to its opposite can be achieved by sending a toggle request via an input argument (request payload) of an UPDATE operation using the write-only OCF Interface, the defined return arguments (in this case on the assumption that the Resource Type has defined a response payload) in the response payload contains information on the success or otherwise of the toggle operation:

```
Request: UPDATE /a/toggle?if=oic.if.w
{
  "do_toggle": true
}

Response: Changed
Payload:
{
  "toggled": "ok"
}
```

7.6.3.11 Property Start-up and Property Revert OCF Interfaces

7.6.3.11.1 Overview

The Property start-up and Property revert OCF Interfaces allow a Client to determine the values that will be set on a Resource following power-up of the Device. The Property start-up OCF Interface allows for the establishment of a specific value that will be set on power-up. The Property revert OCF Interface allows for the Property value on power-up to be the same as the value last set before power-down.

7.6.3.11.2 Support on a Device

A Device shall only expose "oic.if.startup" as part of the "if" Link Parameter or "if" Common Property for a Resource if that Resource also supports either the "oic.if.a" or "oic.if.rw" OCF Interfaces.

A Device shall only expose "oic.if.startup.revert" as part of the "if" Link Parameter or "if" Common Property for a Resource if that Resource also supports either the "oic.if.a" or "oic.if.rw" OCF Interfaces.

Neither the "oic.if.startup" OCF Interface nor the "oic.if.startup.revert" OCF Interface shall be exposed for a Resource that is an Atomic Measurement or that follows the Collection pattern.

7.6.3.11.3 Governing State Machine

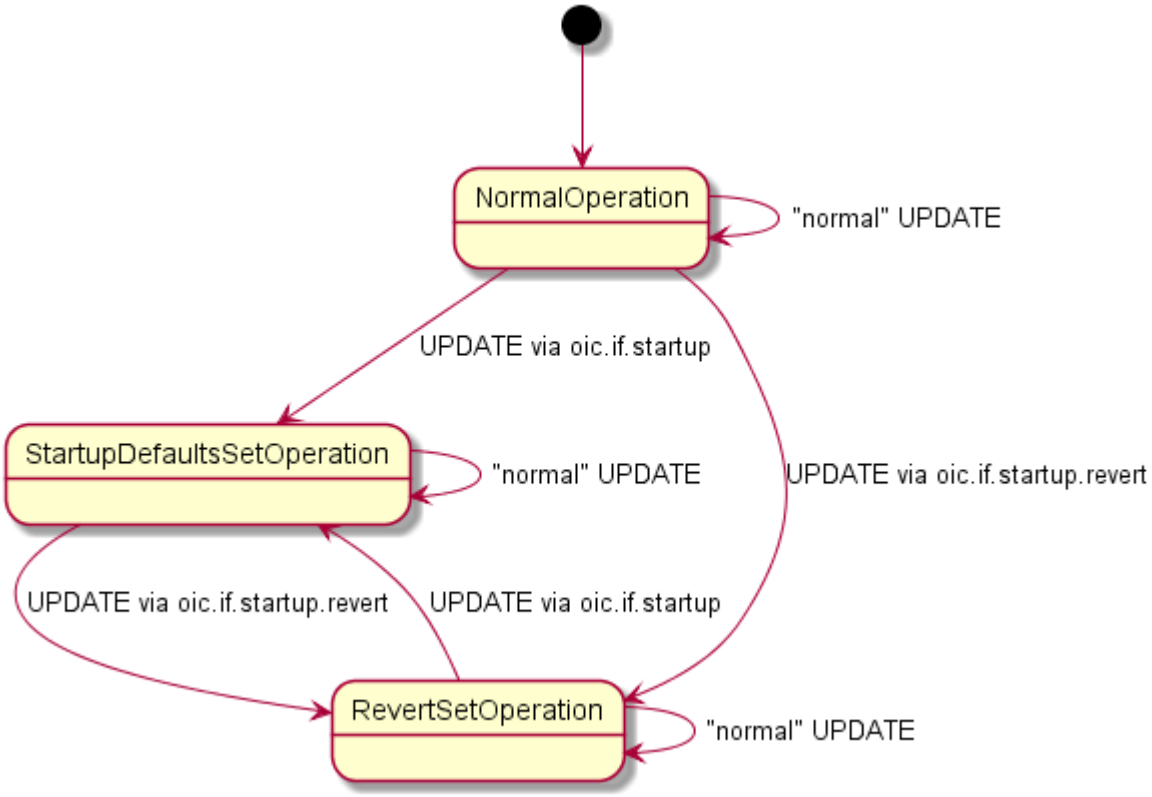


Figure 5 illustrates the state machine supported by all Resources that exposes both the "oic.if.startup" and "oic.if.startup.revert" OCF Interfaces. A Device that supports only one of the defined OCF Interfaces will support only the state transitions appropriate to the OCF Interface that is supported. A Client needs to have an observe relationship established using both the "oic.if.startup" and "oic.if.startup.revert" OCF Interfaces for it to be fully aware of all state transitions that are defined. Note that the entry point into the state machine may be any of the supported states depending on the initial values supported by an implementation (see clause 7.6.3.11.4).

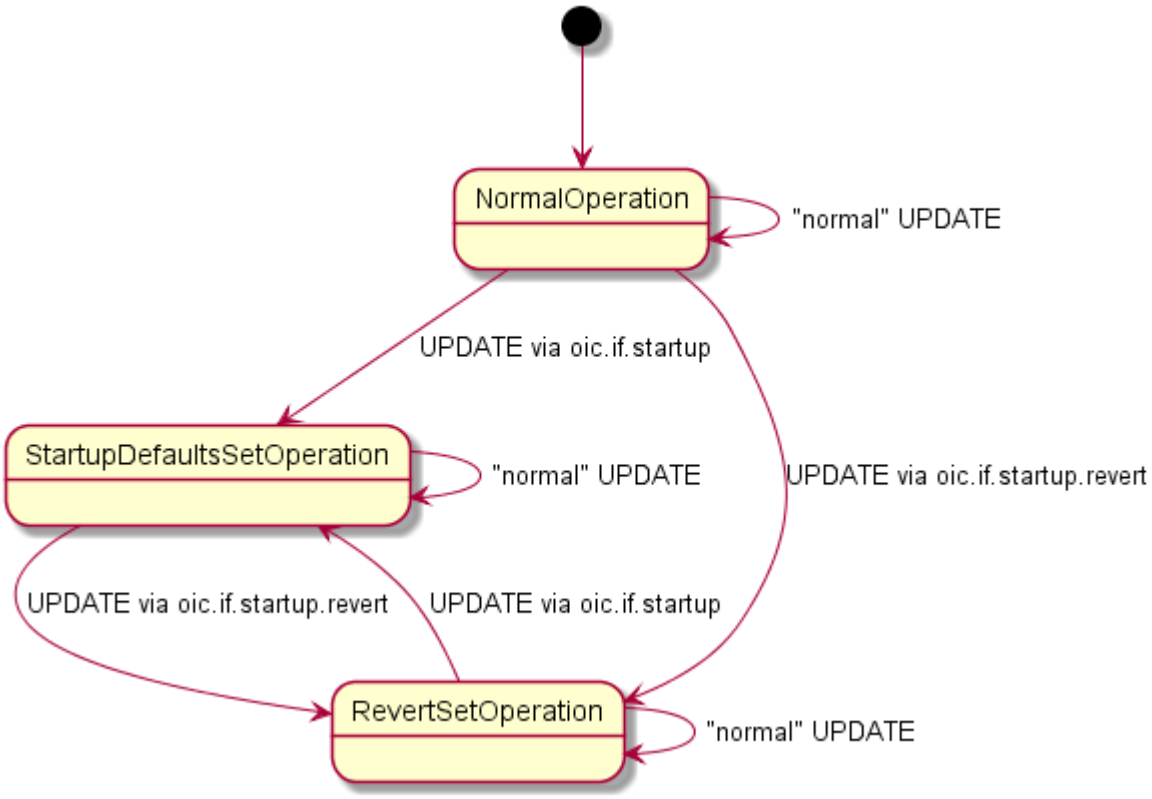


Figure 5 – Property Start-up Defaults/Revert State Machine

Thus on a power cycle:

- If in "NormalOperation' Property values are implementation dependent.
- If in "StartupDefaultsSetOperation' Property Values shall be those set via the last UPDATE operation using an OCF Interface of "oic.if.startup"; values for Properties that are not included in the UPDATE operation are implementation dependent.
- If in "RevertSetOperation" Property Values shall be those prior to the power cycle.
- In all cases, the state of the Device shall be persisted across the power cycle

7.6.3.11.4 Use of RETRIEVE

A Device that supports the "oic.if.startup" or "oic.if.startup.revert" OCF Interfaces may have initial Property values for a Resource for either of these OCF Interfaces, and is thus in either the StartupDefaultsSetOperation or RevertSetOperation state from the point of initial power-up.

If the Device is in StartupDefaultsSetOperation state as defined in clause 7.6.3.11.3, then on reception of a RETRIEVE using the "oic.if.startup" OCF Interface the Device shall respond with success response and a payload containing the currently set default values for the Resource. On reception of a RETRIEVE using the "oic.if.startup" OCF Interface when not in the StartupDefaultsSetOperation state the Device shall respond with a failure response (e.g. service unavailable).

If the Device is in RevertSetOperation state as defined in clause 7.6.3.11.3 then on reception of a RETRIEVE using the "oic.if.startup.revert" OCF Interface the Device shall respond with a "Valid" response with no payload. On reception of a RETRIEVE using the "oic.if.startup.revert" OCF Interface when not in the RevertSetOperation state the Device shall respond with a failure response (e.g. service unavailable).

7.6.3.11.5 Use of UPDATE and Property Start-up Interface

When a Device receives an UPDATE that uses the "oic.if.startup" OCF Interface, the Device shall set the start-up default values for all Properties that are in the payload to be those provided in the payload, the Device shall not change current values of the Resource.

Further, reception of such an UPDATE also moves the Resource into the StartupDefaultsSetOperation state as defined in clause 7.6.3.11.3; such that on completion of a power cycle the Resource shall be populated with the values that were set as defaults, for those Properties that weren't explicitly set by such an UPDATE, their value is implementation dependent.

7.6.3.11.6 Use of UPDATE and Property Revert Interface

When a Device receives an UPDATE that uses the "oic.if.startup.revert" OCF Interface, the Device shall set the current value for all Properties that are in the payload to be those provided in the payload and moves the Resource into the RevertSetOperation state as defined in clause 7.6.3.11.3; such that on completion of a power cycle the Resource shall be populated with the values that were present prior to the initiation of the power cycle.

7.6.3.11.7 Observability of Property Defaults and Property Revert Interfaces

A Client may Observe a Resource using either of the "oic.if.startup" or "oic.if.startup.revert" OCF Interfaces.

When Observing using "oic.if.startup" then the Server shall send a NOTIFICATION whenever there is a change to the Resource using the "oic.if.startup" OCF Interface.

When Observing using "oic.if.startup.revert" the Server shall send a NOTIFICATION whenever there is any change to the Resource as the revert behaviour in essence is a tracker for the currently set Property value(s).

7.7 Resource representation

Resource representation captures the state of a Resource at a particular time. The Resource representation is exchanged in the request and response interactions with a Resource. A Resource representation may be used to retrieve or update the state of a Resource.

The Resource representation shall not be manipulated by the data connectivity protocols and technologies (e.g., CoAP, UDP/IP or BLE).

7.8 Structure

7.8.1 Introduction

In many scenarios and contexts, the Resources may have either an implicit or explicit structure between them. This may be achieved through the use of Collection (7.8.3) and Atomic Measurement (7.8.4) Resources.

7.8.2 Resource relationships (Links)

7.8.2.1 Introduction

Resource relationships are expressed as Links. A Link is a hyperlink, which defines a typed connection between two Resources. Hyperlinks, or web links, have the following components as defined in IETF RFC 8288:

– Link context (URI reference) as defined in 7.8.2.2

– Link relation type as defined in 7.8.2.3

– Link target (URI reference) as defined in 7.8.2.4

– Link target attributes as defined in 7.8.2.5

The Link context is the Resource with which the Link is associated. A Link is viewed as a statement of the form "(Link context) has a (Link relation type) to a Resource at (Link target), which has (Link target attributes)" as per IETF RFC 8288 clause 2.

To paraphrase, the Link target is related to the Link context according to the Link relation type. Additionally, the Link target attributes make semantic statements about the Link target, to identify the content type, physical location, etc.

Links conform to the definitions in IETF RFC 8288, with an example JSON serialization with associated Link Parameters as illustrated:

```
{
  "anchor": "/some/ocf/resource",     // Link context, optional
  "rel": ["hosts"],                   // Link relation Type, optional
  "href": "/some/other/ocf/resource", // Link target, required
  "p": {"bm": 3},                     // Link target attributes, optional
  "if": ["oic.if.baseline"],          // Link target attributes, required
  "rt": ["oic.r.sensor"]              // Link target attributes, required
}
```

Additional items in the Link may be made mandatory based on the use of the Links in different contexts (e.g. in Collections, in discovery, in bridging etc.). The OpenAPI 2.0 file for the Link payload is detailed in Annex A.

Another example of a Link is as illustrated:

```
{"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a",
"oic.if.baseline"], "p": {"bm": 3}, "rel": "item"}
```

7.8.2.2 Link context

The Link context is defined in the Link using the "anchor" Parameter. If the Link doesn't contain an "anchor" Parameter, the Link context shall be the Resource from which the Link was retrieved.

7.8.2.3 Link relation type

The Link relation type conveys the semantics of the Link. The Link relation type is defined in the Link using the "rel" Parameter. If the Link doesn't contain a "rel" Parameter, the Link relation type shall be assumed to have the default value "hosts", which means that the Resource at the Link target is "hosted" by the Resource at the Link context. The set of Link relation types to be used to describe various relationships between Resources are as listed:

- "hosts"
  - The Link target points to a Resource that is hosted at the Link context. This Link relation type indicates that the Resource is allowed to be included in the batch representations of the Link target. This Link relation type is defined by IETF RFC 6690.
- "self"
  - The Link refers to the Link context, which allows a Link to describe the Resource at the Link context, which is to say that the Link can describe the Collection or Atomic Measurement Resource that the Link is retrieved from. The Link target points to the Link context, and the Link target attributes describe the Link context. This Link relation type is defined by IETF RFC 4287.
- "item"
  - The Link target points to a Resource that is a member of the Collection or Atomic Measurement at the Link context, which might not specifically be hosted by the Collection or Atomic Measurement Resource, and is allowed to be contained in batch representations of the Collection or Atomic Measurement. An example is using "rel": "item" to declare that

the Properties of the Collection or Atomic Measurement Resource itself should be included in a batch representation of the Collection or Atomic Measurement. This Link relation type is defined by IETF RFC 6573.

All of these Link relation types are registered in the IANA Registry for Link relations types defined in IANA Link Relations. Other Link relation types may be included in Links, provided that they conform to the requirements in IETF RFC 8288. Other Link relation types may be defined for features contained in other specifications and may not be included in what is defined in this clause. The presence of Link relation types not defined in this document does not affect the processing of Link relation types defined in this document.

When there is more than one Link relation type value in a Link, all of the values apply to describe the relationship between the Link context and the Link target. A Link with multiple Link relation type values is equivalent to a set of Links having the same Link context and Link target, each having one of the Link relation values.

7.8.2.4 Link target

The Link target is a URI reference to a Resource using the "href" Parameter.

7.8.2.5 Parameters for Link target attributes

7.8.2.5.1 Introduction

Link target attributes are specialisations of Link Parameters. Table 10 lists all the Link target attributes defined in this document.

Table 10 – Link target attributes list

Parameter title	Parameter name	Mandatory	Description
Device UUID	"di"	No	Defined in clause 7.8.2.5.5
OCF Endpoint information	"eps"	No	Defined in clause 7.8.2.5.6
OCF Interface	"if"	Yes	Defined in clause 7.6
Link instance	"ins"	No	Defined in clause 7.8.2.5.2
Policy	"p"	No	Defined in clause 7.8.2.5.3
Resource Type	"rt"	Yes	Defined in clause 7.4
Media type	"type"	No	Defined in clause 7.8.2.5.4
Position description Semantic Tag	"tag-pos-desc"	No	Defined in clause 11.5.2.1.2
Relative position Semantic Tag	"tag-pos-rel"	No	Defined in clause 11.5.2.1.3
Function description Semantic Tag	"tag-func-desc"	No	Defined in clause 11.5.2.2.2
Location description Semantic Tag	"tag-locn"	No	Defined in clause 11.5.2.3.2

Note: Other Link target attributes may to defined for features in other specifications and may not be included in this table.

7.8.2.5.2 "ins" or Link instance Parameter

The "ins" Parameter identifies a particular Link instance in a list of Links. The "ins" Parameter may be used to modify or delete a specific Link in a list of Links. The value of the "ins" Parameter is set

at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has been set, the "ins" Parameter shall not be modified for as long as the Link is a member of that list.

7.8.2.5.3 "p" or policy Parameter

The policy Parameter defines various rules for correctly accessing a Resource referenced by a target URI. The policy rules are configured by a set of key-value pairs.

The policy Parameter "p" is defined by:

– "bm" key: The "bm" key corresponds to an integer value that is interpreted as an 8-bit bitmask. Each bit in the bitmask corresponds to a specific policy rule. The rules are specified for "bm" in Table 11:

Table 11 – "bm" Property definition

Bit Position	Policy rule	Comment
Bit 0 (the LSB)	discoverable	The discoverable rule defines whether the Link is to be included in the Resource discovery message via "/oic/res". If the Link is to be included in the Resource discovery message, then "p" shall include the "bm" key and set the discoverable bit to value 1. If the Link is NOT to be included in the Resource discovery message, then "p" shall either include the "bm" key and set the discoverable bit to value 0 or omit the "bm" key entirely.
Bit 1 (2nd LSB)	observable	The Observable rule defines whether the Resource referenced by the target URI supports the NOTIFY operation using an observe pattern (see section 11.3.2).. With the self-link, i.e. the Link with "rel" value of "self", "/oic/res" can have a Link with the target URI of "/oic/res" and indicate itself Observable. The "self" is defined by IETF RFC 4287 and registered in the IANA Registry for "rel" value defined at IANA Link Relations. • If the Resource supports the NOTIFY operation via the use of observe, then "p" shall include the "bm" key and set the Observable bit to value 1. • If the Resource does NOT support the NOTIFY operation via the use of observe, then "p" shall either include the "bm" key and set the Observable bit to value 0 or omit the "bm" key entirely.
Bit 2 (3rd LSB)	pushable	The pushable bit defines whether the Resource referenced by the target URI supports the NOTIFY operation using a push pattern (see section 11.4.1). With the self-link, i.e. the Link with "rel" value of "self", “/oic/res” can have a Link with the target URI of “/oic/res” and indicates itself pushable. The "self" is defined by IETF RFC 4287 and registered in the IANA Registry for "rel" value defined at IANA Link Relations. • If the Resource supports the NOTIFY operation via the use of push, then ”p” shall include the “bm” key and set the pushable bit to value 1. • If the Resource does NOT support the NOTIFY operation via the use of push, then “p” shall either include the “bm” key and set the pushable bit to value 0 or omit the “bm” key entirely.
Bits 2-7	--	Reserved for future use. All reserved bits in "bm" shall be set to value 0.

NOTE If all the bits in "bm" are defined to value 0, then the "bm" key may be omitted entirely from "p" as an efficiency measure. However, if any bit is set to value 1, then "bm" shall be included in "p" and all the bits shall be defined appropriately.

– In a payload sent in response to a request that includes an OCF-Accept-Content-Format-Version option the "eps" Parameter shall provide the information for an encrypted connection.
– Note that access to the Resource is controlled by the ACL for the Resource. A successful encrypted connection does not ensure that the requested action will succeed. See ISO/IEC 30118-2 clause 12 for more information.

This shows the policy Parameter for a Resource that is discoverable but not Observable.

```
"p": {"bm": 1}
```

This shows a self-link, i.e. the "/oic/res" Link in itself that is discoverable and Observable.

```
{
  "href": "/oic/res",
  "rel": "self",
  "rt": ["oic.wk.res"],
  "if": ["oic.if.ll", "oic.if.baseline"],
  "p": {"bm": 3}
}
```

7.8.2.5.4 "type" or media type Parameter

The "type" Parameter may be used to specify the various media types that are supported by a specific target Resource. The default type of "application/vnd.ocf+cbor" shall be used when the "type" element is omitted. Once a Client discovers this information for each Resource, it may use one of the available representations in the appropriate header field of the Request or Response.

7.8.2.5.5 "di" or Device UUID Parameter

The "di" Parameter specifies the Device UUID of the Device that hosts the target Resource defined in the in the "href" Parameter.

The Device UUID may be used to qualify a relative reference used in the "href" or to lookup OCF Endpoint information for the relative reference.

7.8.2.5.6 "eps" Parameter

The "eps" Parameter indicates the OCF Endpoint information of the target Resource.

A Device shall populate all exposed "eps" Link Parameters with an array of items representing OCF Endpoint information as specified in 10.2. Each entry in that array shall include an "ep" Property, and may include the optional "pri" and "lat" Properties.

This is an example of "eps" with multiple OCF Endpoints.

```
"eps": [
  {"ep": "coap://[fe80::b1d6]:1111", "pri": 2, "lat": 240},
  {"ep": "coaps://[fe80::b1d6]:1122", "lat": 240},
  {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
]
```

When "eps" is present in a link, the OCF Endpoint information in "eps" can be used to access the target Resource referred by the "href" Parameter.

Note that the type of OCF Endpoint – Secure or Unsecure – that a Resource exposes merely determines the connection type(s) guaranteed to be available for sending requests to the Resource. For example, if a Resource only exposes a single CoAP "ep", it does not guarantee that the Resource cannot also be accessed via a Secure OCF Endpoint (e.g. via a CoAPS "ep" from another Resource's "eps information). Nor does exposing a given type of OCF Endpoint ensure that access to the Resource will be granted using the "ep" information. Whether requests to the Resource are granted or denied by the Access Control layer is separate from the "eps" information, and is determined by the configuration of the /acl2 Resource (see ISO/IEC 30118-2 clause 13.5.3 for details).

When present, max-age information (e.g. Max-Age option for CoAP defined in IETF RFC 7252) determines the maximum time "eps" values may be cached before they are considered stale.

7.8.2.6 Formatting

When formatting in JSON, the list of Links shall be an array.

7.8.2.7 List of Links in a Collection

A Resource that exposes one or more Properties that are defined to be an array of Links where each Link can be discretely accessed is a Collection. The Property Name "links" is recommended for such an array of Links.

This is an example of a Resource with a list of Links.

```
/Room1
{
  "rt": ["oic.wk.col"],
  "if": ["oic.if.ll", "oic.if.baseline" ],
  "color": "blue",
  "links":
  [
    {
      "href": "/switch",
      "rt": ["oic.r.switch.binary"],
      "if": [ "oic.if.a", "oic.if.baseline" ],
      "p": {"bm": 3}
    },
    {
      "href": "/brightness",
      "rt": ["oic.r.light.brightness"],
      "if": [ "oic.if.a", "oic.if.baseline" ],
      "p": {"bm": 3}
    }
  ]
}
```

7.8.2.8 Properties describing an array of Links

If a Resource Type that defines an array of Links (e.g. Collections, Atomic Measurements) has restrictions on the "rt" values that can be within the array of Links, the Resource Type will define the "rts" Property. The "rts" Property as defined in Table 12 will include all "rt" values allowed for all Links in the array. If the Resource Type does not define the "rts" Property or the "rts" Property is an empty array, then any "rt" value is permitted in the array of Links.

For all instances of a Resource Type that defines the "rts" Property, the "rt" Link Parameter in every Link in the array of Links shall be one of the "rt" values that is included in the "rts" Property.

Table 12 – Resource Types Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Types	"rts"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	No	An array of Resource Types that are supported within an array of Links exposed by a Resource.

If a Resource Type that defines an array of Links has "rt" values which are required to be in the array, the Resource Type will define the "rts-m" Property, as defined in Table 13, which will contain

all of the "rt" values that are required to be in the array of Links. If "rts-m" is defined, and "rts" is defined and is not an empty array, then the "rt" values present in "rts-m" will be part of the values present in "rts". Moreover, if the "rts-m" Property is defined, it shall be mandated (i.e. included in the "required" field of a JSON definition) in the Resource definition and Introspection Device Data (see11.4).

For all instances of a Resource Type that defines the "rts-m" Property, there shall be at least one Link in the array of Links corresponding to each one of the "rt" values in the "rts-m" Property; for all such Links the "rt" Link Parameter shall contain at least one of the "rt" values in the "rts-m" Property.

Table 13 – Mandatory Resource Types Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Mandatory Resource Types	"rts-m"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	No	An array of Resource Types that are mandatory to be exposed within an array of Links exposed by a Resource.

7.8.3 Collections

7.8.3.1 Overview

A Resource that contains one or more references (specified as Links) to other Resources is a Collection. These references may be related to each other or just be a list; the Collection provides a means to refer to this set of references with a single handle (i.e. the URI). A simple Resource is kept distinct from a Collection. Any Resource may be turned into a Collection by binding Resource references as Links. Collections may be used for creating, defining or specifying hierarchies, indexes, groups, and so on.

A Collection shall have at least one Resource Type and at least one OCF Interface bound at all times during its lifetime. During creation time of a Collection the Resource Type and OCF Interfaces are specified. The initial defined Resource Types and OCF Interfaces may be updated during its life time. These initial values may be overridden using mechanism used for overriding in the case of a Resource. Additional Resource Types and OCF Interfaces may be bound to the Collection at creation or later during the lifecycle of the Collection.

A Collection shall define a Property that is an array with zero or more Links. The target URIs in the Links may reference another Collection or another Resource. The referenced Collection or Resource may reside on the same Device as the Collection that includes that Link (called a local reference) or may reside on another Device (called a remote reference). The context URI of the Links in the array shall (implicitly) be the Collection that contains that Property. The (implicit) context URI may be overridden with explicit specification of the "anchor" Parameter in the Link where the value of "anchor" is the new base of the Link.

A Resource may be referenced in more than one Collection, therefore, a unique parent-child relationship is not guaranteed. There is no pre-defined relationship between a Collection and the Resource referenced in the Collection, i.e., the application may use Collections to represent a relationship but none is automatically implied or defined. The lifecycles of the Collection and the referenced Resource are also independent of one another.

In the following example a Property "links" represents the list of Links in a Collection. The "links" Property has, as its value, an array of items and each item is a Link.

```
/my/house   ← This is URI of the Resource
{
```



```
  "rt": ["my.r.house"],   ← This and the next 3 lines are the Properties of the
Resource.
  "color": "blue",
  "n": "myhouse",
  "links": [
    {   ← This and the next 4 lines are the Parameters of a Link
      "href": "/door",
      "rt": ["oic.r.door"],
      "if": ["oic.if.a", "oic.if.baseline"]
    },

    {
      "href": "/door/lock.status",
      "rt": ["oic.r.lock"],
      "if": ["oic.if.a", "oic.if.baseline"]
    },

    {
      "href": "/light",
      "rt": ["oic.r.light"],
      "if": ["oic.if.s", "oic.if.baseline"]
    },

    {
      "href": "/binarySwitch",
      "rt": ["oic.r.switch.binary"],
      "if": ["oic.if.a", "oic.if.baseline"]
    }

  ]
}
```

A Collection may be:

– A pre-defined Collection where the Collection has been defined a priori and the Collection is static over its lifetime. Such Collections may be used to model, for example, an appliance that is composed of other Devices or fixed set of Resources representing fixed functions.

– A Device local Collection where the Collection is used only on the Device that hosts the Collection. Such Collections may be used as a short-hand on a Client for referring to many Servers as one.

– A centralized Collection where the Collection is hosted on a Device but other Devices may access or update the Collection.

– A hosted Collection where the Collection is centralized but is managed by an authorized agent or party.

7.8.3.2 Collection Properties

A Collection shall define a Property that is an array of Links (the Property Name "links" is recommended). In addition, other Properties may be defined for the Collection by the Resource Type. The mandatory and recommended Common Properties for a Collection are shown in Table 14. This list of Common Properties is in addition to those defined for Resources in 7.3.2.

Table 14 – Common Properties for Collections (in addition to Common Properties defined in 7.3.2)

Property	Description	Property Name	Value Type	Mandatory
Links	The array of Links in the Collection	Per Resource Type definition	json Array of Links	Yes
Resource Types	The list of allowed Resource Types for	As defined in Table 12	As defined in Table 12	No

	Links in the Collection. If this Property is not defined or is null string then any Resource Type is permitted			
Mandatory Resource Types	The list of Resource Types for Links that are mandatory in the Collection.	As defined in Table 13	As defined in Table 13	No

7.8.3.3 Default Resource Type

A default Resource Type, "oic.wk.col", is available for Collections. This Resource Type shall be used only when another type has not been defined on the Collection or when no Resource Type has been specified at the creation of the Collection.

The default Resource Type provides support for the Common Properties including an array of Links with the Property Name "links".

7.8.3.4 Default OCF Interface

All instances of a Collection shall support the links list ("oic.if.ll") OCF Interface in addition to the baseline ("oic.if.baseline") OCF Interface. An instance of a Collection may optionally support additional OCF Interfaces that are defined within this document. The Default OCF Interface for a Collection shall be links list ("oic.if.ll") unless otherwise specified by the Resource Type definition.

7.8.4 Atomic Measurement

7.8.4.1 Overview

Certain use cases require that the Properties of multiple Resources are only accessible as a group and individual access to those Properties of each Resource by a Client is prohibited. The Atomic Measurement Resource Type is defined to meet this requirement. This is accomplished through the use of the Batch OCF Interface.

7.8.4.2 Atomic Measurement Properties

An Atomic Measurement shall define a Property that is an array of Links (the Property Name "links" is recommended). In addition, other Properties may be defined for the Atomic Measurement by the Resource Type. The mandatory and recommended Common Properties for an Atomic Measurement are shown in Table 15. This list of Common Properties is in addition to those defined for Resources in 7.3.2.

Table 15 – Common Properties for Atomic Measurement (in addition to Common Properties defined in 7.3.2)

Property	Description	Property Name	Value Type	Mandatory
Links	The array of Links in the Atomic Measurement	Per Resource Type definition	json Array of Links	Yes
Resource Types	The list of allowed Resource Types for Links in the Atomic Measurement. If this Property is not defined or is null string then any Resource Type is permitted	As defined in Table 12	As defined in Table 12	No

Mandatory Resource Types	The list of Resource Types for Links that are mandatory in the Atomic Measurement.	As defined in Table 13	As defined in Table 13	No

7.8.4.3 Normative behaviour

The normative behaviour of an Atomic Measurement is as follows:

- The behaviour of the Batch OCF Interface ("oic.if.b") on the Atomic Measurement is defined as follows:
  - Only RETRIEVE and NOTIFY operations are supported, for Batch OCF Interface, on Atomic Measurement; the behaviour of the RETRIEVE and NOTIFY operations shall be the same as specified in 7.6.3.4, with exceptions as provided for in 7.8.4.3.
  - The UPDATE operation is not allowed, for Batch OCF Interface, on Atomic Measurement; if an UPDATE operation is received, it shall result in a method not allowed error code.
  - An error response shall not include any representation of a linked Resource (i.e. empty response for all linked Resources).
- Any linked Resource within an Atomic Measurement (i.e. the target Resource of a Link in an Atomic Measurement) is subject to the following conditions:
  - Linked Resources within an Atomic Measurement and the Atomic Measurement itself shall exist on a single Server.
  - CRUDN operations shall not be allowed on linked Resources and shall result in a forbidden error code.
  - Linked Resources shall not expose the "oic.if.ll" OCF Interface. Since CRUDN operations are not allowed on linked Resources, the "oic.if.ll" OCF Interface would never be accessible.
- Links to linked Resources in an Atomic Measurement shall only be accessible through the "oic.if.ll" or the "oic.if.baseline" OCF Interfaces of an Atomic Measurement.
  - The linked Resources shall not be listed in "/oic/res".
- A linked Resource in an Atomic Measurement shall have defined one of "oic.if.a", "oic.if.s", "oic.if.r", or "oic.if.rw" as its Default OCF Interface.
- Not all linked Resources in an Atomic Measurement are required to be Observable. If an Atomic Measurement is being Observed using the "oic.if.b" OCF Interface, notification responses shall not be generated when the linked Resources which are not marked Observable are updated or change state.
- All linked Resources in an Atomic Measurement shall be included in every RETRIEVE and Observe response when using the "oic.if.b" OCF Interface.
- An Atomic Measurement shall support the "oic.if.b" and the "oic.if.ll" OCF Interfaces.
- Filtering of linked Resources in an Atomic Measurement is not allowed. Query parameters that select one or more individual linked Resources in a request to an Atomic Measurement shall result in a "forbidden" error code.
- If the "rel" Link Parameter is included in a Link contained in an Atomic Measurement, it shall have either the "hosts" or the "item" value.
- The Default OCF Interface of an Atomic Measurement is "oic.if.b".

7.8.4.4 Security considerations

Access rights to an Atomic Measurement Resource Type is as specified in clause 12.2.7.2 (ACL considerations for batch request to the Atomic Measurement Resource Type) of ISO/IEC 30118-2).

7.8.4.5 Default Resource Type

The Resource Type is defined as "oic.wk.atomicmeasurement" as defined in Table 16.

Table 16 – Atomic Measurement Resource Type

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction	M/CR/O
none	Atomic Measurement	"oic.wk.atomicmeasurement"	"oic.if.ll" "oic.if.baseline" "oic.if.b"	A specialisation of the Collection pattern to ensure atomic RETRIEVAL of its referred Resources	RETRIEVE, NOTIFY	O

The Properties for Atomic Measurement are as defined in Table 17.

Table 17 – Properties for Atomic Measurement (in addition to Common Properties defined in 7.3.2)

Property	Description	Property name	Value Type	Mandatory
Links	The set of links that point to the linked Resources	Per Resource Type definition	json Array of Links	Yes

7.9 Query Parameters

7.9.1 Introduction

A query string is a fundamental part of the definition of a URI (see 6.2.2). The definition of a query may include Properties and Link Parameters by declaring the Property or Link Parameter (i.e. <Property name, Link Parameter name> = <desired Property value, Link Parameter value>) as one of the segments of the query. Only ASCII strings are permitted in queries, and NULL characters are disallowed in queries. This means that only Property and Link Parameter values with ASCII characters may be matched in a query.

When a query is defined as a selector, a Resource is selected when all the declared Properties or Link Parameters in the query match the corresponding Properties or Link Parameters in the target.

The processing of any query parameter by a Server is as specified in this document or other OCF specifications. For any query parameters that are not explicitly specified, the Server may ignore those query parameters and the request is processed as if the query parameter did not exist in the request.

7.9.2 Use of multiple parameters within a query

When a query contains multiple separate query parameters these are delimited by an "&" as described in 6.2.2. Multiple query parameters are only applicable to Collections or Resources with a multi-value "rt".

A Client may select a specific Resource type using separate query parameters, for example "?if=oic.if.b&rt=oic.r.switch.binary". If such queries are supported by the Server this shall be accomplished by matching "all of" the different query parameter types received (i.e. "rt", "if") against the target of the query. In the example, this resolves to a batch response that includes only instances of oic.r.switch.binary. There is no significance applied to the order of the query parameters.

