OIC CORE SPECIFICATION V1.1.0

Part 1

Open Connectivity Foundation (OCF)
admin@openconnectivity.org

Legal Disclaimer

CONTENTS

Figures

Tables

1 Scope

The OCF specifications are divided into two sets of documents:

- Core Specification documents: The Core Specification documents specify 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.
- Vertical Profiles Specification documents: The Vertical Profiles Specification documents specify the OCF profiles to enable IoT usages for different market segments such as smart home, industrial, healthcare, and automotive. The Application Profiles Specification is built upon the interfaces and network security of the OCF core architecture defined in the Core Specification.

This document is the OCF Core specification which specifies the Framework and core architecture.

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

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, August 2008

IETF RFC 1981, *Path MTU Discovery for IP version 6*, August 1996
https://tools.ietf.org/rfc/rfc1981.txt

IETF RFC 2460, *Internet Protocol, version 6 (IPv6), December, 1998*
https://tools.ietf.org/rfc/rfc2460.txt

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999.
http://www.ietf.org/rfc/rfc2616.txt

IETF RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*, June 2004
http://www.ietf.org/rfc/rfc3810.txt

IETF RFC 3986, *Uniform Resource Identifier (URI): General Syntax, January 2005*.
http://www.ietf.org/rfc/rfc3986.txt

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, July 2005
http://www.ietf.org/rfc/rfc4122.txt

IETF RFC 4193, *Unique Local IPv6 Unicast Addresses*, October 2005
http://www.ietf.org/rfc/rfc4193.txt

IETF RFC 4291, IP Version 6 Addressing Architecture, February 2006
http://www.ietf.org/rfc/rfc4291.txt

IETF RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, March 2006
http://www.ietf.org/rfc/rfc4443.txt

IETF RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, September 2007
http://www.ietf.org/rfc/rfc4861.txt

IETF RFC 4862, *IPv6 Stateless Address Autoconfiguration*, September 2007
http://www.ietf.org/rfc/rfc4862.txt

IETF RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, September 2007
http://www.ietf.org/rfc/rfc4944.txt

IETF RFC 5988, *Web Linking: General Syntax, October 2010*
http://www.ietf.org/rfc/rfc5988.txt

IETF RFC 6434, *IPv6 Node Requirements*, December 2011
http://www.ietf.org/rfc/rfc6434.txt

IETF RFC 6455, *The WebSocket Protocol, December 2011*
https://www/ietf.org/rfc/rfc6455.txt

IETF RFC 6690, *Constrained RESTful Environments (CoRE) Link Format*, August 2012
http://www.ietf.org/rfc/rfc6690.txt

IETF RFC 6762, *Multicast DNS* February 2013
http://www.ietf.org/rfc/rfc6762.txt

IETF RFC 6763, *DNS-Based Service Discovery*, February 2013
http://www.ietf.org/rfc/rfc6763.txt

IETF RFC 6775, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, November 2012
http://www.ietf.org/rfc/rfc6775.txt

IETF RFC 7049, *Concise Binary Object Representation (CBOR)*, October 2013
http://www.ietf.org/rfc/rfc7049.txt

IETF RFC 7084, *Basic Requirements for IPv6 Customer Edge Routers*, November 2013
http://www.ietf.org/rfc/rfc7084.txt

IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014
http://www/ietf.org/rfc/rfc7159.txt

IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014
http://tools.ietf.org/html/rfc7252.txt

IETF RFC 7301, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*, July 2014
https://tools.ietf.org/html/rfc7301

IETF RFC 7428, *Transmission of IPv6 Packets over ITU-T G.9959 Networks*, February 2015
http://www.ietf.org/rfc/rfc7428.txt

IETF RFC 7668, *IPv6 over BLUETOOTH(r) Low Energy, October 2015*
https://tools.ietf.org/html/rfc7668

IETF draft-ietf-core-resource-directory-02, CoRE Resource Directory, November 9, 2014
http://www.ietf.org/id/draft-ietf-core-resource-directory-02.txt

IETF draft-ietf-core-observe-16, *Observing Resources in CoAP*, December 30, 2014
http://www.ietf.org/id/draft-ietf-core-observe-16.txt

IETF draft-ietf-core-block-18, *Block-wise transfers in CoAP*, September 14, 2015
http://www.ietf.org/id/draft-ietf-core-block-18.txt

IETF draft-ietf-core-interfaces-02, CoRE Interfaces, November 9, 2014 http://www.ietf.org/id/draft-ietf-core-interfaces-02.txt

IETF draft-tschofenig-core-coap-tcp-tls-04, *A TCP and TLS Transport for the Constrained Application Protocol (CoAP)*, June 10 2015 https://www.ietf.org/id/draft-tschofenig-core-coap-tcp-tls-04.txt

IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00, *Auto-Configuration of a Network of Hybrid Unicast/Multicast DNS-Based Service Discovery Proxy Nodes*, March 5 2015 https://tools.ietf.org/html/draft-ietf-homenet-hybrid-proxy-zeroconf-00

ECMA-4-4, *The JSON Data Interchange Format*, October 2013. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

OCF Security, *Open Connectivity Foundation Security Capabilities*, Version 1.0,

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

3 Terms, definitions, symbols and abbreviations

3.1 Terms and definitions

3.1.1
Client
a logical entity that accesses a Resource on a Server

3.1.2
Collection
a Resource that contains zero or more Links

3.1.3
Configuration Source

a Cloud or Service Network or a local read-only file which contains and provides configuration related information to the Devices

3.1.4
Core Resources
those Resources that are defined in this specification

3.1.5
Default Interface
an Interface used to generate the response when an Interface is omitted in a request

3.1.6
Device
a logical entity that assumes one or more Roles (e.g., Client, Server)

Note 1 to entry: More than one Device can exist on a physical platform.

3.1.7
Device Type
a uniquely named definition indicating a minimum set of Resource Types that a Device supports

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

3.1.8
Entity

an element of the physical world that is exposed through an Device

Note 1 to entry: Example of an entity is an LED.

3.1.9
Framework
a set of related functionalities and interactions defined in this specification, which enable interoperability across a wide range of networked devices, including IoT

3.1.10
Links
extends typed web links as specified in IETF RFC 5988

3.1.11
Non-OCF Device
A device which does not comply with the OCF Device requirements

3.1.12
Notification
the mechanism to make a Client aware of resource state changes in an Resource

3.1.13
Observe
the act of monitoring a Resource by sending a RETRIEVE request which is cached by the Server hosting the Resource and reprocessed on every change to that Resource

3.1.14
Parameter
an element that provides metadata about a Resource referenced by the target URI of a Link

3.1.15
Partial UPDATE
an UPDATE request to a Resource that includes a subset of the Properties that are visible via the Interface being applied for the Resource Type

3.1.16
Platform
a physical device containing one or more Devices

3.1.17
Remote Access Endpoint (RAE) Client
a Client which supports XMPP functionality in order to access a Server from a remote location

3.1.18
Remote Access Endpoint (RAE) Server
a Server which supports XMPP and can publish its resource(s) to an XMPP server in the Cloud, thus becoming remotely addressable and accessible

Note 1 to entry: An RAE Server also supports ICE/STUN/TURN.

3.1.19
Resource
represents an Entity modelled and exposed by the Framework

3.1.20
Resource Directory
a set of descriptions of resources where the actual resources are held on Servers external to the Device hosting the Resource Directory, allowing lookups to be performed for those resources

Note 1 to entry: This functionality can be used by sleeping Servers or Servers that choose not to listen/respond to multicast requests directly.

3.1.21
Resource Interface
a qualification of the permitted requests on a Resource

3.1.22
Resource Property
a significant aspect or parameter of a resource, including metadata, that is exposed through the Resource

3.1.23
Resource Type
a uniquely named definition of a class of Resource Properties and the interactions that are supported by that class

Note 1 to entry: Each Resource has a Property “rt” whose value is the unique name of the Resource Type.

3.1.24
Scene
a static entity that stores a set of defined Resource property values for a collection of Resources

Note 1 to entry: A Scene is a prescribed setting of a set of resources with each having a predetermined value for the property that has to change.

3.1.25
Scene Collection
a collection Resource that contains an enumeration of possible Scene Values and the current Scene Value

Note 1 to entry: The member values of the Scene collection Resource are Scene Members.

3.1.26
Scene Member
a Resource that contains mappings of Scene Values to values of a property in the resource

3.1.27
Scene Value
a Scene enumerator representing the state in which a Resource can be

3.1.28
Server
a Device with the role of providing resource state information and facilitating remote interaction with its resources

Note 1 to entry: A Server can be implemented to expose non-OCF Device resources to Clients (section 5.5)

3.2 Symbols and abbreviations

3.2.1
ACL
Access Control List

Note 1 to entry: The details are defined in OCF Security.

3.2.2
CBOR
Concise Binary Object Representation

3.2.3
CoAP
Constrained Application Protocol

3.2.4
EXI
Efficient XML Interchange

3.2.5
IRI
Internationalized Resource Identifiers

3.2.6
ISP
Internet Service Provider

3.2.7
JSON
JavaScript Object Notation

3.2.8
mDNS
Multicast Domain Name Service

3.2.9
MTU
Maximum Transmission Unit

3.2.10
NAT
Network Address Translation

3.2.11
OCF
Open Connectivity Foundation

the organization that created this specification

3.2.12
URI
Uniform Resource Identifier

3.2.13
URN
Uniform Resource Name

3.2.14
UTC
Coordinated Universal Time

3.2.15
UUID
Universal Unique Identifier

3.2.16
XML
Extensible Markup Language

3.3 Conventions

In this specification 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.

3.4 Data types

Table 1 contains the definitions of data types used to describe a Resource. The data types are derived from JSON values as defined in ECMA-4-4. However a Resource can overload a JSON defined value to specify a particular subset of the JSON value. These specific data types are defined in Table 1. The data types can be adapted for a particular usage, for example the length of a string can be changed for a specific usage.

Table 1. Data type definition

Name	JSON value	JSON format value	Description
boolean	false true	n/a	Binary-value {0, 1}.
BSV	string	bsv	A blank (i.e. space) separated list of values encoded within a string. The value type in the BSV is described by the property where the BSV is used. For example a BSV of integers.

CSV	string	csv	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.
date	string	date-time	As defined in ISO 8601. The format is restricted to [yyyy]-[mm]-[dd].
datetime	string	date-time	As defined in ISO 8601.
enum	enum	n/a	Enumerated type.
float	number	float	Signed IEEE 754 single precision float value.
integer	number	integer	Signed 32 bit integer.
json	object/array	n/a	A data represented using a JSON element which could be an object or array as defined in ECMA-4-4. The JSON object or array needs to be described by means of a JSON schema.
string	string	n/a	UTF-8 character string shall not exceed a max length of 64 octets unless otherwise specified for a Property value in this specification.
time	string	time	As defined in ISO 8601 but restricted to UTC with a trailing “Z”. The format is [hh]:[mm]:[ss]Z.
URI	string	uri	A uniform resource identifier (URI) is a string of characters used to identify a resource according to IETF RFC 3986. The URI value shall not exceed a max length of 256 octets (bytes).
UUID	string	uuid	An identifier formatted according to IETF RFC 4122.

4 Document conventions and organization

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 behavior 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 behavior 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 specification, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current specification 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 specification.

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.

5 Architecture

5.1 Overview

The architecture 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 sections 5.2 through 5.5 respectively. Section 5.2 presents the guiding principles for OCF operations. Section 5.3 defines the functional block diagram and Framework. Section 5.4 provides an example scenario with roles. Section 5.5 provides an example scenario of bridging to non- OCF ecosystem.

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 (section 7.7) 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 (section 5.4). 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, Read, Update, Delete, Notify (CRUDN) defined in section 8, which include representations of Resources.

Figure 1 depicts the architecture.

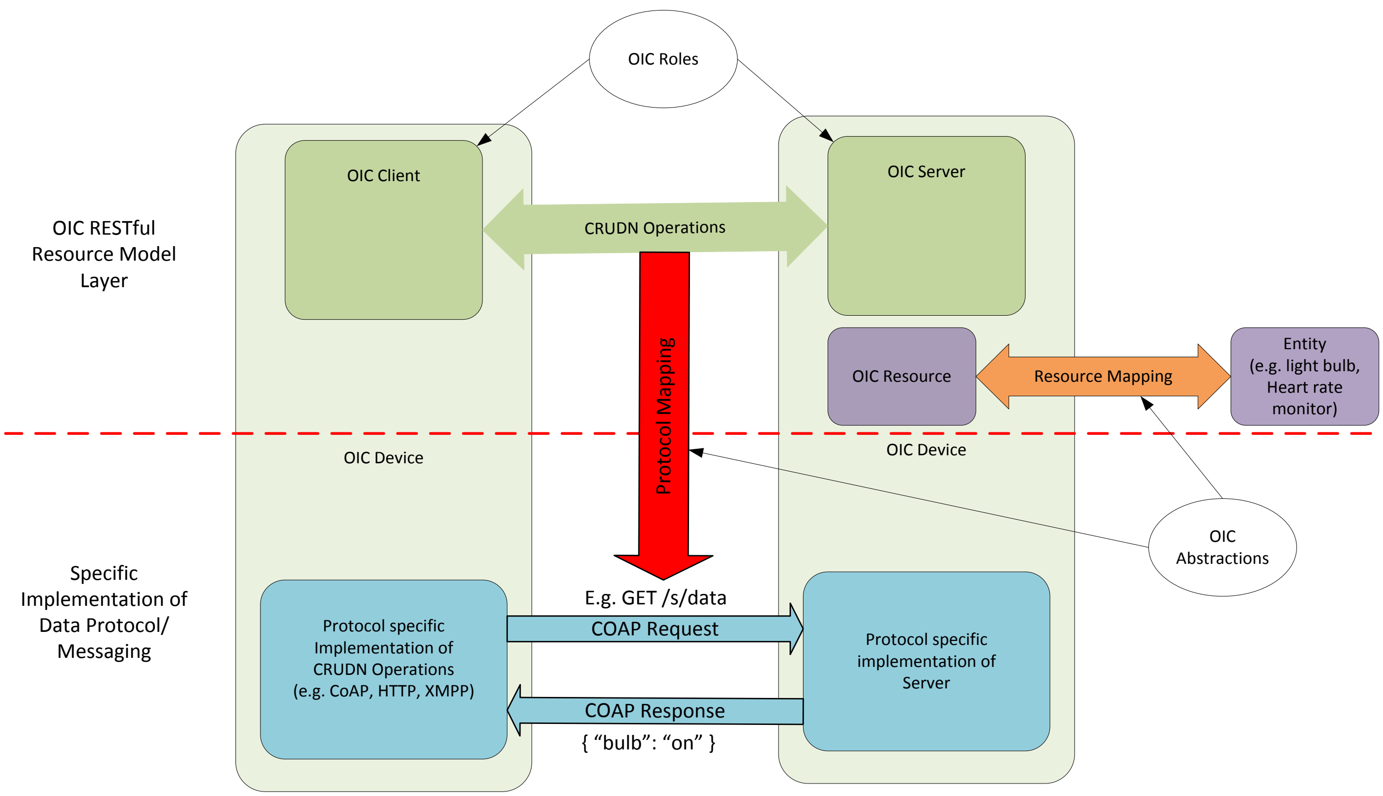

Figure 1: Architecture - concepts

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 behavior.

- 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 section 12.
- 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 and listed below.

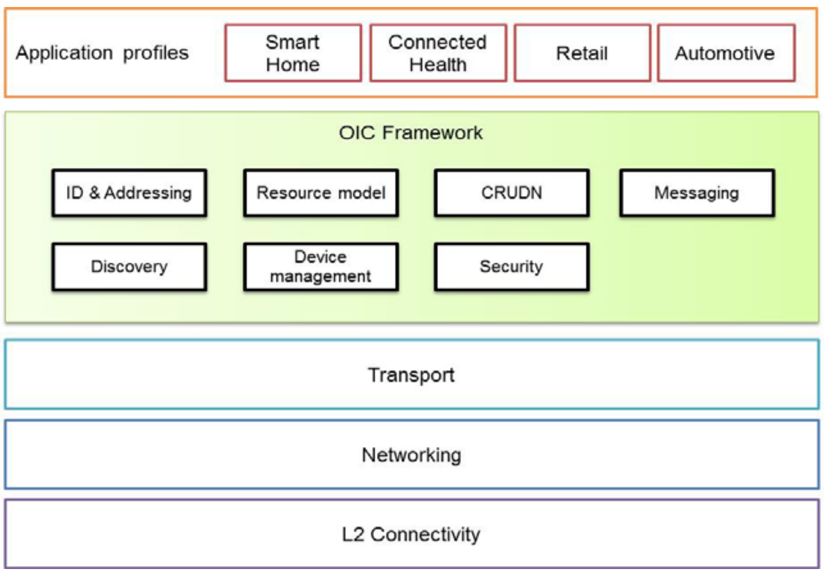

Figure 2: Functional block diagram

- **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 specification. The functional block is the source of requests and responses that are the content of the communication between two Devices.
- **Application profile**: Provides market segment specific data model and functionalities, e.g., smart home data model and functions for the smart home market segment.

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

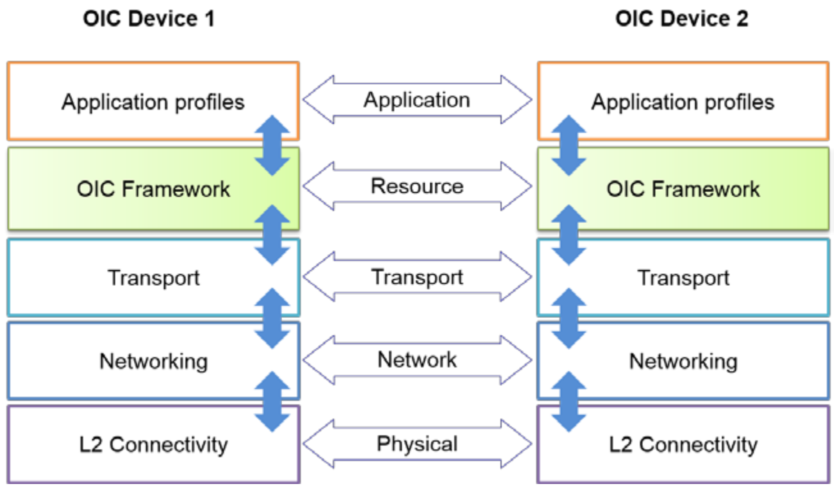

Figure 3: Communication layering model

5.3.1 Framework

Framework consists of functions which provide core functionalities for operation.

1) **Identification and addressing.** Defines the identifier and addressing capability. The Identification and addressing function is defined in section 6.
2) **Discovery**. Defines the process for discovering available
   a) Devices (Endpoint Discovery in section 10) and
   b) Resources (Resource discovery in section 11.3)
3) **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 section 7.
4) **CRUDN**. Provides a generic scheme for the interactions between a Client and Server as defined in section 8.
5) **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 section 12.
6) **Device management.** Specifies the discipline of managing the capabilities of a Device, and includes device provisioning and initial setup as well as device monitoring and diagnostics. The device management function is defined in section 11.5.
7) **Security.** Includes authentication, authorization, and access control mechanisms required for secure access to Entities. The security function is defined in section 13.

5.4 Example Scenario with roles

Interactions are defined between logical entities known as Roles. Three roles are defined: Client, Server and Intermediary.

Figure 4 illustrates an example of the Roles in a scenario where a smart phone sends a request message to a thermostat; the original request is sent over HTTP, but is translated into a CoAP request message by a gateway in between, and then delivered to the thermostat. In this example, the smart phone takes the role of a Client, the gateway takes the role of an Intermediary and the thermostat takes the role of a Server.

