



BSI Standards Publication

Information technology — Open Connectivity Foundation (OCF) Specification

Part 1: Core specification

National foreword

This British Standard is the UK implementation of ISO/IEC 30118-1:2021. It supersedes BS ISO/IEC 30118-1:2018, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee ICT/1, Information systems co-ordination.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 16184 7

ICS 35.200

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2022.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

**Information technology — Open
Connectivity Foundation (OCF)
Specification —**

**Part 1:
Core specification**

*Technologies de l'information — Specification de la Fondation pour la
connectivité ouverte (Fondation OCF) —*

Partie 1: Spécification du coeur



Reference number
ISO/IEC 30118-1:2021(E)

Documents
Document Sharing Hub
© ISO/IEC 2021



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	1
3 Terms, definitions and abbreviated terms.....	3
3.1 Terms and definitions	3
3.2 Symbols and abbreviated terms	7
4 Document conventions and organization.....	8
4.1 Conventions	8
4.2 Notation.....	8
4.3 Data types	9
4.4 Resource notation syntax	10
5 Architecture	11
5.1 Overview.....	11
5.2 Principle	11
5.3 Functional block diagram.....	12
5.4 Framework.....	14
6 Identification and addressing.....	14
6.1 Introduction.....	14
6.2 Identification	15
6.2.1 Device and Platform identification.....	15
6.2.2 Resource identification and addressing.....	15
6.3 Namespace:.....	16
6.4 Network addressing	16
7 Resource model.....	17
7.1 Introduction.....	17
7.2 Resource	17
7.3 Property.....	18
7.3.1 Introduction.....	18
7.3.2 Common Properties	19
7.4 Resource Type	21
7.4.1 Introduction.....	21
7.4.2 Resource Type Property	21
7.4.3 Resource Type definition.....	21
7.4.4 Multi-value "rt" Resource.....	23
7.5 Device Type.....	23
7.6 OCF Interface	24
7.6.1 Introduction.....	24
7.6.2 OCF Interface Property	24
7.6.3 OCF Interface methods	25
7.7 Resource representation.....	42
7.8 Structure	42
7.8.1 Introduction.....	42
7.8.2 Resource relationships (Links).....	42
7.8.3 Collections	48

7.8.4	Atomic Measurement	50
7.9	Query Parameters	52
7.9.1	Introduction	52
7.9.2	Use of multiple parameters within a query.....	52
7.9.3	Application to multi-value "rt" Resources	53
7.9.4	OCF Interface specific considerations for queries	53
8	CRUDN.....	54
8.1	Overview	54
8.2	CREATE.....	54
8.2.1	Overview	54
8.2.2	CREATE request	55
8.2.3	Processing by the Server	55
8.2.4	CREATE response.....	55
8.3	RETRIEVE.....	56
8.3.1	Overview	56
8.3.2	RETRIEVE request	56
8.3.3	Processing by the Server	56
8.3.4	RETRIEVE response	56
8.4	UPDATE.....	57
8.4.1	Overview	57
8.4.2	UPDATE request	57
8.4.3	Processing by the Server	57
8.4.4	UPDATE response.....	58
8.5	DELETE	58
8.5.1	Overview	58
8.5.2	DELETE request.....	59
8.5.3	Processing by the Server	59
8.5.4	DELETE response	59
8.6	NOTIFY	59
8.6.1	Overview	59
8.6.2	NOTIFICATION response	59
9	Network and connectivity.....	60
9.1	Introduction	60
9.2	Architecture	60
9.3	IPv6 network layer requirements	61
9.3.1	Introduction	61
9.3.2	IPv6 node requirements.....	62
10	OCF Endpoint	62
10.1	OCF Endpoint definition	62
10.2	OCF Endpoint information	63
10.2.1	Introduction	63
10.2.2	"ep"	63
10.2.3	"pri".....	64
10.2.4	"lat"	64
10.2.5	OCF Endpoint information in "eps" Parameter	64
10.3	OCF Endpoint discovery	65
10.3.1	Introduction	65
10.3.2	Implicit discovery	65