A Client may select more than one Resource Type using repeated query parameters, for example "?rt=oic.r.switch.binary&rt=oic.r.ramptime". If such queries are supported by the Server, this shall be accomplished by matching "any of" the repeated query parameters against the target of the query. In the example, any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that may exist are selected.

A Client may select multiple Resource Types using multiple repeated "rt" parameters in addition to a separate "if" parameter in a single query, for example "?if=oic.if.b&rt=oic.r.switch.binary&rt=oic.r.ramptime". If such queries are supported by the Server, this shall be accomplished by matching "any of" the repeated query parameters and then matching "all of" the different query parameter types. In the example any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that may exist are selected in a batch response.

NOTE The parameters within a query string are represented within the actual messaging protocol as defined in clause 12.2.2.

7.9.3 Application to multi-value "rt" Resources

An "rt" query for a multi-value "rt" Resource with the Default OCF Interface of "oic.if.a", "oic.if.s", "oic.if.r", "oic.if.rw" or "oic.if.baseline" is an extension of a generic "rt" query.

When a Server receives a RETRIEVE request for a multi-value "rt" Resource with an "rt" query, (i.e. GET /ResExample?rt=oic.r.foo), the Server should respond only when the query value is an item of the "rt" Property Value of the target Resource and should send back only the Properties associated with the query value(s). For example, upon receiving GET /ResExample?rt=oic.r.switch.binary targeting a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"], the Server responds with only the Properties of oic.r.switch.binary.

When a Server receives an UPDATE request for a multi-value "rt" Resource with an "rt" query, (e.g.POST /ResExample?rt=oic.r.foo), the Server should only apply the payload received to the Properties that are part of the "oic.r.foo" Resource.

7.9.4 OCF Interface specific considerations for queries

7.9.4.1 OCF Interface selection

When an OCF Interface is to be selected for a request, it shall be specified as a query parameter in the URI of the Resource in the request message. If no query parameter is specified, then the Default OCF Interface shall be used. If the selected OCF Interface is not one of the permitted OCF Interfaces on the Resource, then selecting that OCF Interface is an error and the Server shall respond with an error response code. A Client shall not include more than one OCF Interface in a query parameter. If a Server receives a request that has more than one OCF Interface included in a query parameter (e.g. "?if=oic.if.ll&if=oic.if.rw") then the Server may either reject the request with an appropriate non-success path response, or the Server may attempt to process the request using the first "if" received

For example, the baseline OCF Interface may be selected by adding "if=oic.if.baseline" to the list of query parameters in the URI of the target Resource. For example: "GET /oic/res?if=oic.if.baseline".

7.9.4.2 Batch OCF Interface

See 7.6.3.4 for details on the batch OCF Interface itself. Query parameters may be used with the batch OCF Interface in order to select particular Resources in a Collection for retrieval or update; these parameters are used to select items in the Collection by matching Link Parameter Values.

When Link selection query parameters are used with RETRIEVE operations applied using the batch OCF Interface, only the Resources in the Collection with matching Link Parameters should be returned.

When Link selection query parameters are used with UPDATE operations applied using the batch OCF Interface, only the Resources having matching Link Parameters should be updated.

See 7.6.3.4.5 for examples of RETRIEVE and UPDATE operations that use Link selection query parameters.

7.9.5 The "action" Query Parameter

A fundamental tenet of this document is the use of REST architectural style and the use of RESTful operations (see clause 5.2). However, there are use cases where a RESTful interaction is either awkward or otherwise problematic to realize, for example, when functionality is needed that does not map to state changes to the properties in the RESTful paradigm. Thus an action pattern provides support for where a Client needs to invoke an action or operation on a Server that is stateless or has an unknown state. A typical example of an action is the operation of a "gang" switch, where the state of the switch (and thus the thing being controlled) is toggled to the inverse of the current value.

The "action" Query Parameter may thus be realized by a Resource definition as part of UPDATE operation handling, where that Resource represents some subsequent action that is then taken. The generalized format of this Query Parameter is "action=<Resource defined value>". A request that includes an "action" Query Parameter may or may not also include a payload; similarly, a response to a request that includes an "action" Query Parameter may or may not also include a payload. The set of possible values that may be present is up to the definition of the Resource. Further, the behaviour of a Server on receipt of a request that contains an "action" Query Parameter is up to the definition of the Resource.

If a Server receives an "action" Query Parameter in a request other than an UPDATE operation, the Server shall reject the request with an appropriate error response.

If a Server receives an "action" Query Parameter in a request to a Resource that does not define such usage, the Server shall reject the request with an appropriate error response.

If a Server receives an "action" Query Parameter in a request to a Resource that supports actions, but the action value is one that the Server does not recognize, the Server shall reject the request with an appropriate error response.

7.10 Error response payload

7.10.1 Overview

Clause 7.10 describes a mechanism and payload to signal additional error information that may be provided in addition to the response code when an error response is sent. The transport specific response for a transport binding (e.g., CoAP) returns a status code that does not always provide enough information on what has gone wrong.

7.10.2 Error response payload content

The error response payload shall be an ASCII string that contains a brief, human-readable diagnostic description as a string describing the details of the transport specific error response code. Standardized messages for the error response payload are defined in Table 27. Vendors may use these standardized messages or define their own messages. The messages contained within an error response payload may be included with any transport specific response code. English text is the only language supported for the message. If the error response payload is not present in the response, a Client deals with the error based on only the transport specific response code.

Table 18 – Standardized error message

Category	Message
Error due to Client	"Invalid parameter"

	"The mandatory parameter is missing"
	"The parameter is not allowed"
	"The token syntax is invalid"
	"The message id syntax is invalid"
	"Invalid permission"
	"The service key is invalid"
	"The token is not issued"
	"The token user is not issued"
	"Terms of service are not agreed"
	"The API is not permitted"
	"The API call count is exceeded"
	"The country is not supported"
	"The Device is inaccessible"
	"The token is invalid"
	"The count of subscription has exceeded the limit"
	"Invalid resource access"
	"The admin is not registered"
	"The user is not registered"
	"The service is not registered"
	"The event is not subscribed"
	"The Device is not registered"
	"The admin is already registered.
	"Internal Server operation error"
	"Device profile error"
	"The model is not supported"
	"Undefined enumeration"
	"The value is out of range"
	"Feature is not supported in the model"
	"Integration Server error"
	"The product is not supported for interworking with other companies"
	"The Device status is abnormal"
	"The Device is not connected (offline)"
	"The Device control failed"
	"The request is required to retry"
	"Time out occurred"
Error due to Server	"Internal Server operation error"
	"Device profile error"
	"The model is not supported"
	"Undefined enumeration"
	"The value is out of range"
	"Feature is not supported in the model"

	"Integration Server error"
	"The product is not supported for interworking with other companies"
	"The Device status is abnormal"
	"The Device is not connected (offline)"
	"The Device control failed"
	"The request is required to retry"
	"Time out occurred"

7.10.3 Example of use

The following example shows an example message exchange for a RETRIEVE operation sent from a proximal Device to an OCF Cloud, with a target URI of: "coaps+tcp://exampleCloudEndPoint//deviceId_001/somehref".

Client request:

```
Target URI: /deviceId_001/somehref
Operation: RETRIEVE
Host: coaps://exampleCloudEndPoint
Accept: application/vnd.ocf+cbor
```

Server response:

```
Status code: 4.04 (Not Found)
Response Body: {
  "The device is not registered"
}
```

With the error response payload, the Client can recognize that the Device it tried to discover is not registered on the OCF Cloud.

7.11 OCF MQTT Proxy

7.11.1 Introduction

An MQTT proxy is an OCF Device that acts as a proxy between MQTT and CoAP transports. An MQTT proxy enables the ability for OCF clients in the MQTT domain to talk to OCF Devices in the CoAP domain.

An MQTT proxy contains the following functionality:

– An OCF Client, talking to OCF Servers on the local network

– An MQTT Client that creates OCF Servers in the MQTT domain, e.g. it represents the proxied devices in MQTT domain.

– This is depicted in Figure 6.

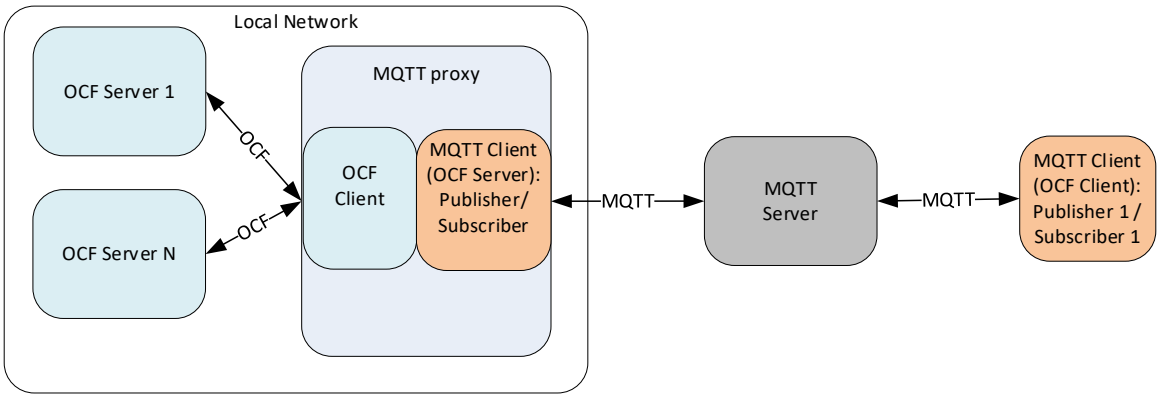


Figure 6 – CoAP domain and OCF MQTT domain interconnected by Proxy

The OCF Servers to be proxied will be provisioned by an OCF (mediator) Client on the MQTT Proxy

7.11.2 Resources for MQTT proxy

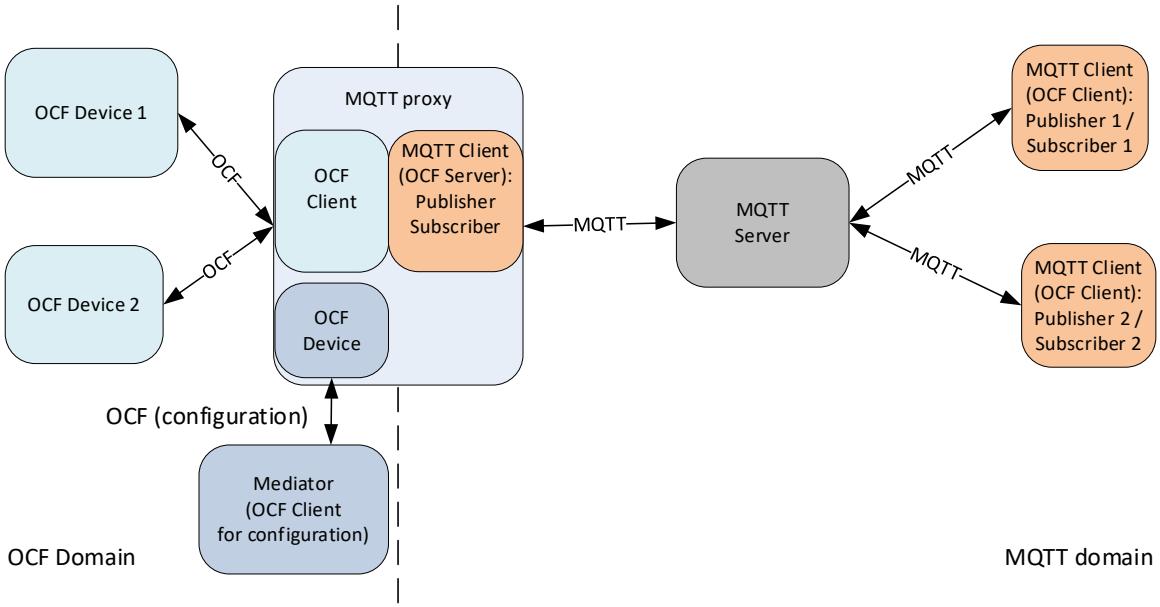


Figure 7 – OCF extended with OCF Clients in the MQTT domain

To represent the OCF CoAP Servers on the MQTT network, the following Resources are exposed by the OCF Device “oic.d.mqttproxy” on the CoAP side of the MQTT proxy:

– "oic.r.mqtt.conf" Resource Type

– "oic.r.d2dserverlist" Resource Type

The "oic.r.mqtt.conf" Resource Type is used to configure the MQTT client to connect to an MQTT server. The Resource Type is available on the CoAP side so that proximal interaction is possible to configure the MQTT proxy to connect to an MQTT server. The information supplied to the MQTT client also includes security information. The connection to the MQTT server should be secured by TLS. The information to be supplied to the Mediator to configure the "oic.r.mqtt.conf" Resource is provided out of band; this information determines how the MQTT proxy is being used in a larger setup. The "oic.r.d2dserverlist" Resource Type is used to list OCF Devices that will be proxied from the CoAP domain to the MQTT domain. This list is maintained from the CoAP OCF domain.

7.11.3 Connecting to an MQTT Server

The information of the MQTT client to connect to the MQTT server may be conveyed by an OCF Resource. This means that the MQTT proxy may be headless and may be configured with a Mediator (OCF client). The configuration information consists of data to contact the MQTT server and also of data to secure the connection.

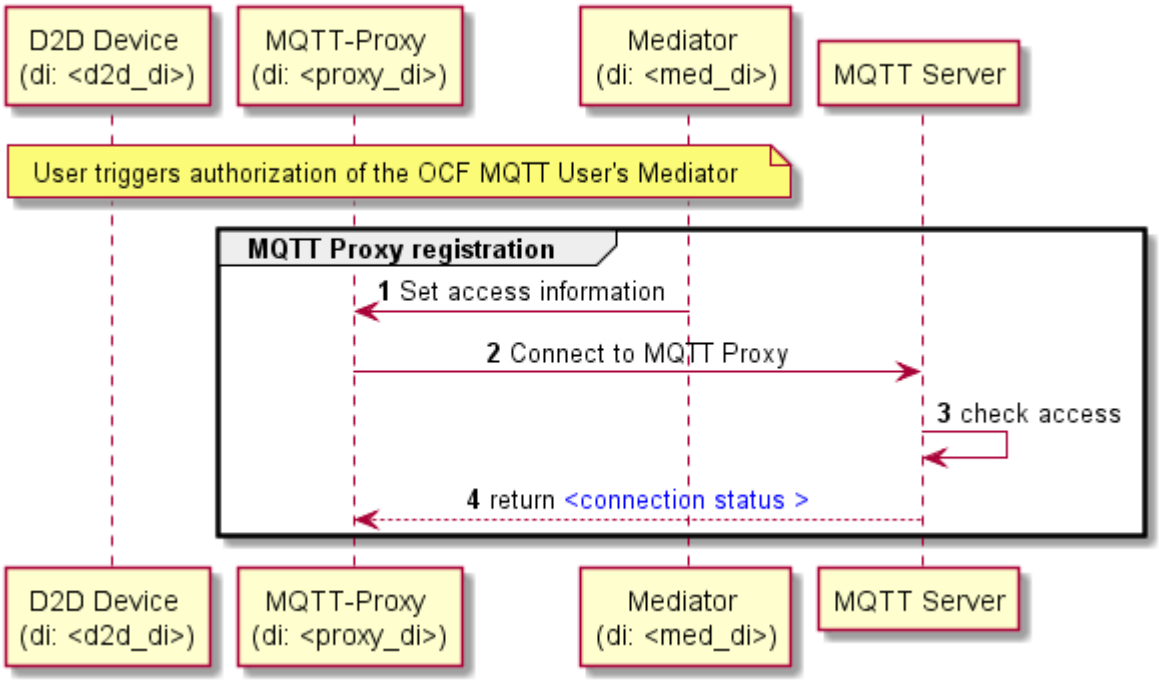


Figure 8 – Registration of the MQTT proxy (as MQTT client) with an MQTT Server

Table 19 – Properties of "oic.r.mqtt.conf" Resource

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory to implement	Description
Server address	"server"	string	N/A	N/A	RW	Yes	The connection information of the MQTT server may be an ip address or a URI e.g: "192.168.178.89" "test.mosquitto.org"
connection	"port"	integer	N/A	N/A	RW	Yes	The port number of the MQTT server e.g. 1883 for unsecured ports or 8883 for secured ports
Keep alive interval	"kai"	integer	N/A	seconds	RW	Yes	The keep alive interval for the MQTT client server connection.
User identifier	"uid"	string	N/A		RW	Yes	User ID, if supported by the MQTT server
password	"pwd"	string	N/A		RW	Yes	Password or token belonging to the user ID.

certificate authority file of the MQTT server	"cacert"	string	As byte array		R	Yes	The credential, if supported by the MQTT server
Client certificate to authenticate the connection	"clcert"	string	As byte array		R	Yes	The credential, if supported by the MQTT server
log	“log”	string	NA		R	Yes	Logging of the connection status
MQTT connection reason codes	“crcode”	integer	NA		R	Yes	See MQTT table 3-1 Note that before connecting, the value should be initialized on -1 indicating, “not yet connected”

All Properties listed in Table 19 are required to be implemented, e.g., listed in the IDD as optional. Not all Properties have to be on the wire though, the Property usage depends on the used MQTT server.

7.11.4 Proxying an OCF Device

The OCF Devices to be proxied are listed in the d2dserver list Resource. Which Vertical Resources are proxied per OCF Device is implementation dependent.

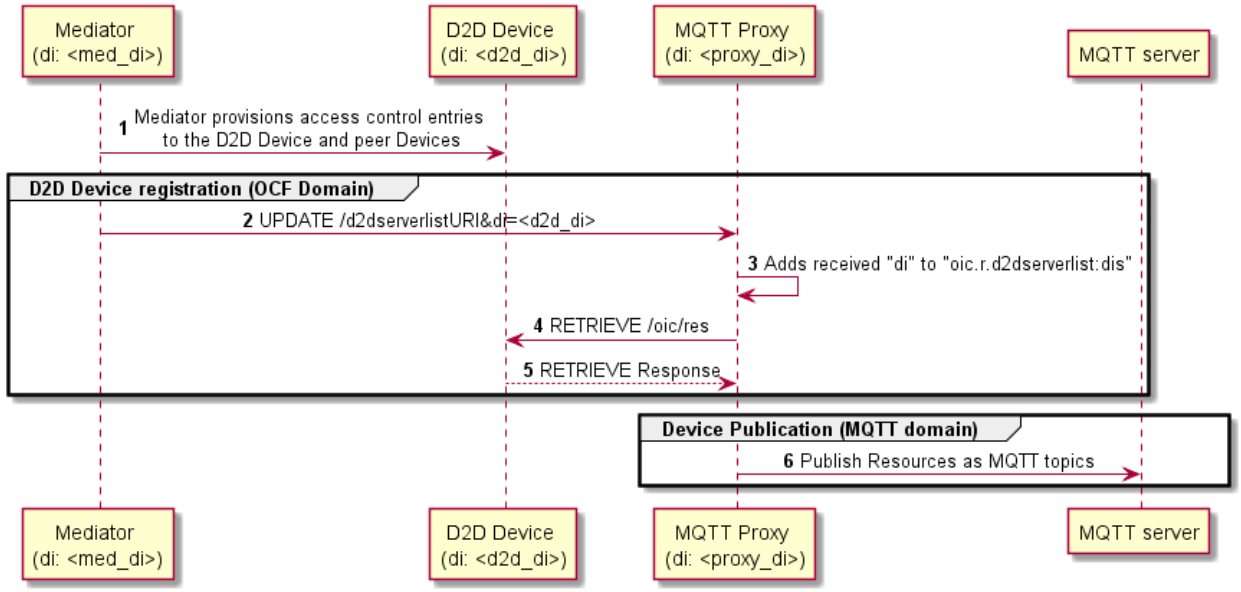


Figure 9 – Device publication to an MQTT server

7.11.5 Security considerations

The OCF Client in the MQTT proxy needs to be granted access to the D2D Device. Even if a privileged OCF Client adds a D2D Device to the "oic.r.d2dserverlist" Resource, the access may still be denied by the D2D Device. This is because the OCF Client in the MQTT proxy needs to be given access to any of the D2D Devices in its list and having the correct access levels set up for the OCF Client in the MQTT proxy.

The connection from a MQTT client to the MQTT server needs to be secure, e.g., using a TLS connection. However, MQTT specifies multiple mechanisms to create a secure connection from an MQTT client to an MQTT server. The used MQTT server should only connect to other MQTT clients via a secure connection.

8 CRUDN

8.1 Overview

CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for manipulating Resources. These operations are performed by a Client on the Resources contained in a Server. All required Properties shall be present in the payloads for which they are defined for the operations for which those payloads apply (see clause 7.1 regarding OpenAPI 2.0 definitions requirement).

On reception of a valid CRUDN operation a Server hosting the Resource that is the target of the request shall generate a response depending on the OCF Interface included in the request; or based on the Default OCF Interface for the Resource Type if no OCF Interface is included.

CRUDN operations utilize a set of parameters that are carried in the messages and are defined in Table 20. A Device shall use CBOR as the default payload (content) encoding scheme for Resource representations included in CRUDN operations and operation responses; a Device may negotiate a different payload encoding scheme (e.g, see in 12.2.4 for CoAP messaging). Clauses 8.2 through 8.6 respectively specify the CRUDN operations and use of the parameters. The type definitions for these terms will be mapped in the clause 12 for each protocol.

Table 20 – Parameters of CRUDN messages

Applicability	Name	Denotation	Definition
All messages	*fr*	From	The URI of the message originator.
	to	To	The URI of the recipient of the message.
	ri	Request Identifier	The identifier that uniquely identifies the message in the originator and the recipient.
	cn	Content	Information specific to the operation.
Requests	*op*	Operation	Specific operation requested to be performed by the Server.
	obs	Observe	Indicator for an Observe request.
Responses	*rs*	Response Code	Indicator of the result of the request; whether it was accepted and what the conclusion of the operation was. The values of the response code for CRUDN operations shall conform to those as defined in clause 5.9 and 12.1.2 in IETF RFC 7252.
	obs	Observe	Indicator for an Observe response.

8.2 CREATE

8.2.1 Overview

The CREATE operation is used to request the creation of new Resources on the Server. The CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 10.

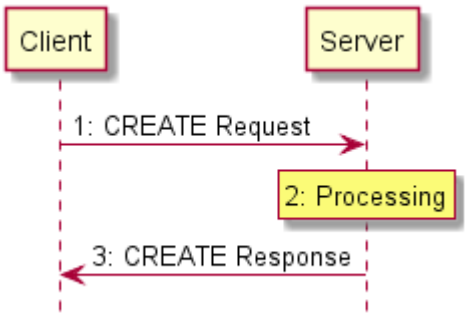


Figure 10 – CREATE operation

8.2.2 CREATE request

The CREATE request message is transmitted by the Client to the Server to create a new Resource by the Server. The CREATE request message will carry the following parameters:

- *fr*: Unique identifier of the Client
- *to*: URI of the target Resource responsible for creation of the new Resource.
- *ri*: Identifier of the CREATE request.
- *cn*: Information of the Resource to be created by the Server.
  - *cn* will include the URI and Resource Type Property of the Resource to be created.
  - *cn* may include additional Properties of the Resource to be created.
- *op*: CREATE

8.2.3 Processing by the Server

Following the receipt of a CREATE request, the Server may validate if the Client has the appropriate rights for creating the requested Resource. If the validation is successful, the Server creates the requested Resource. The Server caches the value of *ri* parameter in the CREATE request for inclusion in the CREATE response message.

8.2.4 CREATE response

The Server shall transmit a CREATE response message in response to a CREATE request message from a Client. The CREATE response message will include the following parameters:

- *fr*: Unique identifier of the Server
- *to*: Unique identifier of the Client
- *ri*: Identifier included in the CREATE request
- *cn*: Information of the Resource as created by the Server.
  - *cn* will include the URI of the created Resource.
  - *cn* will include the Resource representation of the created Resource.
- *rs*: The result of the CREATE operation.

8.3 RETRIEVE

8.3.1 Overview

The RETRIEVE operation is used to request the current state or representation of a Resource. The RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in Figure 11.

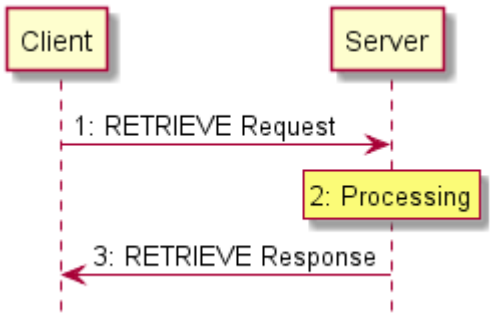


Figure 11 – RETRIEVE operation

8.3.2 RETRIEVE request

RETRIEVE request message is transmitted by the Client to the Server to request the representation of a Resource from a Server. The RETRIEVE request message will carry the following parameters:

– *fr*: Unique identifier of the Client.
– *to*: URI of the Resource the Client is targeting.
– *ri*: Identifier of the RETRIEVE request.
– *op*: RETRIEVE.

8.3.3 Processing by the Server

Following the receipt of a RETRIEVE request, the Server may validate if the Client has the appropriate rights for retrieving the requested data and the Properties are readable. The Server caches the value of *ri* parameter in the RETRIEVE request for use in the response

8.3.4 RETRIEVE response

The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request message from a Client. The RETRIEVE response message will include the following parameters:

– *fr*: Unique identifier of the Server.
– *to*: Unique identifier of the Client.
– *ri*: Identifier included in the RETRIEVE request.
– *cn*: Information of the Resource as requested by the Client.
  – *cn* should include the URI of the Resource targeted in the RETRIEVE request.
– *rs*: The result of the RETRIEVE operation.

8.4 UPDATE

8.4.1 Overview

The UPDATE operation is either a Partial UPDATE or a complete replacement of the information in a Resource in conjunction with the OCF Interface that is also applied to the operation. The UPDATE operation is initiated by the Client and consists of three steps, as depicted in Figure 12.

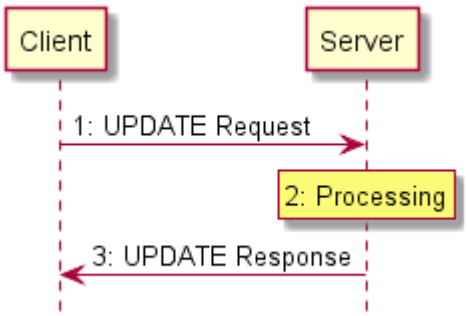


Figure 12 – UPDATE operation

8.4.2 UPDATE request

The UPDATE request message is transmitted by the Client to the Server to request the update of information of a Resource on the Server. The UPDATE request message, as indicated in 8.1, contains all required Properties whether changed or not. The UPDATE request message will carry the following parameters:

– *fr*: Unique identifier of the Client.

– *to*: URI of the Resource targeted for the information update.

– *ri*: Identifier of the UPDATE request.

– *op*: UPDATE.

– *cn*: Information, including Properties, of the Resource to be updated at the target Resource.

8.4.3 Processing by the Server

8.4.3.1 Overview

Following the receipt of an UPDATE request, the Server may validate if the Client has the appropriate rights for updating the requested data. If the validation is successful, the Server updates the target Resource information according to the information carried in *cn* parameter of the UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE request for use in the response.

An UPDATE request that includes Properties that are read-only shall be rejected by the Server with an *rs* indicating a bad request.

An UPDATE request shall be applied only to the Properties in the target Resource visible via the applied OCF Interface that support the operation. An UPDATE of non-existent Properties is ignored.

An UPDATE request shall be applied to the Properties in the target Resource even if those Property Values are the same as the values currently exposed by the target Resource.

8.4.3.2 Resource monitoring by the Server

The Server shall monitor the state the Resource identified in the Observe request from the Client. Anytime there is a change in the state of the Observed Resource or an UPDATE operation applied to the Resource, the Server sends another RETRIEVE response with the Observe indication. The mechanism does not allow the Client to specify any bounds or limits which trigger a notification, the decision is left entirely to the Server.

8.4.3.3 Additional RETRIEVE responses with Observe indication

The Server shall transmit updated RETRIEVE response messages following Observed changes in the state of the Resources requested by the Client. The RETRIEVE response message shall include the parameters listed in 11.3.2.4.

8.4.4 UPDATE response

The UPDATE response message will include the following parameters:

– *fr*: Unique identifier of the Server.

– *to*: Unique identifier of the Client.

– *ri*: Identifier included in the UPDATE request.

– *rs*: The result of the UPDATE request.

The UPDATE response message may also include the following parameters:

– cn: The Resource representation following processing of the UPDATE request.

8.5 DELETE

8.5.1 Overview

The DELETE operation is used to request the removal of a Resource. The DELETE operation is initiated by the Client and consists of three steps, as depicted in Figure 13.

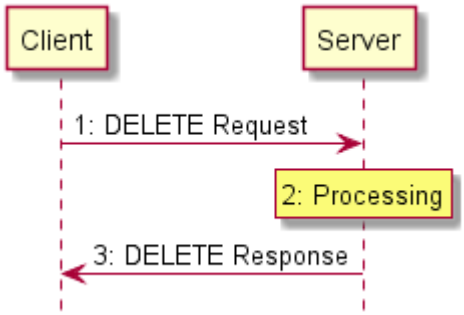


Figure 13 – DELETE operation

8.5.2 DELETE request

DELETE request message is transmitted by the Client to the Server to delete a Resource on the Server. The DELETE request message will carry the following parameters:

– *fr*: Unique identifier of the Client.

– *to*: URI of the target Resource which is the target of deletion.

– *ri*: Identifier of the DELETE request.

– *op*: DELETE.

8.5.3 Processing by the Server

Following the receipt of a DELETE request, the Server may validate if the Client has the appropriate rights for deleting the identified Resource, and whether the identified Resource exists. If the validation is successful, the Server removes the requested Resource and deletes all the associated information. The Server caches the value of *ri* parameter in the DELETE request for use in the response.

8.5.4 DELETE response

The Server shall transmit a DELETE response message in response to a DELETE request message from a Client. The DELETE response message will include the following parameters:

– *fr*: Unique identifier of the Server.

– *to*: Unique identifier of the Client.

– *ri*: Identifier included in the DELETE request.

– *rs*: The result of the DELETE operation.

8.6 NOTIFY

8.6.1 Overview

The NOTIFY operation is used to request asynchronous notification of state changes. Complete description of the NOTIFY operation is provided in 11.2.6. The NOTIFY operation uses the NOTIFICATION response message which is defined here.

8.6.2 NOTIFICATION response

The NOTIFICATION response message is sent by a Server to notify the URLs identified by the Client of a state change. The NOTIFICATION response message carries the following parameters:

– *fr*: Unique identifier of the Server.

– *to*: URI of the Resource target of the NOTIFICATION message.

– *ri*: Identifier included in the CREATE request.

– *op*: NOTIFY.

– *cn*: The updated state of the Resource.

9 Network and connectivity

9.1 Introduction

The Internet of Things is comprised of a wide range of applications which sense and actuate the physical world with a broad spectrum of device and network capabilities: from battery powered nodes transmitting 100 bytes per day and able to last 10 years on a coin cell battery, to mains powered nodes able to maintain Megabit video streams. It is estimated that many 10s of billions of IoT devices will be deployed over the coming years.

It is desirable that the connectivity options be adapted to the IP layer. To that end, IETF has completed considerable work to adapt Bluetooth®, Wi-Fi, 802.15.4, LPWAN, etc. to IPv6. These adaptations, plus the larger address space and improved address management capabilities, make IPv6 the clear choice for the OCF network layer technology.

9.2 Architecture

While the aging IPv4 centric network has evolved to support complex topologies, its deployment was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More complex network topologies, often seen in residential home, are mostly introduced through the acquisition of additional home network devices, which rely on technologies like private Network Address Translation (NAT). These technologies require expert assistance to set up correctly and should be avoided in a home network as they most often result in breakage of constructs like routing, naming and discovery services.

The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices and associated routers, but also new services introducing additional edge routers. All these new requirements require advance architectural constructs to address complex network topologies like the one shown in Figure 14.

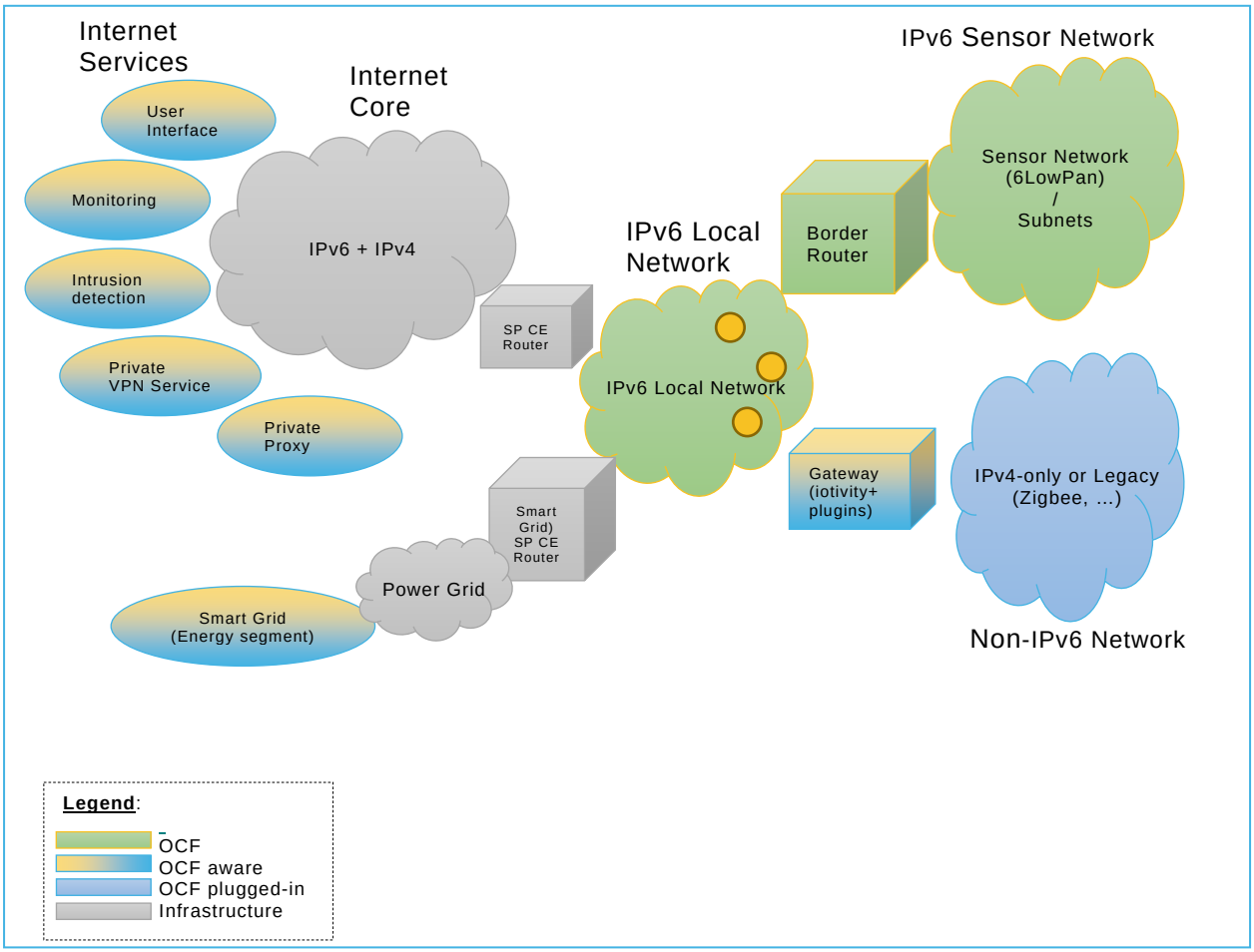


Figure 14 – High level network and connectivity architecture

In terms of IETF RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further implement various specializations of those roles:

– A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.