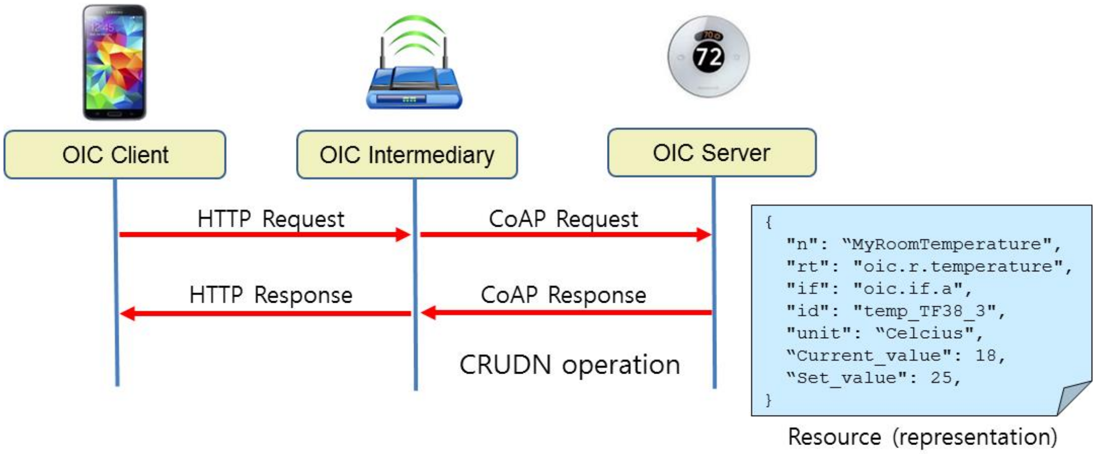

Figure 4: Example illustrating the Roles

5.5 Example Scenario: Bridging to Non- OCF ecosystem

The use case for this scenario is a display (like a wrist watch) that is used to monitor a heart rate sensor that implements a protocol that is not OCF supported.

Figure 5 provides a detailed logical view of the concepts described in Figure 1.

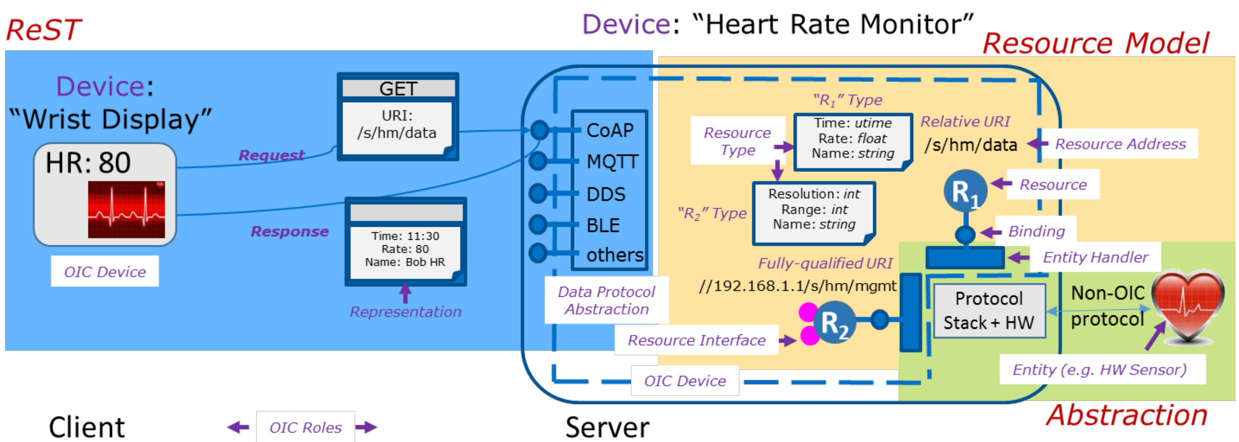

Figure 5: Framework - Architecture Detail

The details may be implemented in many ways, for example, by using a Server with an entity handler to interface directly to a non- OCF device as shown in Figure 6.

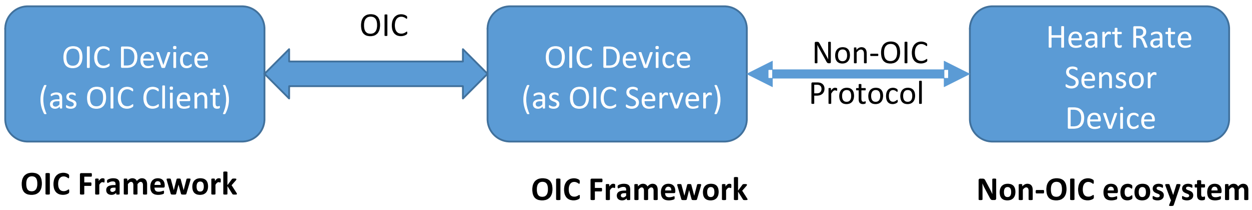

Figure 6: Server bridging to Non- OCF device

On start-up the Server runs the entity handlers which discover the non- OCF systems (e.g., Heart Rate Sensor Device) and create resources for each device or functionality discovered. The entity handler creates a Resource for each discovered device or functionality and binds itself to that Resource. These resources are made discoverable by the Server.

Once the resources are created and made discoverable, then the Display Device can discover these resources and operate on them using the mechanisms described in this specification. The requests to a resource on the Server are then interpreted by the entity handler and forwarded to the non- OCF device using the protocol supported by the non-OCF device. The returned information from the non- OCF device is then mapped to the appropriate response for that resource.

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* shall unambiguously and uniquely identify an element in a context or domain. The context or domain may be determined by the use or the application. The identifier should be immutable over the lifecycle of that element and shall be unique 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 (section 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 section 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 (section 8) on a resource. Also the mapping of these to transport protocols, e.g., CoAP is described.

6.2 Identification

An identifier shall be unique within the context or domain of use. There are many schemes that may be used to generate an identifier that has the required properties. The identifier may be context-specific in that the identifier is expected to be and guaranteed to be unique only within that context or domain. Identifier may also be context- independent where these identifiers are guaranteed to be unique across all contexts and domains both spatially and temporally. The context-specific identifiers could be defined by simple schemes like monotonic enumeration or may be defined by overloading an address or name, for example an IP address may be an identifier within the private domain behind a gateway in a smart home. On the other hand, context-independent identifiers require a stronger scheme that derives universally unique identities, for example any one of the versions of Universally Unique Identifiers (UUIDs). Context independent identifier may also be generated using hierarchy of domains where the root of the hierarchy is identified with a UUID and sub-domains may generate context independent identifier by concatenating context-specific identifiers for that domain to the context-independent identifier of their parent.

6.2.1 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:

oic://<**Authority**>/<**Path**>?<**Query**>

A description of values that each component takes is given below.

The *scheme* for the URI is ‘oic’. The ‘oic’ 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 ‘oic’ 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 to convert from a transport protocol URI to an OCF URI before handing over to the resource model layer on the receiver.

If the authority is the local Device, then ‘oic’ may be used as the authority.

The usual form of the authority is

<host>:<port>, where <host> is the name or endpoint network address and <port> is the network port number. The <host> may be provided as follows:

- For IP networks, the hostname or IP address of <authority>
- For non-IP networks, the name or appropriate identifier.
- If the <authority> is the Device that hosts the resource then the keyword ‘`oic` ‘ may be used for the <host>.

The *path* shall be unique string that unambiguously identifies or references a resource within the context of the Server. In this version of the specification, 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 shall use that path as an opaque string and shall 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 ';' (semicolon). 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 interface, based on some rules or with respect to different roots or bases.

Use of URI:

The absolute path reference of a URI is to be treated as an opaque string and a client shall 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.

6.4 Network addressing

The following are the addresses used in this specification:

- **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, Interfaces, Collections and Links. In addition, the general mechanisms are Create, Update, Retrieve, 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 OCF 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 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 section 12 for transport protocols supported.

The RAML definitions used in this document are normative. This also includes that all defined JSON payloads shall comply with the indicated JSON schema. See Annex D for Resource Types defined in this specification.

7.2 Resource

A Resource shall be defined by one or more Resource Type(s) – see Annex D 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 section 6. The URI may be assigned by the Authority at the creation of the Resource or may be pre-defined by the specification of the Resource Type.

```
/my/resource/example
{
"rt": "oic.r.foobar",
"if": "oic.if.a",
"value": "foo value"
}
```

URI

Properties

Figure 7: Example of a Resource

Core Resources are the Resources defined in this specification to enable functional interactions as defined in section 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 section 3.4 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, 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.

A Property may be used in the query part of an URI as one criterion for selection of a particular Resource. This is done by declaring the Property (i.e. <Property Name> = <desired Property Value>) as one of the segments of the query. In this version of the specification, only ASCII strings are permitted in query filters, and NUL characters are disallowed in query filters. This means that only property values with ASCII characters can be matched in a query filter. The Resource is selected when all the declared Properties in the query match the corresponding Properties in the full Representation of the target Resource. The full Representation is the snapshot that includes the union of all Properties in all Resource Types that define the target Resource. If the Property is declared in the “filter” segment of the query then the declared Property is matched to the Representation defined by the Interface to isolate certain parts of that Representation.

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 with the above fields for all common properties are defined in section 7.3.2.

7.3.2 Common Properties

7.3.2.1 Introduction

The Common Properties defined in this section may be specified for all Resources. The following Properties are defined as Common Properties: “Resource Type”, “Resource Interface”, “Name”, and “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 following Common Properties for all Resources are specified in section 7.3.2.2 through section 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. For example: “rt”: [“oic.wk.d”, “oic.d.airConditioner”] declares that the Resource containing this Property is defined by either the “oic.wk.d” Resource Type or the “oic.d.airConditioner” Resource Type. See section 7.3.2.3 for details.
- Interface ("if") – this Property declares the Interfaces supported by the Resource. The Property Value of the Interface Property can be multi-valued and lists all the Interfaces supported. See section 7.3.2.4 for details.
- Name ("n") – the Property declares “human-readable” name assigned to the Resource. See section 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 section 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 specification:

- **Property Name**– the key in "key=value" pair. Property Name is case sensitive and its data type is "string" but only ASCII characters are permitted, and embedded NUL characters are not permitted.
- **Property Value** – the value in "key=value" pair. Property Value is case sensitive when its data type is "string". Any enum values shall be ASCII only.

7.3.2.3 Resource Type

Resource Type Property is specified in Section 7.4.

7.3.2.4 Interface

Interface Property is specified in Section 7.5.

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			R	no	Human understandable name for the resource; may be set locally or remotely (e.g., by a user)

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. 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	Implementation Dependent		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 Section 7.3.2.3 or in a Link using the Resource Type Parameter.

A Resource Type may either be pre-defined (Core Resource Types in this specification and vertical Resource Types in vertical domain specifications) 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 (like in documentation) or be defined explicitly using a meta-language which may be downloaded and used by APIs or applications. OCF has adopted RAML and JSON Schema as the specification method for OCF's RESTful interfaces and Resource definitions. OCF defined Interfaces and Resource Types are specified using RAML and JSON schema (respectively).

Every Resource Type shall be identified with a Resource Type ID which shall be a lower case string with segments separated by a "." (dot). 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 an 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. The Resource Type ID may also be a reference to an authority similar to IANA that may be used to find the definition of a Resource Type.

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. 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	json	Array of Resource Type IDs		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). A lower case string that has segments separated by a '.' (dot); each segment may represent a name space and in that case later segments (L -> R) would represent sub-name spaces; Implementations shall use these opaquely and use case sensitive string matches.
- **Resource Interfaces** – list of the interfaces that may be supported by the resource type.
- **Resource 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 specification 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 provide 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)	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**			**R**	**yes**	**Resource type**
Interface	**if**	**array**			**R**	**yes**	**Interface**
Foo value	**value**	**string**			**R**	**yes**	**Foo value**

An instance of the foobar resource type is as shown below

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

An example schema for the foobar resource type is shown below

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "rt": {"type": "string"},
    "if": {"type": "string"},
    "value": {"type": "string"}
  },
  "required": ["rt", "if", "value"]
}
```

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 section 11.3.4.

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 (in vertical domain specifications) 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 Interface

7.6.1 Introduction

An 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 Interface defines the context for requests and responses on a Resource. Therefore, the same request to a Resource when targeted to different Interfaces may result in different responses.

An Interface may be defined by either this specification (a Core Interface), the OCF vertical domain specifications (a “vertical Interface) or manufacturers, end users or developers of Devices (a “vendor-defined Interface”).

The Interface Property lists all the Interfaces the Resource support. All resources shall have at least one Interface. The Default Interface shall be defined by an OCF specification and inherited from the resource type definition. The Default Interface associated with all Resource Types defined in this specification shall be the supported Interface listed first within the applicable enumeration in the definition of the Resource Type (see Annex D). All Default Interfaces specified in an OCF specification shall be mandatory.

In addition to any OCF specification defined interface, all Resources shall support the Baseline Interface (oic.if.baseline) as defined in section 7.6.3.2.

When an Interface is to be selected for a Request, it shall be specified as query parameter in the URI of the Resource in the Request message. If no query parameter is specified, then the Default Interface shall be used. If the selected Interface is not one of the permitted Interfaces on the Resource then selecting that Interface is an error.

An Interface may accept more than one media type. An 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 Interface shall have at least one media type.

7.6.2 Interface Property

Table 7. Resource Interface Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface	if	json	Array of Dot separated strings		R	yes	Property to declare the Interfaces supported by a Resource.

The Interfaces supported by a Resource shall be declared using the Interface Common Property (Table 7) as "`if=<array of Interfaces>`". The Property Value of an 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 Interface Property Value, is reserved for OCF-defined Interfaces. The Interface Property Value may also be a reference to an authority similar to IANA that may be used to find the definition of an Interface. A Resource Type shall support one or more of the Interfaces defined in section 7.6.3.

7.6.3 Interface methods

7.6.3.1 Overview

The OCF -defined Interfaces are listed in the table below:

Table 8. OCF standard Interfaces

Interface	Name	Applicable Methods	Description
baseline	oic.if.baseline	RETRIEVE, UPDATE	The baseline Interface defines a view into all Properties of a Resource including the Meta Properties. This Interface is used to operate on the full Representation of a Resource.
links list	oic.if.ll	RETRIEVE	The ‘links list’ Interface provides a view into Links in a Collection (Resource). Since Links represent relationships to other Resources, the links list 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 Interface to allow discovery of Resource “hosted” on a Device.
batch	oic.if.b	RETRIEVE, UPDATE	The batch 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	The read-only Interface exposes the Properties of a Resource that may be ‘read’. This Interface does not provide methods to update Properties or a Resource and so can only be used to ‘read’ Property Values.
read-write	oic.if.rw	RETRIEVE, UPDATE	The read-write Interface exposes only those Properties that may be both ‘read’ and “written” and provides methods to read and write the Properties of a Resource.
actuator	oic.if.a	CREATE, RETRIEVE, UPDATE	The actuator Interface is used to read or write the Properties of an actuator Resource.
sensor	oic.if.s	RETRIEVE	The sensor Interface is used to read the Properties of a sensor Resource.

7.6.3.2 Baseline Interface

7.6.3.2.1 Overview

The Representation that is visible using the “baseline” Interface includes all the Properties of the Resource including the Common Properties. The “baseline” Interface shall be defined for all Resource Types. All Resources shall support the “baseline” Interface.

The “baseline” Interface is selected by adding if=oic.if.baseline to the list of query parameters in the URI of the target Resource. For example: GET /oic/res?if=oic.if.baseline.

7.6.3.2.2 Use of RETRIEVE

The “baseline” Interface is used when a Client wants to retrieve all Properties of a Resource. The Client includes the URI query parameter definition "?if=oic.if.baseline" in a RETRIEVE request. When this query parameter definition is included 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 Interface is shown below:


```
{
"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

Using the baseline Interface, all Properties of a Resource may be modified using an UPDATE request with a list of Properties and their desired values.

7.6.3.3 Link List Interface

7.6.3.3.1 Overview

The links list Interface provides a view into the list of Links in a Collection (Resource). The Representation visible through this Interface has only the Links defined in the Property Value of the “links” Property – so this Interface is used to manipulate or interact with the list of Links in a Collection. The Links list may be RETRIEVEd using this Interface.

The Interface definition and semantics are given as follows:

- The links list Interface name shall be “oic.if.ll”.
- If specified in a request (usually in the request header), the serialization in the response shall be in the format expected in the request.
- In response to a RETRIEVE request on the “links list” Interface, the URIs of the referenced Resources shall be returned as a URI reference.
- If there are no links present in a Resource, then an empty list shall be returned.
- The Representation determined by this Interface view only includes the Property Value of the “links” Property.

7.6.3.3.2 Example: “links list” Interface

Example: Request to a Collection

Request to RETRIEVE the Links in room (the Links could be referencing lights, fans, electric sockets etc)	`GET oic://<devID>/a/room/1?if=oic.if.ll`

7.6.3.4 Batch Interface

7.6.3.4.1 Overview

The batch Interface is used to interact with a collection of Resources using a single/same Request. The batch Interface supports methods of Resources in the Links of the Collection, and can be used to RETRIEVE or UPDATE the Properties of the "linked" Resources with a single Resource representation.

The batch Interface selects a view into the Links in a Collection – the Request is sent to all the Links in this view with potential modifications defined in the Parameters of the Link

The batch Interface is defined as follows:

- The batch Interface name shall be "oic.if.b"
- A Resource with a batch Interface has Links that have Resource references that may be URIs (fully qualified for remote Resources) or relative references (for local Resources).
- If the Link to a Resource does not specify an Interface to use (using the "bp" Link parameter), then the Request shall be forwarded to the Default Interface of the referenced Resource. If the "bp" specifies a query using the "q" key then that query shall be used in the query parameter of the URI formed from the Reference so as to select that Interface in the target Resource. (See "Link" section for more information on "bp" Parameter)
- The original request is modified to create new requests targeting each of the targets in the Resource Links by substituting the URI in the original request with the URI of the target Resource in the Link. The payload in the original request is replicated in the payload of the new Requests.
- All the Responses from the "linked" Resources shall be aggregated into single Response to the Client. The Server may timeout the Response to a time window (if a time window has been negotiated with the Client then the Server shall not timeout within that window; in the absence of negotiated window, the Server may choose any appropriate window based on conditions). If the target Resources cannot process the new request, an empty response or error response shall be returned. These empty/error Responses shall be included in aggregated Response to the original Client Request.
- The aggregate Response is an array of objects with individual responses. Each response in the aggregate shall include at least two items: (1) the URI (fully qualified) as "href": <URI> and (2) the Representation in the Response declared using the keyword "rep" as the key i.e. "rep": { <Representation in individual Response> }.
- The Client may choose to restrict the list of Links to which the Request is forwarded by providing a "filter" in the URI of the Collection to which this original 'batch' Interface Request is made.
- The Representation in the Link-specific Request may not match the Representation from the view exposed by the Interface on the target Resource. In such cases, UPDATE using 'PUT' method will usually fail and so UPDATE using 'POST' method would be appropriate – in this case the 'subset' semantics apply where Properties in the Request which match Properties in the Resource view exposed shall be modified in the target Resource if the Property is writeable.
- A Device that supports the 'batch' Interface shall implement both the Client and Server Roles.