10.3.3	Explicit discovery with "/oic/res" response	65
11	Functional interactions.....	67
11.1	Introduction.....	67
11.2	Resource discovery	68
11.2.1	Introduction.....	68
11.2.2	Resource based discovery: mechanisms	68
11.2.3	Resource based discovery: Finding information.....	69
11.2.4	Resource discovery using "/oic/res"	76
11.2.5	Multicast discovery using "/oic/res".....	77
11.3	Notification.....	78
11.3.1	Overview	78
11.3.2	Observe	78
11.4	Introspection	79
11.4.1	Overview	79
11.4.2	Usage of Introspection.....	83
11.5	Semantic Tags.....	84
11.5.1	Introduction.....	84
11.5.2	Semantic Tag definitions	85
12	Messaging.....	87
12.1	Introduction.....	87
12.2	Mapping of CRUDN to CoAP	87
12.2.1	Overview	87
12.2.2	URIs.....	88
12.2.3	CoAP method with request and response	88
12.2.4	Content-Format negotiation.....	89
12.2.5	OCF-Content-Format-Version information.....	90
12.2.6	Content-Format policy	91
12.2.7	CRUDN to CoAP response codes.....	92
12.2.8	CoAP block transfer.....	92
12.2.9	Generic requirements for CoAP multicast.....	92
12.2.10	Setting timeout on response to a confirmable request	93
12.3	Mapping of CRUDN to CoAP serialization over TCP.....	93
12.3.1	Overview	93
12.3.2	URIs.....	93
12.3.3	CoAP method with request and response	93
12.3.4	Content-Format negotiation.....	93
12.3.5	OCF-Content-Format-Version information.....	94
12.3.6	Content-Format policy	94
12.3.7	CRUDN to CoAP response codes.....	94
12.3.8	CoAP block transfer.....	94
12.3.9	Keep alive (connection health)	94
12.3.10	CoAP using a proxy	94
12.4	Payload Encoding in CBOR.....	94
13	Security.....	95
Annex A (normative)	Resource Type definitions.....	96
A.1	List of Resource Type definitions	96
A.2	Atomic Measurement links list representation	96
A.2.1	Introduction.....	96

A.2.2	Example URI.....	96
A.2.3	Resource type.....	96
A.2.4	OpenAPI 2.0 definition.....	96
A.2.5	Property definition.....	102
A.2.6	CRUDN behaviour.....	103
A.3	Collection.....	103
A.3.1	Introduction.....	103
A.3.2	Example URI.....	103
A.3.3	Resource type.....	104
A.3.4	OpenAPI 2.0 definition.....	104
A.3.5	Property definition.....	110
A.3.6	CRUDN behaviour.....	112
A.4	Device.....	112
A.4.1	Introduction.....	112
A.4.2	Well-known URI.....	112
A.4.3	Resource type.....	112
A.4.4	OpenAPI 2.0 definition.....	112
A.4.5	Property definition.....	115
A.4.6	CRUDN behaviour.....	116
A.5	Introspection Resource.....	116
A.5.1	Introduction.....	116
A.5.2	Well-known URI.....	116
A.5.3	Resource type.....	116
A.5.4	OpenAPI 2.0 definition.....	116
A.5.5	Property definition.....	118
A.5.6	CRUDN behaviour.....	119
A.6	Platform.....	119
A.6.1	Introduction.....	119
A.6.2	Well-known URI.....	119
A.6.3	Resource type.....	119
A.6.4	OpenAPI 2.0 definition.....	119
A.6.5	Property definition.....	122
A.6.6	CRUDN behaviour.....	123
A.7	Discoverable Resources.....	123
A.7.1	Introduction.....	123
A.7.2	Well-known URI.....	123
A.7.3	Resource type.....	123
A.7.4	OpenAPI 2.0 definition.....	123
A.7.5	Property definition.....	128
A.7.6	CRUDN behaviour.....	129
Annex B (informative)	OpenAPI 2.0 Schema Extension.....	130
B.1	OpenAPI 2.0 Schema Reference.....	130
B.2	OpenAPI 2.0 Introspection empty file.....	130
Annex C (normative)	Semantic Tag enumeration support.....	131
C.1	Introduction.....	131
C.2	"tag-pos-desc" supported enumeration.....	131
Bibliography		132

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the Open Connectivity Foundation (OCF) (as OCF Core Specification, version 2.2.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

This second edition cancels and replaces the first edition (ISO/IEC 30118-1:2018), which has been technically revised.