– Nodes limited in processing power, memory, non-volatile storage or transmission capacity requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL). Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU G9959, Bluetooth Low Energy, DECT Ultra Low Energy, and Near Field Communication (NFC).

– A node may translate and route messaging between IPv6 and non-IPv6 networks.

9.3 IPv6 network layer requirements

9.3.1 Introduction

Projections indicate that many 10s of billions of new IoT endpoints and related services will be brought online in the next few years. These endpoint's capabilities will span from battery powered nodes with limited compute, storage, and bandwidth to more richly resourced devices operating over Ethernet and Wi-Fi links.

Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide variety of applications such as Web browsing, email, voice, video, and critical system monitoring and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which is that available address space has been consumed.

The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

– Larger address space. Side-effect: greatly reduce the need for NATs.
– More flexible addressing architecture. Multiple addresses and types per interface: Link-local, ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed networks, better re-numbering capability, etc.
– More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
– Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
– All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
– Major Service Providers around the globe are deploying IPv6.

9.3.2 IPv6 node requirements

9.3.2.1 Introduction

In order to ensure network layer services interoperability from node to node, mandating a common network layer across all nodes is vital. The protocol should enable the network to be: secure, manageable, and scalable and to include constrained and self-organizing meshed nodes. OCF mandates IPv6 as the common network layer protocol to ensure interoperability across all Devices. More capable Devices may also include additional protocols creating multiple-stack Devices. The remainder of this clause will focus on interoperability requirements for IPv6 hosts, IPv6 constrained hosts and IPv6 routers. The various protocol translation permutations included in multi-stack gateway devices may be addresses in subsequent addendums of this document.

9.3.2.2 IP Layer

An IPv6 node shall support IPv6 and it shall conform to the requirements as specified in IETF RFC 6434.

10 OCF Endpoint

10.1 OCF Endpoint definition

The specific definition of an OCF Endpoint depends on the Transport Protocol Suite being used. For the example of CoAP over UDP over IPv6, the OCF Endpoint is identified by an IPv6 address and UDP port number.

Each Device shall associate with at least one OCF Endpoint with which it can exchange request and response messages. When a message is sent to an OCF Endpoint, it shall be delivered to the Device which is associated with the OCF Endpoint. When a request message is delivered to an OCF Endpoint, path component is enough to locate the target Resource.

A Device can be associated with multiple OCF Endpoints. For example, a Device can have several IP addresses or port numbers or support both CoAP and HTTP transfer protocols. Different Resources in a Device may be accessed with the same OCF Endpoint or need different ones. Some Resources may use one OCF Endpoint and others a different one. It depends on the implementation.

On the other hand, an OCF Endpoint can be shared among multiple Devices, only when there is a way to clearly designate the target Resource with a request URI. For example, when multiple CoAP servers use uniquely different URI paths for all their hosted Resources, and the CoAP implementation demultiplexes by path, they can share the same CoAP OCF Endpoint. However, this is not possible in this version of the document, because a pre-determined URI (e.g. "/oic/d") is mandatory for some mandatory Resources (e.g. "oic.wk.d").

10.2 OCF Endpoint information

10.2.1 Introduction

An OCF Endpoint is represented by OCF Endpoint information, which consists of the following key-value pairs, "ep", "pri", and "lat".

10.2.2 "ep"

"ep" represents Transport Protocol Suite and OCF Endpoint Locator specified as follows:

– *Transport Protocol Suite* - a combination of protocols (e.g. CoAP + UDP + IPv6) with which request and response messages can be exchanged for RESTful transaction (i.e. CRUDN). A Transport Protocol Suite shall be indicated by a URI scheme name. All scheme names supported by this document are IANA registered, these are listed in Table 21. A vendor may also make use of a non-IANA registered scheme name for their own use (e.g. "com.example.foo"), this shall follow the syntax for such scheme names defined by IETF RFC 7595. The behaviour of a vendor-defined scheme name is undefined by this document. All OCF defined Resource Types when exposing OCF Endpoint Information in an "eps" (see 10.2.4) shall include at least one "ep" with a Transport Protocol Suite as defined in Table 21.

– *OCF Endpoint Locator* – an address (e.g. IPv6 address + Port number) or an indirect identifier (e.g., DNS name) resolvable to an IP address, through which a message can be sent to the OCF Endpoint and in turn associated Device. The OCF Endpoint Locator for "coap" and "coaps" shall be specified as "IP address: port number". The OCF Endpoint Locator for "coap+tcp" or "coaps+tcp" shall be specified as "IP address: port number" or "DNS name: port number" or "DNS name" such that the DNS name shall be resolved to a valid IP address for the target Resource with a name resolution service (i.e., DNS). For the 3rd case, when the port number is omitted, the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and for "coaps+tcp") scheme respectively as defined in IETF RFC 8323.Temporary addresses should not be used because OCF Endpoint Locators are for the purpose of accepting incoming sessions, whereas temporary addresses are for initiating outgoing sessions (IETF RFC 4941). Moreover, its inclusion in "/oic/res" can cause a privacy concern (IETF RFC 7721).

– *OCF Latency* – the maximum latency in seconds [sec] that the Server may take to respond to a request.

"ep" shall have as its value a URI (as specified in IETF RFC 3986) with the scheme component indicating Transport Protocol Suite and the authority component indicating the OCF Endpoint Locator.

An "ep" example for "coap" and "coaps" is as illustrated:

```
"ep": "coap://[fe80::b1d6]:1111"
```

An "ep" example for "coap+tcp" and "coaps+tcp" is as illustrated:

```
"ep": "coap+tcp://[2001:db8:a::123]:2222"
"ep": "coap+tcp://foo.bar.com:2222"
"ep": "coap+tcp://foo.bar.com"
```

The current list of "ep" with corresponding Transport Protocol Suite is shown in Table 21:

Table 21 – "ep" value for Transport Protocol Suite

Transport Protocol Suite	scheme	OCF Endpoint Locator	"ep" Value example
coap+udp+ip	"coap"	IP address + port number	"coap://[fe80::b1d6]:1111"
coaps + udp + ip	"coaps"	IP address + port number	"coaps://[fe80::b1d6]:1122"
coap + tcp + ip	"coap+tcp"	IP address + port number DNS name: port number DNS name	"coap+tcp://[2001:db8:a::123]:2222" "coap+tcp://foo.bar.com:2222" "coap+tcp://foo.bar.com"
coaps + tcp + ip	"coaps+tcp"	IP address + port number DNS name: port number DNS name	"coaps+tcp://[2001:db8:a::123]:2233" "coaps+tcp://[2001:db8:a::123]:2233" "coaps+tcp://foo.bar.com:2233"

10.2.3 "pri"

When there are multiple OCF Endpoints, "pri" indicates the priority among them.

"pri" shall be represented as a positive integer (e.g. "pri": 1) and the lower the value, the higher the priority.

The default "pri" value is 1, i.e. when "pri" is not present, it shall be equivalent to "pri": 1.

10.2.4 "lat"

"lat" indicates the expected delay of the response. For example, when a Server implements a mode to improve battery performance; the Server can expose this value, thereby providing a Client with the ability to use this for the timeout on the connection. For example, the Thread "rx-off-when-idle" link mode is an implementation of a battery performance improvement mechanism.

"lat" shall be represented as a positive integer (e.g. "lat": 240), and the value is specified in seconds.

10.2.5 OCF Endpoint information in "eps" Parameter

To carry OCF Endpoint information, a new Link Parameter "eps" is defined in 7.8.2.5.6. "eps" has an array of items as its value and each item represents OCF Endpoint information with key-value pairs, "ep", "pri", and "lat", of which "ep" is mandatory and "pri" and "lat" are optional.

OCF Endpoint Information in an "eps" Parameter is valid for the target Resource of the Link, i.e., the Resource referred by "href" Parameter. OCF Endpoint information in an "eps" Parameter may be used to access other Resources on the Device, but such access is not guaranteed.

A Client may resolve the "ep" value to an IP address for the target Resource, i.e., the address to access the Device which hosts the target Resource. A valid (transfer protocol) URI for the target Resource can be constructed with the scheme, host and port components from the "ep" value and the "path" component from the "href" value.

Links with an "eps":

```
{
  "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9 ",
  "href": "/myLightSwitch",
  "rt": ["oic.r.switch.binary"],
```



```
  "if": ["oic.if.a", "oic.if.baseline"],
  "p": {"bm": 3},
  "eps": [
    {"ep": "coap://[fe80::b1d6]:1111", "pri": 2, "lat": 240},
    {"ep": "coaps://[fe80::b1d6]:1122"}
  ]
}

{
  "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
  "href": "/myTemperature",
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "p": {"bm": 3},
  "eps": [
    {"ep": "coap+tcp://foo.bar.com", "pri": 2, "lat": 240},
    {"ep": "coaps+tcp://foo.bar.com:1122"}
  ]
}
```

In the previous example, "anchor" represents the hosting Device, "href", target Resource and "eps" the two OCF Endpoints for the target Resource. The (fully-qualified) URIs for the target Resource are as illustrated:

```
coap://[fe80::b1d6]:1111/myLightSwitch
coaps://[fe80::b1d6]:1122/myLightSwitch
coap+tcp://foo.bar.com:5683/myTemperature
```

coaps+tcp://foo.bar.com:1122/myTemperatureIf the target Resource of a Link requires a secure connection (e.g. CoAPS), "eps" Parameter shall be used to indicate the necessary information (e.g. port number) in OCF 1.0 payload. For optional backward compatibility with OIC 1.1, the "sec" and "port" shall only be used in OIC 1.1 payload.

10.3 OCF Endpoint discovery

10.3.1 Introduction

OCF Endpoint discovery is defined as the process for a Client to acquire the OCF Endpoint information for Device or Resource.

10.3.2 Implicit discovery

If a Device is the source of a CoAP message (e.g. "/oic/res" response), the source IP address and port number may be combined to form the OCF Endpoint Locator for the Device. Along with a "coap" scheme and default "pri" value, OCF Endpoint information for the Device may be constructed.

In other words, a "/oic/res" response message with CoAP may implicitly carry the OCF Endpoint information of the responding Device and in turn all the hosted Resources, which may be accessed with the same transfer protocol of CoAP. In the absence of an "eps" Parameter, a Client shall be able to utilize implicit discovery to access the target Resource.

10.3.3 Explicit discovery with "/oic/res" response

OCF Endpoint information may be explicitly indicated with the "eps" Parameter of the Links in "/oic/res".

As in 10.3.2, an "/oic/res" response may implicitly indicate the OCF Endpoint information for some Resources hosted by the responding Device. However implicit discovery, i.e., inference of OCF Endpoint information from CoAP response message, may not work for some Resources on the same Device. For example, some Resources may allow only secure access via CoAPS which requires the "eps" Parameter to indicate the port number. Moreover "/oic/res" may expose a target Resource which belongs to another Device.

When the OCF Endpoint for a target Resource of a Link cannot be implicitly inferred, the "eps" Parameter shall be included to provide explicit OCF Endpoint information with which a Client can access the target Resource. In the presence of the "eps" Parameter, a Client shall be able to utilize it to access the target Resource. For "coap" and "coaps", a Client may use the IP address in the "ep" value in the "eps" Parameter to access the target Resource. For "coap+tcp" and "coaps+tcp", a Client may use the IP address in the "eps" Parameter or resolve the DNS name in the "eps" Parameter to acquire a valid IP address for the target Resource. If "eps" Parameter omits the port number, then the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and "coaps+tcp") scheme as defined in IETF RFC 8323.To access the target Resource of a Link, a Client may use the "eps" Parameter in the Link, if it is present and fall back on implicit discovery if not.

This is an example of an "/oic/res" response from a Device having the "eps" Parameter in Links.

```
[
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/res",
    "rel": "self",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/d",
    "rt": ["oic.wk.d"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/sec/doxm",
    "rt": ["oic.r.doxm"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
```



```
    "href": "/oic/sec/pstat",
    "rt": ["oic.r.pstat"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/sec/cred",
    "rt": ["oic.r.cred"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/sec/acl2",
    "rt": ["oic.r.acl2"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/myIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/myLight",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coaps://[2001:db8:a::b1d4]:22222"}
    ]
  }
]
```


The exact format of the "/oic/res" response and a way for a Client to acquire a "/oic/res" response message is specified in Annex A and 11.2.4 respectively.

11 Functional interactions

11.1 Introduction

The functional interactions between a Client and a Server are described in 11.1 through 0 respectively. The functional interactions use CRUDN messages (clause 7.11) and include Discovery, Notification, and Device management. These functions require support of core defined Resources as defined in Table 22.

Table 22 – List of Core Resources

Pre-defined URI	Resource Name	Resource Type	Related Functional Interaction	Mandatory
"/oic/res"	Default	"oic.wk.res"	Discovery	Yes
"/oic/p"	Platform	"oic.wk.p"	Discovery	Yes
"/oic/d"	Device	"oic.wk.d"	Discovery	Yes
Implementation defined	Introspection	"oic.wk.introspection"	Introspection	Yes

11.2 Resource discovery

11.2.1 Introduction

Discovery is a function which enables OCF Endpoint discovery as well as Resource based discovery. OCF Endpoint discovery is described in detail in clause 10. This clause mainly describes the Resource based discovery.

11.2.2 Resource based discovery: mechanisms

11.2.2.1 Overview

As part of discovery, a Client may find appropriate information about other OCF peers. This information could be instances of Resources, Resource Types or any other information represented in the Resource model that an OCF peer would want another OCF peer to discover.

At the minimum, Resource based discovery uses the following:

– A Resource to enable discovery shall be defined. The representation of that Resource shall contain the information that can be discovered.

– The Resource to enable discovery shall be specified and commonly known a-priori. A Device for hosting the Resource to enable discovery shall be identified.

– A mechanism and process to publish the information that needs to be discovered with the Resource to enable discovery.

– A mechanism and process to access and obtain the information from the Resource to enable discovery. A query may be used in the request to limit the returned information.

– A scope for the publication.

– A scope for the access.

– A policy for visibility of the information.

Depending on the choice of the base aspects, the Framework defines three Resource based discovery mechanisms:

– Direct discovery, where the Resources are published locally at the Device hosting the Resources and are discovered through peer inquiry.

– Indirect discovery, where Resources are published at a third party assisting with the discovery and peers publish and perform discovery against the Resource to enable discovery on the assisting 3rd party.

– Advertisement discovery, where the Resource to enable discovery is hosted local to the initiator of the discovery inquiry but remote to the Devices that are publishing discovery information.

A Device shall support direct discovery.

11.2.2.2 Direct discovery

In direct discovery,

– The Device that is providing the information shall host the Resource to enable discovery.
– The Device publishes the information available for discovery with the local Resource to enable discovery (i.e. local scope).
– Clients interested in discovering information about this Device shall issue RETRIEVE requests directly to the Resource. The request may be made as a unicast or multicast. The request may be generic or may be qualified or limited by using appropriate queries in the request.
– The Server Device that receives the request shall send a response with the discovered information directly back to the requesting Client Device.
– The information that is included in the request is determined by the policies set for the Resource to be discovered locally on the responding Device.

11.2.3 Resource based discovery: Finding information

The discovery process (Figure 15) is initiated as a RETRIEVE request to the Resource to enable discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the support in the data connectivity layer. The response to the request has the information to be discovered based on the policies for that information. The policies can determine which information is shared, when and to which requesting agent. The information that can be discovered can be Resources, types, configuration and many other standards or custom aspects depending on the request to appropriate Resource and the form of request. Optionally the requester may narrow the information to be returned in the request using query parameters in the URI query.

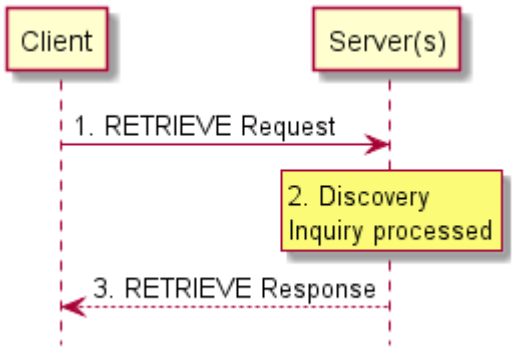


Figure 15 – Resource based discovery: Finding information

Discovery Resources

The following Core Resources shall be implemented on all Devices to support discovery:

– "/oic/res" for discovery of Resources.
– "/oic/p" for discovery of Platform.
– "/oic/d" for discovery of Device information.

Devices shall expose each of "/oic/res", "/oic/d", and "/oic/p" via an unsecured OCF Endpoint. Further details for these mandatory Core Resources are described in Table 23.

Platform Resource

The OCF recognizes that more than one instance of Device may be hosted on a single Platform. Clients need a way to discover and access the information on the Platform. The Core Resource, "/oic/p" exposes Platform specific Properties. All instances of Device on the same Platform shall have the same values of any Properties exposed (i.e. a Device may choose to expose optional Properties within "/oic/p" but when exposed the value of that Property should be the same as the value of that Property on all other Devices on that Platform).

Device Resource

The Device Resource shall have the pre-defined URI "/oic/d", the Device Resource shall expose the Properties pertaining to a Device as defined in Table 26. The Device Resource shall have a default Resource Type that helps in bootstrapping the interactions with the Device (the default type is described in Table 23).The Device Resource may have one or more Resource Type(s) that are specific to the Device in addition to the default Resource Type or if present overriding the default Resource Type. The base Resource Type "oic.wk.d" defines the Properties that shall be exposed by all Devices. The Device specific Resource Type(s) exposed are dependent on the class of Device (e.g. air conditioner, smoke alarm, etc. Since all the Resource Types of "/oic/d" are not known a priori, the Resource Type(s) of "/oic/d" are determined by discovery through the Core Resource "/oic/res".

Table 23 – Mandatory discovery Core Resources

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
"/oic/res"	Default	"oic.wk.res"	"oic.if.ll", "oic.if.b", "oic.if.baseline"	The Resource through which the corresponding Server is discovered and introspected for available Resources. "/oic/res" shall expose the Resources that are discoverable on a Device. When a Server receives a RETRIEVE request targeting "/oic/res" (e.g., "GET /oic/res"), it shall respond with the links list of all the Discoverable Resources of itself. The "/oic/d" and "/oic/p" are Discoverable Resources, hence their links are included in "/oic/res" response. The Properties exposed by "/oic/res" are listed in Table 24.	Discovery
"/oic/p"	Platform	"oic.wk.p"	"oic.if.r"	The Discoverable Resource through which Platform specific information is discovered. The Properties exposed by "/oic/p" are listed in Table 27	Discovery
"/oic/d"	Device	"oic.wk.d" and/or one or more Device Specific Resource Type ID(s)	"oic.if.r"	The discoverable via "/oic/res" Resource which exposes Properties specific to the Device instance. The Properties exposed by "/oic/d" are listed in Table 26.	Discovery

Table 24 defines "oic.wk.res" Resource Type.

Table 24 – "oic.wk.res" Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	"n"	string	N/A	N/A	R	No	Human-friendly name defined by the vendor
Links	"links"	array	See 7.8.2	N/A	R	Yes	The array of Links describes the URI, supported Resource Types and OCF Interfaces, and access policy.
Security Domain UUID	"sduuid"	string	uuid	N/A	R	No	Unique identifier for the Security Domain. This value shall be the same value (i.e. mirror) as the "sdi.uuid" Property as defined in ISO/IEC 30118-2. It shall be exposed if the "sdi.priv" Property is set to "false", and shall not be exposed if the "sdi.priv" Property is set to "true".
Security Domain Name	"sdname"	string	N/A	N/A	R	No	Human-friendly name for the Security Domain. This value shall be the same value (i.e. mirror) as the "sdi.name" Property as defined in ISO/IEC 30118-2. It shall be exposed if the "sdi.priv" Property is set to "false", and shall not be exposed if the "sdi.priv" Property is set to "true".

Note: The "n", "sduuid", and "sdname" Property values for the "oic.wk.res" Resource Type are only in the response payload when used with the "oic.if.baseline" OCF Interface (i.e., RETRIEVE /oic/res?if="oic.if.baseline").

A Device shall support CoAP based discovery as the baseline discovery mechanism (see 11.2.5).

The "/oic/res" shall list all Resources that are indicated as discoverable (see 11.2). Also the following architecture Resource Types shall be listed:

– Introspection Resource indicated with an "rt" value of "oic.wk.introspection".
– "/oic/p" indicated with an "rt" value of "oic.wk.p".
– "/oic/d" indicated with an "rt" value of "oic.wk.d"
– "/oic/sec/doxm" indicated with an "rt" value of "oic.r.doxm" as defined in ISO/IEC 30118-2.
– "/oic/sec/pstat" indicated with an "rt" value of "oic.r.pstat" as defined in ISO/IEC 30118-2.
– "/oic/sec/acl2" indicated with an "rt" value of "oic.r.acl2" as defined in ISO/IEC 30118-2.
– "/oic/sec/cred" indicated with an "rt" value of "oic.r.cred" as defined in ISO/IEC 30118-2.

Conditionally required:

– "/oic/res" with an "rt" value of "oic.wk.res" as self-reference, on the condition that "oic/res" has to signal that it is Observable by a Client.

– if the Device supports batch retrieval of "/oic/res" then "oic.if.b" shall be included in the "if" Property of "/oic/res".

– if the Device supports batch retrieval there shall be a self-reference that includes an "if" Link Parameter containing "oic.if.b"; the self-reference shall expose a secure OCF Endpoint.

The Introspection Resource is only applicable for Devices that host Vertical Resource Types (e.g. "oic.r.switch.binary") or vendor-defined Resource Types. Devices that only host Resources required to onboard the Device as a Client do not have to implement the Introspection Resource.

Table 25 provides an OCF registry for protocol schemes.

Table 25 – Protocol scheme registry

SI Number	Protocol
1	"coap"
2	"coaps"
3	"http"
4	"https"
5	"coap+tcp"
6	"coaps+tcp"

NOTE The discovery of an OCF Endpoint used by a specific protocol is out of scope. The mechanism used by a Client to form requests in a different messaging protocol other than discovery is out of scope.

The following applies to the use of "/oic/d":

– A vertical may choose to extend the list of Properties defined by the Resource Type "oic.wk.d". In that case, the vertical shall assign a new Device Type specific Resource Type ID. The mandatory Properties defined in Table 26 shall always be present.

– A Device may choose to expose a separate, Discoverable Resource with its Resource Type ID set to a Device Type. In this case the Resource is equivalent to an instance of "oic.wk.d" and adheres to the definition thereof. As such the Resource shall at a minimum expose the mandatory Properties of "oic.wk.d". In the case where the Resource tagged in this manner is defined to be an instance of a Collection in accordance with 7.8.3 then the Resources that are part of that Collection shall at a minimum include the Resource Types mandated for the Device Type.

Table 26 "oic.wk.d" Resource Type definition defines the base Resource Type for the "/oic/d" Resource.

Table 26 – "oic.wk.d" Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	"n"	"string:	N/A	N/A	R	Yes	Human friendly name defined by the vendor. In the presence of "n" Property of "/oic/con", both have the same Property Value. When "n" Property Value of "/oic/con" is modified, it shall be reflected to "n" Property Value of "/oic/d".
Spec Version	"icv"	"string "	N/A	N/A	R	Yes	The specification version of this document that a Device is implemented to. The syntax shall be "ocf.<major>.<minor>.<sub-

							version>" where <major>, <minor, and <sub-version> are the major, minor and sub-version numbers of this document respectively. The specification version number (i.e., <major>.<minor>.<sub-version>) shall be obtained from the title page of this document (e.g. "2.0.5"). An example of the string value for this Property is "ocf.2.0.5".
Device UUID	"di"	"uuid"	N/A	N/A	R	Yes	Unique identifier for Device. This value shall be the same value (i.e. mirror) as the "doxm.deviceuuid" Property as defined in ISO/IEC 30118-2. Handling privacy-sensitivity for the "di" Property, refer to clause 13.16 in ISO/IEC 30118-2.
Data Model Version	"dmv"	"csv"	N/A	N/A	R	Yes	Spec version of the Resource specification to which this Device data model is implemented; if implemented against a Vertical specific Device specification(s), then the Spec version of the vertical specification this Device model is implemented to. The syntax is a comma separated list of <res>.<major>.<minor>.<sub-version> or <vertical>.<major>.<minor>.<sub-version>. <res> is the string "ocf.res" and <vertical> is the name of the vertical defined in the Vertical specific Resource specification. The <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. One entry in the csv string shall be the applicable version of the Resource Type Specification for the Device (e.g. "ocf.res.1.0.0"). If applicable, additional entry(-ies) in the csv shall be the vertical(s) being realized (e.g. "ocf.sh.1.0.0"). This value may be extended by the vendor. The syntax for extending this value, as a comma separated entry, by the vendor shall be by adding x.<Domain_Name>.<vendor_string> . For example, "ocf.res.1.0.0, ocf.sh.1.0.0, x.com.example.string", The order of the values in the comma separated string can be in any order (i.e. no prescribed order). This Property shall not exceed 256 octets.
Permanent Immutable ID	"piid"	"uuid"	N/A	N/A	R	Yes	A unique and immutable Device identifier. A Client can detect that a single Device supports multiple communication protocols if it discovers that the Device uses a single Permanent Immutable ID value for all the protocols it supports. Handling privacy-sensitivity for the "piid" Property,

							refer to clause 13.16 in ISO/IEC 30118-2.
Localized Descriptions	"ld"	"array"	N/A	N/A	R	No	Detailed description of the Device, in one or more languages. This Property is an array of objects where each object has a "language" field (containing an IETF RFC 5646 language tag) and a "value" field containing the Device description in the indicated language.
Software Version	"sv"	"string"	N/A	N/A	R	No	Version of the Device software.
Manufacturer Name	"dmn"	"array"	N/A	N/A	R	No	Name of manufacturer of the Device, in one or more languages. This Property is an array of objects where each object has a "language" field (containing an IETF RFC 5646 language tag) and a "value" field containing the manufacturer name in the indicated language.
Model Number	"dmno"	"string"	N/A	N/A	R	No	Model number as designated by manufacturer.
Ecosystem Name	"econame"	“string”	enum	N/A	R	No	This is the name of ecosystem that a Bridged Device belongs to. If a Device has "oic.d.virtual" as one of Resource Type values ("rt") the Device shall contain this Property, otherwise this Property shall not be included. This Property has enumeration values: ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"].
Version of Ecosystem	"ecoversion"	“string”	N/A	N/A	R	No	This is the version of ecosystem that a Bridged Device belongs to. If a Device has "oic.d.virtual" as one of its Resource Type values ("rt") the Device should contain this Property, otherwise this Property shall not be included.

Table 27 defines "oic.wk.p" Resource Type.

Table 27 – "oic.wk.p" Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform ID	"pi"	"uuid"	N/A	N/A	R	Yes	Unique identifier for the physical Platform (UUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC. Handling privacy-sensitivity for the "pi" Property, refer to clause 13.16 in ISO/IEC 30118-2.
Manufacturer Name	"mnmn"	"string"	N/A	N/A	R	Yes	Name of manufacturer.

Manufacturer Details Link	"mnml"	"uri"	N/A	N/A	R	No	Reference to manufacturer, represented as a URI.
Model Number	"mnmo"	"string"	N/A	N/A	R	No	Model number as designated by manufacturer.
Date of Manufacture	"mndt"	"date"	N/A	Time	R	No	Manufacturing date of Platform.
Serial number	"mnsel	"string"	N/A	s	R	No	Serial number of the Platform, may be unique for each Platform of the same model number.
Platform Version	"mnpv"	"string"	N/A	N/A	R	No	Version of Platform – string (defined by manufacturer).
OS Version	"mnos"	"string"	N/A	N/A	R	No	Version of Platform resident OS – string (defined by manufacturer).
Hardware Version	"mnhw"	"string"	N/A	N/A	R	No	Version of Platform hardware.
Firmware version	"mnfv"	"string"	N/A	N/A	R	No	Version of Platform firmware.
Support link	"mnsl"	"uri"	N/A	N/A	R	No	URI that points to support information from manufacturer.
SystemTime	"st"	"date-time"	N/A	N/A	R	No	Reference time for the Platform.
Vendor ID	"vid"	"string"	N/A	N/A	R	No	Vendor defined string for the Platform. The string is freeform and up to the vendor on what text to populate it.
Network Connectivity Type	"mnnct"	"array"	array of integer		R	No	An array of integer where each integer indicates the network connectivity type based on IANAIfType value as defined by IANA ifType-MIB Definitions, e.g., [71, 259] which represents Wi-Fi and Zigbee.

11.2.4 Resource discovery using "/oic/res"

11.2.4.1 General Requirements

Discovery using "/oic/res" is the default discovery mechanism that shall be supported by all Devices. General requirements for use of this mechanism are as follows:

– Every Device updates its local "/oic/res" with the Resources that are discoverable (see 7.3.2.2). Every time a new Resource is instantiated on the Device and if that Resource is discoverable by a remote Device then that Resource is published with the "/oic/res" Resource that is local to the Device (as the instantiated Resource).

After performing discovery using "/oic/res", Clients may discover additional details about the Device by performing discovery using "/oic/p", "/oic/d", etc. If a Client already knows about the Device, it may discover using other Resources without going through the discovery of "/oic/res"

11.2.4.2 Discovery using "oic.if.ll" (Default OCF Interface for "/oic/res")

If a Client does not explicitly include an OCF Interface as a query parameter in the request to "/oic/res" then the OCF Interface is taken to be "oic.if.ll" as that is the Default OCF Interface for "/oic/res". The requirements in this clause are thus applied. The requirements in this clause also apply if an OCF Interface of "oic.if.ll" is explicitly requested by inclusion as a query parameter in the RETRIEVE operation.

– A Device wanting to discover Resources or Resource Types on one or more remote Devices makes a RETRIEVE request to "/oic/res" on the remote Devices. This request may be sent multicast or unicast if a specific Device is to be probed. The RETRIEVE request may optionally be restricted using appropriate clauses in the query portion of the request. Queries may select based on requirements captured in this document (e.g. Resource Types).

– The Device receiving the RETRIEVE request responds with a list of Resources, the Resource Type of each of the Resources and the OCF Interfaces that each Resource supports. Additionally, information on the policies active on the Resource can also be sent (e.g. if the Resource can be Observed, or if the Resource can be discovered).

– The receiving Device may invoke additional operations based on the Resources returned in the request to "/oic/res".

The information that is returned on discovery against "/oic/res" is at minimum:

– The URI (relative or fully qualified URL) of the Resource.

– The Resource Type(s) of each Resource. More than one Resource Type may be returned if the Resource enables more than one type. To access Resources of multiple types, the specific Resource Type that is targeted shall be specified in the request.

– The OCF Interfaces supported by that Resource. Multiple OCF Interfaces may be returned. To access a specific OCF Interface that OCF Interface shall be specified in the request. If the OCF Interface is not specified, then the Default OCF Interface is assumed.

For Clients that do include the OCF-Accept-Content-Format-Version option, an "/oic/res" response includes an array of Links to conform to IETF RFC 6690. Each Link shall use an "eps" Parameter to provide the information for an encrypted connection and carry "anchor" containing an OCF URI where the authority component of is the Device UUID of the Device hosting the target Resource.

The OpenAPI 2.0 file for discovery using "/oic/res" is described in Annex A. Also refer to clause 10 (OCF Endpoint discovery) for details of Multicast discovery using "/oic/res" on a CoAP transport.

An example response from a Device is shown below:

```
[
  {
    "href": "/oic/res",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "rel": "self",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coap://[fe80::b1d6]:44444"}]
  },
  {
    "href": "/oic/p",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
            {"ep": "coaps://[fe80::b1d6]:11111"}
           ]
  },
```



```
  {
    "href": "/oic/d",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "rt": ["oic.wk.d"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
            {"ep": "coaps://[fe80::b1d6]:11111"}
           ]
  },
  {
    "href": "/myLightSwitch",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
            {"ep": "coaps://[fe80::b1d6]:11111"}
           ]
  }
]
```

11.2.5 Multicast discovery using "/oic/res"

11.2.5.1 General requirements

Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices shall support use of CoAP multicast to allow retrieving the "/oic/res" Resource from an unsecured OCF Endpoint on the Device. Clients may support use of CoAP multicast to retrieve the "/oic/res" Resource from other Devices. The CoAP multicast retrieval of "/oic/res" supports filtering Links based on the "rt" Property in the Links:

– If the discovery request is intended for a specific Resource Type including as part of a multi-value Resource Type, the query parameter "rt" shall be included in the request (see 6.2.2) with its value set to the desired Resource Type. Only Devices hosting the Resource Type shall respond to the discovery request.

– When the "rt" query parameter is omitted, all Devices shall respond to the discovery request.

11.2.5.2 Discovery using OCF Security Domain identifier

If OCF Security Domain Properties are exposed by the "/oic/res" Resource (see 11.2.3), the CoAP multicast retrieval of "/oic/res" supports filtering based on the OCF Security Domain Identifier in the "/oic/res" Resource:

– If the discovery request is intended for a specific OCF Security Domain, the query parameter "sduuid" shall be included in the request (see 6.2.2) with its value set to the desired OCF Security Domain UUID. Only Devices exposing the queried "sduuid" Property (i.e., the value matches the OCF Security Domain UUID in the query parameter) in the "/oic/res" Resource shall respond to the discovery request

11.2.6 Multicast discovery using "/.well-known/core"

Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices that join the All CoAP Nodes multicast group as optionally defined in clause 12.2.9 may also support multicast retrieval from "/.well-known/core" (see IETF RFC 7252). A Server node shall join at least both the link-local scoped address FF02::FD and the site-local scoped address FF05::FD. IPv6 addresses of other scopes may also be enabled. A Device responding to a request received on "/.well-known/core" shall encode the payload using the Core link format, which is a Content-Format of "40" (application/link-format) as defined in IETF RFC 6690. Core links in the response payload shall have a Content-Format code ("ct" attribute) of "10000" ("application/vnd.ocf+cbor"). This Content-Format code shall be used in subsequent requests and responses to obtain further Device Resource information.

A Client may send a multicast request to "/.well-known/core" to discover Devices that have joined the All CoAP Nodes multicast group. However, non-OCF Devices may also respond to this request. In order to filter out these non-OCF Devices, a Client may use "rt" query parameters so that only OCF Devices respond. A Server shall support querying for the "oic.wk.res" Resource Type as an "rt" query parameter value. A Client issuing such a request is equivalent to searching for all Devices. The Server shall also support querying for a Device Type as an "rt" query parameter value and respond when the Device Type matches the "rt" query parameter value.