7.6.3.4.2 Examples: Batch Interface

Example 1

Resources	`/a/room/1`
	`{`
	`  "rt": ["acme.room"],`
	`  "if": ["oic.if.baseline", "oic.if.b"],`

<table>
<tr><td></td><td><pre>
  "color": "blue",
  "dimension": "15bx15wx10h",
  "links": [
     {"href": "/the/light/1", "rt"”: ["acme.light"], "if":
["oic.if.a", "oic.if.baseline"], "ins": 1},
     {"href": "/the/light/2", "rt": ["mycorp.light"], if:
["oic.if.a" , "oic.if.baseline"], "p": {"bm": 2, "sec": true, "port": 33270},
"ins": 2},
     {"href": "/my/fan/1", "rt": ["hiscorp.fan"], if:
["oic.if.baseline", "oic.if.a"], "ins": 3 },
     {"href": "/his/fan/2", "rt": ["hiscorp.fan"], if:
["oic.if.baseline", "oic.if.a"], "ins": 4, "bp": {"q":
"if=oic.if.a"}}
  ]
}

/the/light/1
{
  "rt": ["acme.light"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "state": 0,
  "colortemp": "2700K"
}

/the/light/2
{
  "rt": ["mycorp.light"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "state": 1,
  "color": "red"
}

/my/fan/1
{
  "rt": ["hiscorp.fan"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "state": 0,
  "speed": 10
}

/his/fan/2
{
  "rt": ["hiscorp.fan"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "state": 0,
  "speed": 20
}
</pre></td></tr>
<tr><td>Use of batch</td><td>Request: GET /a/room/1?if=oic.if.b

Becomes the following individual responses issued by the Device in the Client role

GET /the/light/1 (NOTE: Uses the default Interface: ‘sensor’)

GET /the/light/2 (NOTE: Uses the default Interface: ‘sensor’)</td></tr>
</table>

	GET /my/fan/1 (NOTE: Uses the default Interface: ‘baseline’)
	
	GET /his/fan/2?if=oic.if.a (NOTE: Interface from “bp” Link parameter: ‘actuator’)
	
	Response:
	
	[
	{
	"href": "oic://<devID>/the/light/1",
	"rep": {"state": 0, "colortemp": "2700K"}
	},
	{
	"href": "oic://<devID>/the/light/2",
	"rep": {"state": 1, "color": "red" }
	},
	{
	"href": "oic://<devID>/my/fan/1",
	"rep": { "rt": ["hiscorp.fan"], "if": ["oic.if.a", "oic.if.baseline"], "state": 0, "speed": "10" }
	},
	{
	"href": "oic://<devID>/his/fan/2",
	"rep": { "state": 0, "speed": "20" }
	}
	]

Use of batch
 	
(UPDATE has POST semantics)
	UPDATE /a/room/1?if=oic.if.b
	{
	"state": 1
	}
	
	becomes
	
	UPDATE /the/light/1 { "state": 1 }
	UPDATE /my/fan/1 { "state": 1 }
	UPDATE /his/fan/2?if=oic.if.a { "state": 1 }
	
	This turns on all the lights (except /the/light/1 Resource) and fans on in the room since all the Resources have “state” as a Property. /the/light/1 has the ‘sensor’ interface as default and so POST is not supported for ‘sensor’ Interface (the Device hosting /a/room/1 does not send this Request)

Use of batch
 	
(UPDATE has POST semantics)
	UPDATE /a/room/1?if=oic.if.b
	{
	"state": 1,
	"color": "blue"
	}
	
	This turns on all the lights (except /the/light/1 Resource) and fans in the room but also sets the color of /the/light/2 to “blue”

Example that further shows the “links list” and “batch” interface

<table>
<tr><td>Example</td><td><pre>/myexample
{
   "rt": ["oic.r.foo"],
    "if": ["oic.if.baseline", "oic.if.ll"],
    "links": [
         {"href": "/acme/switch", "di": "<deviceID1>", "rt":
["oic.r.switc.binary"], "if": ["oic.if.a"]},
         {"href": "oic://<deviceID1>/acme/fan", "rt":
["oic.r.fan"], "if": ["oic.if.a"] }
     ]
}</pre></td></tr>
<tr><td>Use of Baseline</td><td><pre>GET /myexample?if=oic.if.baseline   will return

{
   "rt": ["oic.r.foo"],
    "if": ["oic.if.baseline", "oic.if.ll"],
    "links": [
         {"href": "/acme/switch", "di": "<deviceID1>", "rt":
["oic.r.switc.binary"], "if": ["oic.if.a"]},
         {"href": "oic://<deviceID1>/acme/fan", "rt":
"oic.r.fan", "if”: "oic.if.a"}
     ]
}</pre></td></tr>
<tr><td>Use of Links List</td><td><pre>GET /myexample?if=oic.if.ll.   will return

 [
  {"href": "/acme/switch", "di": "<deviceID1>", "rt":
["oic.r.switc.binary"], "if": ["oic.if.a"]},
   {"href": "oic://<deviceID1>/acme/fan", “rt”:
["oic.r.fan"], "if": ["oic.if.a"]}
 ]</pre></td></tr>
</table>

7.6.3.5 Actuator Interface

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

- The actuator Interface name shall be “oic.if.a”
- The actuator Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable JSON; the actuator interface may also expose in the Resource Representation optional Properties as defined by the applicable JSON schema that are implemented by the target Device.

For the following Resource

NOTE: “prm” is the Property name for ‘parameters’ Property

```
/a/act/heater
{
   "rt": ["acme.gas"],
   "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
   "prm": {"sensitivity": 5, "units": "C", "range": "0 .. 10"},
   "settemp": 10,
   "currenttemp" : 7
```



```
}
```

Figure 8: Example - "Heater" Resource (for illustration only)

NOTE: The example here is with respect to Figure 8

1. Retrieving values of an actuator

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

Response:
           {
   "prm": {"sensitivity": 5, "units": "C", "range": "0 .. 10"},
   "settemp": 10,
   "currenttemp" : 7
           }
```

2. Correct use of actuator:

```
Request: POST /a/act/heater?if="oic.if.a"
           {
             "settemp": 20
           }
Response:
           {
             Ok
           }
```

3. Incorrect use of actuator

```
Request: POST /a/act/heater?if="oic.if.a"
           {
              "if": "oic.if.s"    ← this is visible through baseline
Interface
           }
Response:
           {
              Error
           }
```

Figure 9: Example - Actuator Interface

- A RETRIEVE request using this Interface shall return the Representation for this Resource subject to any query and filter parameters that may also exist
- An UPDATE request using this Interface shall provide a payload or body that contains the Properties that will be updated on the target Resource.

7.6.3.6 Sensor Interface

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

- The sensor Interface name shall be “oic.if.s”
- The sensor Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable JSON; the sensor interface may also expose in the Resource

Representation optional Properties as defined by the applicable JSON schema that are implemented by the target Device.

- A RETRIEVE request using this Interface shall return this Representation for the Resource subject to any query and filter parameters that may also exist
-

NOTE: The example here is with respect to Figure 8

1. Retrieving values of sensor

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

Response:
           {
             "currenttemp": 7
           }
```

2. Incorrect use of sensor

```
Request: PUT /a/act/heater?if="oic.if.s" ← PUT is not allowed
           {
             "settemp": 20    ← this is possible through actuator Interface
           }
Response:
           {
             Error
           }
```

3. Incorrect use of sensor

```
Request: POST /a/act/heater?if="oic.if.s" ← POST is not allowed
           {
              "currenttemp": 15    ← this is possible through actuator
Interface
           }
Response:
           {
              Error
           }
```

7.6.3.7 Read-only Interface

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

7.6.3.8 Read-write Interface

The read-write Interface exposes only the Properties that may be “read” and “written”. The “read-only” Properties shall not be included in Representation for the “read-write” Interface. This is a generic Interface to support “reading” and “setting” Properties in a Resource. The applicable methods that can be applied to a Resource are RETRIEVE and UPDATE only. An attempt by a

Client to apply a method other than RETRIEVE or UPDATE to a Resource shall be rejected with an error response code.

7.7 Resource representation

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

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

7.8 Structure

7.8.1 Introduction

In many scenarios and contexts, the Resources may have either an implicit or explicit structure between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The Framework provides the means to model and map these structures and the relationships among Resources. The primary building block for resource structures in Framework is the collection. A collection represents a container, which is extensible to model complex structures.

Resource RelationshipsResource relationships are expressed as Links. A Link embraces and extends typed web links concept as a means of expressing relationships between Resources. A Link consists of a set of Parameters that define:

- a context URI,
- a target URI,
- a relation from the context URI to the target URI
- elements that provide metadata about the target URI, the relationship or the context of the Link.

The target URI is mandatory and the other items in a Link are optional. Additional items in the Link may be made mandatory based on the use of the links in different contexts (e.g. in collections, in discovery, in bridging etc.). Schema for the Link payload is provided in Annex D.

An example of a Link is shown in

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

Figure 10: Example of a Link

Two Links are distinct from each other when at least one parameter is different. For example the two Links shown in Figure 11 are distinct and can appear in the same list of Links.

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

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

Figure 11: Example of distinct Links

The specification may mandate Parameters and Parameter values as required for certain capabilities. For all Links returned in a response to a RETRIEVE on /oic/res, if a Link does not explicitly include the “rel” Parameter, a value of “rel”=”hosts” shall be assumed . The relation value of “hosts” is defined by IETF RFC 6690 and registered in the IANA Registry for Link Relations at [http://www.iana.org/assignments/link-relations/link-relations.xhtml]

As shown in D.2.8 the relation between the context URI and target URI in a Link is specified using the “rel” JSON element and the value of this element specifies the particular relation.

The context URI of the Link shall implicitly be the URI of the Resource (or specifically a Collection) that contains the Link unless the Link specifies the anchor parameter. The anchor parameter is used to change the context URI of a Link – the relationship with the target URI is based off the anchor URI when the anchor is specified. An example of using anchors in the context of Collections – a floor has rooms and rooms have lights – the lights may be defined in floor as Links but the Links will have the anchor set to the URI of the rooms that contain the lights (the relation is contains). This allows all lights in a floor to be turned on or off together while still having the lights defined with respect to the rooms that contain them (lights may also be turned on by using the room URI too).

```
/a/floor {
  "links": [
      {
          "href": "/x/light1",
          "anchor":  "/a/room1",    ** Note: /a/room1 has the “contains” relationship with
/x/light1; not /a/floor **
          "rel": "contains"
      }
  ]
}

/a/room1 {
  "links": [
       {
                  ** Note: /a/room1 “contains” the /x/light since /a/room1 is the implicit context URI **
"href": "/x/light1",
          "rel": "contains"
       }
   ]
}
```

Figure 12: Example of use of anchor in Link

7.8.1.1 Parameters

7.8.1.1.1 “ins” or Link Instance Parameter

The “ins” parameter identifies a particular Link instance in a list of Links. The "ins" parameter may be used to modify or delete a specific Link in a list of Links. The value of the “ins” parameter is set at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has been set, the “ins” parameter shall not be modified for as long as the Link is a member of that list.

7.8.1.1.2 “p” or Policy Parameter

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

If all the Policy keys are to be omitted, then “p” may optionally be omitted from the Link entirely as an efficiency measure. See each key’s description for the meaning when that key is omitted.

The policy Parameter "p" is defined by:

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

Bit Position	Policy rule	Comment
Bit 0 (the LSB)	discoverable	The discoverable rule defines whether the Link is to be included in the Resource discovery message via /oic/res. • If the Link is to be included in the Resource discovery message, then "p" shall include the "bm" key and set the discoverable bit to value 1. • If the Link is NOT to be included in the Resource discovery message, then "p" shall either include the "bm" key and set the discoverable bit to value 0 or omit the "bm" key entirely.
Bit 1 (2nd LSB)	observable	The observable rule defines whether the Resource referenced by the target URI supports the NOTIFY operation. • If the Resource supports the NOTIFY operation, then "p" shall include the "bm" key and set the observable bit to value 1. • If the Resource does NOT support the NOTIFY operation, then "p" shall either include the "bm" key and set the observable bit to value 0 or omit the "bm" key entirely.
Bits 2-7	--	Reserved for future use. All reserved bits in "bm" shall be set to value 0.

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

- "sec" key: The "sec" key corresponds to a Boolean value that indicates whether the Resource referenced by the target URI is accessed via a secure DTLS over IP connection.
    - If the Resource must be accessed securely via DTLS over IP, then "p" shall include the "sec" key and set the value to true.
    - If the Resource is not required to be accessed securely via DTLS over IP, then "p" shall either include the "sec" key and set the value to false or omit the "sec" key entirely.
- "port" key: The "port" key corresponds to an integer value that is used to indicate the port number where the Resource referenced by the target URI may be accessed.
- If the Resource is available via an encrypted connection (i.e. DTLS over IP), then
    - "p" shall include the "sec" key and its value shall be true.

- "p" shall include the "port" key and its value shall be the port number where the encrypted connection may be established.

- If the Resource is not available via an encrypted connection, then
  - "p" shall include the "sec" key and its value shall be false or "p" shall omit the "sec" key; the default value of "sec" is false.
  - "p" shall omit the "port" key.
- Access to the Resource on the port specified by the “port” key shall be made by an encrypted connection (e.g. coaps://). (Note that unencrypted connection to the Resource may be possible on a separate port discovered thru multicast discovery).
- Note that access to the Resource is controlled by the ACL for the Resource. A successful encrypted connection does not ensure that the requested action will succeed. See OCF Security – Access Control section for more information.
  - If “p” includes the “sec” key and the value of “sec” is set to true, then “p” shall include the “port” key and set the value of “port” to the port number used to access the Resource.
  - If “p” includes the "sec" key and the value is set to false or if “p” omits the “sec” key, then “p” shall omit the "port" key entirely.

Example 1: below shows the Policy Parameter for a Resource that is both discoverable and observable and is accessed via a non-secure connection:

```
"p": { "bm": 2, "sec": true,
"port": 33275 }
```

Example 2 below shows the Policy Parameter for a Resource that is observable but not discoverable and must be accessed via a secure DTLS over IP connection using port 33275:

```
"p": { "bm": 2, "sec": true,
"port": 33275 }
```

7.8.1.1.3 “type” or Media Type Parameter

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

7.8.1.1.4 “bp” or the Batch Interface Parameter

The “batch” Parameter "bp" is used to specify the modifications to the target URI as the "batch" Request is forwarded through this Link. The "q" element in the value defines the query string that shall be appended to the "href" to make the target URI. The "q" query string may contain Property strings that are valid in that context. For example: Given a Collection as follows

```
/room2
{
  "if": "oic.if.b",
  "color": "blue",
  "links": [
```

```
    {"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline" ], "p":
{"bm": 2, "sec": true, "port": 33277}, "rel": "contains", "bp": { "q": "if=oic.if.baseline"} }

  ]

}
```

The following is the sequence for batch request to /room2

1. `GET /room2?if=oic.if.b`
2. This request is transformed to: `GET /switch?if=oic.if.baseline` when the batch request is propagated through the Link to the target `/switch`

See the Interfaces section 7.5 for more details on the "batch" Interface.

7.8.1.1.5 “di” or Device ID parameter

The “di” Parameter specifies the device ID of the Device that hosts the target Resource defined in the in the “href” Parameter.

The device ID may be used to qualify a relative reference used in the “href” or to lookup endpoint information for the relative reference.

7.8.1.1.6 “buri” or base URI Parameter

The “buri” Parameter is the base URI to which the relative reference in “href” is resolved to. The base URI and relative reference may be used to construct the URI to the target for the Link. The base URI shall use the OCF Scheme for the URI defined in section 6.

7.8.1.2 Formatting

When formatting in jSON, the list of Links shall be an array. The first element of the array shall be a JSON object called the “tags block”. This object may be empty or have keys that are the Parameters from the list of Parameters for the Link. The “href” parameter shall not appear in the “tags block”. The second element of this array shall be a list of Links.

For each list of Links the Parameters that appear in the “tags block” shall apply to each of the links in the list of Links array associated with this tags block.

A null list of Links shall have a null “tags block” and both shall not be included.

NOTE: By this organization the list of Links is recursive and the “tags block” allows for a compact representation where Parameters shared by multiple Links don’t need to be repeated in each Links and can be factored into the “tags block”.

For example a list of Links with “tags” block.

```
[
    {
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
    },
    [
      {
        "href": "/oic/d",
        "rt": ["oic.d.light", "oic.wk.d"],
        "if": [ "oic.if.r", "oic.if.baseline" ]
      },
      {
        "href": "/oic/p",
        "rt": ["oic.wk.p"],
```

```
        "if": [ "oic.if.r", "oic.if.baseline" ]
      },
      {
        "href": "/switch",
        "rt": ["oic.r.switch.binary"],
        "if": [ "oic.if.a",  "oic.if.baseline" ],
        "mt": [ "application/cbor", "application/exi+xml" ]
      },
      {
        "href": "/brightness",
        "rt": [ "oic.r.light.brightness" ],
        "if": [ "oic.if.a", "oic.if.baseline" ]
      }
    ]
]
```

Figure 13: Example “list of Links”

7.8.1.3 List of Links in a Collection

A list of Links in a Resource shall be included in that Resource as the value of the “links” Property of that Resource. A Resource that contains Links is a Collection.

A Resource with a list of Links

```
/Room1
{
  "rt": "my.room",
  "if": ["oic.if.ll", "oic.if.baseline" ],
  "color": "blue"
  "links":
  [
    {
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
    },
    [
      {
        "href": "/oic/d",
        "rt": ["oic.d.light", "oic.wk.d"],
        "if": [ "oic.if.r", "oic.if.baseline" ]
      },
      {
        "href": "/oic/p",
        "rt": ["oic.wk.p"],
        "if": [ "oic.if.r", "oic.if.baseline" ]
      },
      {
        "href": "/switch",
        "rt": ["oic.r.switch.binary"],
        "if": [ "oic.if.a", "oic.if.baseline" ],
        "mt": [ "application/cbor", "application/exi+xml" ]
      },
      {
        "href": "/brightness",
        "rt": ["oic.r.light.brightness"],
        "if": [ "oic.if.a", "oic.if.baseline" ]
      }
    ]

  ]
```



```
}
```

Figure 14: List of Links in a Resource

7.8.1.4 Usage Cases – Resource discovery

The OCF architecture utilizes typed Links as a mechanism for bootstrapping Resource discovery through the known Core Resource /oic/res. A RETRIEVE operation on /oic/res returns (among other things) a serialized representation of typed Links to Resources that are discoverable on that Device.

The serialization format should be negotiated using the underlying transport protocol (i.e. using Accept and Content-Type headers in case of CoAP). By default, OCF uses CBOR as the payload. The payload (content) in CBOR for Links is described with the JSON Schema in D.2.8. Other serializations (e.g. XML/EXI) may be defined in future versions of this specification. The JSON Schema that specifies the representation of the response to /oic/res is defined D.8.

7.8.2 Collections

7.8.2.1 Overview

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

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

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

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

If the “drel” property is defined for the Collection then all Links that don’t explicitly specify a relationship shall inherit this default relationship in the context of that Collection. The default relationship defines the implicit relationship between the Collection and the target URI in the Link.

The list of Links defined in a Collection may be either a simple list of Links as illustrated in Figure 16 or may be a list of tagged Links sets as illustrated in Figure 17. For the former, the value of the “links” Property is a simple array of Links. For the later, the value of the “links” Property is an array where each element is a resource containing a Links array and a set of one or more key-value pairs; the key-value pairs are the tags for the Links array (the key is the tag name and the value is the tag value)

```
/my/house
{
 "rt": ["my.r.house"],
 "color": "blue",
 "n" : "myhouse",
 "links": [
  [
   {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
    "n" : "mydoor"
   },
   [
    {
     "href": "/door",
     "rt": ["oic.r.door"],
     "if": ["oic.if.b", "oic.if.ll", "oic.if.baseline]
    },
    {
     "href": "/door/lock",
     "rt": ["oic.r.lock"],
     "if": ["oic.if.b", "oic.if.baseline"],
     "type": ["application/cbor", "application/exi+xml"]
    }
   ]
  ],
  [
   {
   "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
   },
   [
    {
     "href": "/light",
     "rt": ["oic.r.light"],
     "if": ["oic.if.s", "oic.if.baseline"]
    },
    {
     "href": "/binarySwitch",
     "rt": ["oic.r.switch.binary"],
     "if": ["oic.if.a", "oic.if.baseline"],
     "type": ["application/cbor"]
    }
   ]
  ]
 ]
}
```

IRI/URI (resource)

Properties (resource)

Tags (link)

Parameters (link)

Figure 15: Example showing parts of Collection and Links

```
{
 "links": [
   {
      "href": "/door",
      "rt": ["oic.r.door"],
      "if": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
    },
    {
      "href": "/door/lock",
      "rt": ["oic.r.lock"],
      "if": ["oic.if.b", "oic.if.baseline"],
      "type": ["application/cbor", "application/exi+xml"]
    }
 ]
}
```

Figure 16: Example Collection with simple links (JSON)

```
{
 "links": [
  [
   {
   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
   },
   [
     {
      "href": "/door",
      "rt": ["oic.r.door"],
      "if": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
     },
     {
      "href": "/door/lock",
      "rt": ["oic.r.lock"],
      "if": ["oic.if.b", "oic.baseline"],
      "type": ["application/cbor, "application/exi+xml"]
     }
   ]
  ],
  [
   {
   "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9"
   },
   [
     {
      "href": "/light",
      "rt": ["oic.r.light"],
      "if": ["oic.if.s"]
     },
     {
      "href": "/binarySwitch",
      "rt": ["oic.r.switch.binary"],
      "if": ["oic.if.a", "oic.if.baseline"],
      "type": ["application/cbor"]
     }
   ]
  ]
 ]
}
```


Figure 17: Example Collection with tagged Links (JSON)

Note: Example shows only one tag; each tag has the same tag name, i.e., “di”, but have different tag values.