The main changes compared to the previous edition are as follows:

- simplification of specification so that it only includes the core functionality;
- additional reusable infrastructure components are now in the core optional specification;
- addition of semantic tags, sleepy devices based on long latency;
- improvements made on CoAP bindings, Error diagnostic payloads, discovery, reset of device and usage of baseline interfaces;
- addition of clarifications throughout.

A list of all parts in the ISO/IEC 30118 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

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.

ISO/IEC 30118 consists of eighteen parts, under the general title Information technology — Open Connectivity Foundation (OCF) Specification. The parts fall into logical groupings as described herein:

- Core framework
 - Part 1: Core Specification
 - Part 2: Security Specification
 - Part 13: Onboarding Tool Specification
- Bridging framework and bridges
 - Part 3: Bridging Specification
 - Part 6: Resource to Alljoyn Interface Mapping Specification
 - Part 8: OCF Resource to oneM2M Resource Mapping Specification
 - Part 14: OCF Resource to BLE Mapping Specification
 - Part 15: OCF Resource to EnOcean Mapping Specification
 - Part 16: OCF Resource to UPlus Mapping Specification
 - Part 17: OCF Resource to Zigbee Cluster Mapping Specification
 - Part 18: OCF Resource to Z-Wave Mapping Specification
- Resource and Device models
 - Part 4: Resource Type Specification
 - Part 5: Device Specification

- Core framework extensions
 - Part 7: Wi-Fi Easy Setup Specification
 - Part 9: Core Optional Specification
- OCF Cloud
 - Part 10: Cloud API for Cloud Services Specification
 - Part 11: Device to Cloud Services Specification
 - Part 12: Cloud Security Specification

Information technology — Open Connectivity Foundation (OCF) Specification —

Part 1: 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 2016
<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>

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* (3.1.33) of linked *Resources* (3.1.31) 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* (3.1.31) on a *Server* (3.1.36)

3.1.7

Collection

Resource (3.1.31) that contains zero or more *Links* (3.1.21)

3.1.8

Common Properties

Properties (3.1.33) specified for all *Resources* (3.1.31)

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* (3.1.31) that are defined in this document

3.1.12

Default OCF Interface

OCF Interface (3.1.18) used to generate the response when an *OCF Interface* (3.1.18) is omitted in a request

3.1.13

Device

logical entity that assumes one or more roles, e.g., *Client* (3.1.6), *Server* (3.1.36)

Note 1 to entry: More than one *Device* (3.1.13) can exist on a *Platform* (3.1.30).

3.1.14

Device Type

uniquely named definition indicating a minimum set of *Resource Types* (3.1.34) 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* (3.1.31) discovery.

3.1.15

Device UUID

stack instance identifier

3.1.16**Discoverable Resource**

Resource (3.1.31) 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* (3.1.31)

[SOURCE: IETF RFC 6690]

3.1.20**Introspection**

mechanism to determine the capabilities of the hosted *Resources* (3.1.31) of a *Device* (3.1.13)

3.1.21**Introspection Device Data (IDD)**

data that describes the payloads per implemented method of the *Resources* (3.1.31) 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 (3.1.31) that is not listed in *"/oic/res"*

Note 1 to entry: The *Resource* (3.1.31) can be reached by a *Link* (3.1.21) which is conveyed by another *Resource* (3.1.31). For example a *Resource* (3.1.31) 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* (3.1.31) 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* (3.1.31)

3.1.25

Observe

act of monitoring a *Resource* (3.1.31) by sending a RETRIEVE operation which is cached by the *Server* (3.1.36) hosting the *Resource* (3.1.31) and reprocessed on every change to that *Resource* (3.1.31)

3.1.26

OpenAPI 2.0

Resource (3.1.31) and *Intropection Device Data* (3.1.20) definitions used in this document