Devices that support this optional discovery mechanism shall return as a minimum the Core link to the "/oic/res" Resource so that discovery of further Resources may be performed with a RETRIEVE operation to the URL of the discovered "/oic/res" Resource. The returned URL shall be fully qualified.

The "rt" and "if" attribute shall also be included in the response. The "rt" attribute shall include "oic.wk.res" and the "rt" value of the Device Type. The "if" attribute shall include the OCF Interfaces exposed by "/oic/res".

Example of a query for all Devices:

```
Req: GET coap://[FF02::FD]:5683/.well-known/core?rt=oic.wk.res
Res: 2.05 Content, Content-Format: 40
<coap://[fe80::b1d6]:1111/oic/res>;ct=10000;rt="oic.wk.res oic.d.sensor";if="oic.if.11
oic.if.baseline"
```

Example of a query for a specific Device Type:

```
Req: GET coap://[FF02::FD]:5683/.well-known/core?rt=oic.d.sensor
Res: 2.05 Content, Content-Format: 40
<coap://[fe80::b1d6]:1111/oic/res>;ct=10000;rt="oic.wk.res oic.d.sensor"; if="oic.if.ll
oic.if.baseline"
```

11.3 Notification

11.3.1 Overview

A Server shall support NOTIFY operation to enable a Client to request and be notified of desired states of one or more Resources in an asynchronous manner. 11.3.2 specifies the Observe mechanism in which updates are delivered to the requester.

11.3.2 Observe

11.3.2.1 Overview

In the Observe mechanism the Client utilizes the RETRIEVE operation to require the Server for updates in case of Resource state changes. The Observe mechanism consists of five steps which are depicted in Figure 16.

NOTE the Observe mechanism can only be used for a resource with a Property of Observable (see 7.3.2.2).

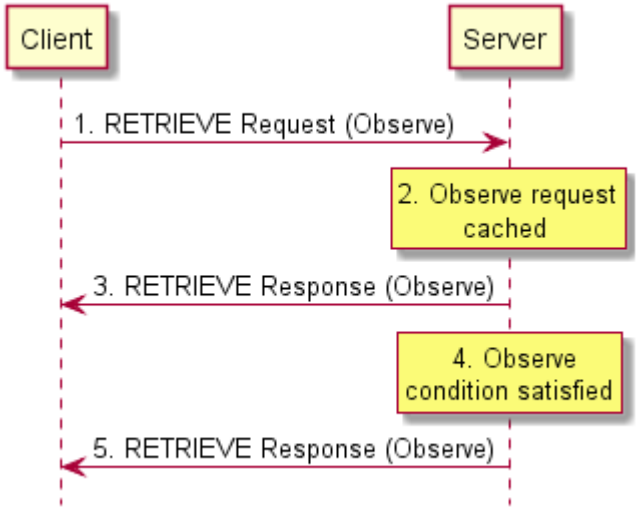


Figure 16 – Observe mechanism

11.3.2.2 RETRIEVE request with Observe indication

The Client transmits a RETRIEVE request message to the Server to request updates for the Resource on the Server if there is a state change. The RETRIEVE request message carries the following parameters:

– *fr*: Unique identifier of the Client.

– *to*: Resource that the Client is requesting to Observe.

– *ri*: Identifier of the RETRIEVE operation.

– *op*: RETRIEVE.

– *obs*: Indication for Observe operation.

11.3.2.3 Processing by the Server

Following the receipt of the RETRIEVE request, the Server may validate if the Client has the appropriate rights for the requested operation and the Properties are readable and Observable. If the validation is successful, the Server caches the information related to the Observe request. The Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial response and future responses in case of a change of state.

11.3.2.4 RETRIEVE response with Observe indication

The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request message from a Client. If validation succeeded, the response includes an Observe indication. If not, the Observe indication is omitted from the response which signals to the requesting Client that registration for notification was not allowed.

The RETRIEVE response message shall include the following parameters:

– *fr*: Unique identifier of the Server.

– *to*: Unique identifier of the Client.

– *ri*: Identifier included in the RETRIEVE operation.

– *cn*: Information Resource representation as requested by the Client.

– *rs*: The result of the RETRIEVE operation.

– *obs*: Indication that the response is made to an Observe operation.

11.3.2.5 Resource monitoring by the Server

The Server shall monitor the state the Resource identified in the Observe request from the Client. Anytime there is a change in the state of the Observed Resource, the Server sends another RETRIEVE response with the Observe indication. The mechanism does not allow the client to specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

11.3.2.6 Additional RETRIEVE responses with Observe indication

The Server shall transmit updated RETRIEVE response messages following Observed changes in the state of the Resources indicated by the Client. The RETRIEVE response message shall include the parameters listed in 11.3.2.4.

11.3.2.7 Cancelling Observe

The Client can explicitly cancel Observe by sending a RETRIEVE request without the Observe indication field to the same Resource on the Server which it was Observing. For certain protocol mappings, the Client may also be able to cancel an Observe by ceasing to respond to the RETRIEVE responses.

11.3.3 Push Notification

11.3.3.1 Overview

A Server may be configured to provide a NOTIFICATION via a push mechanism rather than a pull mechanism (i.e. Observe, see Section 11.3.2). That is rather than a Client establishing one or more discrete Observe transactions with a Server, a Client may configure a Server such that an embedded Client within the Server pushes observable events to a defined destination.

The general principle is that a Server provides the representation of any Resource that is the subject of an UPDATE operation or an internal (to the Server) change in the represented Properties via the use of an embedded Client in an UPDATE operation to a pre-configured target (destination). The set of Resources for which information is pushed in this manner is defined by configuration that exists on the Server.

11.3.3.2 Architectural Model

There are four logical elements that make up the architecture for push notifications. These being: the origin Server that hosts the Resources, an embedded Client within the origin Server that originates the push notifications, a target Server that is the recipient of the push notifications, and a Client that configures both the origin Server and (optionally) the target Server.

The Client providing configuration of the origin and target Servers may be a discrete Client or embedded in the target Server.

This architecture is illustrated in Figure 17.

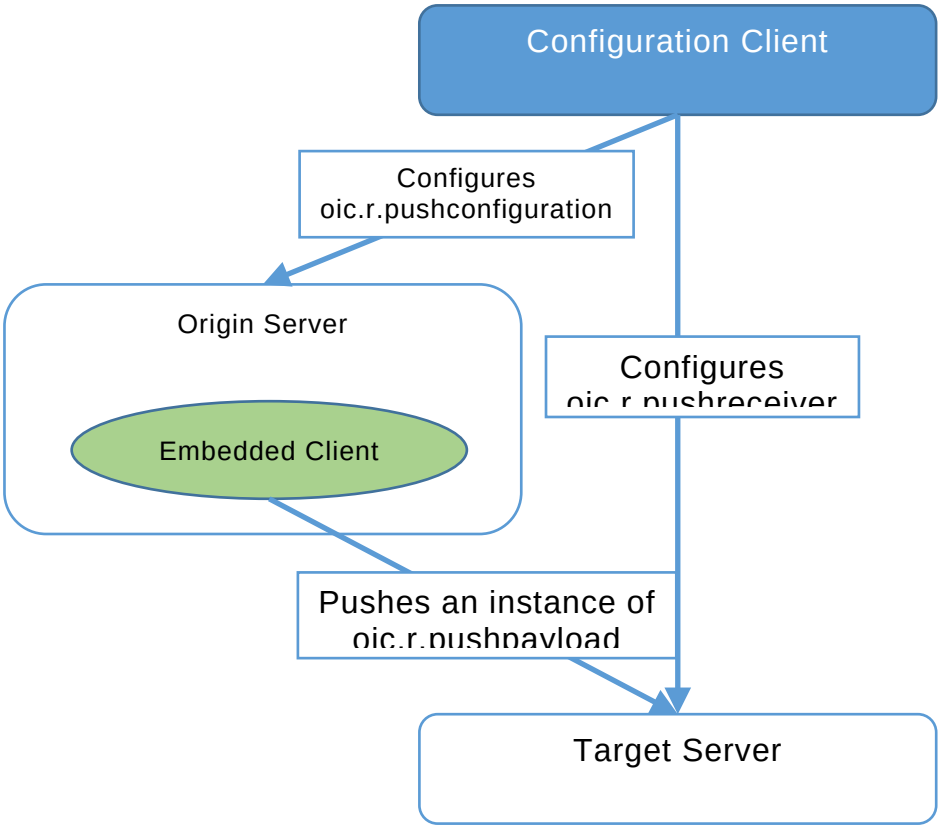


Figure 17 – Push Architecture

As illustrated in Figure 17, a push relationship can be established by a 3rd party, that is by a Client that is not part of the originator of the notifications nor part of the recipient of the notifications. This further distinguishes the push model from an Observe model, for the latter the establishment of the Observe transaction is done by the Client that is the intended recipient of the notifications.

Figure 18 is an example sequence diagram showing how the elements of the architecture may interact in realising push notifications.

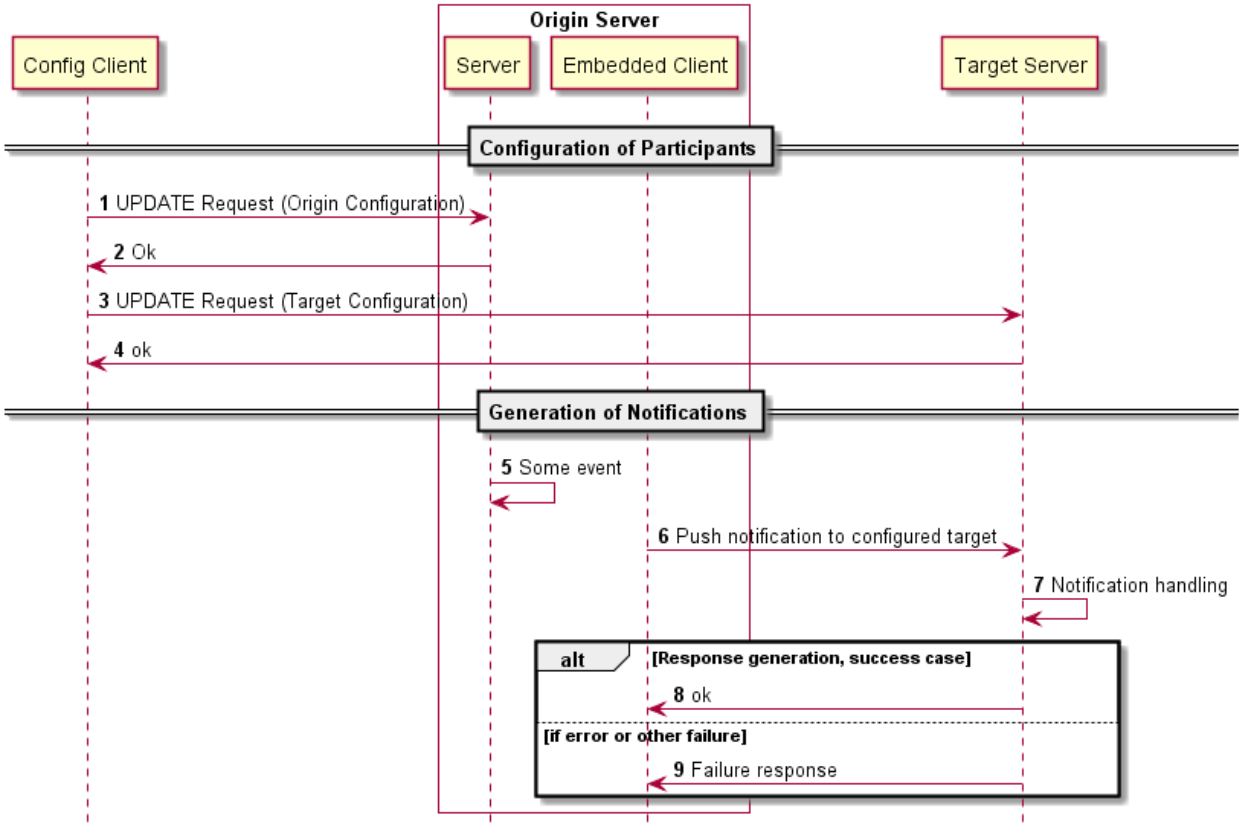


Figure 18 – Example Push Sequence

Table 28 provides additional details for the steps captured in Figure 18.

Table 28 – Example Push Sequence Details

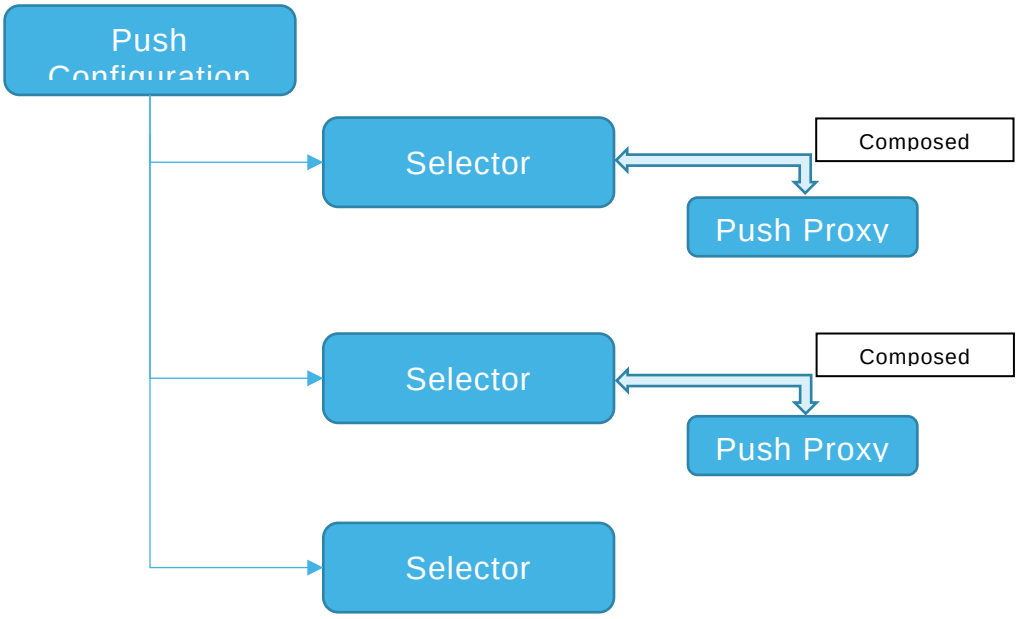
Step	Description
1	The Client configuring the Server that will originate the push notifications via an embedded Client provisions the instance of "oic.r.pushconfiguration" with notification selectors(s); each notification selector identifies a set of candidate Resources from which information may be pushed. The destination for the pushed information is established by composing a Push Proxy with a specific notification selector(s).
2	The Server provides a success path response
3	The Client configuring the Server that will be the recipient of the push notifications provisions the instance of "oic.r.pushreceiver" with the URLs at which notifications may be received and if there are limitations, the Resource Types that may be supported.
4	The Server provides a success path response
5	Some event occurs on the origin server that results in a state change in a Resource identified by one or more notification selectors
6	The embedded Client within the origin Server generates an UPDATE request to destination identified by the composed Push Proxy for the notification selector. The payload for the UPDATE request is an instance of "oic.r.pushpayload"
7	The target Server handles the received push notification in accordance with its own internal application logic
8	If no errors, the target Server provides a success path response
9	If errors occur, the target Server provides a non-success path response

11.3.3.3 Origin Server Configuration

11.3.3.3.1 Overview

A Server that allows for the configuration of one or more target Servers for push notifications shall expose an instance of "oic.r.pushconfiguration", this is a Collection containing instances of a notification selector ("oic.r.notificationselector"). Composed with the instance of a notification selector is a Push Proxy ("oic.r.pushproxy"); the Push Proxy provides the target for the information that is pushed as a result of the application of the notification selector.

Please see Figure 19 – Example Pictorial Push Configuration Collection for a pictorial



representation of "oic.r.pushconfiguration":

Figure 19 – Example Pictorial Push Configuration Collection

The Push Proxy (see Section 11.3.3.3.2) defines the target of the pushed information, the instance of "oic.r.notificationselector" provides Properties that allow for selection of specific Resources to be pushed. Within an instance of "oic.r.notificationselector" are optional Properties "phref", "prt", and "pif" which are applied in a similar manner to a query parameter in a URI (see clause 7.9), and when applied, only Resources that are identified as a result of application of the Properties may be pushed. When multiple Properties are present a logical "and" operation shall be applied between them.

The set of Resources that may be pushed is established when the notification selector is configured. The exception to this are any Resources that have been dynamically created on the Server via use of the "oic.if.create" OCF Interface; these shall not be pushed even though there may be notification selectors that match them.

NOTE: In all cases, only those Resources that are marked as "pushable" in the Policy Link Parameter in “/oic/res” may be pushed.

11.3.3.3.2 Push Proxy Resource Type

The Push Proxy Resource Type is defined in Table 29.

Table 29 – Resource Types for Push Proxy

Example URI	Resource Type Title	Resource Type ID (“rt” value)	OCF Interfaces	Description	Related Functional Interaction
/exampleResourceURI	Push Proxy	"oic.r.pushproxy"	"oic.if.rw", "oic.if.baseline"	Adds Push Proxy functionality to a Source Resource	Notifications

The Push Proxy Resource Type is composed with a Notification Selector Resource Type, this enables notifications to be pushed to a defined target Resource when the Resources identified by applying the Notification Selector are updated, or when their state changes. Therefore an instance of "oic.r.pushproxy" shall always be composed with an instance of "oic.r.notificationselector".

The Properties of the Push Proxy Resource Type are defined in Table 30.

Table 30 – Push Proxy Resource Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Push Target URI	pushtarget	string	URI		RW	yes	Points to the target of the UPDATE operation sent as a notifcation
Source Resource Type	sourcert	array of string			RW	yes	Always set to "oic.r.pushpayload"
State	state	string enumeration	One of ["waitingforprovisioning","waitingforupdate","waitingforresponse", " waitingforupdatemitigation", " waitingforresponsemitigation", "error","timeout"]		RW	yes (RETRIEVE), no (UPDATE)	Current state of the Push Proxy

The "Push Target URI" Property, "pushtarget", contains a pointer to the target Resource, that is the Resource to which the notification shall be sent. The URI reference may be to either a local resource name, for example "output1", or a URI on the local device, for example "/scenes/athome", or a full URI including scheme and authority components, for example "ocf://850faa5d-ccaf-4293-9452-f4fcab2e2c39/scenes/atwork". Note that when the URI uses the "ocf" scheme then the URI shall be resolved to a transport protocol URI as defined in the OCF Core Specification before use in an UPDATE operation. The "pushtarget" may also be an empty string (""); in which case there is no associated target Resource and so no notifications may be sent; this occurs when a Push Proxy is pre-configured with other Resources on a Device.

The "Source Resource Type" Property, "sourcert", shall be set to "oic.r.pushpayload". Future use cases may be defined that add to the set of possible Resource Types, for the purposes of this document, the Property shall only contain "oic.r.pushpayload".

The Property "state" contains the current or desired state of the Push Proxy. When provided in a RETRIEVE response the value shall be one of defined enumeration values in Table 31. When used in an UPDATE operation the only value that shall be accepted by a Device receiving the UPDATE is "waitingforupdate", other values shall result in the UPDATE operation being rejected. Also, an UPDATE that sets "state" is only valid when received by a Push Proxy in a state of "timeout" or "error"; reception in any other state shall result in the operation being rejected.

11.3.3.3.3 Push proxy state machine

The Device that exposes a Push Proxy Resource Type realizes an individual instance of a state machine per Push Proxy.

Table 31 describes the states of which the state machine is composed.

Table 31 – Push Proxy States

State Friendly Name	State enumeration	State Acronym	Description
Waiting for Provisioning	waitingforprovisioning	WFP	Proxy exists but the "pushtarget" Property is an empty string (i.e. not populated).
Waiting for Update	waitingforupdate	WFU	Proxy has been configured and pending Resource state change.
Waiting for Response	waitingforresponse	WFR	Resource state has changed, UPDATE sent to the push target.
Waiting for Update Mitigation	waitingforupdatemitigation	WFUM	UPDATE has been sent to the push target and error situation occurs (transport layer timer expiration or error response) while Proxy has an error mitigation algorithm to handle this situation.
Waiting for Response Mitigation	waitingforresponsemitigation	WFRM	Resource state has changed. UPDATE sent to the push target by the error mitigation algorithm.
Error Response Received	error	ERR	Non-success path response received for the UPDATE operation.
Timeout Condition Detected	timeout	TOUT	Transport layer timeout detected waiting for a response to the UPDATE operation.

The following provides more detail with respect to the behaviour of the push proxy in each defined state:

– **"waiting for provisioning" (WFP)** state shall be entered when the "pushtarget" Property of a Resource that includes the Push Proxy Resource Type is an empty string (i.e. ""). This may be an initial state in the case where a Push Proxy is pre-configured on a Device, or a state that is entered following configuration of the Property by a Client. In the WFP state no action shall be taken on a state change. An existing Push Proxy not initially in this state may only enter this state from the "waiting for update" (WFU) state.

– **"waiting for update" (WFU)** state shall be entered after successful creation of a Resource that includes a Push Proxy Resource Type, and also after the successful completion of an UPDATE operation triggered by a state change of the source Resource. When the source Resource is successfully updated or a state change occurs, an UPDATE request shall be constructed and sent to the target Resource, and the "waiting for response" (WFR) state is entered. An UPDATE request to the source Resource may be acknowledged before the triggered UPDATE request to the target Resource is sent.

– **"waiting for response" (WFR)** state shall be entered when an UPDATE request is sent to a target Resource. If no response is received before any applicable transport layer timers expire, then the "timeout" (TOUT) state is entered. If an error response is received, then the "error" (ERR) state is entered. If a non-error response is received, then the "waiting for update" (WFU) state is entered.

– **"waiting for update mitigation" (WFUM)** state shall be entered from "waiting for response" (WFR) state only if there is an error mitigation algorithm to handle error situations (transport layer timer expiration or error response) when the error situation occurs. The algorithm starts when the Push Proxy enters this state. In this state the Push Proxy may try to send another UPDATE request and enter "waiting for response mitigation" (WFRM) state. The Push Proxy enters this state from "waiting for response mitigation" (WFRM) state when the error situation continues to occur.

– **"waiting for response mitigation" (WFRM)** state shall be entered from "waiting for update mitigation" (WFUM) state after an UPDATE request has been sent to a target Resource. If any error situation (transport layer timer expiration or error response) occurs again, then "waiting for update mitigation" (WFUM) state shall be entered to continue error mitigation algorithm. If a non-error response is received, then the error mitigation algorithm is stopped and the "waiting for update" (WFU) state shall be entered, or if the error mitigation algorithm fails in the end, the proper error state (TOUT or ERR) is entered.

– **"timeout condition detected" (TOUT)** state shall be entered when any applicable transport layer response timer expires. A Client may force a transition from the "timeout condition detected" (TOUT) state to the "waiting for update" (WFU) state by sending an UPDATE request to the Push Proxy Resource that sets the "state" Property to "waiting for update" (WFU). Otherwise, the Push Proxy remains in this state until the Push Proxy composition is deleted. .

– **"error response received" (ERR)** state shall be entered when an error response is received from the target Resource in response to an UPDATE request. A Client may force a transition from the "error response received" (ERR) state to the "waiting for update" (WFU) state by sending an UPDATE request to the Push Proxy Resource that sets the "state" Property to "waiting for update" (WFU). Otherwise, the Push Proxy remains in this state until the Push Proxy composition is deleted.

Figure 20 shows the Push Proxy operational state machine.

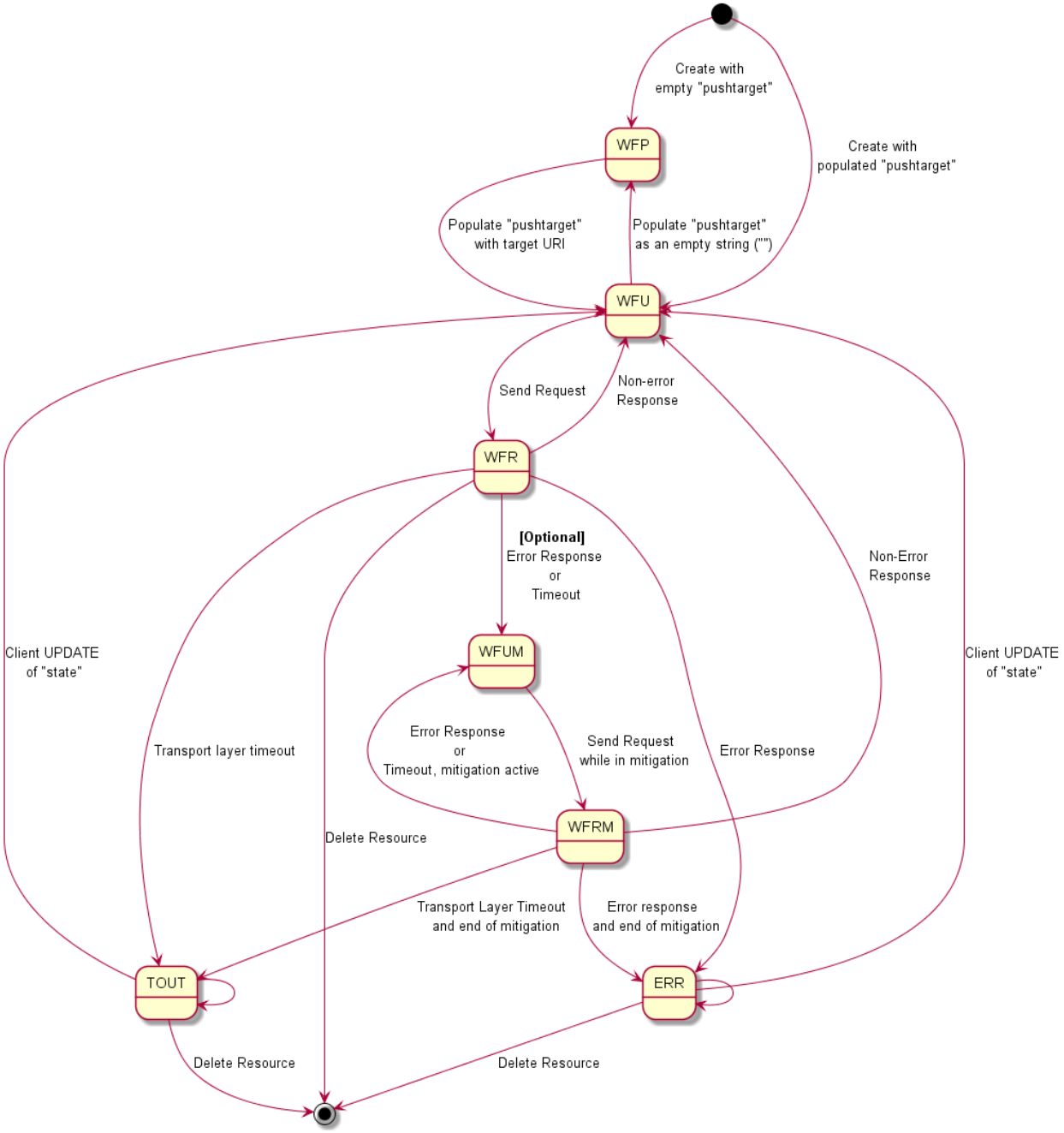


Figure 20 – Push Proxy Operational State Machine

The Push Proxy is initialized when the multi-value "rt" Resource that is composed with the Push Proxy Resource Type is created. This may be done by a Client using the "oic.if.create" OCF Interface or may be already composed with other exposed Resources as part of the Device implementation. With respect to the Client creation case, please see the example in Figure 21. In all cases, a configured Push Proxy is in the "wait for update" state.


```
Method = CREATE
URI = /dynamic/somecollection?if=oic.if.create&rt=oic.wk.col
Payload =

{
 "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
 "if": ["oic.if.baseline", "oic.if.a", "oic.if.rw"],
 "p": {"bm":3},
 "rep": {
  "value": false,
  "pushtarget": " ocf://850faa5d-ccaf-4293-9452-
f4fcab2e2c39/power/switch",
  "sourcert": ["oic.r.pushpayload"]
 }
}

Response = CREATED
Payload =
{
 "href": "02367721",
 "ins": 02367721,
 "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
 "if": ["oic.if.baseline", "oic.if.a", "oic.if.rw"],
 "p": {"bm":3},
 "rep": {
  "value": false,
  "pushtarget": "ocf://850faa5d-ccaf-4293-9452-
f4fcab2e2c39/power/switch",
  "sourcert": ["oic.r.pushpayload"]
 }
}
```

Figure 21 – Creating a Push Proxy Resource

When a notification is triggered due to a change in the set of Resources identified by the Notification Selector, an UPDATE request is sent to the target Resource, and the Push Proxy enters the "wait for response" state. When a non-error response is received, the Push Proxy returns to the "wait for update" state.

11.3.3.3.4 Push Proxy Life Cycle

Figure 22 shows a life cycle example of a Push Proxy

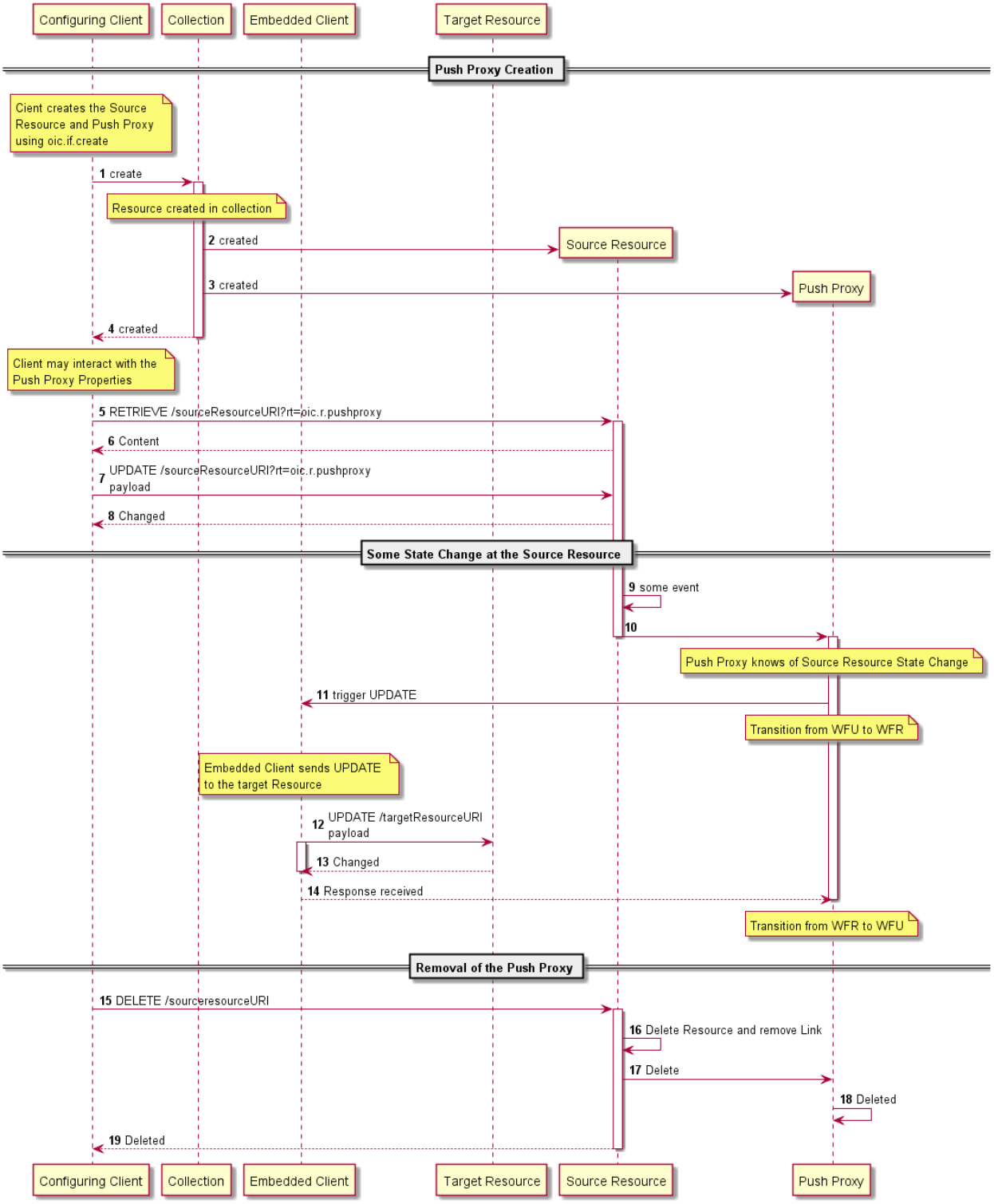


Figure 22 – Push Proxy Life Cycle Example

11.3.3.3.5 Security Considerations

– All requirements with regard to the behaviour of a Push Proxy as defined in the OCF Security Specification shall be met.

11.3.3.3.6 Push Configuration and Notification Selector Resource Types

The Push Configuration and Notification Selector Resource Types are as defined in Table 32.

Table 32 – Optional Push Notification Core Resources for Server Configuration

Example URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
“/example/notificationselectorURI”	Notification Selector	“oic.r.notificationselector”	“oic.if.rw”, ”oic.if.baseline”	The Resource through which the Server is configured with a selector for push notifications. The Properties exposed by the Resource are listed in Table 33.	Notifications
“/example/pushconfigurationURI	Push Configuration	“oic.r.pushconfiguration”	“oic.if.ll”, “oic.if.create”, ”oic.if.baseline”	A specialization of a Collection that contains only instances of "oic.r.notificationselector” with composed Push Proxy.	Notifications

Table 33 defines the details for the “oic.r.notificationselector” Resource Type.

Table 33 – "oic.r.notificationselector” Resource Type definition"

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Pushed Resource URI	phref	string			RW	no	URI of a Resource to be pushed
Pushed Resource Type	prt	array	array of strings		RW	no	Resource type(s) of Resource(s) to be pushed
Pushed Interface	pif	array	array of strings		RW	no	OCF Interface(s) of Resource(s) to be pushed

The Push Configuration (“oic.r.pushconfiguration”) Resource Type defines no Properties additional to those defined for all instances of a Collection in Table 9. However, the Push Configuration does impose restrictions of the values that shall be populated in the “rt” and “rts” Properties. These are described in Table 34 below.

Table 34 – "oic.r.pushconfiguration” Resource Type definition"

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Links	links	Array	See Table 10		RW	yes	See Table 10
Resource Type	rt	array	[“oic.r.pushconfiguration”]		R	yes	See Table 4
Resource Types	rts	array	[“oic.r.notificationselector”,"oic.r.pushproxy"]		R	yes	See Table 12

11.3.3.3.7 Push Configuration Collection Manipulation

Instances of a notification selector with an optionally composed Push Proxy may be created using the "oic.if.create" OCF Interface defined in 7.6.3.9.

A Server may expose an instance of "oic.r.pushconfiguration" with a pre-configured notification selector and composed Push Proxy if the target is known via other means; in this case a Client may be able to update the Push Proxy target via the "oic.if.rw" OCF Interface exposed by the composed Resource.

11.3.3.3.8 Notification Selector Population

The notification selector Resource is an object that contains Properties that may be applied as a selection filter to the set of possible Resources to be pushed. If the Resource is an empty object (i.e. "{}") then nothing shall be pushed. As such, a selector does not necessarily correspond to a single Resource on the origin Server, it may resolve to multiple Resources, and in the absence of a selector, resolution results in no Resources.