A Collection may be:

- A pre-defined Collection where the Collection has been defined a priori and the Collection is static over its lifetime. Such Collections may be used to model, for example, an appliance that is composed of other devices or fixed set of resource representing fixed functions.
- A Device local Collection where the Collection is used only on the Device that hosts the Collection. Such collections may be used as a short-hand on a client for referring to many Servers as one.
- A centralized Collection where the Collection is hosted on an Device but other Devices may access or update the Collection
- A hosted Collection where the collection is centralized but is managed by an authorized agent or party.

7.8.2.2 Collection Properties

An Collection shall define the “links” Property. In addition, other Properties may be defined for the Collection by the Resource Type. The mandatory and recommended Common Properties for Collection are shown in Table 9. This list of Common Properties are in addition to those defined for Resources in section 7.3.2. When a property is repeated in Table 9 , the conditions in this definition shall override those in the general list for Resources.

Table 9: Common Properties for Collections (in addition to Common Properties defined in section 7.3.2)

Property	Description	Property name	Value Type	Mandatory
Links	The set of links in the collection	"links"	json Array of Links	Yes
Name	Human friendly name for the collection	"n"	string	No
Identity	The id of the collection	"id"	UUID	No
Resource Types	The list of allowed resource types for links in the collection. Requests for addition of links using link list or link batch interfaces will be validated against this list. If this property is not defined or is null string then any resource type is permitted	"rts"	json Array of resource type names	No
Default relationship	Specifies the default relationship to use for Links in the collection where the "rel" parameter has not been explicitly defined. It is permissible to have no "drel" property defined for the collection and the Links to also not have "rel" defined either. In such case, the use of the collection is, for example, as a random bag of links	"rel"	string	No

The Properties of a Collection may not be modified.

7.8.2.3 Default resource type

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

The default Resource Type provides support for the Common Properties including the "links" Property. For the default resource type, the value of "links" shall be a simple array of Links and tagging of links shall not be supported.

The default Resource Type shall support the 'baseline' and 'links list' Interfaces. The default Interface shall be the 'links list' Interface.

8 CRUDN

8.1 Overview

CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for manipulating Resources. These operations are performed by a Client on the resources contained in an Server.

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

CRUDN operations utilize a set of parameters that are carried in the messages and are defined in Table 10. A Device shall use CBOR as the default payload (content) encoding scheme for resource representations included in CRUDN operations and operation responses; a Device may negotiate a different payload encoding scheme (e.g, see in section 12.2.4 for CoAP messaging). The following subsections specify the CRUDN operations and use of the parameters. The type definitions for these terms will be mapped in the messaging section for each protocol.

Table 10. Parameters of CRUDN messages

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

8.2 CREATE

The CREATE operation is used to request the creation of new Resources on the Server. The CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 18 and described below.

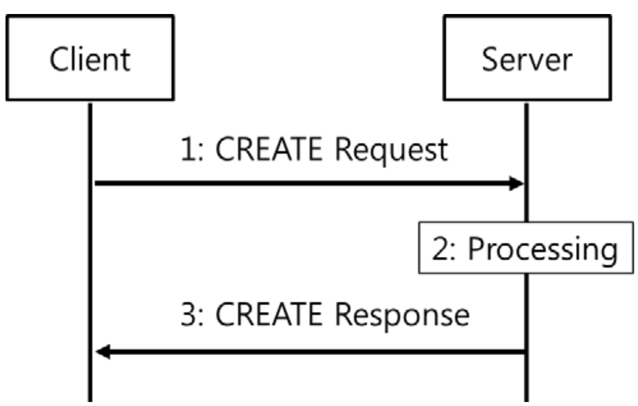

Figure 18. CREATE operation

8.2.1 CREATE request

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

- *fr*: Unique identifier of the Client
- *to*: URI of the target resource responsible for creation of the new resource.
- *ri*: Identifier of the CREATE request
- *cn*: Information of the resource to be created by the Server
    - i) *cn* will include the URI and resource type property of the resource to be created.
    - ii) *cn* may include additional properties of the resource to be created.
- *op*: CREATE

8.2.2 Processing by the Server

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

8.2.3 CREATE response

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

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

8.3 RETRIEVE

The RETRIEVE operation is used to request the current state or representation of a Resource. The RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in Figure 19 and described below.

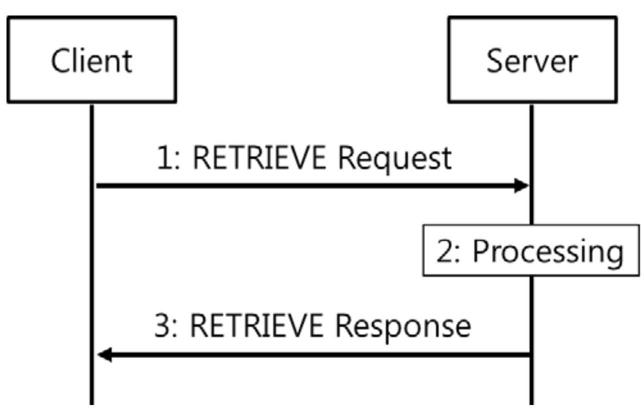

Figure 19. RETRIEVE operation

8.3.1 RETRIEVE request

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

- *fr*: Unique identifier of the Client
- *to*: URI of the resource the Client is targeting
- *ri*: Identifier of the RETRIEVE request
- *op*: RETRIEVE

8.3.2 Processing by the Server

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

8.3.3 RETRIEVE response

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

- *fr*: Unique identifier of the Server
- *to*: Unique identifier of the Client
- *ri*: Identifier included in the RETRIEVE request
- *cn*: Information of the resource as requested by the Client
    i) *cn* should include the URI of the resource targeted in the RETRIEVE request

- *rs*: The result of the RETRIEVE operation

8.4 UPDATE

The UPDATE operation is either a Partial UPDATE or a complete replacement of the information in a Resource in conjunction with the interface that is also applied to the operation. The UPDATE operation is initiated by the Client and consists of three steps, as depicted in Figure 20 and described below.

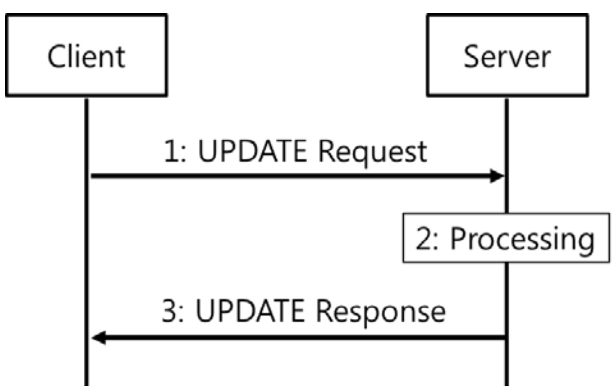

Figure 20. UPDATE operation

8.4.1 UPDATE request

The UPDATE request message is transmitted by the Client to the Server to request the update of information of a Resource on the Server. The UPDATE request message will carry the following parameters.

- *fr*: Unique identifier of the Client
- *to*: URI of the resource targeted for the information update
- *ri*: Identifier of the UPDATE request
- *op*: UPDATE
- *cn*: Information, including properties, of the resource to be updated at the target resource

8.4.2 Processing by the Server

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

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

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

8.4.3 UPDATE response

The UPDATE response message will include the following parameters:

- *fr*: Unique identifier of the Server
- *to*: Unique identifier of the Client
- *ri*: Identifier included in the UPDATE request
- *rs*: The result of the UPDATE request

The UPDATE response message may also include the following parameters:

- cn: The Resource representation following processing of the UPDATE request

8.5 DELETE

The DELETE operation is used to request the removal of a Resource. The DELETE operation is initiated by the Client and consists of three steps, as depicted in Figure 21 and described below.

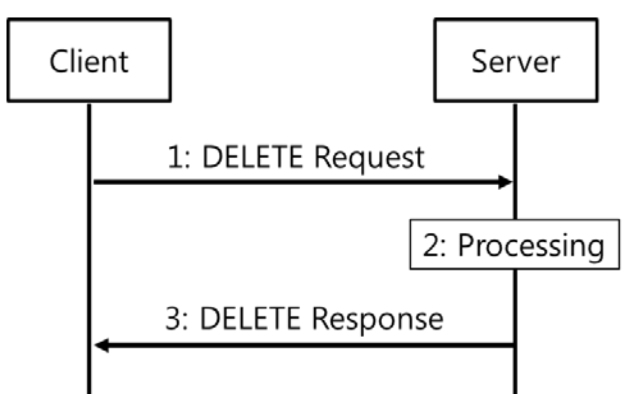

Figure 21. DELETE operation

8.5.1 DELETE request

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

- *fr*: Unique identifier of the Client
- *to*: URI of the target resource which is the target of deletion
- *ri*: Identifier of the DELETE request
- *op*: DELETE

8.5.2 Processing by the Server

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

8.5.3 DELETE response

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

- *fr*: Unique identifier of the Server
- *to*: Unique identifier of the Client
- *ri*: Identifier included in the DELETE request
- *rs*: The result of the DELETE operation

8.6 NOTIFY

The NOTIFY operation is used to request asynchronous notification of state changes. Complete description of the NOTIFY operation is provided in section 11.4. The NOTIFY operation uses the NOTIFICATION response message which is defined here.

8.6.1 NOTIFICATION response

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

- *fr*: Unique identifier of the Server
- *to*: URI of the Resource target of the NOTIFICATION message
- *ri*: Identifier included in the CREATE request
- *op*: NOTIFY
- *cn*: The updated state of the resource

9 Network and connectivity

9.1 Introduction

The IOT environment, which the OCF is addressing, is composed of very heterogeneous systems. Because these systems are often tailored to address dedicated requirements, they are composed of very diverse products and services. Those products span from very constrained devices that run on batteries to every day high end devices available on consumer market shelves. The lack of a global standard and the need to create such a standard has led various groups to work on streamlining those technologies with well-established networking standards.

The IETF recognized the market transition and realized that Ipv4 was no longer adequate. Not only does the new scale call for a new technology, but also the manageability of even more diverse devices, the complexity of multiple subnets and higher security and privacy needs require a whole new set of standards. Cognizant of the existence and need for dedicated physical layer and data link layer, the IETF set up working groups to streamline the various existing technologies at the network layer. In accordance with these market realities, this specification also means to leverage existing radio silicon (e.g., Bluetooth® technology, Wi-Fi, or 802.15.4) and concentrates on the network layer and the associated data link layer adaptations produced by the IETF.

9.2 Architecture

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

The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices and associated routers, but also new services introducing additional edge routers. All these new requirements require advance architectural constructs to address complex network topologies like the one shown in Figure 22.

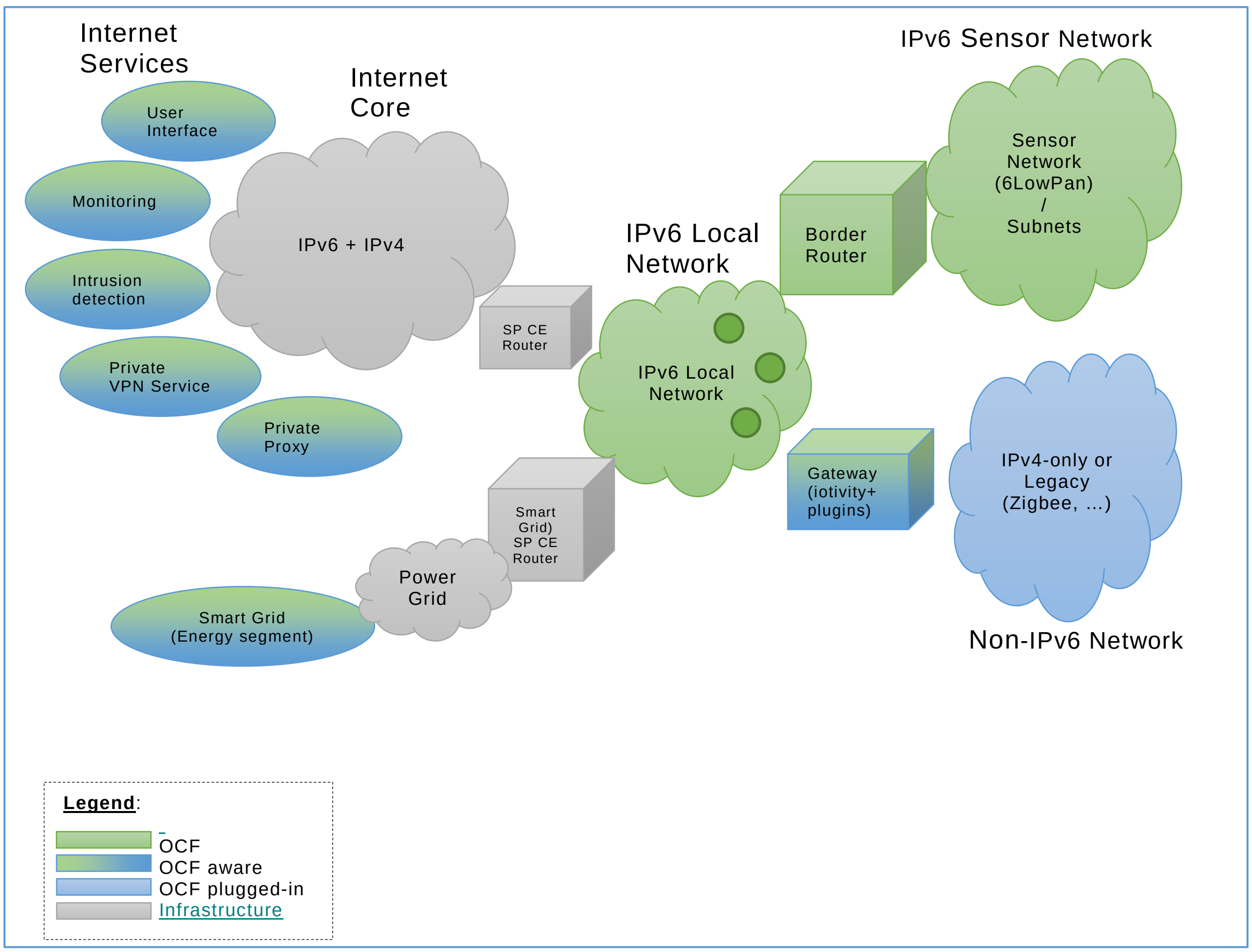

Figure 22. High Level Network & Connectivity Architecture

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

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

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

• A node may translate and route messaging between IPv6 and non-IPv6 networks.se roles:

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

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

9.3 • A node may translate and route messaging between IPv6 and non-IPv6 networks.IPv6 network layer requirements

9.3.1 Introduction

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

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

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

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

9.3.2 IPv6 node requirements

9.3.2.1 Introduction

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

9.3.2.2 IP Layer

An IPv6 node should support IPv6. If a node supports IPv6, then it shall conform to the requirements for communication on the local network as follows:

- Shall support IETF RFC 2460 “Internet Protocol version 6 Specification” and related updates as defined in section 5.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 4291 “IP Version 6 Addressing Architecture” and related updates as defined in section 5.9.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 4861 “Neighbor Discovery for IPv6” and related updates as defined in section 5.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 4862 “IPv6 Stateless Address Autoconfiguration” and related updates as defined in section 5.9.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 4443 “Internet Control Message Protocol (ICMPv6) for IPv6” [RFC4443] and related updates as defined in section 5.8 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 1981 “Path MTU Discovery” and related updates as defined in section 5.6 of IETF RFC 6434 “IPv6 Node Requirements”.
- Shall support IETF RFC 4193 “Unique Local IPv6 Unicast Addresses” and related updates.
- Shall support IETF RFC 3810 “Multicast Listener Discovery Version 2 (MLDv2) for IPv6” and related updates. In particular, shall generate new MLDv2 Report messages for every “All OCF Nodes” address FF0X::158 joined on an interface.

.

9.3.3 IPv6 constrained nodes

9.3.3.1 Requirements

An IPv6 constrained node shall support all node requirements defined in section 9.3.2. If a constrained node supports IPv6, it should use the adaptations defined as follows in order to support IPv6.

9.3.3.2 IP layer

An IPv6 constrained node should support the neighbour discovery optimization as defined in IETF RFC 6775 “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)”.

9.3.3.3 Sub IP layer

- An IPv6 constrained node on an ITU-T G.9959 network should support IETF RFC 7428 and related updates.
- An IPv6 constrained node on an IEEE 802.15.4 network should support IETF RFC 4944 and related updates.
- An IPv6 constrained node on a BLUETOOTH(R) Low Energy network should support IETF RFC 7668 and related updates.

10 Endpoint discovery

10.1 Introduction

This section describes how an OCF Endpoint is discovered by another OCF Endpoint in a network. An OCF Endpoint shall support CoAP discovery..

10.2 CoAP based Endpoint discovery

The following describes CoAP based Endpoint discovery:

a) Advertising or publishing Devices shall join the ‘All OCF Nodes’ multicast groups (as defined in [IANA IPv6 Multicast Address Space Registry]) and listen on the port 5683.

b) Clients intending to discover resources shall join the ‘All OCF Nodes’ multicast groups (as defined in [IANA IPv6 Multicast Address Space Registry]).

c) Clients shall senddiscovery requests (GET request) to the 'All OCF Nodes’ multicast group address at port 5683. The requested URI shall be /oic/res.

d) If the discovery request is intended for a specific resource type, the Query parameter "rt" shall be included in the request (section 6.2.1) with its value set to the desired resource type. Only Devices hosting the resource type shall respond to the discovery request.

e) When the “rt” Query parameter is omitted, all Devices shall respond to the discovery request.

f) Handling of multicast requests shall be as described in section 8 of IETF RFC 7252 and section 4.1 in IETF RFC 6690.

g) Devices which receive the request shall respond using CBOR payload encoding. A Device shall indicate support for CBOR payload encoding for multicast discovery as described in section 12.2.3. Later versions of the specification may support alternate payload encodings (JSON, XML/EXI, etc.).

Below are a few examples to search for Devices on the network:

To search for all Devices on the network a Client can issue:

Request

```
GET /oic/res
```

Response


```
[
  {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
    "links": [
      {
        "href": "/oic/d",
        "rt":["oic.d.light", "oic.wd.d"],
        "if": ["oic.if.r oic.if.baseline",]
      },
      {
        "href": "/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"]
      },
      {
        "href": "/switch",
        "rt": ["oic.r.switch.binary"],
        "if": ["oic.if.a", "oic.if.baseline"]
      },
      {
        "href": "/brightness",
        "rt": ["oic.r.light.brightness"],
```



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

To search for oic.r.switch.binary resources on the network a Client can issue:

Request

```
GET /oic/res?rt=oic.r.switch.binary
```

Response

```
[
  {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
    "links": [
      {
        "href": "/switch",
        "rt": ["oic.r.switch.binary"],
        "if": ["oic.if.a", "oic.if.baseline"]
      }
    ]
  }
]
```

Note that the examples do not indicate the multicast address and port number. The examples also do not include the accept header.

11 Functional interactions

11.1 Introduction

The functional interactions between a Client and n Server are described in section 11.2 through section 11.6 respectively. The functional interactions use CRUDN messages (section 8) and include Discovery, Notification, and Device management. These functions require support of core defined resources as defined in Table 11. More details about these resources are provided later in this section.