[SOURCE: OpenAPI specification]

3.1.27

Parameter

element that provides metadata about a *Resource* (3.1.31) referenced by the target URI of a *Link* (3.1.21)

3.1.28

Partial UPDATE

UPDATE operation to a *Resource* (3.1.31) that includes a subset of the *Properties* (3.1.33) that are visible via the *OCF Interface* (3.1.18) being applied for the *Resource Type* (3.1.34)

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 (3.1.29) containing one or more *Devices* (3.1.13)

3.1.32

Resource

represents an entity modelled and exposed by the *Framework* (3.1.17)

3.1.33

Resource Interface

qualification of the permitted requests on a *Resource* (3.1.31)

3.1.34

Property

significant aspect or *Parameter* (3.1.26) of a *Resource* (3.1.31), including metadata, that is exposed through the *Resource* (3.1.31)

3.1.35

Resource Type

uniquely named definition of a class of *Properties* (3.1.33) and the interactions that are supported by that class

Note 1 to entry: Each *Resource* (3.1.31) has a *Property* (3.1.33) "rt" whose value is the unique name of the *Resource Type* (3.1.34).

3.1.36**Secure OCF Endpoint**

OCF Endpoint (3.1.16) with a secure connection (e.g., CoAPS)

3.1.37**Semantic Tag**

meta-information that provides additional contextual information with regard to the *Resource* (3.1.31) that is the target of a *Link* (3.1.21)

3.1.38**Server**

Device (3.1.13) with the role of providing *Resource* (3.1.31) state information and facilitating remote interaction with its *Resources* (3.1.31)

3.1.39**Sleepy Server**

Server (3.1.37) that will have latency in responding to requests

3.1.40**Unsecure OCF Endpoint**

OCF Endpoint (3.1.16) with an unsecure connection (e.g., CoAP)

3.1.41**Vertical Resource Type**

Resource Type (3.1.34) in a vertical domain specification

Note 1 to entry: An example of a Vertical *Resource Type* (3.1.40) 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, RETRIVE, 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 documenttext 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)?))?)\$ ^([0-9]+W)\$ ^([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])\$ ^([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.

Pattern Name	Pattern	Description
"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 8, which include representations of Resources.

Figure 1 depicts the architecture.