The Properties within the notification selector align with the set of Link Parameters that may exist for a Resource, thus matching semantically the actions of a query parameter.

Figure 23 shows an example notification selector that limits the Resources to be pushed to those that have the Resource Type that is listed. The result of the use of this selector is that all Resources on the origin Server, with an "rt" of "oic.r.sensor.carbonmonoxide", that are marked as "pushable", may have push notifications generated to the configured target Server whenever there is a state change in those Resources.


```
{"prt":["oic.r.sensor.carbonmonoxide"]}
```


Figure 23 – notification selector example for the given "prt"

Figure 24 shows an example notification selector that limits the Resources to be pushed to the Resource at the Resource URI that is listed. The result of the use of this selector is that the Resource on the origin Server identified with an "href" of "/myDevice/mySelectedResource", if it is marked as "pushable", may have push notifications generated to the configured target Server whenever there is a state change in that Resource.


```
{"phref":"/myDevice/mySelectedResource"}
```


Figure 24 – notification selector for the given "phref"

11.3.3.3.9 Notification Selector Operational Considerations

11.3.3.3.9.1 No Resources Match the Selector

If no Resources match the provisioned notification selector, then the set of Resources to be pushed is effectively an empty set and no notifications shall be generated by the Server.

If a Resource that was selected via application of the notification selector is deleted, leaving the set of Resources to be pushed as an empty set, no notifications are generated by the Server.

11.3.3.3.9.2 Push Proxy Population Considerations

Associated with each Resource linked from the Push Configuration there shall be a composed Push Proxy, and the following requirements apply to the Push Proxy Resource in such cases.

For each Push Proxy, the "pushtarget" Property is populated with the target URI for the push. The target provided by the Push Proxy shall not be a multicast address. If a Client attempts to configure such an address the Server shall reject the request with a response code indicating bad request.

For each Push Proxy, if a Client attempts to configure the "sourcert" Property to any value other than "oic.r.pushpayload", then the Server shall reject the request with a response code indicating bad request.

Figure 25 provides an example of a notification selector Resource composed with a Push Proxy that results the sending of notifications to the target identified by the "pushtarget" Property.


```
{
    "rt": ["oic.r.notificationselector","oic.r.pushproxy"],
    "phref":"/myDevice/mySelectedResource",
    "pushtarget": "ocf://myTarget/myTargetURI",
    "sourcert": "oic.r.pushpayload",
    "state": "waitingforprovisioning"
}
```

Figure 25 – Example composed notificationselector and pushproxy

11.3.3.4 Target Server Configuration

All push notifications contain a payload of type "oic.r.pushpayload" (see clause 11.3.3.4.2). Within the payload shall be the complete representation (i.e. all Properties including Common Properties) of the Resource at the origin Server which experienced the state change that generated the notification. Should the target Server have limitations with respect to the Resource Types that can be received as part of an "oic.r.pushpayload" representation that the Server may optionally be explicitly configured to define which Resource Types it is capable of receiving. This is realized by exposing an instance of "oic.r.pushreceiver" which is an object array, each object containing the URI to which information may be pushed and an instance of the “rts” Property (see 7.8.2.8) listing the Resource Types that can be received within a pushed payload representation. If the “rts” Property is an empty array then there is no restriction in the Resource Types than can be received in an "oic.r.pushpayload" representation. A target Server shall expose an instance of "oic.r.pushreceiver" Resource.

The Push Receiver Resource Type is as defined in Table 35.

Table 35 – Optional Push Receiver Core Resources for Target Server Configuration

Example URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
“/example/pushreceiverURI”	Push Receiver	“oic.r.pushreceiver”	“oic.if.rw”, ”oic.if.baseline”	The Resource through which a Device can be configured as a target for push notifications. The Properties exposed by the Resource are listed in Table 36.	Notifications

Table 36 defines the details for the “oic.r.pushreceiver” Resource Type.

Table 36 – "oic.r.pushreceiver” Resource Type definition"

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Push Receivers	receivers	array of object			RW	yes	Resource set that can be received by the push target

Each object within the "receivers" array is made up of two Properties, these are defined in Table 37.

Table 37 – "receivers" object definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Receiver URI	receiveruri	URI			RW	yes	URI at which push notifications may be received
Resource Types	rts	array of string			RW	yes	Resource Type values that may be received at the "receiveruri"

When the configuration Client sends an UPDATE to the target Server it may use a "receiveruri" query parameter, otherwise the configuration Client should provide the whole list of the "receivers" object array which shall replace existing "receivers" Property. When the push target receives an UPDATE with a "receiveruri" query parameter, it shall only update receiver object which matches the "receiveruri" query parameter. If there is no receiver object which matches the "receiveruri" query parameter, a new receiver object shall be created.

When the configuration Client sends a DELETE to the target Server it may use a "receiveruri" query parameter, otherwise the configuration Client shall remove whole receiver object array. When the push target receives a DELETE with a "receiveruri" query parameter, it shall only remove the receiver object which matches the "receiveruri" query parameter.

An example of an instance of "oic.r.pushreceiver" is provided in Figure 26.

```
"receivers": [
    {"receiveruri": "/mylocaltargeturiforthermostats",
     "rts": ["oic.r.temperature","oic.r.humidity"]
    },
    {"receiveruri": "/mylocaltargeturifordontcare",
     "rts": []
    }
]
```

Figure 26 – example push receiver configuration

11.3.3.4.1 Target Server Behaviour

If all "rt" values in the pushpayload are part of the configured "rts" Property against the target URI, or if the "rts" Property is empty, then the push target shall handle the operation as an UPDATE against the target URI with the caveat that the entirety of the received pushpayload is handled as a writeable entity.

If a push target receives an UPDATE operation containing a pushpayload that it is not configured to handle (i.e. an “rt” in the pushpayload is not part of the configured “rts” against the target URI) then the push target shall respond with a response of "Forbidden".

If a push target receives an UPDATE operation containing a pushpayload that it is configured to handle (i.e. all “rt” values in the pushpayload are part of the configured “rts” against the target URI) but it is unable to parse a provided Resource Representation (contents of the “rep” Property) then the push target shall respond with a response of "Bad Request".

A Server may choose to make the "receiveruri" discoverable or non-discoverable. However, if a RETRIEVE to the "receiveruri" would return an empty payload (i.e. nothing has been pushed to it) then the Resource shall be non-discoverable.

11.3.3.4.2 Notification Payload

Anytime there is a change in the state of the Resources that are subject to a push notification as defined by the application of a specific notification selector, the Server shall send an UPDATE operation to the target defined in the "pushtarget" Property of the Push Proxy that is composed with the notification selector. The representation provided in the UPDATE operation shall be an instance of an "oir.r.pushpayload" Resource. The Push Payload Resource is defined in Table AA; this is an array of representations where for each representation the Resource also provides additional meta-information to enable the target Server to understand the contents of the representation itself (e.g. the Resource Type, the OCF Interface that has been applied). An array is used as more than one Resource may be providing state information at any one time.

Table 38 – Push Payload Resource

Example URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
“/example/pushpayloadURI”	Push Notification Payload	“oic.r.pushpayload”	“oic.if.r”, ”oic.if.baseline”	The Resource through which pushed notification information is provided to the push target The Resource exposes an array of JSON objects, the Properties exposed by the objects are listed in Table 39.	Notifications

Table 39 defines the details for the JSON object that is carried as an array item within the “oic.r.pushpayload” Resource Type.

Table 39 – "oic.r. pushpayload” array entry definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Context URI	anchor	uri			R	yes	Context URI of the Resource being pushed
URI	href	string			R	no	Server URI of the Resource being pushed
Resource Type	rt	array			R	yes	Resource Type of the Resource being pushed
Interface	if	array			R	yes	OCF Interface(s) that are valid for the Resource being pushed
Representation	rep	object			R	yes	Resource representation

Note that the “href”, if included, is always a relative URI to the root of the source Device (i.e. the Server that is originating the push request).

The "anchor" Property contains an OCF URI with the authority component set to the <deviceID> of the Device hosting the source Resource.

An example of an instance of "oic.r.pushpayload" is provided in Figure 27.

```
[{
   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
   "href": "/mysensor",
   "rt": ["oic.r.sensor.carbonmonoxide"],
   "if": ["oic.if.s"],
   "rep": {
      "value": true
   }
}]
```

Figure 27 – Example pushpayload content

11.3.3.4.3 Observability of Notification Selectors

Notification selector does not support Observation because all notification selectors are dynamically created and deleted ones.

11.3.3.5 Common requirement for origin Server and target Server

Origin Server and target Server shall expose "oic.d.push" as a value of "rt" of "/oic/d" to show that they can push or receive push notifications.

11.4 Introspection

11.4.1 Overview

Introspection is a mechanism to announce the capabilities of Resources hosted on the Device.

The intended usage of the Introspection Device Data (IDD) is to enable dynamic Clients e.g. Clients that can use the IDD) to generate dynamically a UI or dynamically create translations of the hosted Resources to another eco-system. Another use of Introspection is that the information can be used to generate Client code. The IDD is designed to augment the existing data already on the wire. This means that existing mechanisms need to be used to get a full overview of what is implemented in the Device. For example, the IDD does not convey information about Observability, since that is already conveyed with the "p" Property on the Links in "/oic/res" (see 7.8.2.5.3).

The IDD is recommended to be conveyed as static data. Meaning that the data does not change during the uptime of a Device. However, when the IDD is not static, the Introspection Resource shall be Observable and the url Property Value of "oic.wk.introspection" Resource shall change to indicate that the IDD is changed.

The IDD describes the Resources that make up the Device. For the complete list of included Resources see Table 22. The IDD is described as a OpenAPI 2.0 in JSON format file. The text in the following bulleted list contains OpenAPI 2.0 terms, such as paths, methods etc. The OpenAPI 2.0 file shall contain the description of the Resources:

- The IDD will use the HTTP syntax, e.g., define the CRUDN operation as HTTP methods and use the HTTP status codes.
- The IDD does not have to define all the status codes that indicate an error situation.
- The IDD does not have to define a schema when the status code indicates that there is no payload (see HTTP status code 204 as an example).
- The paths (URLs) of the Resources in the IDD shall be without the OCF Endpoint description, e.g. it shall not be a fully-qualified URL but only the relative path from the OCF Endpoint, aka the "href". The relative path may include a query parameter (e.g. "?if=oic.if.ll"), in such cases the text following (and including) the "?" delimiter shall be removed before equating to the "href" that is conveyed by "/oic/res".
- The following Resources shall be excluded in the IDD:
  - Resource with Resource Type: "oic.wk.res" unless 3rd party defined or optional Properties are implemented.
  - Resource with Resource Type: "oic.wk.introspection".
  - Resources explicitly identified within other specifications working in conjunction with this document (e.g. Resources that handle Wi-Fi Easy Setup, see [2]).
- The following Resources shall be included in the IDD when optional or 3rd party defined Properties are implemented:
  - Resources with type: "oic.wk.p" and "oic.wk.d" (e.g. discovery related Resources).
  - Security Virtual Resources from ISO/IEC 30118-2.
- When the Device does not expose instances of Vertical Resource Types, and does not have any 3rd party defined Resources (see 7.8.4.4), and does not need to include Resources in the IDD due to other clauses in this clause, then the IDD shall be an empty OpenAPI 2.0 file. An example of an empty OpenAPI 2.0 file can be found in found in Annex **B.2**.

- All other Resources that are individually addressable by a Client (i.e. the "href" can be resolved and at least one operation is supported with a success path response) shall be listed in the IDD.
- Per Resource the IDD shall include:
  - All implemented methods
    - For an OCF defined Resource Type, only the methods that are listed in the OpenAPI 2.0 definition are allowed to exist in the IDD. For an OCF defined Resource Type, methods not listed in the OpenAPI 2.0 definition shall not exist in the IDD. The supported methods contained in the IDD shall comply with the listed OCF Interfaces. For example, if the POST method is listed in the IDD, then an OCF Interface that allows UPDATE will be listed in the IDD.
  - Per supported method:
    - Implemented query parameters per method.
      - This includes the supported OCF Interfaces ("if") as enum values.
    - Schemas of the payload for the request and response bodies of the method.
    - Where the schema provides the representation of a batch request or response ("oic.if.b") the schema shall contain the representations for all Resource Types that may be included within the batch representation. The representations shall be provided within the IDD itself.
    - The schema data shall be conveyed by the OpenAPI 2.0 schema.
    - The OpenAPI 2.0 schema object shall comply with:
      - The schemas shall be fully resolved, e.g. no references shall exist outside the OpenAPI 2.0 file.
      - The schemas shall list which OCF Interfaces are supported on the method.
      - The schemas shall list if a Property is optional or required.
      - The schemas shall include all Property validation keywords. Where an enum is defined the enum shall contain the values supported by the Device. When vendor defined extensions exist to the enum (defined in accordance to 7.8.4.4) these shall be included in the enum.
      - The schemas shall indicate if a Property is read only or read-write.
        - By means of the readOnly schema tag belonging to the Property.
        - Default value of readOnly is false as defined by OpenAPI 2.0.
      - The default value of the "rt" Property shall be used to indicate the supported Resource Types.
      - oneOf and anyOf constructs are allowed to be used as part of a OpenAPI 2.0 schema object. The OpenAPI 2.0 schema with oneOf and anyOf constructs can be found in Annex **B.1**.
- For Atomic Measurements (see clause 7.8.4), the following apply:
  - The "rts" Property Value in the IDD shall include only the Resource Types the instance contains and not the theoretical maximal set allowed by the schema definition.
  - The Resources that are part of an Atomic Measurement, excluding the Atomic Measurement Resource itself, shall not be added to their own individual path in the IDD, as they are not individually addressable; however, the schemas for the composed Resource Types shall be provided in the IDD as part of the batch response definition along with the "href" for the Resource.

Dynamic Resources (e.g. Resources that can be created on a request by a Client) shall have a URL definition which contains a URL identifier (e.g. using the {} syntax). A URL with {} identifies

that the Resource definition applies to the whole group of Resources that may be created. The actual path may contain the Collection node that links to the Resource.

Example of a URL with identifiers:

`/SceneListResURI/{SceneCollectionResURI}/{SceneMemberResURI}:`

When different Resource Types are allowed to be created in a Collection, then the different schemas for the CREATE method shall define all possible Resource Types that may be created. The schema construct oneOf allows the definition of a schema with selectable Resources. The oneOf construct allows the integration of all schemas and that only one existing sub schema shall be used to indicate the definition of the Resource that may be created.

Example usage of oneOf JSON schema construct is shown in Figure 28:

```
{
  "oneOf": [
    { <<subschema 1 definition>> },
    { << sub schema 2 definition >> }
…
  ]
}
```

Figure 28 – Example usage of oneOf JSON schema

A Client using the IDD of a Device should check the version of the supported IDD of the Device. The OpenAPI 2.0 version is indicated in each file with the tag "swagger". Example of the 2.0 supported version of the tag is: "swagger": "2.0". Later versions of this document may reference newer versions of the OpenAPI specification, for example 3.0.

A Device shall support one Resource with a Resource Type of "oic.wk.introspection" as defined in Table 40. The Resource with a Resource Type of "oic.wk.introspection" shall be included in the Resource "/oic/res".

An empty IDD file, e.g. no URLs are exposed, shall still have the mandatory OpenAPI 2.0 fields. See OpenAPI specification. An example of an empty OpenAPI 2.0 file can be found in found in Annex B.2.

Table 40 – Introspection Resource

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
none	Introspection	"oic.wk.introspection"	"oic.if.r"	The Resource that announces the URL of the Introspection file.	Introspection

Table 41 defines "oic.wk.introspection" Resource Type.

Table 41 – "oic.wk.introspection" Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
urlInfo	"urlInfo"	"array"	N/A	N/A	R	Yes	array of objects
url	"url"	"string"	"uri"	N/A	R	Yes	URL to the hosted payload
protocol	"protocol"	"string"	"enum"	N/A	R	Yes	Protocol definition to retrieve the Introspection Device Data from the url.
content-type	"content-type"	"string"	"enum"	N/A	R	No	content type of the url.

version	"version"	"integer"	"enum"	N/A	R	No	Version of the Introspection protocol, indicates which rules are applied on the Introspection Device Data regarding the content of the OpenAPI 2.0 file. Current value is 1.

If the IDD is hosted on the local Device, then an additional "url" specified as an OCF URI should be listed.

Example payload with an entry using a scheme of "coap" and an entry with an OCF URI:

```
{
"rt": ["oic.wk.introspection"],
"urlInfo": [
  {
    "content-type": "application/cbor",
    "protocol": "coap",
    "url": "coap://[fe80::1]:1234/IntrospectionExampleURI"
   },
   {
    "content-type": "application/cbor",
    "protocol": "coap",
    "url": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9/IntrospectionExampleURI"
   }
  ]
}
```

11.4.2 Usage of Introspection

The Introspection Device Data is retrieved in the following steps and as depicted in Figure 29:

- Check if the Introspection Resource is supported and retrieve the URL of the Resource.
- Retrieve the contents of the Introspection Resource
- Download the Introspection Device Data from the URL specified the Introspection Resource.
- Usage of the Introspection Device Data by the Client

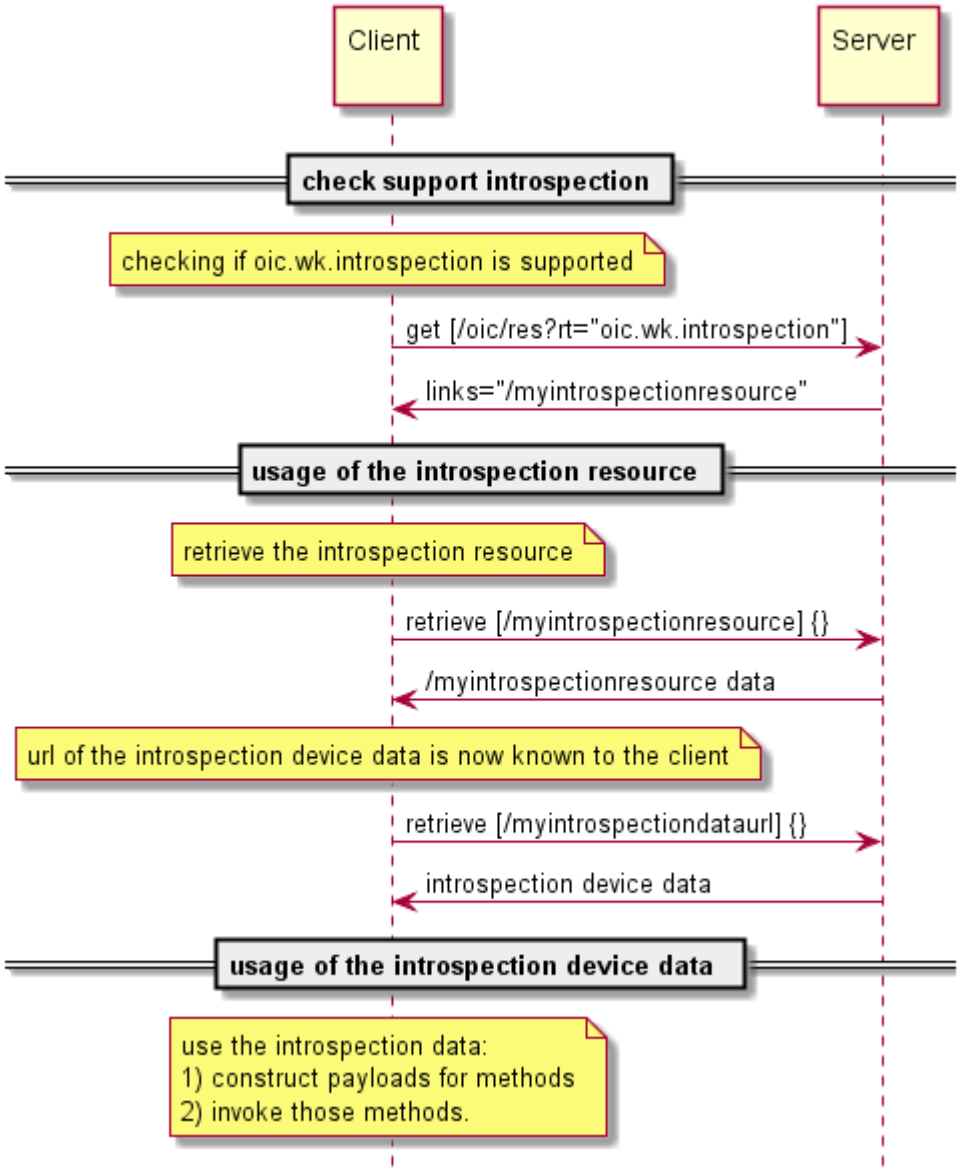


Figure 29 – Interactions to check Introspection support and download the Introspection Device Data.

11.5 Semantic Tags

11.5.1 Introduction

Semantic Tags are meta-information associated with a specific Resource instance that are represented as both Link Parameters and Resource Properties that provide a mechanism whereby the Resource be annotated with additional contextual metadata that helps describe the Resource.

When a Semantic Tag is defined for a Resource, it shall be present as a Link Parameter in all Links that are present that target the Resource, including Links in "/oic/res" if the Resource is a Discoverable Resource. The Semantic Tag is further treated as a Common Property associated with the Resource and so shall be returned as part of the "baseline" response for the Resource if a Semantic Tag has been populated.

11.5.2 Semantic Tag definitions

11.5.2.1 Relative and descriptive position Semantic Tags

11.5.2.1.1 Introduction

Consider where there may be multiple instances of the same Resource Type exposed by a Device; or a case where there may be potentially ambiguity with regard to the physical attribute that a Resource is representing. In such a case the ability to annotate the Links to the Resource with information pertaining to the relative position of the Resource within the Physical Device becomes useful.

11.5.2.1.2 "tag-pos-desc" or position description Semantic Tag

The "tag-pos-desc" Semantic Tag as defined in Table 42 describes the position of the Resource as a descriptive position. If the tag is not exposed it conveys the same meaning as if the tag is exposed with a value of "unknown". The value for the "tag-pos-desc" Semantic Tag if exposed, shall be a string containing a value from the enumeration detailed in Annex C. The population of the Semantic Tag is defined by the Device vendor and shall not be mutable by a Client.

Table 42 – "tag-pos-desc" Semantic Tag definition

Link Parameter name	Type	Contents	Value example
"tag-pos-desc"	enum	See Annex C	"tag-pos-desc": "topleft"

11.5.2.1.3 "tag-pos-rel" or relative position Semantic Tag

The "tag-pos-rel" Semantic Tag describes the position of the Resource as a relative position in 3D space against a known point defined by the Device vendor. The known point is defined using [x,y,z] form as [0.0,0.0,0.0]. The position itself is then represented by the x-, y-, and z- plane relative position from this known point using a bounded box of size +1.0/-1.0 in each plane.

Figure 30 illustrates the definition of "tag-pos-rel".

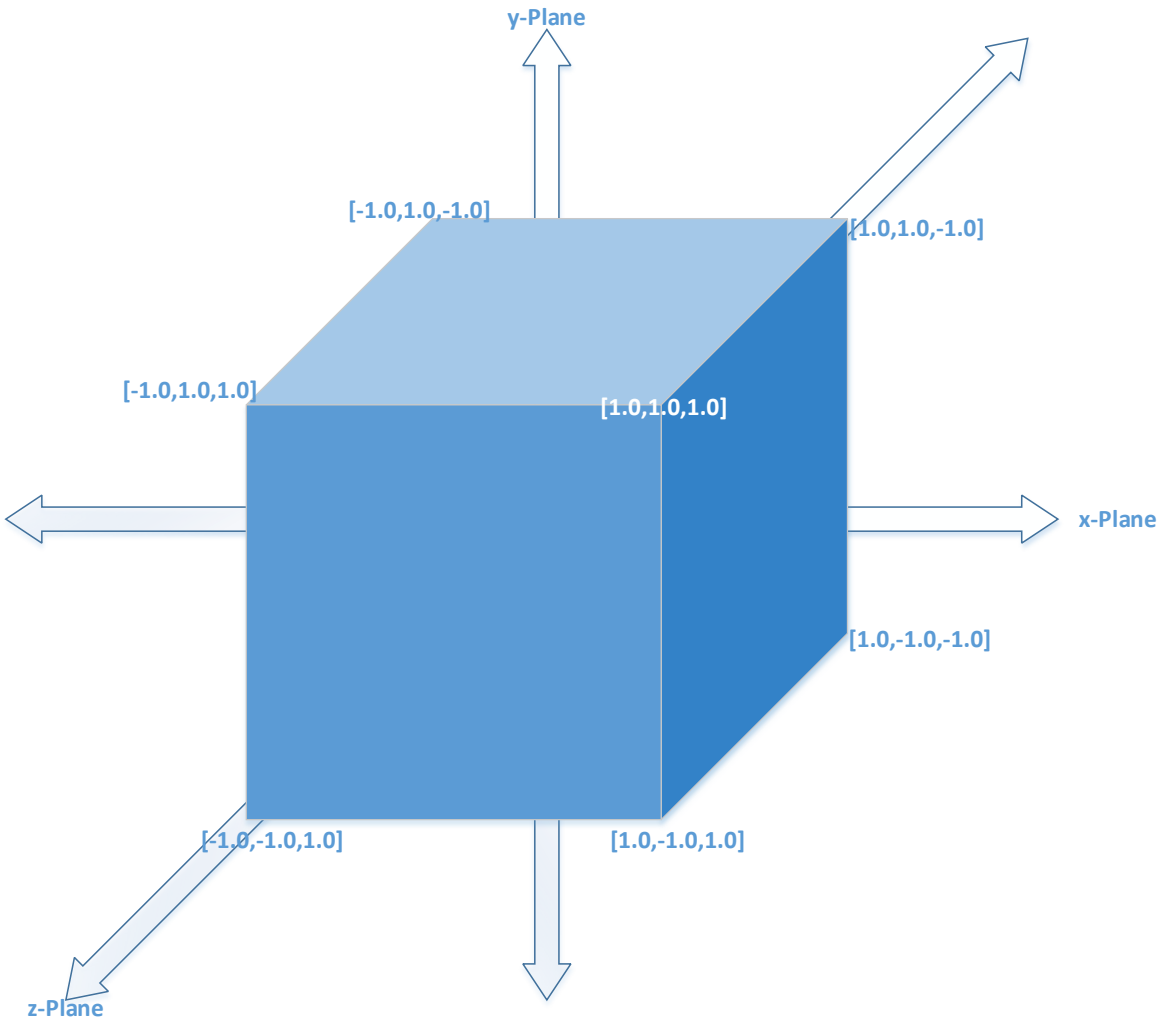


Figure 30 – "tag-pos-rel" definition

The "tag-pos-rel" Semantic Tag value is defined by the Device vendor and shall not be mutable by a Client. This is detailed in Table 43.

Table 43 – "tag-pos-rel" Semantic Tag definition

Link Parameter name	Type	Contents	Value example
"tag-pos-rel"	array	Three element array of numbers defining the position relative to a known [0,0,0] point within the context of an abstract box [-1,-1,-1],[1,1,1].	"tag-pos-rel": [0.5,0.5,0.5]

11.5.2.2 Functional behaviour Semantic Tags

11.5.2.2.1 Introduction

Consider, for example, the case of a Device that supports two target temperatures simultaneously for different modes of operation, for example a temperature for heating and a separate temperature for cooling.

There is then an ambiguity with respect to the target mode of the specific temperature Resource; it isn't explicit which instance of temperature is associated with which Device function. In such a case the ability to annotate the Links to the Resource with information pertaining to the function of the Resource within the Physical Device becomes useful.

11.5.2.2.2 "tag-func-desc" or function description Semantic Tag

The "tag-func-desc" Semantic Tag describes the function of the Resource, if exposed it shall be populated with a value from the currently supported set of standardized enumeration values defined by the Device ecosystem specifications. If the tag is not exposed it conveys the same meaning as if the tag is exposed with a value of "unknown". The value for the "tag-func-desc" Semantic Tag, if exposed, is defined by the Device vendor and shall not be mutable by a Client.

This "tag-func-desc" Semantic Tag is detailed in Table 44.

Table 44 – "tag-func-desc" Semantic Tag definition

Link Parameter name	Type	Contents	Value example
"tag-func-rel"	enum	Defined by Device ecosystem	"tag-func-desc": "cool"

11.5.2.3 Location Semantic Tags

11.5.2.3.1 Introduction

Consider a Bridge, Resource Directory or other similar concept whereby the Link to the Device Resource ("oic.wk.d") that is exposed may reference or relate to a physically separate Device. In such a case the ability to annotate the Link to the Device Resource with location information becomes useful. Additionally, in a deployment of multiple similar or identical Devices, the ability to annotate the Device with where it is deployed assists in disambiguation.

11.5.2.3.2 "tag-locn" or location description Semantic Tag

The “tag-locn” Semantic Tag may be exposed as a Link Parameter for the Device Resource, it describes the physical location of the target Device, it shall not be exposed as a Link Parameter for any other Resource Type. If the tag is not exposed it conveys the same meaning as if the tag is exposed with a value of “unknown”. The initial value for the “tag-locn” Semantic Tag if exposed shall be “unknown”. This Link Parameter shall not contain any 3rd party defined values.

The "tag-locn" shall be exposed as string containing a value from the enumeration ("locn-descriptions") defined in Annex C. The tag is detailed in Table 45.

An instance of "tag-locn" may be updated by a Client by modifying the reflected instance of this value that is present in the Configuration Resource, see [1].

Table 45 – "tag-locn" Semantic Tag definition

Semantic Tag Name	Type	Contents	Value example
tag-locn	Enumeration	See Annex C	“tag-locn”: “familyroom”

12 Messaging

12.1 Introduction

This clause specifies the protocol messaging mapping to the CRUDN messaging operations (clause 7.11) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols is expected in later version of this document. All the Property information from the Resource model shall be carried within the message payload. This payload shall be generated in the Resource model layer and shall be encapsulated in the data connectivity layer. The message header shall

only be used to describe the message payload (e.g., verb, mime-type, message payload format), in addition to the mandatory header fields defined in a messaging protocol (e.g., CoAP) specification. If the message header does not support this, then this information shall also be carried in the message payload. Resource model information shall not be included in the message header structure unless the message header field is mandatory in the messaging protocol specification.

When a Resource is specified with a RESTful description language like OpenAPI 2.0 then the HTTP syntax definitions are used in the description (e.g., HTTP syntax for the CRUDN operations, status codes, etc.). The HTTP syntax will be mapped to the actual used web transfer protocol (e.g., CoAP).

The communication is largely based on UDP and UDP has defined the Maximum Transmission Unit (MTU). All UDP payload size communications shall not exceed the MTU size as per by the IETF RFC 8085 clause 3.2. This is to avoid being dependent on package reassembly by the operating systems.

12.2 Mapping of CRUDN to CoAP

12.2.1 Overview

A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in clause 12.2.3. A Device implementing CoAP shall conform to IETF RFC 7641 to implement the CoAP Observe option. Support for CoAP block transfer when the payload is larger than the MTU is defined in 12.2.8.

12.2.2 URIs

An OCF: URI is mapped to a coap: URI by replacing the scheme name "ocf" with "coap" if unsecure or "coaps" if secure before sending over the network by the requestor. Similarly, on the receiver side, the scheme name is replaced with "ocf".

Any query string that is present within the URI is encoded as one or more URI-Query Options as defined in IETF RFC 7252 clause 6.4.

12.2.3 CoAP method with request and response

12.2.3.1 Overview

Every request has a CoAP method that realizes the request. The primary methods and their meanings are shown in Table 46, which provides the mapping of GET/POST/DELETE methods to CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides the generic behaviours when using these methods, however Resource OCF Interfaces may modify these generic semantics. The HTTP codes in the RESTful descriptions will be translated as described in IETF RFC 8075 clause 7 Response Code Mapping. CoAP methods not listed in Table 46 are not supported.

Table 46 – CoAP request and response

Method for CRUDN	(mandatory) Request data	(mandatory) Response data
GET for RETRIEVE	- **Method code**: GET (0.01). - **Request URI**: an existing URI for the Resource to be retrieved	- **Response code**: success (2.xx) or error (4.xx or 5.xx). - **Payloa**d: Resource representation of the target Resource (when successful).
POST for CREATE	- **Method code**: POST (0.02). - **Request URI**: an existing URI for the Resource responsible for the creation. - **Payload**: Resource presentation of the Resource to be created.	- **Response code**: success (2.xx) or error (4.xx or 5.xx). - **Payload**: the URI of the newly created Resource (when successful).

POST for UPDATE	- **Method code**: POST (0.02). - **Request URI**: an existing URI for the Resource to be updated. - **Payload**: representation of the Resource to be updated.	- **Response Code**: success (2.xx) or error (4.xx or 5.xx).
DELETE for DELETE	**- Method code**: DELETE (0.04). - **Request URI**: an existing URI for the Resource to be deleted.	- **Response code**: success (2.xx) or error (4.xx or 5.xx).

12.2.3.2 CREATE with POST

POST with the "oic.if.create" OCF Interface query parameter (i.e., "POST ?if=oic.if.create") shall be used only in situations where the request URI is valid, that is it is the URI of an existing Resource on the Server that is processing the request. If no such Resource is present, the Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall use an existing request URI which identifies the Resource on the Server responsible for creation. The URI of the created Resource is determined by the Server and provided to the Client in the response.

A Client shall include the representation of the new Resource in the request payload. The new resource representation in the payload shall have all the necessary Properties to create a valid Resource instance, i.e. the created Resource should be able to properly respond to the valid Request with mandatory OCF Interface (e.g., "GET with ?if=oic.if.baseline").

Upon receiving the POST request, the Server shall either:

– Create the new Resource with a new URI, respond with the new URI for the newly created Resource and a success response code (2.xx); or

– respond with an error response code (4.xx or 5.xx).

12.2.3.3 RETRIEVE with GET

GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of the target Resource identified by the request URI.

Upon receiving the GET request, the Server shall either:

– Send back the response with the representation of the target Resource with a success response code (2.xx); or

– respond with an error response code (4.xx or 5.xx) or ignore it (e.g. non-applicable multicast GET).

GET is a safe method and is idempotent.

12.2.3.4 UPDATE with POST

POST shall be used only in situations where the request URI is valid, that is it is the URI of an existing Resource on the Server that is processing the request. If no such Resource is present, the Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE Property values of an existing Resource.

Upon receiving the request, the Server shall either:

– Apply the request to the Resource identified by the request URI in accordance with the applied OCF Interface (i.e. POST for non-existent Properties is ignored) and send back a response with a success response code (2.xx); or

– respond with an error response code (4.xx or 5.xx). Note that if the representation in the payload is incompatible with the target Resource for POST using the applied OCF Interface (i.e. the overwrite semantic cannot be honoured because of read-only Property in the payload), then the error response code 4.xx shall be returned.

12.2.3.5 DELETE with DELETE

DELETE shall be used for DELETE operation. The DELETE method requests that the Resource identified by the request URI be deleted.

Upon receiving the DELETE request, the Server shall either:

– Delete the target Resource and send back a response with a success response code (2.xx); or

– respond with an error response code (4.xx or 5.xx).

DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

12.2.4 Content-Format negotiation

The Framework mandates support of CBOR, however it allows for negotiation of the payload body if more than one Content-Format (e.g. CBOR and JSON) is supported by an implementation. In this case the Accept Option defined in clause 5.10.4 of IETF RFC 7252 shall be used to indicate which Content–Format (e.g. JSON) is requested by the Client.

The Content-Formats supported are shown in Table 47.

Table 47 – OCF Content-Formats

Media Type	ID
"application/vnd.ocf+cbor"	10000

Clients shall include a Content-Format Option in every message that contains a payload. Servers shall include a Content-Format Option for all success (2.xx) responses with a payload body. Per IETF RFC 7252 clause 5.5.1, Servers shall include a Content-Format Option for all error (4.xx or 5.xx) responses with a payload body unless they include a Diagnostic Payload; error responses with a Diagnostic Payload do not include a Content-Format Option. The Content-Format Option shall use the ID column numeric value from Table 47. An OCF vertical may mandate a specific Content-Format Option.

Clients shall also include an Accept Option in every request message. The Accept Option shall indicate the required Content-Format as defined in Table 47 for response messages. The Server shall return the required Content-Format if available. If the required Content-Format cannot be returned, then the Server shall respond with an appropriate error message.

12.2.5 OCF-Content-Format-Version information

Servers and Clients shall include the OCF-Content-Format-Version Option in both request and response messages with a payload. Clients shall include the OCF-Accept-Content-Format-Version Option in request messages. The OCF-Content-Format-Version Option and OCF-Accept-Content-Format-Version Option are specified as Option Numbers in the CoAP header as shown in Table 48.

Table 48 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option Numbers

CoAP Option Number	Name	Format	Length (bytes)
2049	OCF-Accept-Content-Format-Version	uint	2

2053	OCF-Content-Format-Version	uint	2

The value of both the OCF-Accept-Content-Format-Version Option and the OCF-Content-Format-Version Option is a two-byte unsigned integer that is used to define the major, minor and sub versions. The major and minor versions are represented by 5 bits and the sub version is represented by 6 bits as shown in Table 49.

Table 49 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version Representation

	Major Version					**Minor Version**					**Sub Version**					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 50 illustrates several examples:

Table 50 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Representation

OCF version	Binary representation	Integer value
"1.0.0"	"0000 1000 0000 0000"	2048
"1.1.0"	"0000 1000 0100 0000"	2112

The OCF-Accept-Content-Format-Version Option and OCF-Content-Format-Version Option for this version of the document shall be "1.0.0" (i.e. "0b0000 1000 0000 0000").

12.2.6 Content-Format policy

All Devices shall support the current Content-Format Option, "application/vnd.ocf+cbor", and OCF-Content-Format-Version "1.0.0".

For backward compatibility with previous OCF-Content-Format-Version Options:

– All Client Devices shall support OCF-Content-Format-Version Option set to "1.0.0" and higher.

– All Client Devices shall support OCF-Accept-Content-Format-Version Option set to "1.0.0" and higher.

– A Client shall send a discovery request message with its Accept Option set to "application/vnd.ocf+cbor", and its OCF-Accept-Content-Format-Version Option matching its highest supported version.

– A Server shall respond to a Client's discovery request that is higher than its OCF-Content-Format-Version by responding with its Content-Format Option set to "application/vnd.ocf+cbor", and OCF-Content-Format-Version matching its highest supported version. The response representation shall be encoded with the OCF-Content-Format-Version matching the Server's highest supported version.

– A Server may support previous Content-Formats and OCF-Content-Format-Versions to support backward compatibility with previous versions.

– For a Server that supports multiple OCF-Content-Format-Version Options, the Server should attempt to respond with an OCF-Content-Format-Version that matches the OCF-Accept-Content-Format-Version of the request.

To maintain compatibility between Devices implemented to different versions of this document, Devices should follow the policy as described in Figure 31.

The OCF Clients in Figure 31 support sending Content-Format Option set to "application/vnd.ocf+cbor", Accept Option set to "application/vnd.ocf+cbor", OCF-Content-Format-Version Option set to "1.0.0", and OCF-Accept-Content-Format-Version Option set to "1.0.0" (representing OCF 1.0 and later Clients). The OCF Servers in Figure 31 support sending Content-Format Option set to "application/vnd.ocf+cbor" and OCF-Content-Format-Version Option set to "1.0.0" (representing OCF 1.0 and later Servers).

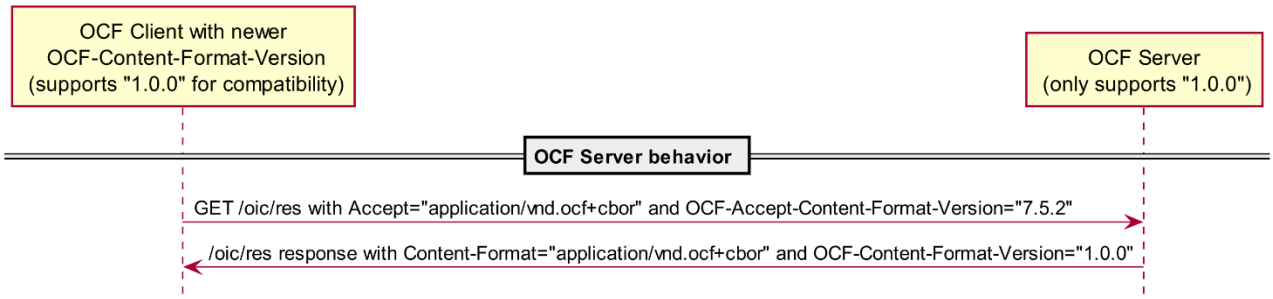


Figure 31 – Content-Format Policy for backward compatible OCF Clients negotiating lower OCF Content-Format-Version

12.2.7 CRUDN to CoAP response codes

The mapping of CRUDN operations response codes to CoAP response codes are identical to the response codes defined in IETF RFC 7252.

A Client that receives a CoAP response with a response code of 5.03 that also includes a Max-Age option, should re-attempt the original request after waiting for a period of at least that provided in the Max-Age option.

12.2.8 CoAP block transfer

Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT devices. However, scenarios can be envisioned in which an application needs to transfer larger payloads.

CoAP block-wise transfer as defined in IETF RFC 7959 shall be used by all Servers which generate a content payload that would exceed the size of a CoAP datagram as the result of handling any defined CRUDN operation.

Similarly, CoAP block-wise transfer as defined in IETF RFC 7959 shall be supported by all Clients. The use of block-wise transfer is applied to both the reception of payloads as well as transmission of payloads that would exceed the size of a CoAP datagram.

A Client may support both the block1 (as descriptive) and block2 (as control) options as described by IETF RFC 7959. A Server may support both the block1 (as control) and block2 (as descriptive) options as described by IETF RFC 7959.

12.2.9 Generic requirements for CoAP multicast

A Client may use CoAP multicast to retrieve a target Resource with a fixed local path from multiple other Devices. This clause provides generic requirements for this mechanism.

– Devices shall join the All OCF Nodes multicast groups (as defined in [IANA IPv6 Multicast Address Space Registry]) with scopes 2, 3, and 5 (i.e., ff02::158, ff03::158 and ff05::158) and

shall listen on the port 5683. For compliance to IETF RFC 7252 a Device may additionally join the All CoAP Nodes multicast groups.

– Clients intending to discover Resources shall join the multicast groups as defined in the first bullet.

– Clients shall send multicast requests to the All OCF Nodes multicast group address with scope 2 ("ff02::158") or with scope 5 ("ff05::158") at port "5683". The requested URI shall be the fixed local path of the target Resource optionally followed by query parameters. For compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast groups.

– To discover Devices on a low-rate wireless personal area network (LR-WPAN) [see IETF RFC 7346], Clients should send additional discovery requests (GET request) to the *All OCF Nodes* multicast group address with REALM_LOCAL scope 3 ("ff03::158") at port "5683". The set of replying Devices then can be used to distinguish if the Device is SITE_LOCAL or REALM_LOCAL to the Client discovering the Devices. Such request shall use the IPv6 hop limit with a value of 255. If the Client sends discovery requests to *All OCF Nodes*, then for compliance to IETF RFC 7252 a Client may additionally send to the *All CoAP Nodes* multicast groups with the same REALM_LOCAL scope with the IPv6 hop limit value of 255.

– Clients should send discovery requests (GET request) to the *All OCF Nodes* multicast group address with SITE_LOCAL scope 5 ("ff05::158") at port "5683". Such request shall use the IPv6 hop limit with a value of 255. If the Client sends discovery requests to *All OCF Nodes*, then for compliance to IETF RFC 7252 a Client may additionally send to the *All CoAP Nodes* multicast groups with the same SITE_LOCAL scope with the IPv6 hop limit value of 255.

– The multicast request shall be permitted by matching the request to an ACE which permits unauthenticated access to the target Resource as described in ISO/IEC 30118-2.

– Handling of multicast requests shall be as described in clause 8 of IETF RFC 7252 and clause 4.1 in IETF RFC 6690.

– Devices which receive the request shall respond, subject to query parameter processing specific to the requested Resource.

– A Device may expose the All OCF Nodes multicast address as part of an OCF Endpoint "eps" Link Parameter for a Resource, see clause 10. The behaviour of a Device that receives a CRUDN operation on the exposed multicast address for such a Resource (outside of those Resources explicitly defined as being for the purposes of discovery, see clause 11.2.3) is not specified by this document.

12.2.10 Setting timeout on response to a confirmable request

The timeout specified by "oic.wk.res:eps[]:lat", when present, should only be taken into account by the Client when the Server is in the "ready for normal operation state" [see clause 8.5 in ISO/IEC 30118-2] and the request made is a confirmable request. The Server should only enable the state that will cause latency when in "ready for normal operation state" [see clause 8.5 in ISO/IEC 30118-2]. In all other states the Server should respond with timeouts as identified in IETF RFC 7252.

12.2.11 Mapping the error response payload

The error response payload as defined in clause 7.10 shall be included as a diagnostic payload as described in IETF RFC 7252 clause 5.5.2. The diagnostic payload shall be encoded in ASCII.

12.2.12 Handling of non-confirmable requests

IETF RFC 7252 explicitly notes that non-confirmable requests are appropriate in cases where reliability of the delivery of the request is not an issue. However, all requests that are sent as a result of a CRUDN operation (see clause 12.2.3) defined in this document should be sent as confirmable requests, and non-confirmable requests should not be used.

If a Client makes use of a non-confirmable message in a request, , then the Client should realize the mechanism defined in IETF RFC 7252 clause 4.3 to reduce the possibility of message loss.

Further, a Client that makes use of non-confirmable requests shall not depend on a response being provided for that request.

12.3 Mapping of CRUDN to CoAP serialization over TCP

12.3.1 Overview

In environments where TCP is already available, CoAP can take advantage of it to provide reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For example, consider a cloud application acting as a Client and the Server is located at the user's home. A Server which already support CoAP as a messaging protocol could easily support CoAP serialization over TCP rather than utilizing another messaging protocol. A Device implementing CoAP Serialization over TCP shall conform to IETF RFC 8323.

12.3.2 URIs

When UDP is blocked, Clients are dependent on pre-configured details of the Device to determine if the Device supports CoAP serialization over TCP. When UDP is not-blocked, a Device which supports CoAP serialization over TCP shall populate the "eps" Parameter in the "/oic/res" response, as defined in 10.2, with the URI scheme(s) as defined in clause 8.1 or 8.2 of IETF RFC 8323. For the "coaps+tcp" URI scheme, as defined in clause 8.2 of IETF RFC 8323, IETF RFC 7301 shall be used. In addition, the URIs used for CoAP serialization over TCP shall conform to 12.2.2 by substituting the scheme names with the scheme names defined in clauses 8.1 and 8.2 of IETF RFC 8323 respectively.

12.3.3 CoAP method with request and response

The CoAP methods used for CoAP serialization over TCP shall conform to 12.2.3.

12.3.4 Content-Format negotiation

The Content Format negotiation used for CoAP serialization over TCP shall conform to 12.2.4.

12.3.5 OCF-Content-Format-Version information

The OCF Content Format Version information used for CoAP serialization over TCP shall conform to 12.2.5.

12.3.6 Content-Format policy

The Content Format policy used for CoAP serialization over TCP shall conform to 12.2.6.

12.3.7 CRUDN to CoAP response codes

The CRUDN to CoAP response codes for CoAP serialization over TCP shall conform to 12.2.7.

12.3.8 CoAP block transfer

The CoAP block transfer for CoAP serialization over TCP shall conform to clause 6 of IETF RFC 8323.

12.3.9 Keep alive (connection health)

The Device that initiated the CoAP over TCP connection shall send a Ping message as described in clause 5.4 in IETF RFC 8323. The Device to which the connection was made may send a Ping message. The recipient of any Ping message shall send a Pong message as described in clause 5.4 in IETF RFC 8323.

Both sides of an established CoAP over TCP connection may send subsequent Ping (and corresponding Pong) messages.

12.3.10 CoAP using a proxy

In cases that a request is made to a forwarding proxy, the option proxy-uri (clause 5.10.2 of IETF RFC 7252) shall be used. The format of the information in the proxy-uri option includes the OCF Device information. The proxy-uri shall have the format of an OCF URI as described in clause 6.2.2. The authority will have the same value as "oic.wk.d:uuid" of the targeted Device.

12.3.11 Mapping the error response payload

The mapping of the error response payload for CoAP serialization over TCP shall conform to clause 12.2.11.

12.3.12 Handling of non-confirmable requests

The requirements defined in clause 12.2.12 with regard to non-confirmable requests do not apply to CoAP serialization over TCP as TCP itself is inherently reliable.

12.4 Mapping of CRUDN to MQTT

12.4.1 Overview

MQTT contains the following entities:

– Client: Publisher

– Server

– Client: Subscriber

This is depicted in Figure 32.

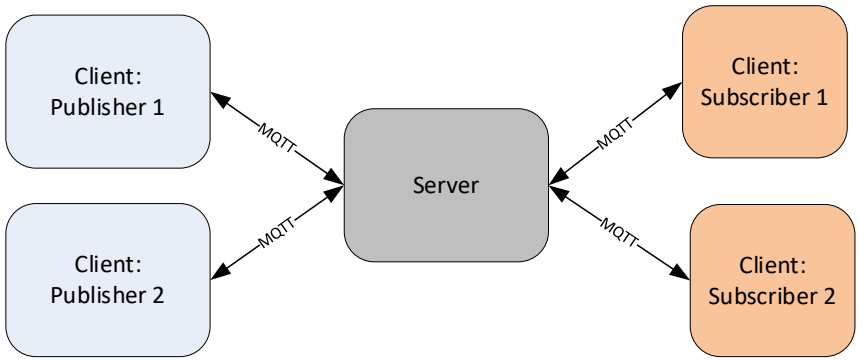


Figure 32 – Typical MQTT

The interaction model involves a Client publishing data on a topic to the Server. The Server knows which Clients subscribed to the topic and forwards the data to those Clients. This is depicted in Figure 33.

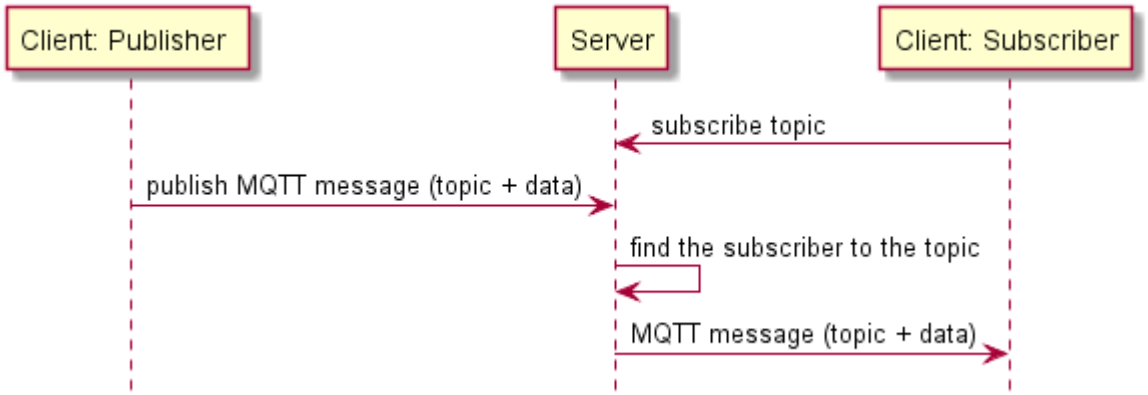


Figure 33 – Publish interaction model

An MQTT message contains payload data, a Quality of Service (QoS), a collection of Properties, and a topic name. A more complex interaction involves a request which expects a return payload. This is achieved by tagging the publish MQTT message with a return topic and correlation identifier. The publisher tracks the correlation identifier and thus can match up the response with the request. This is depicted in Figure 34.

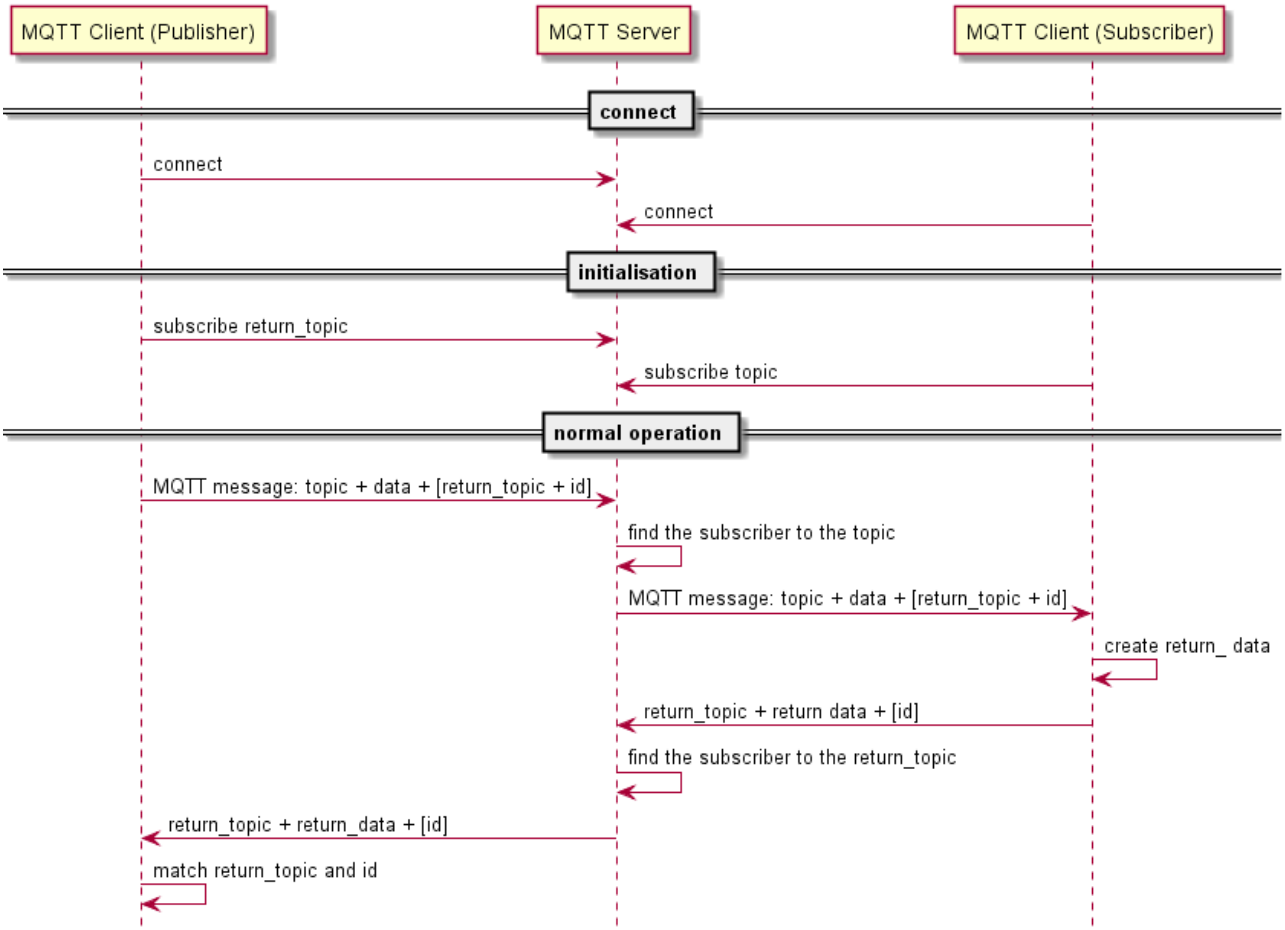


Figure 34 – MQTT Request and Response interaction model

12.4.2 Mapping OCF Devices and Resources to MQTT topics

The topic defined by MQTT is used to simulate the RESTful operations and Resource addressing of the OCF architecture. To do this, the Resource address and CRUDN operation need to be defined. The MQTT message data consists of the payload of the CRUDN operation. This allows the payload defined in the Resource Types to be transported in the MQTT message.

MQTT topics are organized as a folder structure with "/" as a topic level separator.

Table 51 – Command usage

OCF operations	operation in topic	MQTT message contents	return topic	correlation ID
CREATE	"C"	Request payload	Required	Required
RETRIEVE	"R"	-	Required	Required
UPDATE	"U"	Request payload	Required	Required
DELETE	"D"	-	Required	Required
NOTIFY	"N"	-	Required	Required

Topics are represented as follows:

Topic = OCF/<OCF Device UUID>/<path?query>/<CRUDN operation>

Since the topic uses the same "/" separator as the path, the path is escaped.

The path?query is escaped by replacing the "/" character with "%2F". The first "/" of the local path?query may be omitted. For example; the Resource path?query for "/oic/res" is conveyed as "oic%2Fres".

The operation is added to the topic when a MQTT actor wants to communicate with an OCF Device. Since all subscribers do not know the origin of the publisher, there is a need to distinguish topics as an operation. If an operation does not require a payload (like RETRIEVE) the request MQTT message does not contain any data. If a response is expected, then the return topic and correlation ID are required in the MQTT message (see Table 51 for more information).

Table 52 contains examples of path?query and operation to the MQTT topic.

Table 52 – Sending operations as topic

Addressing scope	Topic for publishing
Retrieve of "/oic/res" on a specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/oic%2Fres/R
Retrieve of "/oic/d" on a specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/oic%2Fd/R
Retrieve of "/oic/p" on a specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/oic%2Fp/R
Sending the RETRIEVE to Resource URL "blah" using the baseline OCF Interface	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/blah?if=oic.if.baseline/R
Sending the UPDATE to change a Property value to a specific Resource URL "blah" on a specific Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/blah/U

A level may have a wild card. This is the "+" symbol. Wildcards, identified by the "#" symbol, may be used in place of the level name to receive all topics at that level of the folder structure. Wild cards may be used to simplify subscriptions on a topic. See clause 4.7 of [MQTT] for more details.

Table 53 – topic wild cards

MQTT wild card	description
+	Single level wild card
#	Multi-level wild card

Table 54 – Subscription addressing scope and topic wild cards

Addressing scope	Topic subscription
All OCF Devices	OCF/#
Specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/#
"/oic/res" on a specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/oic%2Fres/#
"/oic/d" on a specific OCF Device	OCF/6ed4963b-5770-44ab-b61f-f6b7b0681933/oic%2Fd/#

The NOTIFY operation in OCF is based on Observe, see 11.3.2 [CORE]. The MQTT Client as OCF Client subscribes (registers) to receive notifications by sending a NOTIFY operation to the topic. The MQTT Client as OCF Server sends updates to the MQTT Client as OCF Client by means of the return topic and correlation id of the initial NOTIFY operation. The sequence of subscribing for notifications with MQTT messages is depicted in Figure 33.

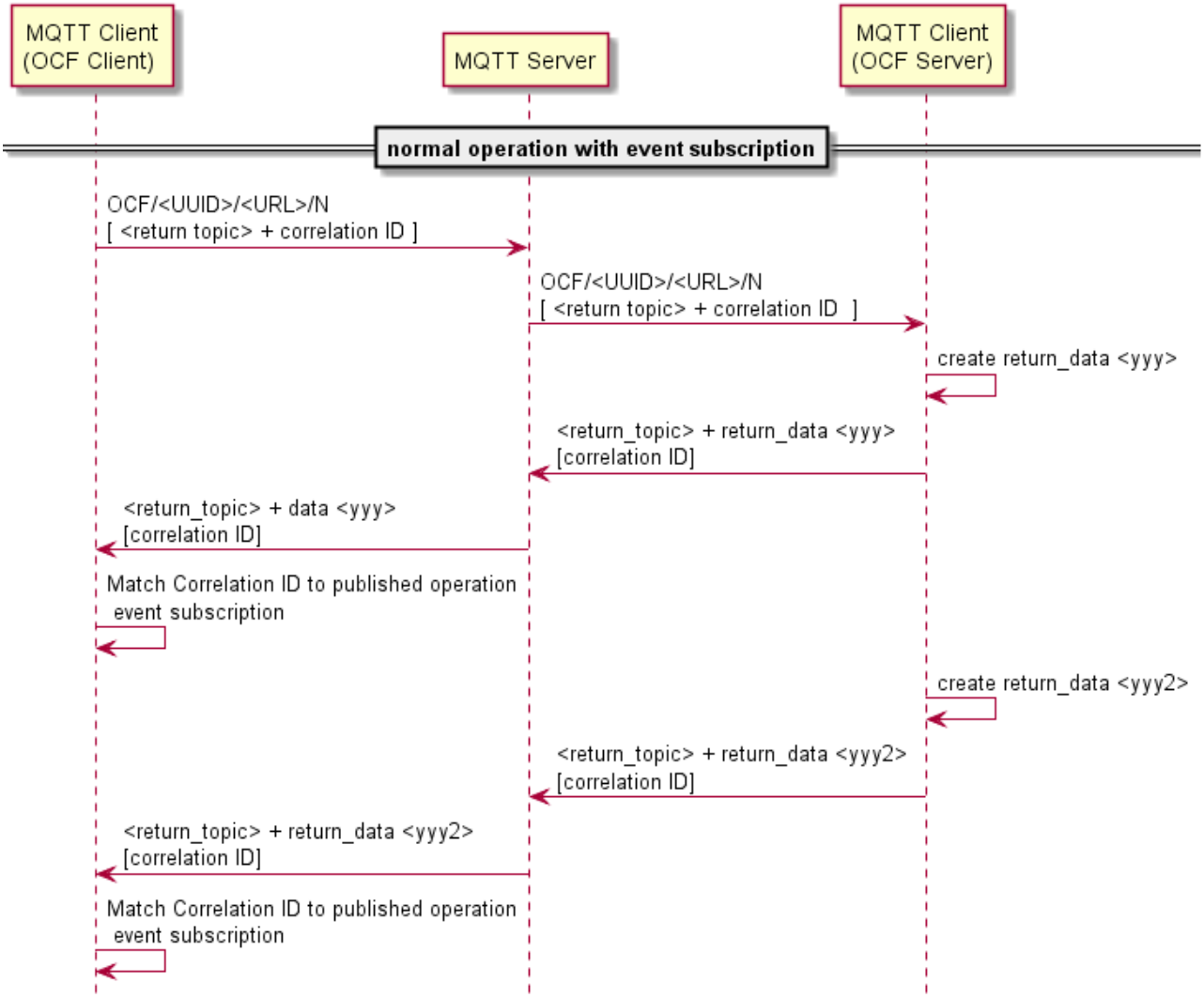


Figure 35 – Example interaction model with an event subscription

12.4.3 Mapping OCF Data to MQTT Data

The OCF payload data is encoded in CBOR. Using MQTT as transport will use the OCF payload data as MQTT messages (content in BOR). Transporting the data in CBOR simplifies the changing in transports, it does not alter the data between the conversion of CoAP to MQTT as transport. Furthermore, keeping the payload data unchanged enables end-to-end encryption.
Note that the MQTT Content Type is not used, the topic naming convention determines already the content type of the payload This is achieved by having the "OCF" prefix on the topics.

12.4.4 Mapping OCF Discovery to MQTT

MQTT does not support multicast, however a single MQTT message may be received by multiple subscribers. Hence a topic where all OCF devices on the MQTT has to listen can simulate multicast discovery.

Discovery of OCF Devices is achieved by sending a RETRIEVE MQTT message to the topic "OCF/*/<discovery resource url>/R". The asterisk symbol ("*") is replacing the actual UUID of the device in the topic. The asterisk is not an MQTT wild card and therefore this topic is a valid MQTT topic that the MQTT client of the OCF device should subscribe and respond to. For example, the MQTT client of the OCF Server should subscribe to the topics "OCF/*/#" and "OCF/<UUID>/#" to

be able to receive the operations targeted with the UUID and the operation targeted as discovery operation.

Table 55 – Examples of Discovery topics

Description	Topic in the request
RETRIEVE of "/oic/res"	OCF/*/oic%2Fres/R
RETRIEVE of "/oic/res" with baseline	OCF/*/oic%2Fres?if=oic.baseline/R
RETRIEVE of "/oic/d"	OCF/*/oic%2Fres/R
RETRIEVE of "/oic/p"	OCF/*/oic%2Fres/R

12.4.5 Error condition behaviour

MQTT may signal errors between the client and server connection, see 3.4.2.1 [MQTT v5 Ref] . However, the OCF request and response communication may have errors as well, Receiving CBOR encoded MQTT messages will be regarded as a 2.xx OK response. When an actual error occurs, the Error response payload should be used, See 7.10 [CORE SPEC REF]. The actual transport binding status (error) code may be included in the error response payload.

Example CoAP error response payload:

```
Status code: 4.04 (Not Found)
Response Body:
"The device is not registered"
```

Translates to MQTT error response payload:

```
Publish MQTT message:
"4.04 The device is not registered"
```

12.4.6 MQTT considerations

The MQTT configuration, e.g., where the MQTT server resides and how the security credentials are distributed, is out of scope of the MQTT transport. This information is highly dependent on the actual deployment, e.g., which MQTT server is being used and thus which security mechanisms are available to protect the MQTT client-server connection.

The usage of the request-response sequences determines that version v5.0 MQTT and higher can be used. The versions predating v5.0 does not have the functionality to implement the request-response MQTT messages.

The deployment may use secure TLS (TCP port 8883) as indicated in clause 4.2 of the MQTT specification. For additional security recommendations, see the security clause 4.12 and 5 of the MQTT specification.

12.5 Payload Encoding in CBOR

OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to JSON from CBOR in accordance with IETF RFC 7049 clause 4 unless otherwise specified in this clause.

Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer numbers shall be within the closed interval [-2^53, 2^53]. Properties defined as a JSON number should be encoded as integers whenever possible; if this is not possible Properties defined as a JSON number should use single-precision if the loss of precision does not affect the quality of service, otherwise the Property shall use double-precision.

On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values in any position. If a Property defined as a JSON integer is received encoded other than as an integer, the implementation may reject this encoding using a final response as appropriate for the underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a Property is defined as a JSON number an implementation shall accept integers, single- and double-precision floating point.

13 Security

The details for handling security and privacy are specified in ISO/IEC 30118-2.

Annex A
(normative)

Resource Type definitions

A.1 List of Resource Type definitions

All the clauses in Annex A describe the Resource Types with a RESTful API definition language. The Resource Type definitions presented in Annex A are formatted for readability, and so may appear to have extra line breaks. Table A.1 contains the list of defined Core Common Resources in this document.

Table A.1 – Alphabetized list of Core Resources

Friendly Name (informative)	Resource Type (rt)	Clause
Atomic Measurement	"oic.wk.atomicmeasurement"	A.2
Collections	"oic.wk.col"	A.3
Device	"oic.wk.d"	A.4
Discoverable Resource	"oic.wk.res"	A.7
Introspection	"oic.wk.introspection"	A.5
Platform	"oic.wk.p"	A.6
MQTT Configuration	"oic.r.mqtt.conf"	A.8

A.2 Atomic Measurement links list representation

A.2.1 Introduction

The oic.if.baseline OCF Interface exposes a representation of the links and the Common Properties of the Atomic Measurement Resource.

A.2.2 Example URI

/AtomicMeasurementResURI

A.2.3 Resource type

The Resource Type is defined as: "oic.wk.atomicmeasurement".