Table 11. List of Core Resources

Pre-defined URI	Resource Name	Resource Type	Related Functional Interaction	Mandatory
/oic/res	Default	oic.wk.res	Discovery	Yes
/oic/p	Platform	oic.wk.p	Discovery	Yes
/oic/d	Device	oic.wk.d	Discovery	Yes
/oic/con	Configuration	oic.wk.con	Device Management	No
/oic/mnt	Maintenance	oic.wk.mnt	Device Management	No

11.2 Provisioning

Provisioning in Framework includes two distinct processes: onboarding and Configuration.

onboarding is the process which delivers required information to a Device for joining the OCF network. When onboarding process is completed, the Device has necessary information and is able to join the OCF network (State #1 in Figure 23). Further details about provisioning can be found in OCF Security specification (Owner PSK).

Configuration is the process which delivers required information to a device for accessing OCF services. At the end of the configuration process, the Device has all the necessary information and is able to access OCF services (State #2 in Figure 23).

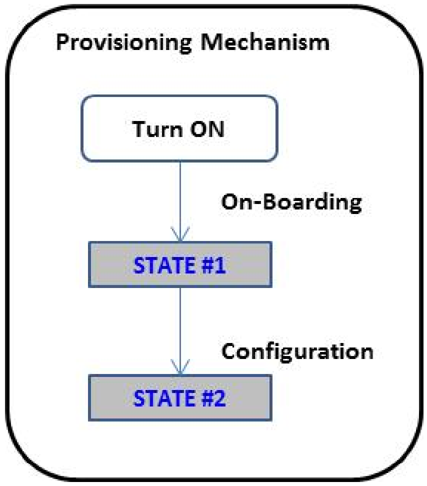

Figure 23. Provisioning State Changes

#1 onboarding

Framework is applicable to many different types of devices with different capabilities, including devices with a rich user interface that can take inputs from the users, e.g., smartphones, as well as headless devices that have no means for receiving user inputs, e.g., sensors. Additionally, the Devices may support different communication and connectivity technologies, e.g., Bluetooth, Wi-Fi, etc. Different communication and connectivity technologies provide different onboarding mechanisms specific to that technology.

Due to these differences and diversity of device capabilities, this version of specification does not mandate a particular process for onboarding, instead, specifies the state of the Device upon completion of the onboarding process.

As part of the onboarding process the device acquires detailed information and required parameter values to be able to connect to the network, resulting in successful establishment of a connection to the network at the end of the onboarding process. The required information and parameters values include for example, SSID for Wi-Fi as well as authentication credentials.

Later versions of this specification may specify a common process for onboarding across different communication and connectivity technologies.

#2 Configuration

Once a Device is successfully connected to the OCF network, it needs additional configuration information for accessing the OCF services or to subscribe for OCF services. The information required may include geographical location, time zone, security requirements, etc. This information may be pre-loaded on an Device, or may be acquired from a configuration service that can be located on another Device, e.g., the Configuration Source. The information regarding the configuration service resource, e.g., the URI of the Configuration Source, is pre-configured on the Device.

The configuration information is also in core resource /oic/con. Upon completion of the onboarding process and as soon as the Device is connected to the network, if the configuration information is not pre-loaded, it shall initiate the configuration process, as a result of which the Device acquires the relevant configuration information, through either a pull or a push interaction, and populates its designated configuration resource with its current configured state information. The designated configuration resource maintains the latest configuration state and is the designated resource through which updates to the configuration are made.

If the configuration information is not pre-loaded the Device retrieves them from the Configuration Source. During the lifetime of a Device a Client may retrieve or update the configuration state of the Device. Some of the configuration information is read only and some may be modified by Configuration Sources depending on the ‘Access Modes’ of properties in /oic/con resource.

Figure 24 depicts the interactions triggered by a Device to retrieve its configuration information from the Configuration Source which may be located on a remote Device or locally. These interactions occur instantly following completion of onboarding process; the Device may retrieve its configuration at any time during its lifetime.

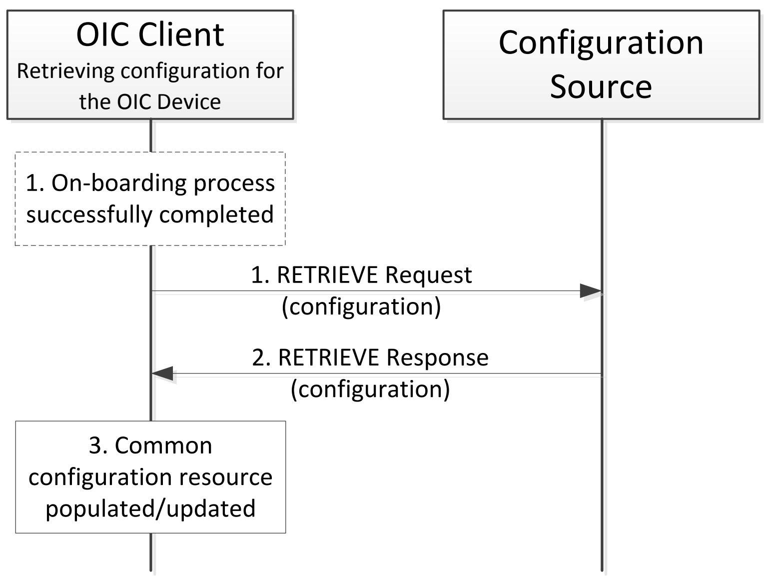

Figure 24. Interactions initiated by the Device to retrieve its configuration from a configuration source

Figure 25 depicts the interactions when the retrieve of configuration information is done by a Client.

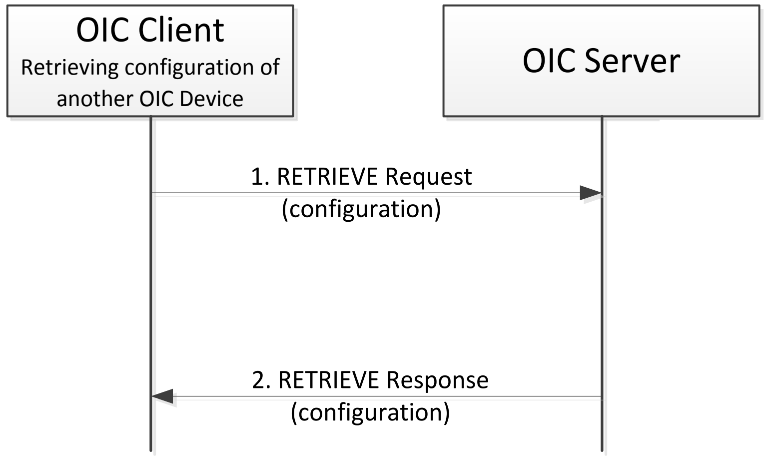

Figure 25. Interactions for retrieving the configuration state of an Device.

Figure 26 depicts the interactions when the configuration information of an Device is updated by a Client, e.g., the Configuration Source.

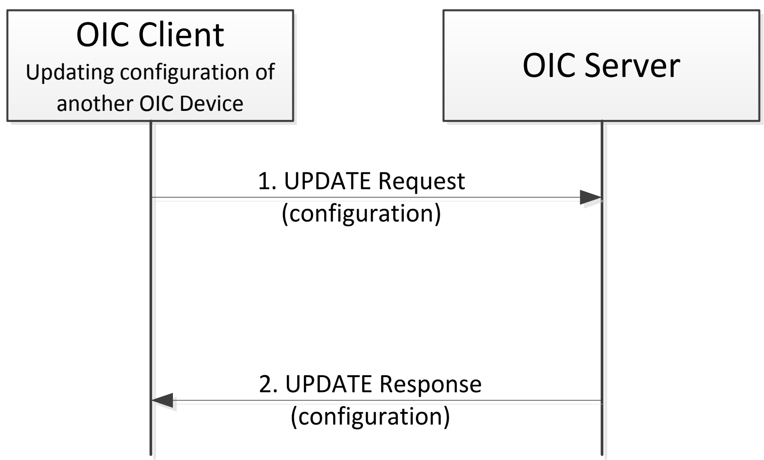

Figure 26. Update of and Device configuration

If Configuration is supported by a Device, i.e., the configuration information may be dynamically updated, the Core Resource /oic/con shall be supported as the designated configuration resource as described in Table 12.

Configuration Resource

A Device or a Platform may be initially configured from information that is set or provisioned at bootstrap. In addition, the Device and Platform may be configured further by an external agent post bootstrap depending on changing conditions or context. The core resource /oic/con exposes properties that may be used to effect changes in the configuration.

A configuration is determined by setting all the properties that collectively pertain to that configuration. The outcome of setting a new configuration is determined by the value of the specific properties in that set. Setting a new configuration through /oic/con may lead to initiation of processes that affect or create side effects in other resources.

Table 12. Configuration Resources

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/con	Configuration	oic.wk.con	oic.if.rw	The resource through which configurable information specific to the Device is exposed. The **resource properties** exposed by /oic/con are listed in Table 13.	Configuration

Table 13 defines the oic.wk.con resource type.

Table 13. oic.wk.con resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R, W	yes	Human friendly name configurable by the end user (e.g. Bob's thermostat).
Location	loc	json (has two attributes one with longitude and latitude and also a name for a location)			R, W	no	Provides location information where available.
Location Name	locn	string			R, W	no	Human friendly name for location For example, "Living Room".
Currency	c	string			R,W	no	Indicates the currency that is used for any monetary transactions
Region	r	string			R,W	no	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").

11.3 Resource discovery

11.3.1 Introduction

Discovery is a function which enables endpoint discovery as well as resource based discovery. Endpoint discovery is described in detail in section 10. This section mainly describes the resource based discovery.

11.3.2 Resource based discovery: mechanisms

11.3.2.1 Overview

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

At the minimum, Resource based discovery uses the following:

1) A resource to enable discovery shall be defined. The representation of that resource shall contain the information that can be discovered.
2) The resource to enable discovery shall be specified and commonly known a-priori.A Device for hosting the resource to enable discovery shall be identified.
3) A mechanism and process to publish the information that needs to be discovered with the resource to enable discovery.
4) A mechanism and process to access and obtain the information from the resource to enable discovery. A query may be used in the request to limit the returned information.
5) A scope for the publication
6) A scope for the access.
7) A policy for visibility of the information.

Depending on the choice of the base aspects defined above, the Framework defines three resource based discovery mechanisms:

- Direct discovery, where the Resources are published locally at the Device hosting the resources and are discovered through peer inquiry.
- Indirect discovery, where Resources are published at a third party assisting with the discovery and peers publish and perform discovery against the resource to enable discovery on the assisting 3rd party.
- Advertisement discovery, where the resource to enable discovery is hosted local to the initiator of the discovery inquiry but remote to the Devices that are publishing discovery information.

A Device shall support direct discovery.

11.3.2.2 Direct discovery

In direct discovery,

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

11.3.2.3 Indirect discovery of Resources (resource directory based discovery)

In indirect discovery the information about the resource to be discovered is hosted on a Server that is not hosting the resource. See section 11.3.6 for details on resource directory based discovery.

In indirect discovery:

a) The resource to be discovered is hosted on a Device that is neither the client initiating the discovery nor the Device that is providing or publishing the information to be discovered. This Device may use the same resource to provide discovery for multiple agents looking to discover and for multiple agents with information to be discovered.
b) The Device to be discovered or with information to discover, publishes that information with resource to be discovered on a different Device. The policies on the information shared including the lifetime/validity are specified by the publishing Device. The publishing Device may modify these policies as required.
c) The client doing the discovery may send a unicast discovery request to the Device hosting the discovery information or send a multicast request that shall be monitored and responded to by the Device. In both cases, the Device hosting the discovery information is acting on behalf of the publishing Device.
d) The discovery policies may be set by the Device hosting the discovery information or by the party that is publishing the information to be discovered. The discovery information that is returned in the discovery response shall adhere to the policies that are in effect at the time of the request.

11.3.2.4 Advertisement Discovery

In advertisement discovery:

a) The resource to enable discovery is hosted local to the Device that is initiating the discovery request (client). The resource to enable discovery may be an Core Resource or discovered as part of a bootstrap.
b) The request could be an implementation dependent lookup or be a local RETRIEVE request against the resource that enables discovery.
c) The Device with information to be discovered shall publish the appropriate information to the resource that enables discovery.
d) The publishing Device is responsible for the published information. The publishing Device may UPDATE the information at the resource to enable discovery based on its needs by sending additional publication requests. The policies on the information that is discovered including lifetime is determined by the publishing Device.

11.3.3 Resource based discovery: Information publication process

The mechanism to publish information with the resource to enable discovery can be done either locally or remotely. The publication process is depicted in Figure 27. The Device which has discovery information to publish shall a) either update the resource that enables discovery if hosted locally or b) issue an UPDATE request with the information to the Device which hosts the resource that enables discovery. The Device hosting the resource to enable discovery adds/updates the resource to enable discovery with the provided information and then responds to the Device which has requested the publication of the resource with an UPDATE response.

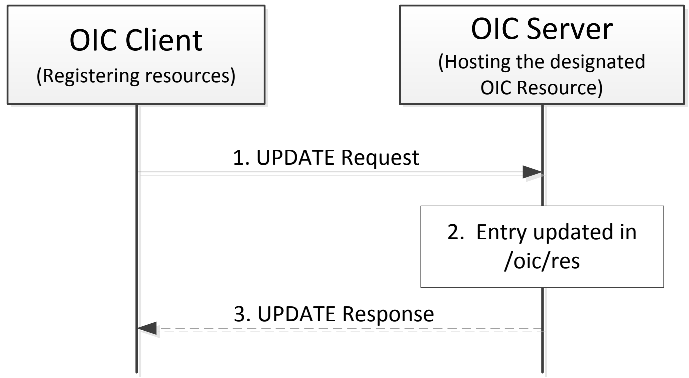

Figure 27. Resource based discovery: Information publication process

11.3.4 Resource based discovery: Finding information

The discovery process (Figure 28) is initiated as a RETRIEVE request to the resource to enable discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the support in the data connectivity layer. The response to the request has the information to be discovered based on the policies for that information. The policies can determine which information is shared, when and to which requesting agent. The information that can be discovered can be resources, types, configuration and many other standards or custom aspects depending on the request to appropriate resource and the form of request. Optionally the requester may narrow the information to be returned in the request using query parameters in the URI query.

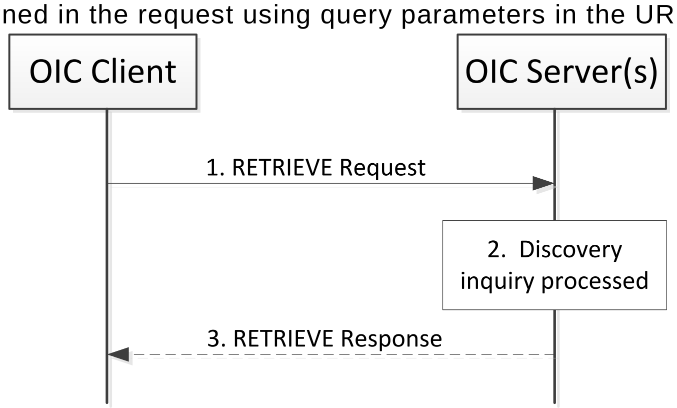

Figure 28. Resource based discovery: Finding information

Discovery Resources

Some of the Core Resources shall be implemented on all Devices to support discovery. The Core Resources that shall be implemented to support discovery are:

- /oic/res for discovery of resources
- /oic/p for discovery of platform
- /oic/d for discovery of device information

Details for these mandatory Core Resources are described in Table 14

Platform resource –

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

Device resource

The device resource shall have the pre-defined URI /oic/d. The resource /oic/d exposes the properties pertaining to a Device as defined in Table 14. The properties exposed are determined by the specific instance of Device and defined by the resource type(s) of /oic/d on that Device. Since all the resource types of /oic/d are not known a priori, the resource type(s) of /oic/d shall be determined by discovery through the core resource /oic/res. The device resource /oic/d shall have a default resource type that helps in bootstrapping the interactions with this device (the default type is described in Table 14.)

Protocol indication

A Device may need to support different messaging protocols depending on requirements for different application profiles. For example, the Smart Home profile may use CoAP and the Industrial profile may use DDS. To enable interoperability, a Device uses the protocol indication to indicate the transport protocols they support and can communicate over.

Table 14. Mandatory discovery Core Resources

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	**Interfaces**	**Description**	Related **Functional Interaction**
/oic/res	Default	oic.wk.res	oic.if.ll	The resource through which the corresponding Server is discovered and introspected for available resources. /oic/res shall expose the resources that are discoverable on a Device. When an Server receives a RETRIEVE request targeting /oic/res (e.g., GET /oic/res), it shall respond with the link list of all the discoverable resources of itself. The /oic/d and /oic/p are discoverable resources, hence their links are included in /oic/res response. The resource properties exposed by /oic/res are listed in Table 15.	Discovery
/oic/p	Platform	oic.wk.p	oic.if.r	The discoverable resource through which platform specific information is discovered. The **resource properties** exposed by /oic/p are listed in Table 17	Discovery
/oic/d	Device	oic.wk.d and/or one or more Device Specific resource type IDs	oic.if.r	The discoverable via /oic/res resource which exposes properties specific to the Device instance. The **resource properties** exposed by /oic/d are listed in Table 17 /oic/d may have one or more resource types that are specific to Device in addition to the default resource type or if present overriding the default resource type. The base type oic.wk.d defines the properties that shall be exposed by all Devices. The device specific resource type(s) exposed are dependent on the class of device (e.g. air conditioner, smoke alarm); applicable values are defined by the vertical specifications.	Discovery

Table 15 defines oic.wk.res resource type.

Table 15. oic.wk.res resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R	no	Human-friendly name defined by the vendor
Device Identifier	di	UUID			R	yes	The device identifier as indicated by the /oic/d resource of the Device. There may be multiple "di" instances in /oic/res but each "di" shall have a unique value. This "di" value uniqueness implies that the resources of a device shall be grouped together under a single "di".
Links	links	array	See 0		R	yes	The array of Links describes the URI, supported resource types and interfaces, and access policy.
Messaging Protocol	mpro	SSV			R	No	String with Space Separated Values (SSV) of messaging protocols supported as a SI Number from Table 16 For example, “1 and 3” indicates that the Device supports coap and http as messaging protocols.

A Device which wants to indicate its messaging protocol capabilities may add the property ‘mpro’ in response to a request on /oic/res. A Device shall support CoAP based discovery as the baseline discovery mechanism (see section 10.2). A Client which sees this property in a discovery response can choose any of the supported messaging protocols for communicating with the Server for further messages. For example, if a Device supporting multiple protocols indicates it supports a value of “1 3” for the ‘mpro’ property in the discovery response, then it cannot be assumed that there is an implied ordering or priority. But a vertical service specification may choose to specify an implied ordering or priority. If the ‘mpro’ property is not present in the response, A Client shall use the default messaging protocol as specified in the vertical specification for further communication. Table 16 provides an OCF registry for protocol schemes.

Table 16. Protocol scheme registry

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

Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by a Client to form requests in a different messaging protocol other than discovery is out of scope.

The following applies to the use of /oic/d as defined above:

- A vertical may choose to expose its Device Type (e.g., refrigerator or A/C) by adding the Device Type to the list of Resource Types associated with /oic/d.
    - For example; rt of /oic/d becomes ["oic.wk.d", "oic.d.<thing>"]; where “oic.d.<thing>” is defined in another spec such as the Smart Home vertical.
    - This implies that the properties exposed by /oic/d are by default the mandatory properties in Table 17.
- A vertical may choose to extend the list of properties defined by the Resource Type 'oic.wk.d'. In that case, the vertical shall assign a new Device Type specific Resource Type ID. The mandatory properties defined in Table 17 shall always be present.

Note:

As per existing Core specification definitions the resource type ID may be a list of resource type IDs; when that is the case the default resource type ID for /oic/d is the first resource type ID listed. So a vertical can list ‘oic.d.thing’ first. This then means a GET /oic/d returns the properties for oic.d.thing and a GET /oic/d?rt=<some rt> returns the properties for the rt listed in the query.