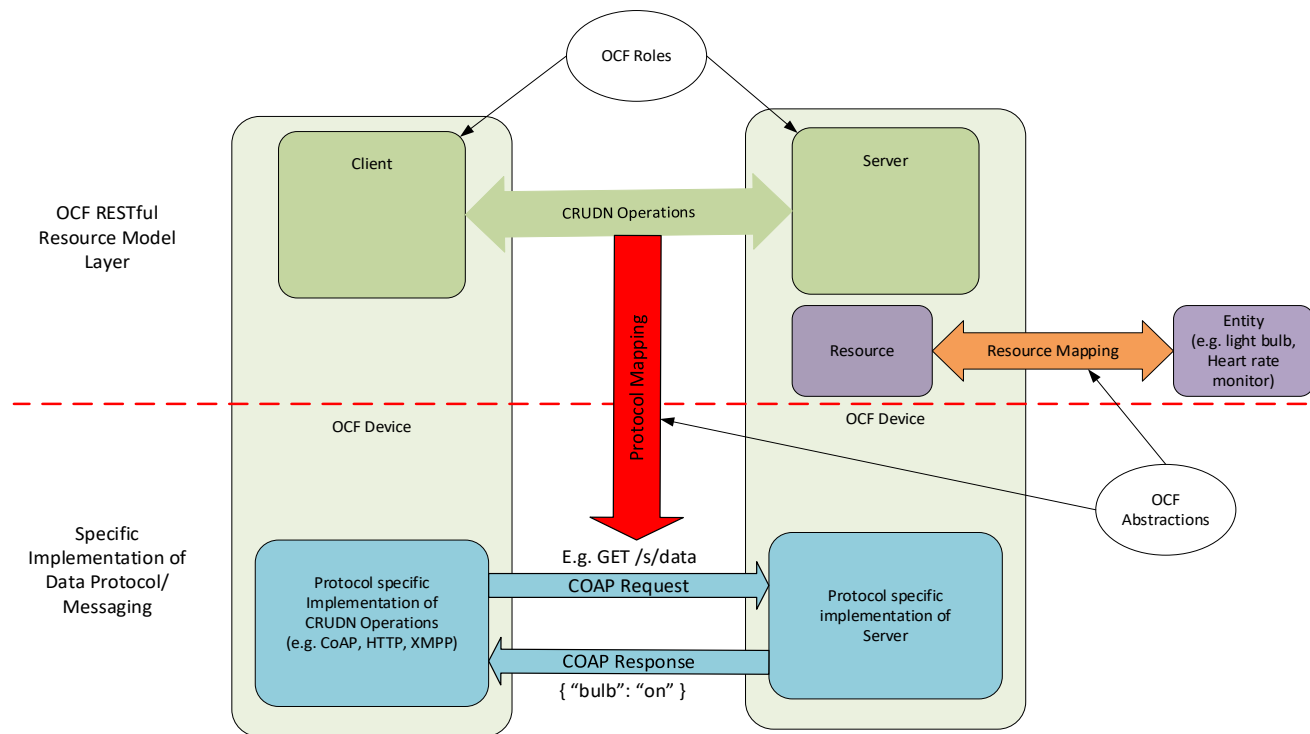


Figure 1 – Architecture - concepts

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 11.4.
- **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.

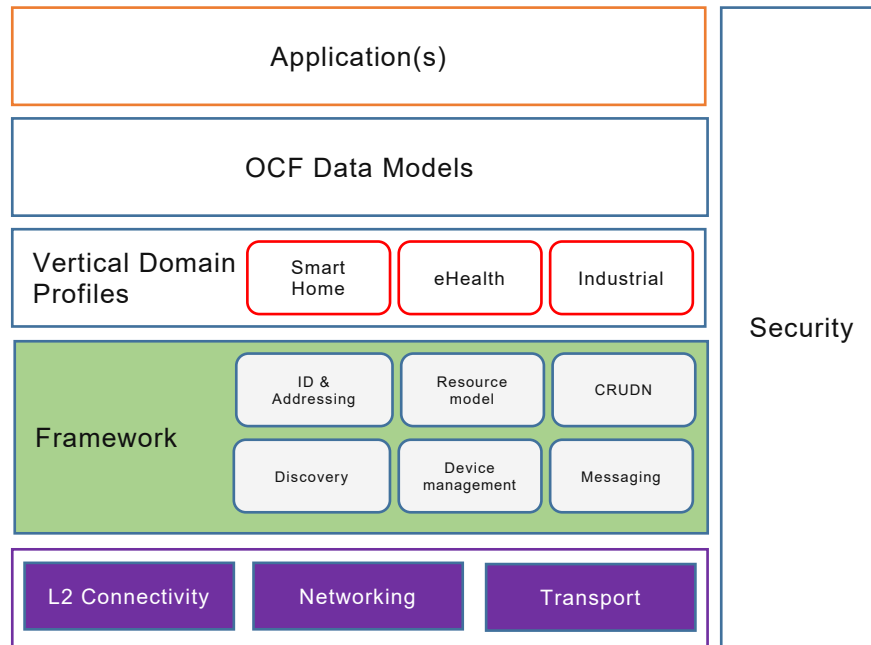


Figure 2 – Functional block diagram

- **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.