A.2.4 OpenAPI 2.0 definition

```
{
        "swagger": "2.0",
        "info": {
                "title": "Atomic Measurement links list representation",
                "version": "2019-03-04",
                "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2018-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
        },
        "schemes": ["http"],
        "consumes": ["application/json"],
        "produces": ["application/json"],
        "paths": {
                "/AtomicMeasurementResURI?if=oic.if.ll": {
                        "get": {
                                "description": "The oic.if.ll OCF Interface exposes a representation
of the Links",
```

```
                        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
                        "responses": {
                                "200": {
                                        "description": "",
                                        "x-example": [{
                                                "href": "/temperature",
                                                "rt": ["oic.r.temperature"],
                                                "if": ["oic.if.s", "oic.if.baseline"]
                                        },
                                        {
                                                "href": "/bodylocation",
                                                "rt": ["oic.r.body.location.temperature"],
                                                "if": ["oic.if.s", "oic.if.baseline"]
                                        },
                                        {
                                                "href": "/timestamp",
                                                "rt": ["oic.r.time.stamp"],
                                                "if": ["oic.if.s", "oic.if.baseline"]
                                        }],
                                        "schema": {
                                                "$ref": "#/definitions/links"
                                        }
                                }
                        }
                }
        },
        "/AtomicMeasurementResURI?if=oic.if.b": {
                "get": {
                        "description": "The oic.if.b OCF Interface returns data items
retrieved from Resources pointed to by the Links.\n",
                        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
                        "responses": {
                                "200": {
                                        "description": "Normal response, no errors, all
Properties are returned correctly\n",
                                        "x-example": [{
                                                "href": "/temperature",
                                                "rep": {
                                                        "temperature": 38,
                                                        "units": "C",
                                                        "range": [25,  45]
                                                }
                                        },
                                        {
                                                "href": "/bodylocation",
                                                "rep": {
                                                        "bloc": "ear"
                                                }
                                        },
                                        {
                                                "href": "/timestamp",
                                                "rep": {
                                                        "timestamp": "2007-04-05T14:30+09:00"
                                                }
                                        }],
                                        "schema": {
                                                "$ref": "#/definitions/batch-retrieve"
                                        }
                                }
                        }
                }
        },
        "/AtomicMeasurementResURI?if=oic.if.baseline": {
                "get": {
```



```
                        "description": "The oic.if.baseline OCF Interface exposes a
representation of the links and\nthe Common Properties of the Atomic Measurement Resource.\n",
                        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
                        "responses": {
                                "200": {
                                        "description": "",
                                        "x-example": {
                                                "rt": ["oic.wk.atomicmeasurement"],
                                                "if": ["oic.if.b", "oic.if.ll",
       "oic.if.baseline"],
                                                "rts": ["oic.r.temperature",
"oic.r.body.location.temperature",    "oic.r.time.stamp"],
                                                "rts-m": ["oic.r.temperature",
"oic.r.body.location.temperature",    "oic.r.time.stamp"],
                                                "links": [{
                                                        "href": "/temperature",
                                                        "rt": ["oic.r.temperature"],
                                                        "if": ["oic.if.s", "oic.if.baseline"]
                                                },
                                                {
                                                        "href": "/bodylocation",
                                                        "rt":
["oic.r.body.location.temperature"],
                                                        "if": ["oic.if.s", "oic.if.baseline"]
                                                },
                                                {
                                                        "href": "/timestamp",
                                                        "rt": ["oic.r.time.stamp"],
                                                        "if": ["oic.if.s", "oic.if.baseline"]
                                                }]
                                        },
                                        "schema": {
                                                "$ref": "#/definitions/baseline"
                                        }
                                }
                        }
                }
        },
        "parameters": {
                "interface-all": {
                        "in": "query",
                        "name": "if",
                        "type": "string",
                        "enum": ["oic.if.b", "oic.if.ll",     "oic.if.baseline"]
                }
        },
        "definitions": {
                "links": {
                        "type": "array",
                        "items": {
                                "$ref": "#/definitions/oic.oic-link"
                        }
                },
                "batch-retrieve": {
                        "title": "Collection Batch Retrieve Format (auto merged)",
                        "minItems": 1,
                        "items": {
                                "additionalProperties": true,
                                "properties": {
                                        "href": {
                                                "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
                                        },
                                        "rep": {
                                                "oneOf": [{
                                                        "description": "The response payload from a
```



```
single Resource",
                                                "type": "object"
                                        },
                                        {
                                                "description": " The response payload from a
Collection (batch) Resource",
                                                "items": {
                                                        "properties": {
                                                                "anchor": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/anchor"
                                        },
                                        "di": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
                                        },
                                        "eps": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
                                        },
                                        "href": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
                                        },
                                                                "if": {
                                                                        "description": "The OCF
Interface set supported by this Resource",
                                                                        "items": {
                                                                                "enum": [

        "oic.if.baseline",
                                                                                        "oic.if.ll",
                                                                                        "oic.if.b",
                                                                                        "oic.if.rw",
                                                                                        "oic.if.r",
                                                                                        "oic.if.a",
                                                                                        "oic.if.s"],
                                                                                        "type":
"string"
                                                                        },
                                                                        "minItems": 1,
                                                                        "uniqueItems": true,
                                                                        "type": "array"
                                                                },
                                                                "ins": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
                                        },
                                        "p": {
                                          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
                                        },
                                        "rel": {
                                          "description": "The relation of the target URI
referenced by the Link to the context URI",
                                          "oneOf": [
                                            {
                                              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
                                            },
                                            {
                                              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_string"
                                            }
```



```
                                                ]
                                              },
                                                                    "rt": {
                                                                           "description":
"Resource Type of the Resource",
                                                                           "items": {
                                                                                  "maxLength":
64,
                                                                                  "type":
"string"
                                                                           },
                                                                           "minItems": 1,
                                                                           "uniqueItems": true,
                                                                           "type": "array"
                                                                    },
                                                                    "title": {
                                                 "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
                                              },
                                              "type": {
                                                "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
                                              }
                                                                           },
                                                                           "required": [
                                                                             "href",
                                                                             "rt",
                                                                             "if"
                                                                           ],
                                                                           "type": "object"
                                                                    },
                                                                    "type": "array"
                                              }]
                                                      }
                                             },
                                             "required": [
                                               "href",
                                               "rep"
                                             ],
                                             "type": "object"
                                      },
                                      "type": "array"
                              },
                              "baseline": {
                                      "properties": {
                                             "links": {
                                                    "description": "A set of simple or individual Links.",
                                                    "items": {
                                                           "$ref": "#/definitions/oic.oic-link"
                                                    },
                                                    "type": "array"
                                             },
                                             "n": { "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"},
                                      "id": { "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"},
                                             "rt": {
                                "description": "Resource Type of this Resource",
                                "items": {
                                  "enum": ["oic.wk.atomicmeasurement"],
                                  "type": "string",
                                  "maxLength": 64
                                },
                                "minItems": 1,
                                "readOnly": true,
                                "uniqueItems": true,
                                "type": "array"
                              },
```



```
                                "rts": {
                                        "description": "An array of Resource Types that are supported
within an array of Links exposed by the Resource",
                                        "items": {
                                                "maxLength": 64,
                                                "type": "string"
                                        },
                                        "minItems": 1,
                                        "readOnly": true,
                                        "uniqueItems": true,
                                        "type": "array"
                                },
                                "rts-m": {
                                        "description": "An array of Resource Types that are mandatory
to be exposed within an array of Links exposed by the Resource",
                                        "items": {
                                                "maxLength": 64,
                                                "type": "string"
                                        },
                                        "minItems": 1,
                                        "readOnly": true,
                                        "uniqueItems": true,
                                        "type": "array"
                                },
                                "if": {
                                        "description": "The OCF Interface set supported by this
Resource",
                                        "items": {
                                                "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"],
                                                "type": "string"
                                        },
                                        "minItems": 3,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
                                }
                        },
                        "type": "object",
                        "required": [
        "rt",
        "if",
        "links"
      ]
                },
                "oic.oic-link": {
                        "properties": {
                                "anchor": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/anchor"
        },
        "di": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
        },
        "eps": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
        },
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
                                "if": {
                                        "description": "The OCF Interface set supported by this
Resource",
                                        "items": {
                                                "enum": [
                                                "oic.if.baseline",
```



```
                                        "oic.if.ll",
                                        "oic.if.b",
                                        "oic.if.rw",
                                        "oic.if.r",
                                        "oic.if.a",
                                        "oic.if.s"],
                                        "type": "string"
                                },
                                "minItems": 1,
                                "uniqueItems": true,
                                "type": "array"
                        },
                        "ins": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
        },
        "p": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
        },
        "rel": {
          "description": "The relation of the target URI referenced by the Link to the context URI",
          "oneOf": [
            {
              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
            },
            {
              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_string"
            }
          ]
        },
                        "rt": {
                                "description": "Resource Type of the Resource",
                                "items": {
                                        "maxLength": 64,
                                        "type": "string"
                                },
                                "minItems": 1,
                                "uniqueItems": true,
                                "type": "array"
                        },
                        "title": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
        },
        "type": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
        }
                },
                "required": [
                        "href",
                  "rt",
                  "if"
                ],
                "type": "object"
        }
    }
}
```

A.2.5 Property definition

Table A.2 defines the Properties that are part of the "oic.wk.atomicmeasurement" Resource Type.

Table A.2 – The Property definitions of the Resource with type "rt" = "oic.wk.atomicmeasurement".

Property name	Value type	Mandatory	Access mode	Description
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
links	array: see schema	Yes	Read Write	A set of simple or individual Links.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource
rts	array: see schema	No	Read Only	An array of Resource Types that are supported within an array of Links exposed by the Resource
rts-m	array: see schema	No	Read Only	An array of Resource Types that are mandatory to be exposed within an array of Links exposed by the Resource
if	array: see schema	Yes	Read Only	The OCF Interface set supported by this Resource
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interface set supported by this Resource
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI
rt	array: see schema	Yes	Read Write	Resource Type of the Resource
title	multiple types: see schema	No	Read Write	

type	multiple types: see schema	No	Read Write	

A.2.6 CRUDN behaviour

Table A.3 defines the CRUDN operations that are supported on the "oic.wk.atomicmeasurement" Resource Type.

Table A.3 – The CRUDN operations of the Resource with type "rt" = "oic.wk.atomicmeasurement".

Create	Read	Update	Delete	Notify
	get			observe

A.3 Collection

A.3.1 Introduction

Collection Resource Type contains Properties and Links.
The oic.if.baseline OCF Interface exposes a representation of the Links and the Properties of the Collection Resource itself

A.3.2 Example URI

/CollectionResURI

A.3.3 Resource type

The Resource Type is defined as: "oic.wk.col".

A.3.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Collection",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/CollectionResURI?if=oic.if.ll" : {
      "get": {
        "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.ll OCF
Interface exposes a representation of the Links\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
        "responses": {
          "200": {
            "description" : "",
```

```
            "x-example": [
              {
                "href": "/switch",
                "rt":   ["oic.r.switch.binary"],
                "if":   ["oic.if.a", "oic.if.baseline"],
                "eps": [
                  {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
                  {"ep": "coaps://[fe80::b1d6]:1122"},
                  {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
                ]
              },
              {
                "href": "/airFlow",
                "rt":   ["oic.r.airflow"],
                "if":   ["oic.if.a", "oic.if.baseline"],
                "eps": [
                  {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
                  {"ep": "coaps://[fe80::b1d6]:1122"},
                  {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
                ]
              }
            ],
            "schema": {
              "$ref": "#/definitions/slinks"
            }
          }
        }
      }
    },
    "/CollectionResURI?if=oic.if.baseline" : {
      "get": {
        "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.baseline
OCF Interface exposes a representation of\nthe Links and the Properties of the Collection Resource
itself\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
        "responses": {
          "200": {
            "description" : "",
            "x-example": {
              "rt": ["oic.wk.col"],
              "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
              "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
              "rts-m": [ "oic.r.switch.binary" ],
              "links": [
                {
                  "href": "/switch",
                  "rt":   ["oic.r.switch.binary"],
                  "if":   ["oic.if.a", "oic.if.baseline"],
                  "eps": [
                      {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
                      {"ep": "coaps://[fe80::b1d6]:1122"},
                      {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
                  ]
                },
                {
                  "href": "/airFlow",
                  "rt":   ["oic.r.airflow"],
                  "if":   ["oic.if.a", "oic.if.baseline"],
                  "eps": [
                      {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
                      {"ep": "coaps://[fe80::b1d6]:1122"},
                      {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
                  ]
                }
              ]
            },
            "schema": {
              "$ref": "#/definitions/sbaseline"
```



```
          }
        }
      }
    },
    "post": {
      "description": "Update on Baseline OCF Interface\n",
      "parameters": [
        {
          "$ref": "#/parameters/interface-update"
        },
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/sbaseline-update"
          }
        }
      ],
      "responses": {
        "200": {
          "description" : "",
          "schema": {
            "$ref": "#/definitions/sbaseline"
          }
        }
      }
    }
  },
  "/CollectionResURI?if=oic.if.b" : {
    "get": {
      "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.b OCF
Interfacce exposes a composite representation of the\nResources pointed to by the Links\n",
      "parameters": [
        {
          "$ref": "#/parameters/interface-all"
        }
      ],
      "responses": {
        "200": {
          "description" : "All targets returned OK status",
          "x-example": [
            {
              "href": "/switch",
              "rep": {
                "value": true
              }
            },
            {
              "href": "/airFlow",
              "rep": {
                "direction": "floor",
                "speed":     3
              }
            }
          ],
          "schema": {
            "$ref": "#/definitions/sbatch-retrieve"
          }
        },
        "404": {
          "description" : "One or more targets did not return an OK status, return a
representation containing returned Properties from the targets that returned OK",
          "x-example": [
            {
              "href": "/switch",
              "rep": {
                "value": true
              }
            }
          ],
          "schema": {
```



```
            "$ref": "#/definitions/sbatch-retrieve"
          }
        }
      }
    },
    "post": {
      "description": "Update on Batch OCF Interface\n",
      "parameters": [
        {
          "$ref": "#/parameters/interface-update"
        },
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/sbatch-update"
          },
          "x-example": [
            {
              "href": "/switch",
              "rep": {
                "value": true
              }
            },
            {
              "href": "/airFlow",
              "rep": {
                "direction": "floor",
                "speed": 3
              }
            }
          ]
        }
      ],
      "responses": {
        "200": {
          "description" : "All targets returned OK status, return a representation of the current
state of all targets",
          "x-example": [
            {
              "href": "/switch",
              "rep": {
                "value": true
              }
            },
            {
              "href": "/airFlow",
              "rep": {
                "direction": "demist",
                "speed": 5
              }
            }
          ],
          "schema": {
            "$ref": "#/definitions/sbatch-retrieve"
          }
        },
        "403": {
          "description" : "One or more targets did not return OK status; return a retrieve
representation of the current state of all targets in the batch",
          "x-example": [
            {
              "href": "/switch",
              "rep": {
                "value": true
              }
            },
            {
              "href": "/airFlow",
              "rep": {
                "direction": "floor",
```



```
                "speed": 3
              }
            }
          ],
          "schema": {
            "$ref": "#/definitions/sbatch-retrieve"
          }
        }
      }
    }
  },
  "parameters": {
    "interface-all" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
    },
    "interface-update" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.b", "oic.if.baseline"]
    }
  },
  "definitions": {
    "sbaseline" : {
      "properties": {
        "links" : {
          "description": "A set of simple or individual Links.",
          "items": {
            "$ref": "#/definitions/oic.oic-link"
          },
          "type": "array"
        },
        "n": {
          "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
              "rt": {
          "$ref": "#/definitions/oic.core.rt-col"
        },
              "rts": {
          "$ref": "#/definitions/oic.core.rt"
        },
              "rts-m": {
          "$ref": "#/definitions/oic.core.rt"
        },
              "if": {
                "description": "The OCF Interfaces supported by this Resource",
                "items": {
                        "enum": [
              "oic.if.ll",
              "oic.if.baseline",
                              "oic.if.b"
            ],
                                                "type": "string",
            "maxLength": 64
                                      },
                                      "minItems": 2,
          "uniqueItems": true,
          "readOnly": true,
                                      "type": "array"
                              }
      },
```



```
      "additionalProperties": true,
      "type" : "object",
      "required": [
        "rt",
        "if",
        "links"
      ]
    },
    "sbaseline-update": {
      "additionalProperties": true
    },
          "oic.core.rt-col": {
      "description": "Resource Type of the Resource",
                        "items": {
                                "enum": ["oic.wk.col"],
                                "type": "string",
        "maxLength": 64
                        },
                        "minItems": 1,
      "uniqueItems": true,
                        "readOnly": true,
                        "type": "array"
                },
    "oic.core.rt": {
      "description": "Resource Type or set of Resource Types",
                        "items": {
                                "type": "string",
        "maxLength": 64
                        },
                        "minItems": 1,
      "uniqueItems": true,
                        "readOnly": true,
                        "type": "array"
                },
    "sbatch-retrieve" : {
      "minItems" : 1,
      "items" : {
        "additionalProperties": true,
        "properties": {
          "href": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
          },
          "rep": {
            "oneOf": [
              {
                "description": "The response payload from a single Resource",
                "type": "object"
              },
              {
                "description": " The response payload from a Collection (batch) Resource",
                "items": {
                  "$ref": "#/definitions/oic.oic-link"
                },
                "type": "array"
              }
            ]
          }
        },
        "required": [
          "href",
          "rep"
        ],
        "type": "object"
      },
      "type" : "array"
    },
    "sbatch-update" : {
      "title" : "Collection Batch Update Format",
      "minItems" : 1,
      "items" :  {
```



```
        "$ref": "#/definitions/sbatch-update.item"
      },
      "type" :  "array"
    },
    "sbatch-update.item" : {
      "additionalProperties": true,
      "description": "Array of Resource representations to apply to the batch Collection, using href
to indicate which Resource(s) in the batch to update. If the href Property is empty, effectively
making the URI reference to the Collection itself, the representation is to be applied to all
Resources in the batch",
      "properties": {
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "rep": {
          "oneOf": [
            {
              "description": "The payload for a single Resource",
              "type": "object"
            },
            {
              "description": " The payload for a Collection (batch) Resource",
              "items": {
                "$ref": "#/definitions/oic.oic-link"
              },
              "type": "array"
            }
          ]
        }
      },
      "required": [
        "href",
        "rep"
      ],
      "type": "object"
    },
    "slinks" : {
      "type" : "array",
      "items" : {
        "$ref": "#/definitions/oic.oic-link"
      }
    },
    "oic.oic-link": {
      "properties": {
        "if": {
          "description": "The OCF Interfaces supported by the Linked target",
          "items": {
            "enum": [
              "oic.if.baseline",
              "oic.if.ll",
              "oic.if.b",
              "oic.if.rw",
              "oic.if.r",
              "oic.if.a",
              "oic.if.s"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "rt": {
          "$ref": "#/definitions/oic.core.rt"
        },
        "anchor": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
```



```
schema.json#/definitions/anchor"
        },
        "di": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
        },
        "eps": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
        },
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "ins": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
        },
        "p": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
        },
        "rel": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
        },
        "title": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
        },
        "type": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
        },
        "tag-pos-desc": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-pos-desc"
        },
        "tag-pos-rel": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-pos-rel"
        },
        "tag-func-desc": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-func-desc"
        }
      },
      "required": [
        "href",
        "rt",
        "if"
      ],
      "type": "object"
    }
  }
}
```


A.3.5 Property definition

Table A.4 defines the Properties that are part of the "oic.wk.col" Resource Type.

Table A.4 – The Property definitions of the Resource with type "rt" = "oic.wk.col".

Property name	Value type	Mandatory	Access mode	Description
links	array: see schema	Yes	Read Write	A set of simple or individual Links.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	multiple types: see schema	Yes	Read Write	
rts	multiple types: see schema	No	Read Write	
rts-m	multiple types: see schema	No	Read Write	
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by the Linked target
rt	multiple types: see schema	Yes	Read Write	
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
tag-pos-desc	multiple types: see schema	No	Read Write	
tag-pos-rel	multiple types: see schema	No	Read Write	

tag-func-desc	multiple types: see schema	No	Read Write	

A.3.6 CRUDN behaviour

Table A.5 defines the CRUDN operations that are supported on the "oic.wk.col" Resource Type.

Table A.5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col".

Create	Read	Update	Delete	Notify
	get	post		observe

A.4 Device

A.4.1 Introduction

Known Resource that is hosted by every Server.
Allows for logical Device specific information to be discovered.

A.4.2 Well-known URI

/oic/d

A.4.3 Resource type

The Resource Type is defined as: "oic.wk.d".

A.4.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Device",
    "version": "2019-03-13",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/oic/d" : {
      "get": {
        "description": "Known Resource that is hosted by every Server.\nAllows for logical Device
specific information to be discovered.\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface"
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example":
              {
                "n":    "Device 1",
                "rt":   ["oic.wk.d"],
```

```
              "di":   "54919CA5-4101-4AE4-595B-353C51AA983C",
              "icv":  "ocf.2.0.2",
              "dmv":  "ocf.res.1.0.0, ocf.sh.1.0.0",
              "piid": "6F0AAC04-2BB0-468D-B57C-16570A26AE48"
            },
            "schema": {
               "$ref": "#/definitions/Device"
             }
          }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in": "query",
      "name": "if",
      "type": "string",
      "enum": ["oic.if.r", "oic.if.baseline"]
    }
  },
  "definitions": {
    "Device": {
      "properties": {
        "rt":  {
          "description": "Resource Type of the Resource",
          "items": {
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        },
        "ld":  {
          "description": "Localized Descriptions.",
          "items": {
            "properties": {
              "language": {
                "allOf": [
                  {
                    "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/language-tag"
                  },
                  {
                    "description": "An RFC 5646 language tag.",
                    "readOnly": true
                  }
                ]
              },
              "value": {
                "description": "Device description in the indicated language.",
                "maxLength": 64,
                "readOnly": true,
                "type": "string"
              }
            },
            "type": "object"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "piid": {
          "allOf": [
            {
              "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/uuid"
            },
            {
              "description": "Protocol independent unique identifier for the Device that is
```



```
immutable.",
              "readOnly": true
            }
          ]
        },
        "di": {
          "allOf": [
            {
              "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/uuid"
            },
            {
              "description": "Unique identifier for the Device",
              "readOnly": true
            }
          ]
        },
        "dmno":  {
          "description": "Model number as designated by manufacturer.",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "sv":  {
          "description": "Software version.",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "dmn":  {
          "description": "Manufacturer Name.",
          "items": {
            "properties": {
              "language": {
                "allOf": [
                  {
                    "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/language-tag"
                  },
                  {
                    "description": "An RFC 5646 language tag.",
                    "readOnly": true
                  }
                ]
              },
              "value": {
                "description": "Manufacturer name in the indicated language.",
                "maxLength": 64,
                "readOnly": true,
                "type": "string"
              }
            },
            "type": "object"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "icv": {
          "description": "The version of the Device",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "dmv":  {
          "description": "Specification versions of the Resource and Device Specifications to which
this device data model is implemented",
          "maxLength": 256,
          "readOnly": true,
          "type": "string"
        },
        "n":  {
```



```
          "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref" :
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "if":  {
          "description": "The OCF Interfacces supported by this Resource",
          "items": {
            "enum": [
              "oic.if.r",
              "oic.if.baseline"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 2,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "econame" : {
          "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD.",
          "type": "string",
          "enum": ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"],
          "readOnly": true
        },
        "ecoversion" : {
          "description": "Version of ecosystem that a Bridged Device belongs to. Typical version
string format is like n.n (e.g. 5.0).",
          "type": "string",
          "maxLength": 64,
          "readOnly": true
        }
      },
      "type": "object",
      "required": ["n", "di", "icv", "dmv", "piid"]
    }
  }
}
```

A.4.5 Property definition

Table A.6 defines the Properties that are part of the "oic.wk.d" Resource Type.

Table A.6 – The Property definitions of the Resource with type "rt" = "oic.wk.d".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
ld	array: see schema	No	Read Only	Localized Descriptions.
piid	multiple types: see schema	Yes	Read Write	
di	multiple types: see schema	Yes	Read Write	
dmno	string	No	Read Only	Model number as designated by manufacturer.
sv	string	No	Read Only	Software version.
dmn	array: see schema	No	Read Only	Manufacturer Name.

icv	string	Yes	Read Only	The version of the Device
dmv	string	Yes	Read Only	Specification versions of the Resource and Device Specifications to which this device data model is implemented
n	multiple types: see schema	Yes	Read Write	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfacces supported by this Resource
econame	string	No	Read Only	Ecosystem Name of the Bridged Device which is exposed by this VOD.
ecoversion	string	No	Read Only	Version of ecosystem that a Bridged Device belongs to. Typical version string format is like n.n (e.g. 5.0).

A.4.6 CRUDN behaviour

Table A.7 defines the CRUDN operations that are supported on the "oic.wk.d" Resource Type.

Table A.7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d".

Create	Read	Update	Delete	Notify
	get			observe

A.5 Introspection Resource

A.5.1 Introduction

This Resource provides the means to get the Introspection Device Data (IDD) specifying all the OCF Endpoints of the Device.
The url hosted by this Resource is either a local or an external url.

A.5.2 Well-known URI

/IntrospectionResURI

A.5.3 Resource type

The Resource Type is defined as: "oic.wk.introspection".

A.5.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Introspection Resource",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
```



```
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/IntrospectionResURI": {
      "get": {
        "description": "This Resource provides the means to get the Introspection Device Data (IDD)
specifying all the OCF Endpoints of the Device.\nThe url hosted by this Resource is either a local
or an external url.\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface"
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": {
              "rt": ["oic.wk.introspection"],
              "urlInfo": [
                {
                  "content-type": "application/cbor",
                  "protocol": "coap",
                  "url": "coap://[fe80::1]:1234/IntrospectionExampleURI"
                }
              ]
            },
            "schema": {
              "$ref": "#/definitions/oic.wk.introspectionInfo"
            }
          }
        }
      }
    }
  },
  "parameters": {
    "interface": {
      "in": "query",
      "name": "if",
      "type": "string",
      "enum": ["oic.if.r", "oic.if.baseline"]
    }
  },
  "definitions": {
    "oic.wk.introspectionInfo": {
      "properties": {
        "rt":  {
          "description": "Resource Type of the Resource",
          "items": {
              "enum": ["oic.wk.introspection"],
              "type": "string",
              "maxLength": 64
          },
          "minItems": 1,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        },
        "n":  {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
```

```
schema.json#/definitions/n"
        },
        "urlInfo":  {
          "description": "Information on the location of the Introspection Device Data (IDD).",
          "items": {
            "properties": {
              "content-type": {
                "default": "application/cbor",
                "description": "content-type of the Introspection Device Data",
                "enum": [
                  "application/json",
                  "application/cbor"
                ],
                "type": "string"
              },
              "protocol": {
                "description": "Identifier for the protocol to be used to obtain the Introspection
Device Data",
                "enum": [
                  "coap",
                  "coaps",
                  "http",
                  "https",
                  "coap+tcp",
                  "coaps+tcp"
                ],
                "type": "string"
              },
              "url": {
                "description": "The URL of the Introspection Device Data.",
                "format": "uri",
                "type": "string"
              },
              "version": {
                "default": 1,
                "description": "The version of the Introspection Device Data that can be
downloaded",
                "enum": [
                  1
                ],
                "type": "integer"
              }
            },
            "required": [
              "url",
              "protocol"
            ],
            "type": "object"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "id":  {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "if": {
          "description": "The OCF Interfaces supported by this Resource",
          "items": {
            "enum": [
              "oic.if.r",
              "oic.if.baseline"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 2,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
```



```
        }
      },
      "type" : "object",
      "required": ["urlInfo"]
    }
  }
}
```

A.5.5 Property definition

Table A.8 defines the Properties that are part of the "oic.wk.introspection" Resource Type.

Table A.8 – The Property definitions of the Resource with type "rt" = "oic.wk.introspection".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
n	multiple types: see schema	No	Read Write	
urlInfo	array: see schema	Yes	Read Only	Information on the location of the Introspection Device Data (IDD).
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource

A.5.6 CRUDN behaviour

Table A.9 defines the CRUDN operations that are supported on the "oic.wk.introspection" Resource Type.

Table A.9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".

Create	Read	Update	Delete	Notify
	get			observe

A.6 Platform

A.6.1 Introduction

Known Resource that is defines the Platform on which an Server is hosted.
Allows for Platform specific information to be discovered.

A.6.2 Well-known URI

/oic/p

A.6.3 Resource type

The Resource Type is defined as: "oic.wk.p".

A.6.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Platform",
    "version": "2021-02-02",
```



```
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2016-2019, 2021 Open Connectivity Foundation, Inc. All rights
reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/oic/p" : {
      "get": {
        "description": "Known Resource that is defines the Platform on which an Server is
hosted.\nAllows for Platform specific information to be discovered.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
          "200": {
            "description" : "",
            "x-example": {
              "pi":   "54919CA5-4101-4AE4-595B-353C51AA983C",
              "rt":   ["oic.wk.p"],
              "mnmn": "Acme, Inc"
            },
            "schema": { "$ref": "#/definitions/Platform" }
          }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.r", "oic.if.baseline"]
    }
  },
  "definitions": {
    "Platform" : {
      "properties": {
        "rt" : {
          "description": "Resource Type of the Resource",
          "items": {
            "enum":   ["oic.wk.p"],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "pi" : {
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$",
          "type": "string",
          "description": "Platform Identifier",
          "readOnly": true
        },
        "mnfv" : {
          "description": "Manufacturer's firmware version",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "vid" : {
```



```
          "description": "Manufacturer's defined information for the Platform. The content is
freeform, with population rules up to the manufacturer",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "mnmn" : {
          "description": "Manufacturer name",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "mnmo" : {
          "description": "Model number as designated by the manufacturer",
          "maxLength": 128,
          "readOnly": true,
          "type": "string"
        },
        "mnhw" : {
          "description": "Platform Hardware Version",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "mnos" : {
          "description": "Platform Resident OS Version",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "mndt" : {
          "pattern": "^([0-9]{4})-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$",
          "type": "string",
          "description": "Manufacturing Date.",
          "readOnly": true
        },
        "id" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "mnsl" : {
          "description": "Manufacturer's Support Information URL",
          "format": "uri",
          "maxLength": 256,
          "readOnly": true,
          "type": "string"
        },
        "mnpv" : {
          "description": "Platform Version",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "st" : {
          "description": "The date-time format pattern according to IETF RFC 3339.",
          "format": "date-time",
          "readOnly": true,
          "type": "string"
        },
        "n" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "mnml" : {
          "description": "Manufacturer's URL",
          "format": "uri",
          "maxLength": 256,
          "readOnly": true,
          "type": "string"
        },
```



```
        "mnsel" : {
          "description": "Serial number as designated by the manufacturer",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "if" : {
          "description": "The OCF Interfaces supported by this Resource",
          "items": {
            "enum": [
              "oic.if.r",
              "oic.if.baseline"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 2,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        },
        "mnnct" : {
           "description": "An array of integers and each integer indicates the network connectivity
type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-
mib/ianaiftype-mib, e.g., [71, 259] which represents Wi-Fi and Zigbee.",
           "items": {
             "type": "integer",
             "minimum": 1,
             "description": "The network connectivity type based on IANAIfType value as defined by:
https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib."
           },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["pi", "mnmn"]
    }
  }
}
```

A.6.5 Property definition

Table defines the Properties that are part of the "oic.wk.p" Resource Type.

Table A.10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
pi	string	Yes	Read Only	Platform Identifier
mnfv	string	No	Read Only	Manufacturer's firmware version
vid	string	No	Read Only	Manufacturer's defined information for the Platform. The content is freeform, with population rules up to the manufacturer
mnmn	string	Yes	Read Only	Manufacturer name
mnmo	string	No	Read Only	Model number as designated by the manufacturer
mnhw	string	No	Read Only	Platform Hardware Version
mnos	string	No	Read Only	Platform Resident OS Version
mndt	string	No	Read Only	Manufacturing Date.

id	multiple types: see schema	No	Read Write	
mnsl	string	No	Read Only	Manufacturer's Support Information URL
mnpv	string	No	Read Only	Platform Version
st	string	No	Read Only	The date-time format pattern according to IETF RFC 3339.
n	multiple types: see schema	No	Read Write	
mnml	string	No	Read Only	Manufacturer's URL
mnsel	string	No	Read Only	Serial number as designated by the manufacturer
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource
mnnct	array: see schema	No	Read Only	An array of integers and each integer indicates the network connectivity type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib, e.g., [71, 259] which represents Wi-Fi and ZigBee.

A.6.6 CRUDN behaviour

Table A.11 defines the CRUDN operations that are supported on the "oic.wk.p" Resource Type.

Table A.11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p".

Create	Read	Update	Delete	Notify
	get			observe

A.7 Discoverable Resources

A.7.1 Introduction

Baseline representation of /oic/res; list of discoverable Resources

A.7.2 Well-known URI

/oic/res

A.7.3 Resource type

The Resource Type is defined as: "oic.wk.res".