Table 17 oic.wk.d resource type definition defines the base resource type for the /oic/d resource.

Table 17. oic.wk.d resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R	no	Human friendly name defined by the vendor.”
Spec Version	icv	string			R	yes	Spec version of the core specification this device is implemented to, The syntax is "core.<major>.<minor>.<sub-version>” where <major>, <minor, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. This version of the specification the string value shall be “core.1.1.0”.
Device ID	di	UUID			R	yes	Unique identifier for Device. This value shall be as defined in [OCF Security] for DeviceID.
Data Model Version	dmv	CSV			R	yes	Spec version of the Resource Specification to which this device data model is implemented; if implemented against a Vertical specific resource specification, then the Spec version of the vertical specification this device model is implemented to. The syntax is a comma separated list of ” <res>.<major>.<minor>.<sub-version> or <vertical>.<major>.<minor>.<sub-version>. <res> is the string “res” and <vertical> is the name of the vertical defined in the Vertical specific resource specification. The <major>, <minor, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. This

							version of the specification the string value shall be “res.1.1.0”.

The additional resource type(s) of the /oic/d resource are defined by the vertical specification.

Table 18 defines oic.wk.p resource type.

Table 18. oic.wk.p resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform ID	pi	string			R	yes	Unique identifier for the physical platform (UIUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.
Manufacturer Name	mnmn	string			R	yes	Name of manufacturer
Manufacturer Details Link	mnml	URI			R	no	Reference to manufacturer, represented as a URI
Model Number	mnmo	string			R	no	Model number as designated by manufacturer
Date of Manufacture	mndt	date		Time *(show RFC)*	R	no	Manufacturing date of device
Platform Version	mnpv	string			R	no	Version of platform – string (defined by manufacturer)
OS Version	mnos	string			R	no	Version of platform resident OS – string (defined by manufacturer)
Hardware Version	mnhw	string			R	no	Version of platform hardware
Firmware version	mnfv	string			R	no	Version of device firmware
Support link	mnsl	URI			R	no	URI that points to support information from manufacturer
SystemTime	st	datetime			R	no	Reference time for the device. The format is restricted to the concatenation of “date” and “time”

							with the “T” as a delimiter between “date” and “time”. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.
Vendor ID	vid	string			R	no	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.

Composite Device

A physical device may be modelled as a single device or as a composition of other devices. For example a refrigerator may be modelled as a composition, as such part of its definition of may include a sub-tending thermostat device which itself may be composed of a sub-tending thermometer device.

There may be more than one way to model an server as a composition. One example method would be to have Platform which represents the composite device to have more than one instance of a Device on the Platform. Each Device instance represents one of the distinct devices in the composition. Each instance of Device may itself have or host multiple instances of other resources.

An implementation irrespective of how it is composed shall only expose a single instance of /oic/d with an ‘rt’ of choice for each logical Server.

Thus, for the above refrigerator example if modeled as a single Server; /oic/res would expose /oic/d with a resource type name appropriate to a refrigerator. The sub-tending thermostat and thermometer devices would be exposed simply as instances of a resource with a device appropriate resource type with an associated URI assigned by the implementation; e.g., /MyHost/MyRefrigerator/Thermostat and /MyHost/MyRefrigerator/Thermostat/Thermometer.

11.3.5 Resource discovery using /oic/res

Discovery using /oic/res is the default discovery mechanism that shall be supported by all Devices as follows:

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

b) An Device wanting to discover resources or resource types on one or more remote Devices makes a RETRIEVE request to the /oic/res on the remote Devices. This request may be sent multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request may optionally be restricted using appropriate clauses in the query portion of the request. Queries may select based on resource types, interfaces, or properties.

c) Query applies to the representation of the resources. /oic/res is the only resource whose representation has “rt”. So /oic/res is the only resource that can be used for Multicast discovery at the transport protocol layer.

d) The Device receiving the RETRIEVE request responds with a list of resources, the resource type of each of the resources and the interfaces that each resource supports. Additionally information on the policies active on the resource can also be sent. The policy supported includes observability and discoverability. (More details below)

e) The receiving Device may do a deeper discovery based on the resources returned in the request to /oic/res.

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

- The URI (relative or fully qualified URL) of the resource
- The Resource Type of each resource. More than one Resource Type may be returned if the resource enables more than one type. To access resources of multiple types, the specific resource type that is targeted shall be specified in the request.
- The Interfaces supported by that Resource. Multiple interfaces may be returned. To access a specific interface that interface shall be specified in the request. If the interface is not specified, then the Default Interface is assumed.
- Policies defined against that resource. These policies may be security related, access modes, types of interactions, etc. In addition to the request/response type of interaction, the specification allows the resource to be "observed" (section 11.4.2).

The JSON schemas for discovery using /oic/res are described in D.8. Also refer to Section 10 (Endpoint Discovery) for details of Multicast discovery using /oic/res on a CoAP transport.

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

11.3.6 Resource directory (RD) based discovery

11.3.6.1 Introduction

11.3.6.1.1 Indirect discovery for lookup of the resources

Direct discovery is the mechanism used currently to find resources in the network. When needed, resources are queried at a particular node directly or a multicast packet is sent to all nodes. Each queried node responds directly with its discoverable resources to the discovering device. Resources available locally are registered on the same device.

In some situations, one of the other mechanisms described in section 11.3.2.3, called indirect discovery, may be required. Indirect discovery is when a 3rd party device, other than the discovering device and the discovered device, assists with the discovery process. The 3rd party only provides information on resources on behalf of another device but does not host resources on part of that device.

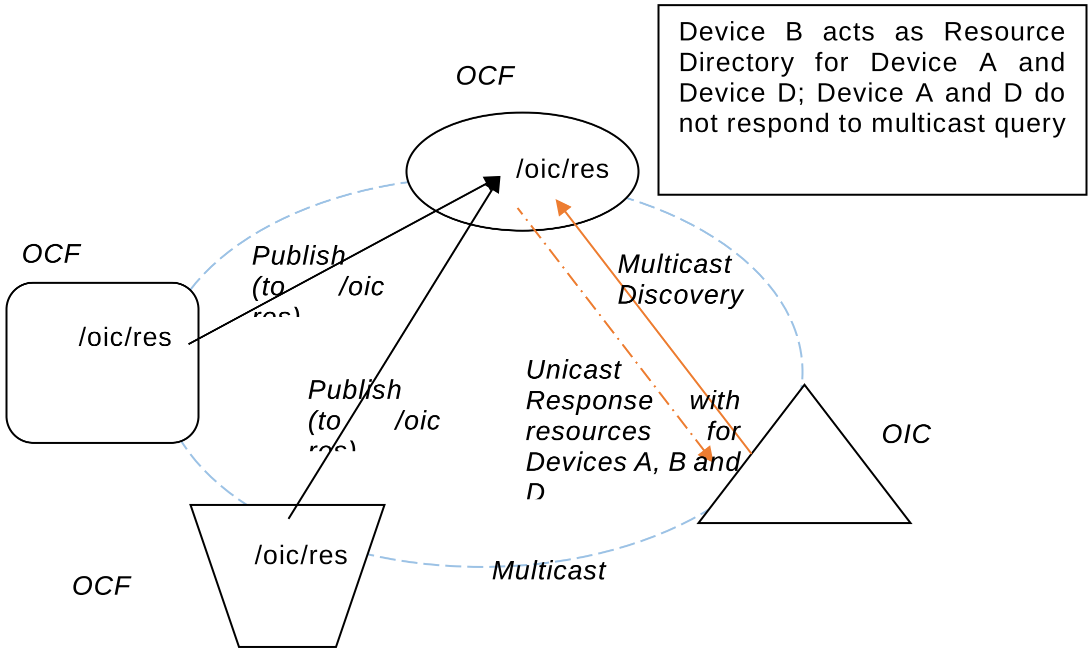

Figure 29. Indirect discovery of resource by resource directory

Indirect discovery is useful for a resource constrained device that needs to sleep to manage power and cannot process every discovery request, or when devices may not be on the same network and requires optimization for discovery. Once resources are discovered using indirect discovery then the access to the resource is done by a request directly to the Device that hosts that resource.

11.3.6.1.2 Resource directory

A resource directory (RD) is an Device that assists with indirect discovery. A RD can be queried at its /oic/res resource to find resources hosted on other Devices. These Devices can be sleepy nodes or any other device that cannot or may not respond to discovery requests. Device can publish all or partial list of resources they host to a RD. The RD then responds to queries for Resource discovery on behalf of the publishing Device (for example: when a Device may go to sleep). For general Resource discovery, the RD behaves like any other Server in responding to requests to /oic/res.

Any Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The Devices that want to discover RDs shall use this resource and one of the Resource discovery mechanisms to discover the RD and get the parameters of the RD. The information discovered through this resource shall be used to select the appropriate RD to use for resource publication. The bias information shall include the following criteria: power source (AC, battery powered or safe/reliable), connectivity (wireless, wired), CPU, memory, load statistics (processing publishing and query from the devices). In addition, the RD shall return a bias factor that ranges from 0 to 100. Optionally, the RD may also return a context - the value which shall be a string and semantics of the context are not discussed in this document but it is expected that the context will be used to establish a domain, region or some such scope that is meaningful to the application, deployment or usage.

Using these criteria or the bias factor, the Device shall select one RD (per context) to publish its resources. A context describes the state of an OCF Device with respect to Resource discovery. A context is usually determined at deployment and from application requirements. An example of a context could be a multicast group- a Device that is a member of more than one multicast group may have to find and select a RD in each of the multicast groups (i.e. per context) to publish its information. The Device may choose other RDs during its lifetime but a Device shall not publish its resource information to more than one RD Devices such as TV, network router, desktop will have higher weightage or bias factor compared to mobile phone device.

11.3.6.2 The remainder of this section is divided into two parts. The first part covers discovering of the RD and publishing, updating and deleting of resources for the constrained/sleepy device. The second part covers the replies of the RD to queries from devices with the aim to discover resources. Resource directory discovery

11.3.6.2.1 Discovering a resource directory

A RD in the OCF network shall support RD discovery, shall provide the facility to allow devices to publish their resource information to a RD, to update resource information in a RD and to delete resource information from a RD.

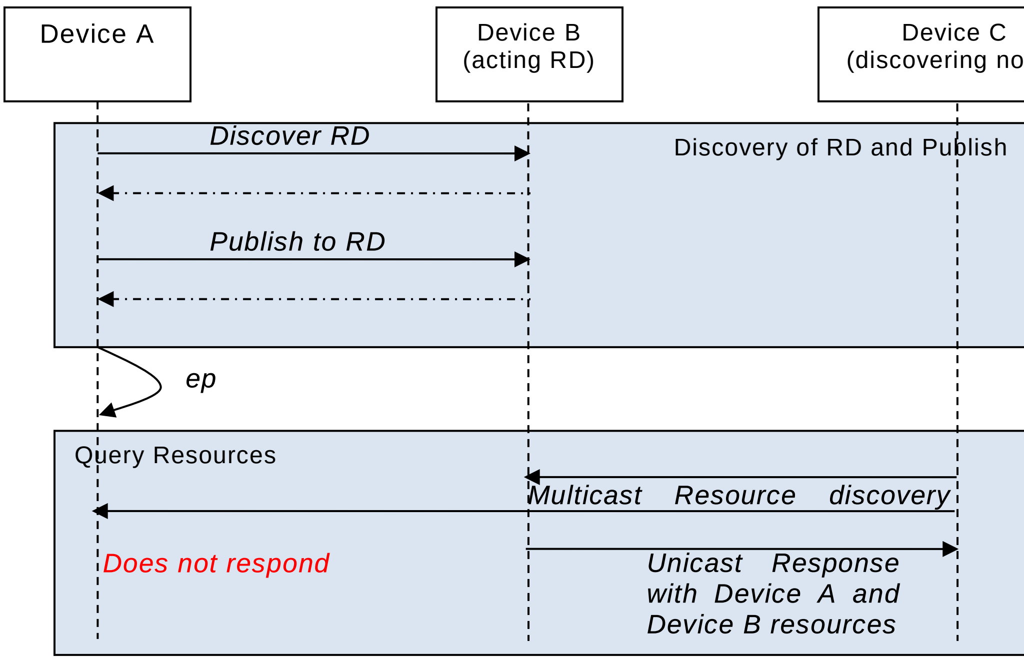

Figure 30. RD discovery and RD supported query of resources support

As shown in Figure 30, the Device that wishes to advertise its resources: first discovers a resource directory and then publishes the desired resource information. Once a set of resources have been published to a RD then the publishing device shall not respond to multicast Resource discovery queries for those published resources when the RD is on the same multicast domain. In that case, only the RD shall respond to multicast Resource discovery requests on the resource published to it.

An OCF network allows for more than one device acting as a RD. The reason to have multiple RD support is to make network scalable, handle network failures and centralized device failure bottleneck. This does not preclude a scenario where a use case or deployment environment may require single device in the environment to be deployed as the only resource directory (e.g. gateway model). There may be more than one Device acting as RD on a Platform.

Discovering of an RD may result in responses from more than one RD. The discovering device shall select a RD. The selection may be based on the weightage parameter(s) provided in the response from the RD.

An RD will be application agnostic i.e., application should not be aware whether resource directory was queried to get the resource information. All the handling of the retrieval is kept opaque to the application. A Client that performs Resource discovery uses an RD just like it may use any other Server for discovery. It may send a unicast request to the RD when it needs only the resource advertised on the RD or do a multicast query when it does not require or have explicit knowledge of an RD.

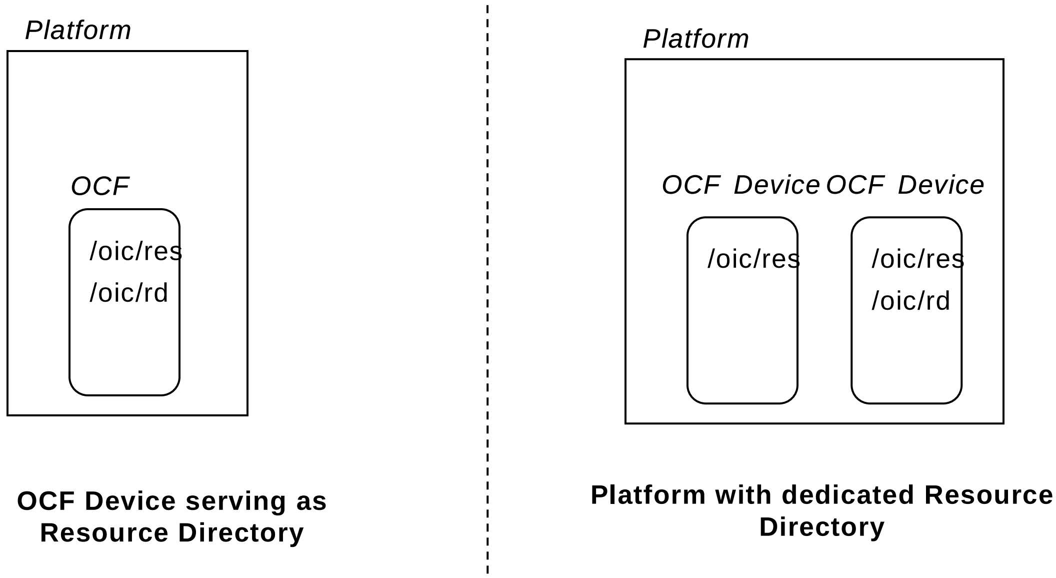

Figure 31. Resource Direction Deployment Scenarios

Resource directory can also be discovered in the following manners:

- Pre-configuration: Devices wishing to publish resource information may be configured a priori with the information (e.g. IP address, port, transport etc.) of a specific resource directory. This pre-configuration may be done at onboarding or may be updated on the device using an out-of-band method. This pre-configuration may be done by the manufacturer or by the user/device manager.
- Query-oriented: A Client wanting to discover resource directories using query-oriented discovery (i.e. pull) shall issue multicast Resource discovery request directed to the /oic/rd resource. Only the devices that hosts a /oic/rd resource shall respond to this query. The response shall include information about the RD (as defined by the resource type) and weightage parameters to allow the discovering device to select between RDs (see details in RD selection section). The /oid/rd resource shall be instantiated on the OCF Devices acting as a resource directory. . The /oic/rd schema is as defined in D.12**.**.
- Advertisement: An RD may advertise about itself to devices. It is an advertisement packet. The devices that are already publishing to a RD may use this as a heartbeat message of the RD. If the RD advertisement does not arrive at a stipulated interval, publishing device starts searching for other RDs in the network, as this is a signal that RD is not online. Other usage of this message is it serves as an advertisement for a device seeking a RD to publish their resources. The details from the advertisement can then be used to query directly to a RD to get weightage details instead of sending a multicast packet in a network. As it is intended this is sent at a regular interval and does not include weightage information to keep packet sizes small.
- One of the important benefits of an RD is to make services discoverable in networks that don't support site wide multicast but do support site wide routing. An example of such a network is Homenet .To enable an RD function across such a network a site discovery mechanism is needed to discover the RD service (IP address & port number). Homenets that support hybrid

proxy (IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00) allow site wide discovery based on dns-sd/mDNS. In order to make itself discoverable beyond the link local scope, an RD with a routable ip address shall implement the mDNS responder requirements defined in IETF RFC 6762. The RD shall respond to mDNS queries of type PTR and with a service name equal to "_rd._sub._oic._udp.local". The response shall include all routable IP addresses. Devices with a routable ip address shall discover all available RD instances by issuing a DNS-SD's PTR lookup as defined in IETF RFC 6763 with as service name service name "_rd._sub._oic._udp.local". The response shall include all routable addresses/port pair through which the RD service is made accessible.

11.3.6.2.2 Resource directory selection process

11.3.6.2.2.1 Selection criteria

When a device discovers more than one RD then it shall decide to use one of these RDs based on the selection criteria described here. A device shall use or publish information to only one RD within a multicast domain at a given time. This is to minimize the burden of processing duplicate information in the Resource discovery phase.

There two ways to select an RD. One is based on a bias factor (RD generated) and the other is based on clients determination based on granular parameters provided by the server (client/device generated). Devices may use one or both methods to select an RD.

Bias factor: The bias factor is a server generated positive number in the range of 0 to 100, where 0 is the lowest to 100 being the highest. If two RDs have the same bias factor then the selecting device may choose either based auxiliary criteria or at random. Either way only one RD shall be selected and used at a time. No specific method is defined in this specification to determine the bias factor for an RD. The number may be a pre-configured value at the time of onboarding or subsequent configuration of the RD or may be based on a formula determined by the implementation of the RD. (OCF will provide a standard formula for this calculation in a future version or release of specification).

The bias factor shall be calculated by the RD by adding the contribution values determined for each of the parameters in Table 19 and divided by the number of parameters. An RD may advertise a bias factor larger than the calculated value when there is reason to believe that the RD is highly capable for example an installed service provider gateway.

Parameters: Optionally, parameters defined in Table 19 (like direct power supply, network connectivity, load conditions, CPU power, memory, etc.) may be returned in the discovery response. Discovering device may use the details to make granular selection decisions based on client defined policies and criteria that use the RD parameters. For example, a device in an industrial deployment may not weight power connectivity high but another in home environments may give more weightage for power.