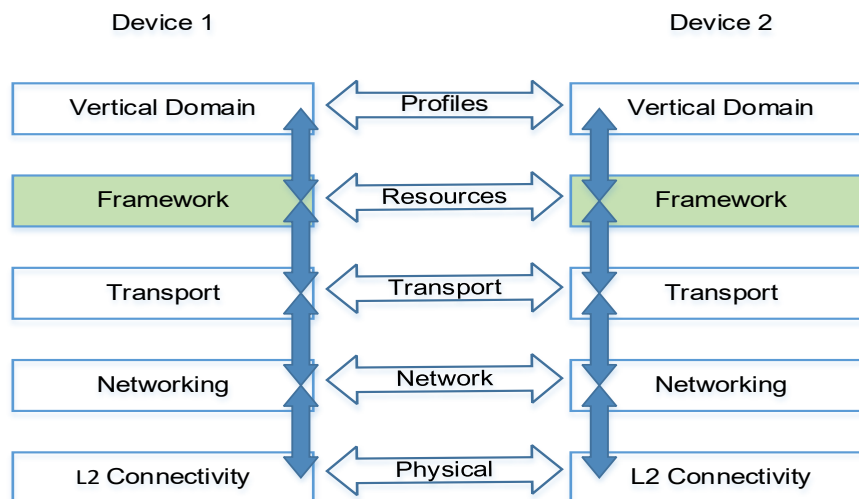


Figure 3 – Communication layering model

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 8.
- *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 13.

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 8) 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:

```
<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.

A query string shall contain a list of "<name>=<value>" segments (aka name-value pair) each separated by a "&" (ampersand). 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.

In addition, the concepts in the Resource model support modelling of the primary artefacts and their relationships to one and another and capture the semantic information required for interoperability in a context. In this way, OCF goes beyond simple protocol interoperability to capture the rich semantics required for true interoperability in Wearable and Internet of Things ecosystems.

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 11.4 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.

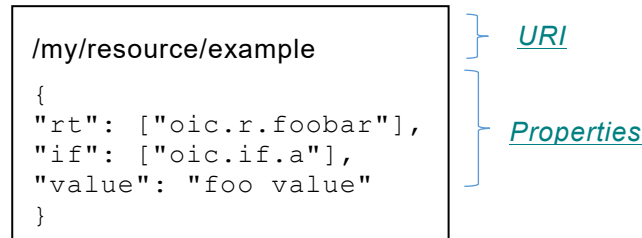


Figure 4 – Example Resource

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 Common Properties defined in this clause may be specified for all Resources. The following Properties are defined as Common Properties:

- Resource Type
- Resource Interface
- Name
- Resource Identity.

The name of a Common Property shall be unique and shall not be used by other Properties. When defining a new Resource Type, its non-common Properties shall not use the name of existing Common Properties (e.g., "rt", "if", "n", "id"). When defining a new "Common Property", it should be ensured that its name has not been used by any other Properties. The uniqueness of a new Common Property name can be verified by checking all the Properties of all the existing OCF defined Resource Types. However, this may become cumbersome as the number of Resource Types grow. To prevent such name conflicts in the future, OCF may reserve a certain name space for Common Property. Potential approaches are (1) a specific prefix (e.g. "oic") may be designated and the name preceded by the prefix (e.g. "oic.psize") is only for Common Property; (2) the names consisting of one or two letters are reserved for Common Property and all other Properties shall have the name with the length larger than the 2 letters; (3) Common Properties may be nested under specific object to distinguish themselves.

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.

The following Common Properties for all Resources are specified in 7.3.2.2 through 7.3.2.6 and summarized as follows:

- *Resource Type* ("rt") – this Property is used to declare the Resource Type of that Resource. Since a Resource could be define by more than one Resource Type the Property Value of the Resource Type Property can 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 Property declares the OCF Interfaces supported by the Resource. The Property Value of the OCF Interface Property can be multi-valued and lists all the OCF Interfaces supported. See 7.3.2.4 for details.
- *Name* ("n") – the Property declares human-readable name assigned to the Resource. See 7.3.2.5.
- *Resource Identity* ("id"): its Property Value 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. See 7.3.2.6 for details.

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.

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 read-write).

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)

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.

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.
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.
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.

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 Common Properties. The baseline OCF Interface shall be defined for all Resource Types. All Resources shall support the baseline OCF Interface.

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

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/scenel?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/scenel?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:

```
[
]
```

7.6.3.3.4 Use 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 realiation 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 choose to restrict the linked Resources to which the request is forwarded by including additional query parameters in the request. The Server should process any additional query parameters in a request that includes "oic.if.b" as selectors for 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 Behavior 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	<pre> /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/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"], </pre>
------------------	--

```

    "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/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 }
      }
    ]
  }
}

```