A.7.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Discoverable Resources",
    "version": "2019-04-22",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
```



```
"consumes": [
  "application/json"
],
"produces": [
  "application/json"
],
"paths": {
  "/oic/res?if=oic.if.ll": {
    "get": {
      "description": "Links list representation of /oic/res; list of discoverable Resources\n",
      "parameters": [
        {
          "$ref": "#/parameters/interface-all"
        }
      ],
      "responses": {
        "200": {
          "description" : "",
          "x-example": [
            {
              "href": "/oic/res",
              "rt":   ["oic.wk.res"],
              "if":   ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
              "rel":  ["self"],
              "p":    {"bm": 3},
              "eps": [
                {"ep": "coaps://[fe80::b1d6]:1122"}               ]
            },
            {
              "href": "/humidity",
              "rt":   ["oic.r.humidity"],
              "if":   ["oic.if.s", "oic.if.baseline"],
              "p":    {"bm": 3},
              "eps": [
                {"ep": "coaps://[fe80::b1d6]:1111", "pri": 2},
                {"ep": "coaps://[fe80::b1d6]:1122"},
                {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
              ]
            },
            {
              "href": "/temperature",
              "rt":   ["oic.r.temperature"],
              "if":   ["oic.if.s", "oic.if.baseline"],
              "p":    {"bm": 3},
              "eps": [
                {"ep": "coaps://[[2001:db8:a::123]:2222"}
              ]
            }
          ],
          "schema": {
            "$ref": "#/definitions/slinklist"
          }
        }
      }
    }
  },
  "/oic/res?if=oic.if.b" : {
    "get": {
      "description": "Batch representation of /oic/res; list of discoverable Resources\n",
      "parameters": [
        {"$ref": "#/parameters/interface-all"}
      ],
      "responses": {
        "200": {
          "description" : "",
          "x-example": [
            {
              "href": "/humidity",
              "rep":{
                "rt":           ["oic.r.humidity"],
                "humidity": 40,
                "desiredHumidity": 40
```



```
            }
          },
          {
            "href": "/temperature",
            "rep":{
              "rt":          ["oic.r.temperature"],
              "temperature": 20.0,
              "units": "C"
            }
          }
        ],
        "schema": { "$ref": "#/definitions/sbatch" }
      }
    }
  }
},
"/oic/res?if=oic.if.baseline": {
  "get": {
    "description": "Baseline representation of /oic/res; list of discoverable Resources\n",
    "parameters": [
      {
        "$ref": "#/parameters/interface-all"
      }
    ],
    "responses": {
      "200": {
        "description": "",
        "x-example": [
          {
            "rt": ["oic.wk.res"],
            "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
            "links": [
              {
                "href": "/humidity",
                "rt":   ["oic.r.humidity"],
                "if":   ["oic.if.s", "oic.if.baseline"],
                "p":    {"bm": 3},
                "eps": [
                  {"ep": "coaps://[fe80::b1d6]:1111", "pri": 2},
                  {"ep": "coaps://[fe80::b1d6]:1122"},
                  {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
                ]
              },
              {
                "href": "/temperature",
                "rt":   ["oic.r.temperature"],
                "if":   ["oic.if.s", "oic.if.baseline"],
                "p":    {"bm": 3},
                "eps": [
                  {"ep": "coaps://[[2001:db8:a::123]:2222"}
                ]
              }
            ]
          }
        ],
        "schema": {
          "$ref": "#/definitions/sbaseline"
        }
      }
    }
  }
},
"parameters": {
  "interface-all": {
    "in": "query",
    "name": "if",
    "type": "string",
    "enum": ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
  }
},
"definitions": {
```



```
    "oic.oic-link": {
      "type": "object",
      "properties": {
        "anchor": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/anchor"
        },
        "di": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
        },
        "eps": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
        },
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "if": {
          "description": "The OCF Interfaces supported by the Linked Resource",
          "items": {
            "enum": [
              "oic.if.baseline",
              "oic.if.ll",
              "oic.if.b",
              "oic.if.rw",
              "oic.if.r",
              "oic.if.a",
              "oic.if.s"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "type": "array"
        },
        "ins": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
        },
        "p": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
        },
        "rel": {
          "description": "The relation of the target URI referenced by the Link to the context URI",
          "oneOf": [
            {
              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
            },
            {
              "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_string"
            }
          ]
        },
        "rt": {
          "description": "Resource Type of the Linked Resource",
          "items": {
            "maxLength": 64,
            "type": "string"
```



```
          },
          "minItems": 1,
          "uniqueItems": true,
          "type": "array"
        },
        "title": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
        },
        "type": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
        },
        "tag-pos-desc": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-pos-desc"
        },
        "tag-pos-rel": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-pos-rel"
        },
        "tag-func-desc": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/tag-func-desc"
        }
      },
      "required": [
        "href",
        "rt",
        "if"
      ]
    },
    "slinklist": {
      "type" : "array",
      "readOnly": true,
      "items": {
        "$ref": "#/definitions/oic.oic-link"
      }
    },
    "sbaseline": {
      "type": "array",
      "minItems": 1,
      "maxItems": 1,
      "items": {
        "type": "object",
        "properties": {
          "n": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
          },
          "id": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
          },
          "rt": {
            "description": "Resource Type of this Resource",
            "items": {
              "enum": ["oic.wk.res"],
              "type": "string",
              "maxLength": 64
            },
            "minItems": 1,
            "readOnly": true,
            "uniqueItems": true,
            "type": "array"
```



```
          },
          "if": {
            "description": "The OCF Interfaces supported by this Resource",
            "items": {
              "enum": [
                "oic.if.ll",
                "oic.if.b",
                "oic.if.baseline"
              ],
              "type": "string",
              "maxLength": 64
            },
            "minItems": 2,
            "readOnly": true,
            "uniqueItems": true,
            "type": "array"
          },
          "links": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/oic.oic-link"
            }
          },
          "sduuid": {
            "description": "A UUID that identifies the Security Domain.",
            "type": "string",
            "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$",
            "readOnly": true
          },
          "sdname": {
            "description": "Human-friendly name for the Security Domain.",
            "type": "string",
            "readOnly": true
          }
        },
        "required": [
          "rt",
          "if",
          "links"
        ]
      }
    },
    "sbatch" : {
      "type" : "array",
      "minItems" : 1,
      "items" : {
        "type": "object",
        "additionalProperties": true,
        "properties": {
          "href": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
          },
          "rep": {
            "oneOf": [
              {
                "description": "The response payload from a single Resource",
                "type": "object"
              },
              {
                "description": " The response payload from a Collection (batch) Resource",
                "items": {
                  "$ref": "#/definitions/oic.oic-link"
                },
                "type": "array"
              }
            ]
          }
        },
        "required": [
```



```
          "href",
          "rep"
        ]
      }
    }
  }
}
```

A.7.5 Property definition

Table A.12 defines the Properties that are part of the "oic.wk.res" Resource Type.

Table A.12 – The Property definitions of the Resource with type "rt" = "oic.wk.res".

Property name	Value type	Mandatory	Access mode	Description
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the Linked Resource
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI
rt	array: see schema	Yes	Read Write	Resource Type of the Linked Resource
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
tag-pos-desc	multiple types: see schema	No	Read Write	
tag-pos-rel	multiple types: see schema	No	Read Write	
tag-func-desc	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource

links	array: see schema	Yes	Read Write	
sduuid	string	No	Read Only	A UUID that identifies the Security Domain.
sdname	string	No	Read Only	Human-friendly name for the Security Domain.
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	

A.7.6 CRUDN behaviour

Table A.13 defines the CRUDN operations that are supported on the "oic.wk.res" Resource Type.

Table A.13 – The CRUDN operations of the Resource with type "rt" = "oic.wk.res".

Create	Read	Update	Delete	Notify
	get			observe

A.8 MQTT configuration

A.8.1 Introduction

The Resource through which the MQTT server information can be set.

A.8.2 Example URI

/mqttconfResURI

A.8.3 Resource type

The Resource Type is defined as: "oic.r.mqtt.conf".

A.8.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "MQTT configuration",
    "version": "2022-01-11",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2022 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/mqttconfResURI" : {
      "get": {
        "description": "The Resource through which the MQTT server information can be set.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-all"}
        ],
        "responses": {
            "200": {
```

```
            "description" : "",
            "x-example": {
              "rt":   ["oic.r.mqtt.conf"],
              "server":    "",
              "port":   8,
              "kai" : 60,
              "uid" : "",
              "pwd" : "",
              "cacert" : "",
              "clcert" : "",
              "log" : "not connected",
              "crcode" : -1
            },
            "schema": { "$ref": "#/definitions/mqttconf" }
          }
        }
      },
      "post": {
        "description": "Set information to connect to an MQTT server\n",
        "parameters": [
          {"$ref": "#/parameters/interface-all"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/mqttconf" },
            "x-example": {
              "server": "test.mosquitto.org",
              "port":   1883 ,
              "kai" :  60
            }
          }
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example": {
                "rt":   ["oic.r.mqtt.conf"],
                "server":   "test.mosquitto.org",
                "port":   8,
                "kai" : 60,
                "uid" : "",
                "pwd" : "",
                "cacert" : "",
                "clcert" : "",
                "log" : "connected",
                "crcode" : 0
              },
              "schema": { "$ref": "#/definitions/mqttconf" }
            }
        }
      }
    }
  },
  "parameters": {
    "interface-all" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.rw", "oic.if.baseline"]
    }
  },
  "definitions": {
    "mqttconf" : {
      "properties": {
        "server" : {
          "description": "The connection information of the MQTT server. Can be a URI or IP
address",
          "type": "string"
        },
        "port" : {
          "description": "The port to connect too",
```



```
          "type": "integer"
        },
        "kai" : {
          "description": "The keep alive interval, in seconds",
          "type": "integer"
        },
        "uid" : {
          "description": "The user id to be supplied when connecting to the MQTT server",
          "type": "string"
        },
        "pwd" : {
          "description": "The password to be supplied when connecting to the MQTT server",
          "type": "string"
        },
        "cacert" : {
          "description": "The certificate authority certificate to be supplied when connecting to
the MQTT server",
          "type": "string"
        },
        "clcert" : {
          "description": "The client certificate to be supplied when connecting to the MQTT server",
          "type": "string"
        },
        "log" : {
          "description": "Logging information, giving status information back, formatting not
defined",
          "type": "string"
        },
        "crcode" : {
          "description": "MQTT connection reason codes, see MQTT v5 table 3-1 for values. note that
-1 indicates 'not yet connected' ",
          "type": "integer"
        },
        "rt" : {
          "description": "Resource Type of the Resource",
          "items": {
            "enum": ["oic.r.mqtt.conf"],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "id" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "n" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "if" : {
          "description": "The OCF Interfaces supported by this Resource",
          "items": {
            "enum": [
              "oic.if.rw",
              "oic.if.baseline"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        }
      },
      "type" : "object",
```



```
      "required": ["server", "port"]
    }
  }
}
```

A.8.5 Property definition

Table A.14 defines the Properties that are part of the "oic.r.mqtt.conf" Resource Type.

Table A.14 – The Property definitions of the Resource with type "rt" = "oic.r.mqtt.conf".

Property name	Value type	Mandatory	Access mode	Description
server	string	Yes	Read Write	The connection information of the MQTT server. Can be a URI or IP address
port	integer	Yes	Read Write	The port to connect too
kai	integer	No	Read Write	The keep alive interval, in seconds
uid	string	No	Read Write	The user id to be supplied when connecting to the MQTT server
pwd	string	No	Read Write	The password to be supplied when connecting to the MQTT server
cacert	string	No	Read Write	The certificate authority certificate to be supplied when connecting to the MQTT server
clcert	string	No	Read Write	The client certificate to be supplied when connecting to the MQTT server
log	string	No	Read Write	Logging information, giving status information back, formatting not defined
crcode	integer	No	Read Write	MQTT connection reason codes, see MQTT v5 table 3-1 for values. note that -1 indicates 'not yet connected'
rt	array: see schema	No	Read Only	Resource Type of the Resource
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource

A.8.6 CRUDN behaviour

Table defines the CRUDN operations that are supported on the "oic.r.mqtt.conf" Resource Type.

Table A.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.mqtt.conf".

Create	Read	Update	Delete	Notify
	get	post		observe

A.9 Push Configuration Resources

A.9.1 Introduction

A specialization of a Collection that contains only instances of "oic.r.notificationselector" composed with "oic.r.pushproxy". Each instance of them includes filtering parameters for the Resources to be pushed and Target Resource to which updated Resource of origin Server will be pushed.

A.9.2 Well-known URI

None

A.9.3 Resource type

The Resource Type is defined as: "oic.r.pushconfiguration".

A.9.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Push Configuration",
    "version": "2022-06-15",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/PushConfigurationResURI" : {
      "get": {
        "description": "Collection of oic.r.notificationselector with associated push
proxies.\nAllows a Server to be configured with one or more Push Notification destinations.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-ll"}
        ],
        "responses": {
          "200": {
            "description" : "",
            "x-example": [
              {"href": "/pushconfig/1", "rt": ["oic.r.notificationselector","oic.r.pushproxy"],
"if": ["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[fe80::b1d6]:1122"}]},
              {"href": "/pushconfig/2", "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
"if": ["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[fe80::b1d6]:1122"}]}
            ],
```

```
            "schema": { "$ref": "#/definitions/links-in-response" }
          }
        }
      }
    },
    "/PushConfigurationResURI?if=oic.if.baseline" : {
      "get": {
        "description": "Collection of oic.r.notificationselector and associated push
proxies.\nAllows a Server to be configured with one or more Push Notification destinations.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-baseline"}
        ],
        "responses": {
          "200": {
            "description" : "",
            "x-example": {
              "rt": ["oic.r.pushconfiguration"],
              "if": ["oic.if.ll","oic.if.create","oic.if.baseline"],
              "rts": ["oic.r.notificationselector", "oic.r.pushproxy"],
              "links": [
                {"href": "/pushconfig/1", "rt": ["oic.r.notificationselector","oic.r.pushproxy"],
"if": ["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[fe80::b1d6]:1122"}]},
                {"href": "/pushconfig/2", "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
"if": ["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[fe80::b1d6]:1122"}]}
              ]
            },
            "schema": { "$ref": "#/definitions/get-baseline-response" }
          }
        }
      }
    },
    "/PushConfigurationResURI?if=oic.if.create" : {
      "post": {
        "description": "Collection of oic.r.notificationselector and associated push
proxies.\nAllows a Server to be configured with one or more Push Notification destinations.\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface-create"
          },
          {
            "$ref": "#/parameters/body-create"
          }
        ],
        "responses": {
          "201": {
            "description" : "new link and corresponding target Resource are created",
            "x-example": {
              "href": "/pushconfig/1",
              "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
              "if": ["oic.if.baseline", "oic.if.rw"],
              "ins": 4213291245,
              "p": { "bm": 3 },
              "rep": {
                "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
                "if": ["oic.if.rw", "oic.if.baseline"],
                "phref": "/myAirquality",
                "prt": [
                  "oic.r.airquality"
                ],
                "pushtarget" : "coaps://[2001::200]:49355/pushed-resource-airquality",
                "sourcert": [
                  "oic.r.pushpayload"
                ],
                "state": "waitingforupdate"
              }
            },
            "schema": { "$ref": "#/definitions/post-create-response" }
          }
        }
      }
    }
  },
```



```
"parameters": {
  "interface-baseline" : {
    "in" : "query",
    "name" : "if",
    "type" : "string",
    "enum" : ["oic.if.baseline"]
  },
  "interface-ll" : {
    "in" : "query",
    "name" : "if",
    "type" : "string",
    "enum" : ["oic.if.ll"]
  },
  "interface-create" : {
    "in" : "query",
    "name" : "if",
    "type" : "string",
    "enum" : ["oic.if.create"]
  },
  "body-create" : {
    "in" : "body",
    "name" : "notificationselector-pushproxy",
    "required" : true,
    "schema" : {
      "$ref" : "#/definitions/post-create-request"
    },
    "x-example" : {
      "rt": [ "oic.r.notificationselector", "oic.r.pushproxy" ],
      "if": [ "oic.if.rw", "oic.if.baseline" ],
      "rep": {
        "phref": "/myFilterResURI",
        "prt": [
          "oic.r.airquality"
        ],
        "pushtarget" : "coaps://[2001::200]:49355/pushed-resource-filter",
        "sourcert": [
          "oic.r.pushpayload"
        ]
      }
    }
  }
},
"definitions": {
  "oic.oic-link": {
    "type": "object",
    "properties": {
      "if": {
        "description": "The OCF Interface set supported by the target Resource",
        "type": "array",
        "minItems": 1,
        "uniqueItems": true,
        "readOnly": true,
        "items": {
          "type": "string",
          "maxLength": 64,
          "enum": [
            "oic.if.baseline",
            "oic.if.rw"
          ]
        }
      },
      "rt": {
        "description": "Resource Type of the target Resource",
        "type": "array",
        "minItems": 1,
        "uniqueItems": true,
        "readOnly": true,
        "items": {
          "enum": [
            "oic.r.notificationselector",
            "oic.r.pushproxy"
          ],
```



```
            "type": "string",
            "maxLength": 64
          }
        },
        "anchor": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/anchor"
        },
        "di": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
        },
        "eps": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
        },
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "ins": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
        },
        "p": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
        },
        "rel": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
        },
        "title": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
        },
        "type": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
        }
      },
      "required": [
        "href",
        "rt",
        "if"
      ]
    },
    "links-in-response": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/oic.oic-link"
      }
    },
    "get-baseline-response" : {
      "type" : "object",
      "properties": {
        "rt": {
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "items": {
            "type": "string",
            "maxLength": 64,
```

```
            "enum": ["oic.r.pushconfiguration"]
          }
        },
        "if": {
          "description": "The OCF Interface set supported by this Resource",
          "type": "array",
          "minItems": 1,
          "readOnly": true,
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": [
              "oic.if.ll",
              "oic.if.create",
              "oic.if.baseline"
            ]
          }
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "rts": {
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": [
              "oic.r.notificationselector",
              "oic.r.pushproxy"
            ]
          }
        },
        "links": {
          "$ref": "#/definitions/links-in-response"
        }
      },
      "required": [
        "rt",
        "if",
        "links"
      ]
    },
    "rep-in-request": {
      "type": "object",
      "properties": {
        "phref": {
          "description": "URI of a Resource to be pushed",
          "type": "string",
          "maxLength": 256
        },
        "prt": {
          "description": "Resource type(s) of Resource(s) to be pushed",
          "type": "array",
          "uniqueItems": true,
          "items": {
            "type": "string",
            "maxLength": 64
          }
        },
        "pif": {
          "description": "OCF Interface(s) of Resource(s) to be pushed",
          "type": "array",
          "uniqueItems": true,
```



```
      "items": {
        "type": "string",
        "maxLength": 64
      }
    },
    "pushtarget": {
      "description": "Points to the target of the UPDATE operation sent as a notifcation",
      "type": "string",
      "maxLength": 256
    },
    "sourcert": {
      "description" : "Always set to oic.r.pushpayload",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type": "string",
        "maxLength": 64,
        "enum" : [
          "oic.r.pushpayload"
        ]
      }
    },
    "state": {
      "description": "Current state of the Push Proxy",
      "type": "string",
      "enum": [
        "waitingforprovisioning",
        "waitingforupdate",
        "waitingforresponse",
        "waitingforupdatemitigation",
        "waitingforresponsemitigation",
        "error",
        "timeout"
      ]
    }
  },
  "required": [
    "pushtarget",
    "sourcert"
  ]
},
"rep-in-response": {
  "type": "object",
  "properties": {
    "rt": {
      "description": "Resource Type of the target Resource",
      "type": "array",
      "minItems": 1,
      "uniqueItems": true,
      "readOnly": true,
      "items": {
        "enum": [
          "oic.r.notificationselector",
          "oic.r.pushproxy"
        ],
        "type": "string",
        "maxLength": 64
      }
    },
    "if": {
      "description": "The OCF Interface set supported by the target Resource",
      "type": "array",
      "minItems": 1,
      "uniqueItems": true,
      "readOnly": true,
      "items": {
        "type": "string",
        "maxLength": 64,
        "enum": [
          "oic.if.rw",
          "oic.if.baseline"
        ]
```



```
        }
      },
      "phref": {
        "description": "URI of a Resource to be pushed",
        "type": "string",
        "maxLength": 256
      },
      "prt": {
        "description": "Resource type(s) of Resource(s) to be pushed",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string",
          "maxLength": 64
        }
      },
      "pif": {
        "description": "OCF Interface(s) of Resource(s) to be pushed",
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string",
          "maxLength": 64
        }
      },
      "pushtarget": {
        "description": "Points to the target of the UPDATE operation sent as a notifcation",
        "type": "string",
        "maxLength": 256
      },
      "sourcert": {
        "description" : "Always set to oic.r.pushpayload",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type": "string",
          "maxLength": 64,
          "enum" : [
            "oic.r.pushpayload"
          ]
        }
      },
      "state": {
        "description": "Current state of the Push Proxy",
        "type": "string",
        "enum": [
          "waitingforprovisioning",
          "waitingforupdate",
          "waitingforresponse",
          "waitingforupdatemitigation",
          "waitingforresponsemitigation",
          "error",
          "timeout"
        ]
      }
    },
    "required": [
      "rt",
      "if",
      "pushtarget",
      "sourcert",
      "state"
    ]
  },
  "post-create-request": {
    "type" : "object",
    "properties": {
      "rt": {
        "description": "Resource Type of the target Resource",
        "type": "array",
        "minItems": 1,
        "uniqueItems": true,
```



```
          "readOnly": true,
          "items": {
            "enum": [
              "oic.r.notificationselector",
              "oic.r.pushproxy"
            ],
            "type": "string",
            "maxLength": 64
          }
        },
        "if": {
          "description": "The OCF Interface set supported by the target Resource",
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": [
              "oic.if.rw",
              "oic.if.baseline"
            ]
          }
        },
        "rep": {
          "$ref": "#/definitions/rep-in-request"
        }
      },
      "required": [
        "rt",
        "if",
        "rep"
      ]
    },
    "post-create-response": {
      "type" : "object",
      "properties": {
        "href": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "rt": {
          "description": "Resource Type of the target Resource",
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "items": {
            "enum": [
              "oic.r.notificationselector",
              "oic.r.pushproxy"
            ],
            "type": "string",
            "maxLength": 64
          }
        },
        "if": {
          "description": "The OCF Interface set supported by the target Resource",
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": [
              "oic.if.rw",
              "oic.if.baseline"
            ]
          }
```



```
        },
        "ins": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
        },
        "p": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
        },
        "rep": {
          "$ref": "#/definitions/rep-in-response"
        }
      },
      "required": [
        "href",
        "rt",
        "if",
        "ins",
        "rep"
      ]
    }
  }
}
```

A.10 Composition Resource of Notification Selector and Push Proxy

A.10.1 Introduction

Each instance of composition Resource of Notification Selector and Push Proxy includes filtering parameters for the Resources to be pushed and Target Resource to which updated Resource of origin Server will be pushed.

A.10.2 Well-known URI

None

A.10.3 Resource type

The Resource Type is defined as: ["oic.r.pushconfiguration", "oic.r.pushproxy"]

A.10.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Notification Selector-Push Proxy",
    "version": "2022-06-15",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/NotificationSelectorPushproxyResURI?if=oic.if.rw" : {
      "get": {
        "description": "Resource that defines the selector for Push Notifications",
```



```
      "parameters": [
        {"$ref": "#/parameters/interface-rw"}
      ],
      "responses": {
        "200": {
          "description" : "",
          "x-example": {
            "phref": "/myAirquality",
            "prt": [
              "oic.r.airquality"
            ],
            "pushtarget" : "coaps://[2001::200]:49355/pushed-resource-airquality",
            "sourcert": [
              "oic.r.pushpayload"
            ],
            "state": "waitingforupdate"
          },
          "schema": { "$ref": "#/definitions/get-nspp-rw-response" }
        }
      }
    },
    "post": {
      "description": "Updates the current notification selector information.\n",
      "parameters": [
        { "$ref": "#/parameters/interface-rw" },
        { "$ref": "#/parameters/body-update" }
      ],
      "responses": {
        "204": {
          "description" : "the notification selector-push proxy is updated successfully\n"
        }
      }
    }
  },
  "/NotificationSelectorPushproxyResURI?if=oic.if.baseline" : {
    "get": {
      "description": "Resource that defines the selector for Push Notifications",
      "parameters": [
        {"$ref": "#/parameters/interface-baseline"}
      ],
      "responses": {
        "200": {
          "description" : "",
          "x-example": {
            "rt": ["oic.r.notificationselector", "oic.r.pushproxy"],
            "if": ["oic.if.rw", "oic.if.baseline"],
            "phref": "/myAirquality",
            "prt": [
              "oic.r.airquality"
            ],
            "pushtarget" : "coaps://[2001::200]:49355/pushed-resource-airquality",
            "sourcert": [
              "oic.r.pushpayload"
            ],
            "state": "waitingforupdate"
          },
          "schema": { "$ref": "#/definitions/get-nspp-baseline-response" }
        }
      }
    }
  }
},
"parameters": {
  "interface-rw" : {
    "in" : "query",
    "name" : "if",
    "type" : "string",
    "enum" : ["oic.if.rw"]
  },
  "interface-baseline" : {
    "in" : "query",
    "name" : "if",
```



```
      "type" : "string",
      "enum" : ["oic.if.baseline"]
    },
    "body-update": {
      "name": "notificationselector-pushproxy",
      "in": "body",
      "required": true,
      "schema": { "$ref": "#/definitions/post-nspp-rw-request" },
      "x-example": {
        "phref": "/myFilterResURI",
        "pushtarget" : "coaps://[2001::200]:49355/pushed-resource-filter",
        "sourcert": [
          "oic.r.pushpayload"
        ]
      }
    }
  },
  "definitions": {
    "get-nspp-baseline-response" : {
         "type": "object",
      "properties": {
        "rt": {
          "description": "Resource Type of the Resource",
          "items": {
            "enum": ["oic.r.notificationselector", "oic.r.pushproxy"],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "if": {
          "description": "The interface set supported by this resource",
          "items": {
            "enum": [
              "oic.if.rw",
              "oic.if.baseline"
            ],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "phref" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "prt": {
          "description": "Resource Type(s) of the Resource(s) to be pushed",
          "type": "array",
          "items" : {
            "type" : "string",
            "maxLength": 64
          },
          "minItems" : 1
        },
```



```
        "pif": {
          "description": "The OCF Interface(s) of the Resource(s) to be pushed",
          "type": "array",
          "items": {
            "type" : "string",
            "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
"oic.if.r", "oic.if.a", "oic.if.s" ]
          },
          "minItems": 1
        },
        "pushtarget": {
          "description": "Points to the target of the UPDATE operation sent as a notification",
          "type": "string",
          "maxLength": 256
        },
        "sourcert": {
          "description" : "Always set to oic.r.pushpayload",
          "type" : "array",
          "uniqueItems" : true,
          "items" : {
            "type": "string",
            "maxLength": 64,
            "enum" : [
              "oic.r.pushpayload"
            ]
          }
        },
        "state": {
          "description": "Current state of the Push Proxy",
          "type": "string",
          "enum": [
            "waitingforprovisioning",
            "waitingforupdate",
            "waitingforresponse",
            "waitingforupdatemitigation",
            "waitingforresponsemitigation",
            "error",
            "timeout"
          ]
        }
      },
      "required": [
        "rt",
        "if",
        "pushtarget",
        "sourcert",
        "state"
      ]
    },
    "get-nspp-rw-response" : {
      "type": "object",
      "properties": {
        "phref" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "prt": {
          "description": "Resource Type(s) of the Resource(s) to be pushed",
          "type": "array",
          "items" : {
            "type" : "string",
            "maxLength": 64
          },
          "minItems" : 1
        },
        "pif": {
          "description": "The OCF Interface(s) of the Resource(s) to be pushed",
          "type": "array",
          "items": {
            "type" : "string",
```



```
            "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
"oic.if.r", "oic.if.a", "oic.if.s" ]
          },
          "minItems": 1
        },
        "pushtarget": {
          "description": "Points to the target of the UPDATE operation sent as a notification",
          "type": "string",
          "maxLength": 256
        },
        "sourcert": {
          "description" : "Always set to oic.r.pushpayload",
          "type" : "array",
          "uniqueItems" : true,
          "items" : {
            "type": "string",
            "maxLength": 64,
            "enum" : [
              "oic.r.pushpayload"
            ]
          }
        },
        "state": {
          "description": "Current state of the Push Proxy",
          "type": "string",
          "enum": [
            "waitingforprovisioning",
            "waitingforupdate",
            "waitingforresponse",
            "waitingforupdatemitigation",
            "waitingforresponsemitigation",
            "error",
            "timeout"
          ]
        }
      },
      "required": [
        "pushtarget",
        "sourcert",
        "state"
      ]
    },
    "post-nspp-rw-request" : {
      "type": "object",
      "properties": {
        "phref" : {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
        },
        "prt": {
          "description": "Resource Type(s) of the Resource(s) to be pushed",
          "type": "array",
          "items" : {
            "type" : "string",
            "maxLength": 64
          },
          "minItems" : 1
        },
        "pif": {
          "description": "The OCF Interface(s) of the Resource(s) to be pushed",
          "type": "array",
          "items": {
            "type" : "string",
            "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
"oic.if.r", "oic.if.a", "oic.if.s" ]
          },
          "minItems": 1
        },
        "pushtarget": {
          "description": "Points to the target of the UPDATE operation sent as a notification",
          "type": "string",
```



```
          "maxLength": 256
        },
        "sourcert": {
          "description" : "Always set to oic.r.pushpayload",
          "type" : "array",
          "uniqueItems" : true,
          "items" : {
            "type": "string",
            "maxLength": 64,
            "enum" : [
              "oic.r.pushpayload"
            ]
          }
        },
        "state": {
          "description": "Current state of the Push Proxy",
          "type": "string",
          "enum": [
            "waitingforprovisioning",
            "waitingforupdate",
            "waitingforresponse",
            "waitingforupdatemitigation",
            "waitingforresponsemitigation",
            "error",
            "timeout"
          ]
        }
      },
      "required": [
        "pushtarget",
        "sourcert"
      ]
    }
  }
}
```

A.11 Push Receiver Resource

A.11.1 Introduction

The Resource through which a Device can be configured as a target Server for push notifications.

A.11.2 Well-known URI

None

A.11.3 Resource type

The Resource Type is defined as: "oic.r.pushreceiver"

A.11.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Push Receiver",
    "version": "2022-06-15",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/PushReceiverResURI": {
```

```
    "get": {
      "description": "Resource that defines the receiver for Push Notifications",
      "parameters": [
        {
          "$ref": "#/parameters/interface-rw"
        }
      ],
      "responses": {
        "200": {
          "description": "",
          "x-example": {
            "receivers": [
              {
                "receiveruri": "/mylocaltargeturiforthermostats",
                "rts": ["oic.r.temperature", "oic.r.humidity"]
              },
              {
                "receiveruri": "/mylocaltargeturifordontcare",
                "rts": []
              }
            ]
          },
          "schema": { "$ref": "#/definitions/get-rw-response" }
        }
      }
    },
    "post": {
      "description": "replace whole push receiver objects\n",
      "parameters": [
        { "$ref": "#/parameters/interface-rw" },
        { "$ref": "#/parameters/body-receivers-update" }
      ],
      "responses": {
        "204": {
          "description": "whole receiver objects are replaced successfully\n"
        }
      }
    },
    "delete": {
      "description": "delete whole push receiver objects\n",
      "parameters": [
        {
          "$ref": "#/parameters/interface-rw"
        }
      ],
      "responses": {
        "204": {
          "description": "whole receiver objects are removed successfully\n"
        }
      }
    }
  },
  "/PushReceiverResURI?receiveruri=<uri>": {
    "post": {
      "description": "Updates a push receiver object which has `receiveruri`\n",
      "parameters": [
        { "$ref": "#/parameters/interface-rw" },
        { "$ref": "#/parameters/receiveruri" },
        { "$ref": "#/parameters/body-receiver-update" }
      ],
      "responses": {
        "204": {
          "description": "a receiver object is updated successfully\n"
        }
      }
    },
    "delete": {
      "description": "delete a push receiver object which has `receiveruri`\n",
      "parameters": [
        { "$ref": "#/parameters/interface-rw" },
        { "$ref": "#/parameters/receiveruri" }
      ],
```



```
      "responses": {
        "204": {
          "description": "a receiver object was removed successfully\n"
        }
      }
    }
  },
  "/PushReceiverResURI?if=oic.if.baseline": {
    "get": {
      "description": "Resource that defines the receiver for Push Notifications",
      "parameters": [
        { "$ref": "#/parameters/interface-baseline" }
      ],
      "responses": {
        "200": {
          "description": "",
          "x-example": {
            "rt": ["oic.r.pushreceiver"],
            "if": ["oic.if.rw", "oic.if.baseline"],
            "receivers": [
              {
                "receiveruri": "/mylocaltargeturiforthermostats",
                "rts": ["oic.r.temperature", "oic.r.humidity"]
              },
              {
                "receiveruri": "/mylocaltargeturifordontcare",
                "rts": []
              }
            ]
          },
          "schema": { "$ref": "#/definitions/get-baseline-response" }
        }
      }
    }
  }
},
"parameters": {
  "interface-baseline": {
    "in": "query",
    "name": "if",
    "type": "string",
    "enum": ["oic.if.baseline"]
  },
  "interface-rw": {
    "in": "query",
    "name": "if",
    "type": "string",
    "enum": ["oic.if.rw"]
  },
  "receiveruri": {
    "in": "query",
    "name": "receiveruri",
    "type": "string"
  },
  "body-receiver-update": {
    "in": "body",
    "name": "receiver",
    "required": true,
    "schema": {
      "$ref": "#/definitions/receiver"
    },
    "x-example": {
      "receiveruri": "/mylocaltargeturifordontcare",
      "rts": []
    }
  },
  "body-receivers-update": {
    "in": "body",
    "name": "receivers",
    "required": true,
    "schema": {
      "$ref": "#/definitions/post-rw-request"
```



```
      },
      "x-example": {
        "receivers": [
          {
            "receiveruri": "/mylocaltargeturifordontcare",
            "rts": []
          },
          {
            "receiveruri": "/mylocaltargeturifordontcare-2",
            "rts": []
          }
        ]
      }
    }
  },
  "definitions": {
    "receiver": {
      "description": "a definition of URIs at which push payloads may be received",
      "type": "object",
      "properties": {
        "receiveruri": {
          "format": "uri",
          "type": "string"
        },
        "rts": {
          "description": "The list of allowable Resource Types for this instance of a push
receiver",
          "type": "array",
          "items": {
            "type": "string",
            "maxLength": 64
          },
          "minItems": 0
        }
      },
      "required": ["receiveruri", "rts"]
    },
    "receivers": {
      "description": "Definitions of URIs at which push payloads may be received",
      "type": "array",
      "items": {
        "$ref": "#/definitions/receiver"
      },
      "minItems": 0
    },
    "get-rw-response": {
      "type": "object",
      "properties": {
        "receivers": {
          "$ref": "#/definitions/receivers"
        }
      },
      "required": ["receivers"]
    },
    "get-baseline-response": {
      "type": "object",
      "properties": {
        "rt": {
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": ["oic.r.pushreceiver"]
          }
        },
        "if": {
          "description": "The OCF Interface set supported by this Resource",
          "type": "array",
          "minItems": 1,
          "readOnly": true,
```



```
          "items": {
            "type": "string",
            "maxLength": 64,
            "enum": ["oic.if.rw", "oic.if.baseline"]
          }
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "receivers": {
          "$ref": "#/definitions/receivers"
        }
      },
      "required": [
        "rt", "if", "receivers"
      ]
    },
    "post-rw-request": {
      "type": "object",
      "properties": {
        "receivers": {
          "$ref": "#/definitions/receivers"
        }
      },
      "required": ["receivers"]
    }
  }
}
```


Annex B
(informative)

OpenAPI 2.0 Schema Extension

B.1 OpenAPI 2.0 Schema Reference

OpenAPI 2.0 does not support allOf and anyOf JSON schema valiation constructs; this document has extended the underlying OpenAPI 2.0 schema to enable these, all OpenAPI 2.0 files are valid against the extended schema. Reference the following location for a copy of the extended schema:

– https://github.com/openconnectivityfoundation/OCFswagger2.0-schema

B.2 OpenAPI 2.0 Introspection empty file

Reference the following location for a copy of an empty OpenAPI 2.0 file:

– https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-examples/introspection-empty.txt

Annex C
(normative)

Semantic Tag enumeration support

C.1 Introduction

This Annex defines the enumerations that are applicable to defined Semantic Tags.

C.2 "tag-pos-desc" supported enumeration

Figure C.1 defines the enumeration from which a value populated within an instance of the "tag-pos-desc" Semantic Tag is taken.

```
"pos-descriptions": {
"enum":
["unknown","top","bottom","left","right","centre","topleft","bottomleft","centreleft"
,"centreright","bottomright","topright","topcentre","bottomcentre"]
}
```

Figure C.1 – Enumeration for "tag-pos-desc" Semantic Tag

Figure C.2 provides an illustrative representation of the definition of the values that can be represented within an instance of "tag-pos-desc".

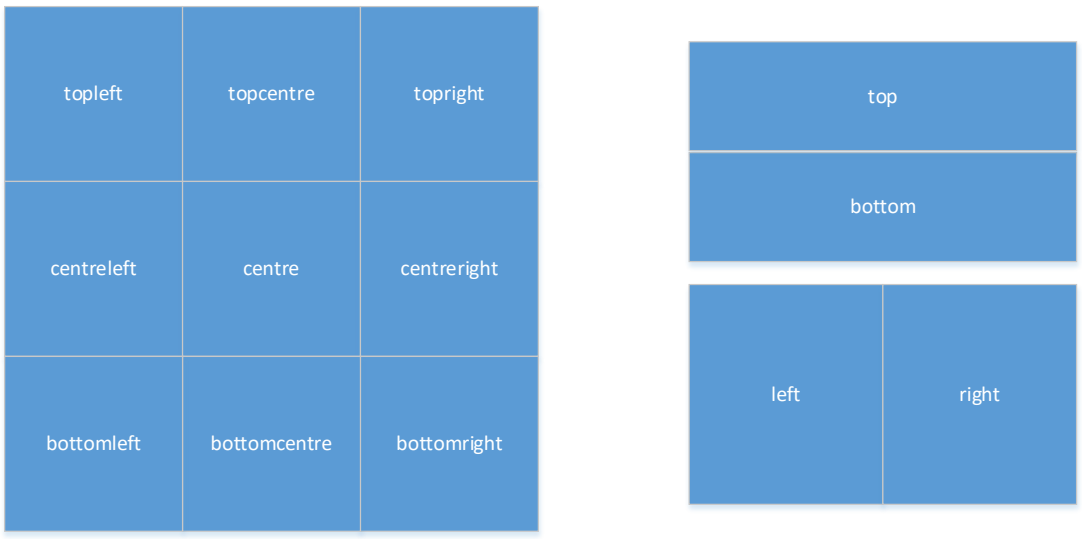


Figure C.2 – Definition of "tag-pos-desc" Semantic Tag values

C.3 "tag-loc" supported enumeration

Figure C.3 defines the enumeration from which a value populated within an instance of the "tag-locn" Semantic Tag is taken.

```
"locn-descriptions": {

"enum":
["unknown","attic","balcony","ballroom","bathroom","bedroom","border","boxroom","cellar","cloakr
oom","conservatory","corridor","deck","den","diningroom","drawingroom","driveway","dungeon","ens
uite","entrance","familyroom","garage","garden","guestroom","hall","indoor","kitchen","larder","
lawn","library","livingroom","lounge","mancave","masterbedroom","musicroom","office","outdoor","
pantry","parkinglot","parlour","patio","receiptionroom","restroom","roof","roofterrace","sauna",
```



```
 "scullery","shed","sittingroom","snug","spa","studio","suite","swimmingpool","terrace","toilet",
 "utilityroom","vegetableplot","ward","yard"]


}
```

Figure C.3 – Enumeration for "tag-locn" Semantic Tag

Bibliography

The following are documents that are informatively (but not normatively) referenced herein.

[1] OCF Core - Optional, Information technology – Open Connectivity Foundation (OCF) Specification – Part 9: Core - Optional specification
Latest version available at:
https://openconnectivity.org/specs/OCF_Core_Optional_Specification.pdf

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