Table 19: Selection parameters

Parameter	Values (Contribution)	Description
Power	Safe (100) AC (70) Batt (40)	• Safe implies that the power supply is reliable and is backed up with battery for power outages etc. • Implementation may lower the number for Batt based on the type of battery the RD device runs on. If battery conservation is important then this number should be lowered.
Mobility	Fixed (100) Mobile (50)	• Implementation may further grade the mobility number based on how mobile the RD device is; lower number for highly mobile and larger numbers for limited mobility • The mobility number shall not be larger than 80

Network Product	Type: • Wired (10) • Wireless (4) Bandwidth: • High (10) • Low (5) • Lossy (3) Interfaces	• Network product = [sum of (type * bandwidth per network interface)]/[number of interfaces] • Normalized to 100
Memory Factor	Available Total	• Memory is the volatile or non-volatile storage used to store the resource information • Memory Factor = [Available]/[Total] • Normalized to 100 (i.e. expressed as percentage)
Request Load Factor	1-minute 5-minute 15-minutes	• Current request loading of the RD • Similar to UNIX load factor (using observable, pending and processing requests instead of runnable processes) • Expressed as a load factor 3-tuple (up to two decimal points each). Factor is based on request processed in a 1-minute (L1), 5-minute (L5) and 15-minute (L15) windows • See http://www.teamquest.com/import/pdfs/whitepaper/ldavg1.pdf • Factor = 100 – ([L1*3 + L5*7 + L15*10]/3)

11.3.6.2.2.2 Selection scenarios

The device that wants to use an RD will use the endpoint discovery to find zero or more RDs on the network. After discovering the RDs, the device needs to select an RD of all found RDs on the network. The selection based on the bias factor will ensure that an Device can judge if the found RD is suitable for its needs.

The following situation can occur during the selection of an RD:

1) A single or multiple RDs are present in the network

2) No RD is present in the network

3) an additional RD arrives on the network

In the first scenario the RDs are already present. If a single RD is detected then that RD can be used . When multiple RDs are detected the Device uses the bias information to select the RD.

In the second scenario, device will listen to the advertisement of the devices that hosts the RDs. Once an RD advertisement packet is received it judges if the bias criteria are met and starts using the RDs.

In the third scenario the Device has already published its resources to an existing RD. In this scenario it discovers a new RD on the network.

After judging the bias factor the Device may choose to move to the new RD.

11.3.6.3 If the decision is made to select the new RD, the then Device shall delete its resource information from the current used RD and then after removal publish the information to the new RD. During the transition period the Device itself shall respond to Resource discovery requests. Resource publishing

11.3.6.3.1 Publish resources

11.3.6.3.1.1 Overview

After the selection process of a RD, a device may choose one of the following mechanisms:

- Push its resources information to the selected RD or
- Request the RD to pull the resource information by doing a unicast discovery request against its /oic/res

The publishing device may decide to publish all resources or few resources on the resource directory. The publishing device shall only publish resources that are otherwise published to its own /oic/res. A publishing device may respond to discovery requests (on its /oic/res resource) for the resources it does not publish to a RD. Nonetheless, it is highly recommended that when an RD is used, all discoverable resources on the publisher be published to the RD.

11.3.6.3.1.2 Publish: Push resource information

Resource information is published using an UPDATE CRUDN operation to /oic/rd using the resource type oic.wk.rdpub and the oic.if.baseline interface.

Once a publishing device has published resources to a RD, it may not respond to the multicast discovery queries for the same resources against its own /oic/res, especially when on the same multicast domain as the RD. After publishing resources, it is a RD responsibility to reply to the queries for the published resources.

If the publishing device is in sleep mode and a RD has replied on behalf of the publishing device, then a discovering device will try to access resource on the provided URI.

There is another possibility that the resource directory and the publishing device both respond to the multicast query from the discovering device. This will create a duplication of the packet but is an alternate that may be used for non-robust network. It is not a recommended option but for industrial scenarios, this is one of the possibilities. Either way, discovering clients shall always be prepared to process duplicate information in responses to multicast discovery request. The /oic/rd schema is as defined in D.12 to specify publishing (oic.rd.publish) to the /oic/rd resrouce.

11.3.6.3.2 Update resource information

Server will hold the publish resource information till the time specified in the ttl field. A device can send update if it seeks a RD to keep holding resources and reply to queries on its behalf. Update can be used for updating about all resources that are published on a RD or can use to do per resource published.

Updates are done using the same resource type and interface as for the initial publish but only the information to be updated is provided in the payload.

11.3.6.3.3 Delete resource information

A resource information hold at the resource directory can be removed anytime by the publishing device. It can be either for the whole device information or for a particular resource. This resource should be only allowed when device meets a certain requirement, as it can create potential security issue.

The delete is done using the device ID “id” as the tag in DELETE request query when all the resource information from the device is to be deleted. In the case of a specific resource then the DELETE request shall include the instance “ins” tag along with the device ID in the query.

Selective deletion of information for individual resources is not possible the case where the RD pull the resource information. The publishing device can request a delete but only for all the resource information that the RD has pulled from that device. In this case, the DELETE request has the device ID “id” tag in the query.

11.3.6.3.4 Transfer resource information from one RD to another

When a publishing device identifies an RD that is better suited, it may decide to publish to that RD. Since the device shall publish to only one RD at a time, the client shall ensure that previously published information is deleted from the currently used RD before publishing to the newly selected RD. The deletion of the resource may be done either by allowing the TTL to expire or explicitly deleting the resource information.

RDs shall not communicate resource information between themselves. It is the client’s responsibility to choose the RD and to manage the published resources.

11.3.6.4 Resource discovery

11.3.6.4.1 Query and retrieving of the resources

The query based discovery process remains the same as that in the absence of an RD. Resources may be discovered by querying the /oic/res resource by sending a multicast or unicast request. In the case of a multicast discovery request, an RD will respond for the device that hosts the resources. Clients shall be prepared to process duplicate resource information from more than one RD responding with the same information or from an RD and the hosting device (publishing the resource information) both responding to the request. Interaction with resources discovered using the RD is done using the same mechanism and methods as with resources discovered by querying the /oic/res resource of the device hosting the resources (e.g., connect to the resource and perform CRUDN operations on the resource).

11.4 Notification

11.4.1 Overview

An Server shall support NOTIFY operation to enable a Client to request and be notified of desired states of one or more Resources in an asynchronous manner. Section 11.4.2 specifies the observe mechanism in which updates are delivered to the requester.

11.4.2 Observe

In observe mechanism the Client utilizes the RETRIEVE operation to require the Server for updates in case of Resource state changes. The Observe mechanism consists of five steps which are depicted in Figure 32 and described below.

> Note: the observe mechanism can only be used for a resource with a property of observable (section 7.3.2.2).

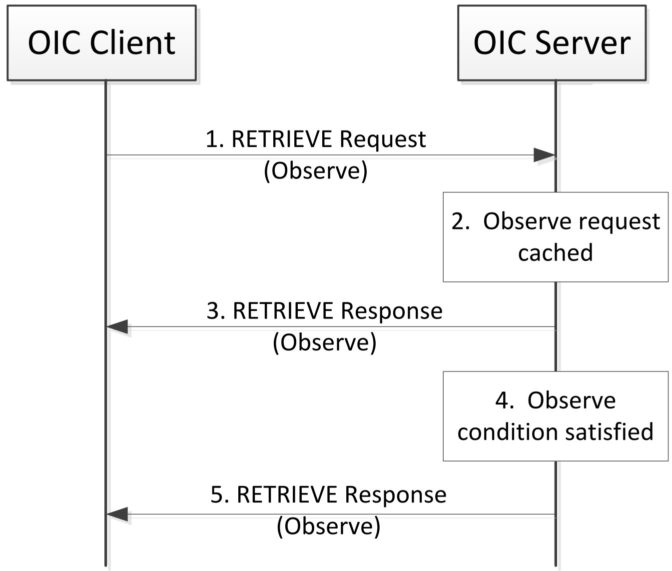

Figure 32. Observe Mechanism

11.4.2.1 RETRIEVE request with observe indication

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

- *fr*: Unique identifier of the Client
- *to*: Resource that the Client is requesting to observe
- *ri*: Identifier of the RETRIEVE request
- *op*: RETRIEVE
- *obs*: Indication for observe request

11.4.2.2 Processing by the Server

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

11.4.2.3 RETRIEVE response with observe indication

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

The RETRIEVE response message shall include the following parameters:

- *fr*: Unique identifier of the Server
- *to*: Unique identifier of the Client
- *ri*: Identifier included in the RETRIEVE request
- *cn*: Information resource representation as requested by the Client

- *rs*: The result of the RETRIEVE operation
- *obs*: Indication that the response is made to an observe request

11.4.2.4 Resource monitoring by the Server

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

11.4.2.5 Additional RETRIEVE responses with observe indication

The Server shall transmit updated RETRIEVE response messages following observed changes in the state of the Resources indicated by the Client. The RETRIEVE response message shall include the parameters listed in section 11.4.2.3.

11.4.2.6 Cancelling Observe

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

11.5 Device management

The Device Management includes the following functions:

- Diagnostics and maintenance

The device management functionalities specified in this version of specification are intended to address the basic device management features. Addition of new device management features in the future versions of the specification is expected.

11.5.1 Diagnostics and maintenance

The Diagnostics and Maintenance function in the Framework is intended for use by the administrators to resolve issues encountered with the Devices while operating in the field. If diagnostics and maintenance is supported by a Device, the Core Resource '/oic/mnt' shall be supported as described in Table 20.

Table 20. Optional diagnostics and maintenance device management Core Resources

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	**Interfaces**	**Description**	Related **Functional Interaction**
/oic/mnt	Maintenance	oic.wk.mnt	oic.if.rw	The resource through which the device is maintained and can be used for diagnostic purposes. The **resource properties** exposed by /oic/mnt are listed in Table 21.	Device Management

Table 21 defines the oic.wk.mnt resource type. At least one of the Factory_Reset, and Reboot properties shall be implemented.

Table 21. oic.wk.mnt resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	

Factory_Reset	fr	boolean			R, W	no	When writing to this Property: 0 – No action (Default*) 1 – Start Factory Reset After factory reset, this value shall be changed back to the default value (i.e., 0). After factory reset all configuration and state data will be lost. When reading this Property, a value of “1” indicates a pending factory reset, otherwise the value shall be “0” after the factory reset.
Reboot	rb	boolean			R, W	no	When writing to this Property: 0 – No action (Default) 1 – Start Reboot After Reboot, this value shall be changed back to the default value (i.e., 0)

Note: * - Default indicates the value of this property as soon as the device is rebooted or factory reset

The Framework specifies the following commands to be executed on the designated diagnostic resource of Devices over the network:

- Factory_Reset: Updates the device configuration to its original (default) state (factory state and equivalent to hard reboot)
- Reboot: Triggers a soft reboot of a Device maintaining most of the configurations intact

Execution of these commands may result in a change in the configuration state of a Device. The configuration information in the configuration resource is expected to be updated following execution of these commands by the Device, if needed. A Client invokes operations on the Server for executing the Diagnostic functions by sending an UPDATE message to the Server.

11.6 Scenes

11.6.1 Introduction

Scenes are a mechanism for automating certain operations.

A scene is a static entity that stores a set of defined resource property values for a collection of resources. Scenes provide a mechanism to store a setting over multiple Resources that may be hosted by multiple separate Servers. Scenes, once set up, can be used by multiple Clients to recall a setup.

Scenes can be grouped and reused, a group of scences is also a scene.

In short, scenes are bundled user settings.

11.6.2 Scenes

11.6.2.1 Introduction

Scenes are described by means of resources. The scene resources are hosted by a Server and the top level resource is listed in /oic/res. This means that a Client can determine if the scene functionality is hosted on a Server via a RETRIEVE on /oic/res or via Resource discovery. The setup of scenes is driven by Client interactions. This includes creating new scenes, and mappings of Server resource properties that are part of a scene.

The scene functionality is created by multiple resources and has the structure depicted in Figure 33. The sceneList and sceneCollection resources are overloaded collection resources. The sceneCollection contains a list of scenes. This list contains zero or more scenes. The sceneMember resource contains the mapping between a scene and what needs to happen according to that scene on an indicated resource.

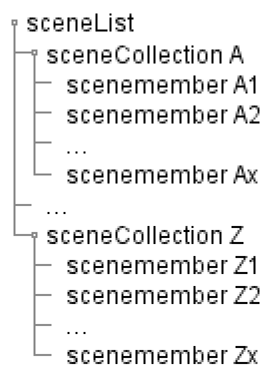

Figure 33 Generic scene resource structure

11.6.2.2 Scene creation

A Client desiring to interact with scenes needs to first determine if the server supports the scene feature; the sceneMembers of a scene do not have to be co-located on the server supporting the scene feature. This can be done by checking if /oic/res contains the rt of the sceneList resource. This is depicted in first steps of Figure 34. The sceneCollection is created by the Server using some out of bound mechanism, Client creation of scenes is not supported at this time. This will entail defining the scene with an applicable list of scene values and the mappings for each Resource being part of the scene. The mapping for each resource being part of the sceneCollection is described by a resource called sceneMember. The sceneMember resource contains the link to a resource and the mapping between the scene listed in the sceneValues property and the actual resource property value of the Resource indicated by the link.

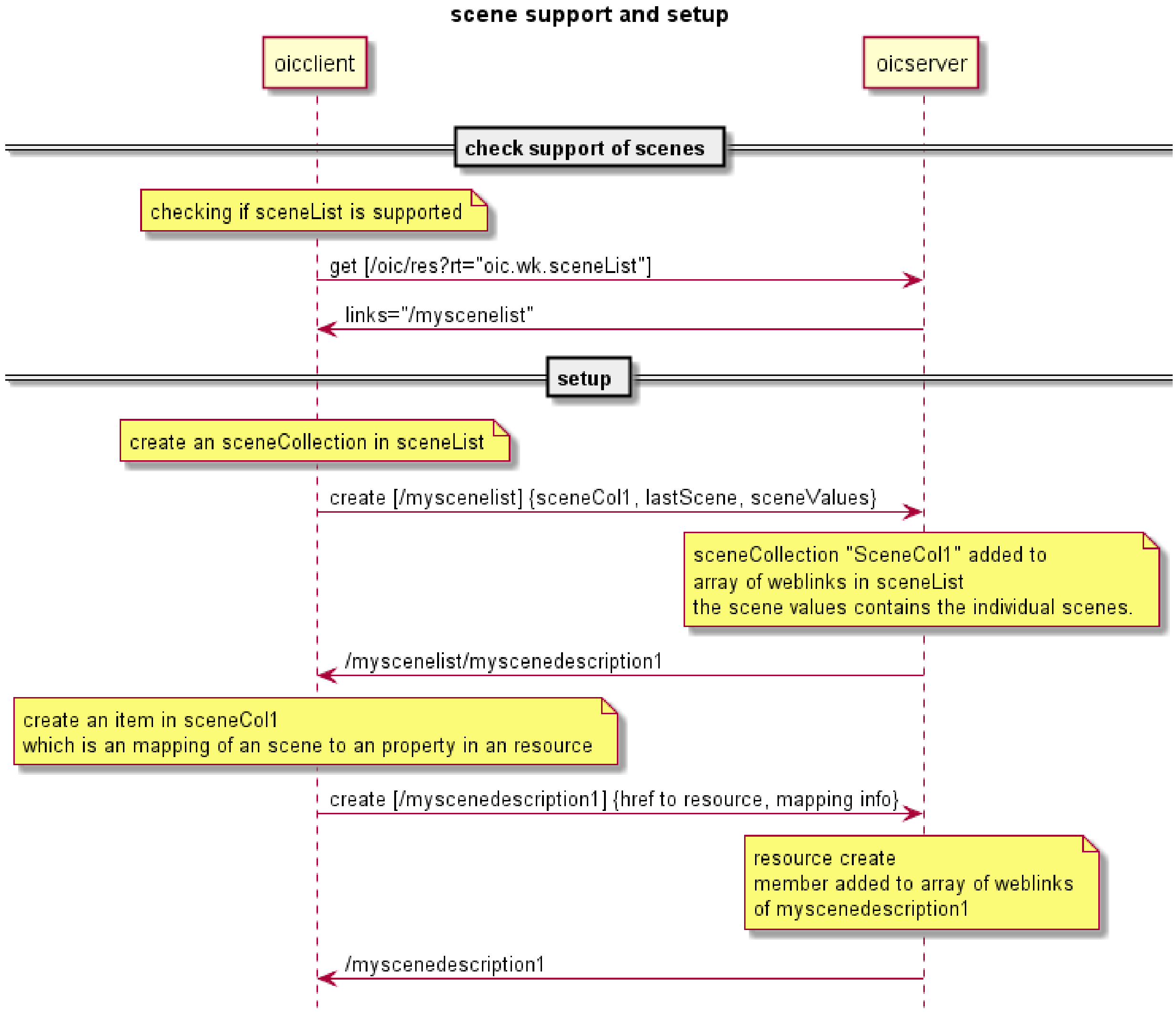

Figure 34 Interactions to check Scene support and setup of specific scenes

11.6.2.3 Interacting with Scenes

All capable Clients can interact with scenes. The allowed scene values and the last applied scene value can be retrieved from the server hosting the scene. The scene value shall be changed by issuing an UPDATE operation with a payload that sets the lastScene property to one of the listed allowed scene values. These steps are depicted in Figure 35. Note that the lastScene value does not imply that the current state of all resources that are part of the scene will be at the mapped value. This is due to that the setting the scene values are not modelled as actual states of the system. This means that another Client can change just one resource being part of the scene without having feedback that the state of the scene is changed.

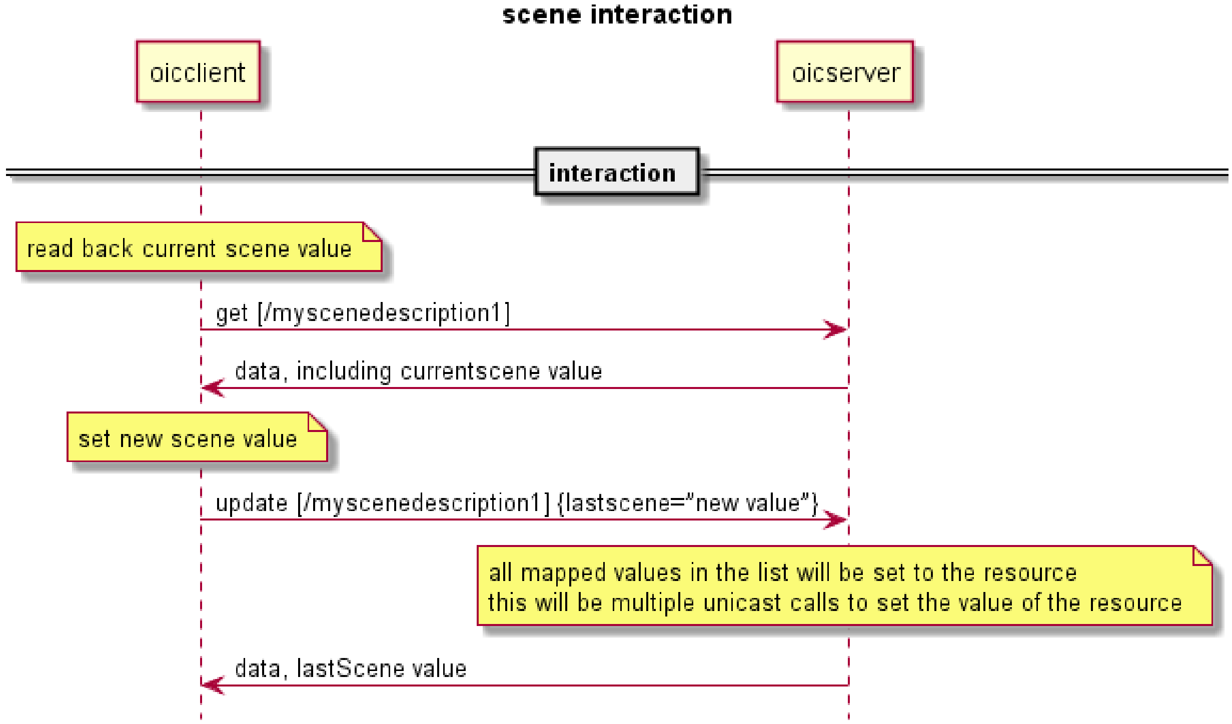

Figure 35 Client interactions on a specific scene

As described previously, a scene can reference one or more resources that are present on one or more Servers. The scene members are re-evaluated each time a scene change takes place. This evaluation is triggered by a Client that is either embedded as part of the Server hosting the scene, or separate to the server having knowledge of the scene via a RETRIEVE operation, observing the referenced resources using the mechanism described in section 11.4.2. During the evaluation the mappings for the new scene value will be applied to the Server. This behaviour is depicted in Figure 36.

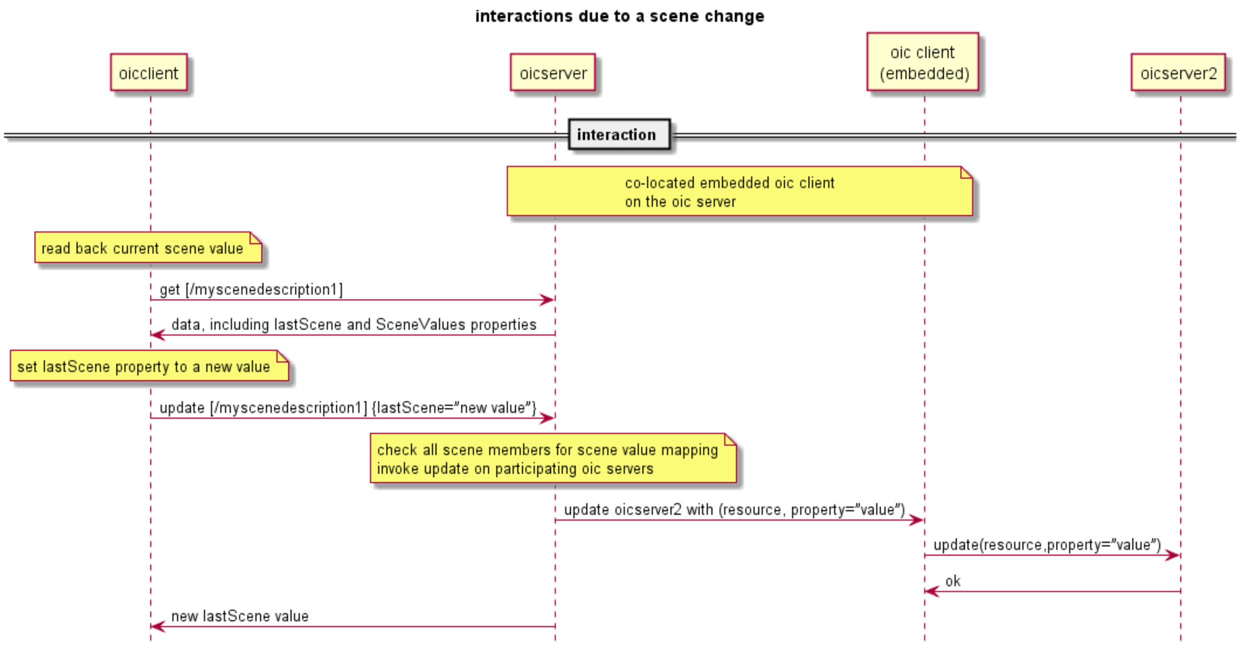

Figure 36 Interaction overview due to a Scene change

11.6.2.4 Summary of resource types defined for Scene functionality

Table 22 summarizes the list of resource types that are part of Scenes.

Table 22 list of resource types for Scenes

Friendly Name (informative)	Resource Type (rt)	Short Description	Section
sceneList	oic.wk.sceneList	Top Level collection containing sceneCollections	
sceneCollection	oic.wk.sceneCollection	Description of zero or more scenes	
sceneMember	oic.wk.sceneMember	Description of mappings for each specific resource part of the sceneCollection	

11.6.3 Security considerations

Creation of Scenes on a Server that is capable of this functionality is dependent on the ACLs applied to the resources and the Client having the appropriate permissions. Interaction between a Client (embedded or separate) and a Server that hosts the resource that is referenced as a scene member is contingent on the Client having appropriate permissions to access the resource on the host Server.

See OCF Security for details on the use of ACLs and also the mechanisms around Device Authentication that are necessary to ensure that the correct permissions exist for the Client to access the scene member resource(s) on the Server.

12 Messaging

12.1 Introduction

This section specifies the protocol messaging mapping to the CRUDN messaging operations (Section 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols is expected in later version of this specification. All the property information from the resource model shall be carried within the message payload. This payload shall be generated in the resource model layer and shall be encapsulated in the data connectivity layer. The message header shall only be used to describe the message payload (e.g., verb, mime-type, message payload format), in addition to the mandatory header fields defined in messaging protocol (e.g., CoAP) specification. If the message header does not support this, then this information shall also be carried in the message payload. Resource model information shall not be included in the message header structure unless the message header field is mandatory in the messaging protocol specification.

12.2 Mapping of CRUDN to CoAP

12.2.1 Overview

A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in section 12.2.3. A Device implementing CoAP shall conform to IETF draft-ietf-core-observe-16 to implement the CoAP Observe option. Support for CoAP block transfer when the payload is larger than the MTU is defined in section 12.2.6.

12.2.2 URIs

An OCF: URI is mapped to a coap: URI by replacing the scheme name ‘oic’ with ‘coap’ if unsecure or ‘coaps’ if secure before sending over the network by the requestor. Similarly on the receiver side, the scheme name is replaced with ‘oic’.

12.2.3 CoAP method with request and response

12.2.3.1 Overview

Every request has a CoAP method that realizes the request. The primary methods and their meanings are shown in Table 23, which provides the mapping of GET/PUT/POST/DELETE methods to CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides the generic behaviours when using these methods, however resource interfaces may modify these generic semantics.

Table 23. CoAP request and response

Method for CRUDN	(mandatory) Request data	(mandatory) Response data
GET for RETRIEVE	- **Method code**: GET (0.01) - **Request URI**: an existing URI for the Resource to be retrieved	- **Response code**: success (2.xx) or error (4.xx) - **Payloa**d: Resource representation of the target Resource (when successful)
POST for CREATE	- **Method code**: POST (0.02) - **Request URI**: an existing URI for the Resource responsible for the creation - **Payload**: Resource presentation of the Resource to be created	- **Response code**: success (2.xx) or error (4.xx) - **Payload**: the URI of the newly created Resource (when successful).
PUT for CREATE	- **Method code**: PUT (0.03) - **Request URI**: a new URI for the Resource to be created. - **Payload**: Resource presentation of the Resource to be created.	- **Response code**: success (2.xx) or error (4.xx)
POST for UPDATE	- **Method code**: POST (0.02)	- **Response Code**: success (2.xx) or error (4.xx)

	- **Request URI**: an existing URI for the Resource to be updated. - **Payload**: representation of the Resource to be updated.	
DELETE for DELETE	**- Method code**: DELETE (0.04) - **Request URI**: an existing URI for the Resource to be deleted.	- **Response code**: success (2.xx) or error (4.x)

12.2.3.2 CREATE with POST or PUT

12.2.3.2.1 With POST

POST shall be used only in situations where the request URI is valid, that is it is the URI of an existing Resource on the Server that is processing the request. If no such Resource is present, the Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall use an existing request URI which identifies the Resource on the Server responsible for creation. The URI of the created Resource is determined by the Server and provided to the Client in the response.

A Client shall include the representation of the new Resource in the request payload. The new resource representation in the payload shall have all the necessary properties to create a valid Resource instance, i.e. the created Resource should be able to properly respond to the valid Request with mandatory Interface (e.g., GET with ?if=oic.if.baseline).

Upon receiving the POST request, the Server shall either

- create the new Resource with a new URI, respond with the new URI for the newly created Resource and a success response code (2.xx); or
- respond with an error response code (4.xx).

POST is unsafe and is the supported method when idempotent behaviour cannot be expected or guaranteed.

12.2.3.2.2 With PUT

PUT shall be used to create a new Resource or completely replace the entire representation of an existing Resource. The resource representation in the payload of the PUT request shall be the complete representation. PUT for CREATE shall use a new request URI identifying the new Resource to be created.

The new resource representation in the payload shall have all the necessary properties to create a valid Resource instance, i.e. the created Resource should be able to properly respond to the valid Request with mandatory Interface (e.g. GET with ?if=oic.if.baseline).

Upon receiving the PUT request, the Server shall either

- create the new Resource with the request URI provided in the PUT request and send back a response with a success response code (2.xx); or
- respond with an error response code (4.xx).

PUT is an unsafe method but it is idempotent, thus when a PUT request is repeated the outcome is the same each time.

12.2.3.3 RETRIEVE with GET

GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of the target Resource identified by the request URI.

Upon receiving the GET request, the Server shall either

- send back the response with the representation of the target Resource with a success response code (2.xx); or
- respond with an error response code (4.xx) or ignore it (e.g. non-applicable multicast GET).

GET is a safe method and is idempotent.

12.2.3.4 UPDATE with POST

POST shall be used only in situations where the request URI is valid, that is it is the URI of an existing Resource on the Server that is processing the request. If no such Resource is present, the Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE Property values of an existing Resource (see Sections 3.1.32 and 8.4.2).

Upon receiving the request, the Server shall either

- apply the request to the Resource identified by the request URI in accordance with the applied interface (i.e. POST for non-existent Properties is ignored) and send back a response with a success response code (2.xx); or
- respond with an error response code (4.xx). Note that If the representation in the payload is incompatible with the target Resource for POST using the applied interface (i.e. the "overwrite" semantic cannot be honored because of read-only property in the payload), then the error response code 4.xx shall be returned.

POST is unsafe and is the supported method when idempotent behaviour cannot be expected or guaranteed.

12.2.3.5 DELETE with DELETE

DELETE shall be used for DELETE operation. The DELETE method requests that the resource identified by the request URI be deleted.
Upon receiving the DELETE request, the Server shall either

- delete the target Resource and send back a response with a success response code (2.xx); or
- respond with an error response code (4.xx).

DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

12.2.4 Content Type negotiation

The Device framework mandates support of CBOR, however it allows for negotiation of the payload body if more than one encoding type is supported by an implementation. In this case the accept option defined in section 5.10.4 of IETF RFC 7252 shall be used to indicate which content encodings are requested by the Client.

Content types supported are as shown in Table 24.

Table 24. Content Types and Content Formats

Content Type	Content Format
application/xml	41

application/exi	47
application/json defined in IETF RFC 7159	50
application/cbor defined in IETF RFC 7049	60

Note: An OCF vertical can mandate a specific content type.

Server and Client shall send a Content-Format option every time in a message with a payload body. The Content Format option shall use the Content Format numeric value from Table 24.

12.2.5 CRUDN to CoAP response codes

The mapping of CRUDN operations response codes to CoAP response codes are identical to the response codes defined in IETF RFC 7252.

12.2.6 CoAP block transfer

Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT devices. However scenarios can be envisioned in which an application needs to transfer larger payloads.

CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be used by all Servers which generate a content payload that would exceed the size of a CoAP datagram as the result of handling any defined CRUDN operation.

Similarly, CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be supported by all Clients. The use of block-wise transfer is applied to both the reception of payloads as well as transmission of payloads that would exceed the size of a CoAP datagram.

All blocks that are sent using this mechanism for a single instance of a transfer shall all have the same reliability setting (i.e. all confirmable or all non-confirmable).

A Client may support both the block1 (as descriptive) and block2 (as control) options as described by IETF draft-ietf-core-block-18. A Server may support both the block1 (as control) and block2 (as descriptive) options as described by IETF draft-ietf-core-block-18.

12.2.7 CoAP serialization over TCP

12.2.7.1 Introduction

In environments where TCP is already available, CoAP can take advantage of it to provide reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For example, consider a cloud application acting as a Client and the Server is located at the user's home. The Server which already support CoAP as a messaging protocol (e.g., Smart Home vertical profile) could easily support CoAP serialization over TCP rather than adding another messaging protocol. A Device implementing CoAP Serialization over TCP shall conform to IETF draft-tschofenig-core-coap-tcp-tls-04.

12.2.7.2 Indication of support

If UDP is blocked, clients depend on the pre-configured details on the device to find support for CoAP over TCP. If UDP is not-blocked, a Device which supports CoAP serialization over TCP shall populate the Messaging Protocol (mpro) property in oic/res with the value "coap+tcp" or "coaps+tcp" to indicate that the device supports messaging protocol as specified by section 11.3.4.

12.2.7.3 Message type and header

The message type transported between Client and Server shall be a non-confirmable message (NON). The protocol stack used in this scenario shall be as described in section 3 in IETF draft-tschofenig-core-coap-tcp-tls-04.

The CoAP header as described in figure 6 in IETF draft-tschofenig-core-coap-tcp-tls-04 shall be used for messages transmitted between a Client and a Server. A Device shall use “Alternative L3” as defined in IETF draft-tschofenig-core-coap-tcp-tls-04.

12.2.7.4 URI scheme

The URI scheme used shall be as defined in section 6 in IETF draft-tschofenig-core-coap-tcp-tls-04].

For the “coaps+tcp” URI scheme the “TLS Application Layer Protocol Negotiation Extension” IETF RFC 7301 shall be used.

12.2.7.5 KeepAlive

12.2.7.5.1 Overview

In order to ensure that the connection between a Device is maintained, when using CoAP serialization over TCP, a Device that initiated the connection should send application layer KeepAlive messages. The reasons to support application layer KeepAlive are as follows:

- TCP KeepAlive only guarantees that a connection is alive at the network layer, but not at the application layer
- Interval of TCP KeepAlive is configurable only using kernel parameters, and is OS dependent (e.g., 2 hours by default in Linux)

12.2.7.5.2 KeepAlive Mechanism

Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. A Server shall support a resource of type oic.wk.ping as defined in Table 25.

Table 25. Ping resource

Pre-defined URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
/oic/ping	Ping	oic.wk.ping	oic.if.rw	The resource using which a Client keeps its Connection with a Server active. The resource properties exposed by /oic/ping are listed in Table 26.	KeepAlive

Table 26 defines oic.wk.ping resource type.

Table 26. oic.wk.ping resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Interval	in	integer	minutes		R,W	yes	The time interval for which connection shall be kept alive and not closed.

The following steps detail the KeepAlive mechanisms for a Client and Server:

1) A Client which wants to keep the connection with a Server alive shall send a PUT request to /oic/ping resource on the Server updating its connection Interval.
    a) This time interval shall start from 2 minutes and increases in multiples of 2 up to a maximum of 64 minutes. It stays at 64 minutes from that point.
2) An Server receiving this ping request shall respond within 1 minute.
3) If a Client does not receive the response within 1 minute, it shall terminate the connection.
4) If an Server does not receive a PUT request to ping resource within the specified "interval" time, the Server shall terminate the connection.

An example of the KeepAlive mechanism is as follows:

- Client → Server: PUT /oic/ping {interval: 2}
- Server → Client: 2.03 valid

12.3 Payload Encoding in CBOR

OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to JSON from CBOR in accordance with IETF RFC 7049 section 4 unless otherwise specified in this section.

Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer numbers shall be within the open range (-2^53, 2^53). Properties defined as a JSON number should be encoded as integers whenever possible; if this is not possible Properties defined as a JSON number should use single-precision if the loss of precision does not affect the quality of service, otherwise the Property shall use double-precision.

On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values in any position. If a property defined as a JSON integer is received encoded other than as an integer, the implementation may reject this encoding using a final response as appropriate for the underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a property is defined as a JSON number an implementation shall accept integers, single- and double-precision floating point.

13 Security

The details for handling security and privacy are specified in [OCF Security].

14 Multi resource model support

14.1 Interoperability issue

14.1.1 Multiple IoT Standards

Note: Alignment and interoperability between models will be added in a later version of the specification.

IoT requires standardization for interoperability among diverse devices and multiple standards are under development currently. IETF defines network and web transfer protocol (e.g. 6lowpan [RFC6775] and CoAP [RFC6690], [RFC7252]), oneM2M [oneM2M] produces technical specifications for a common M2M Service Layer [oneM2M-TS0001], [oneM2M-TS0004] and IPSO Alliance [IPSO] publishes Smart Object Guideline [IPSOSmartObjects].

Multitude of IoT standards are based on "Representational State Transfer (REST)", which is a software architecture style with a coordinated set of constraints for the design of components in a distributed hypermedia system [REST]. In REST based IoT, a real world entity is represented as resource in a server, which a client accesses and manipulates the resource through representations to interact with the entity, i.e. sensing and controlling the physical environments. Moreover several IoT standards adopt the common network and web transfer protocols. oneM2M, IPSO and OCF all use CoAP and IP/ UDP, [oneM2M-TS0008], [IPSO], [OCF] so any client and server supporting those standards can exchange request and response messages.

However in order to interact properly, it's not sufficient for IoT devices to be able to transfer CoAP messages. IoT devices should understand each other's resources and be aware of their semantic meaning and syntactic form. Currently each standard defines its own "resource model" and specifies a different scheme to construct resources from physical entities such as light [OCF], [IPSOFramework], [IPSOSmartObjects], [oneM2M-TS0001]. Hence client and server adopting different standards can't perform meaningful interaction, i.e. the client can't manipulate the resource representation in the server.

For wider interoperability among multiple standards, IoT devices need to understand each other's resource model to process CoAP request and response message properly. To interpret resources correctly, client and server need to determine which resource model each other follows in the first place. The client should be aware of whether its corresponding server adopts oneM2M or OCF model and vice versa.

14.1.2 Different resource models

OCF specification follows a resource oriented architecture with RESTful architectural style. Without common understanding on resource model, two IoT devices can’t interact with each other.

Currently multiple organizations such as OCF, IPSO Alliance or oneM2M, define their own resource model in difference ways, which may restrict interoperability to the respective ecosystems. The main discrepancies are as follows

- **Resource structure:** Some define resource to have attributes (e.g. oneM2M), whereas others define it atomic and not decomposed into attributes (e.g. IPSO alliance). For example, a smart light may be represented as a resource with on-off attribute or a resourcecollection with on-off resource. In the former, on-off attribute doesn’t have URI and should be accessed indirectly via the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and can be directly manipulated.

- **Resource name & type:** Some allow resource to be named freely and indicate its characteristic with separate resource type attribute (e.g. oneM2M). Whereas others fix the name ofresource a priori and indicate its characteristic with the name itself (e.g IPSOalliance). For example, smart light can be named anyway such as ‘LivingRoomLight_1” in oneM2M but should have the fixed Object name with numerical Object ID of “IPSO Light Control (3311)” in IPSO alliance. Furthermore, in consequence, it’s likely that data path in URI is freely defined in the former and predetermined for the latter.

- **Resource hierarchy:** Some allow resource to be organized in hierarchy so that resource includes another resource in itself with parent-child relationship (e.g. oneM2M). Whereas others mandate resource to be of flat structure and associate with other resources only by referencing their links.

In addition to the above, different organizations use different syntax and have different features (e.g. resource interface), which will inhibit IoT interoperability. When IoT client and server don’t understand the resource model each supports, they can’t perform RESTful transaction.

For example, a smart light can be represented as an IPSO Smart Object in JSON as below:

```
{
  "3311": {
    "description":     "IPSO     light
control",
    "instances": {
      "0": {
        "resources": {
          "5850": {
            "description": "On/Off",
            "value": 0
          },
          "5851": {
            "description": "Dimmer",
            "value": 70
          }
        }
      }
    }
  }
}
```

In the above, "3311" is an "Object ID" defining object type, 0" an "Object Instance", designating one or more instances, "5850", "5851", "Resource ID", defining resource type. Also IPSO embeds resource information in data path, so "On/Off" resource has predetermined data path of "3311/0/5850" and "Dimmer" resource datapath of "3311/0/5851"

Whereas the same smart light may be represented in OCF as two Resources.

```
{
  "n": "myLightSwtich",
  "rt": "oic.r.switch.binary",
  "value": True
}
```

```
{
"n": "myLightBrightness",
  "rt":
"oic.r.light.brightness",
  "brightness": 70
}
```


14.2 A scheme to exchange resource model information

14.2.1 A scheme to exchange resource model information

IoT devices, i.e. client and server, need to understand the resource model which their corresponding device supports to be able to interoperate each other.

For the initial step, it would help for IoT devices to indicate resource model each device supports. Then client and server may choose a common resource model for interaction, or in the absence of such a common model, rely on translation between the models, possibly with the assistance of 3rd party such as intermediary. Alignment and interoperability between models will be added in a later version of the specification.

This document presents a scheme for CoAP endpoints, client and server, to exchange resource model they support.

First, the Internet media type and Content-Format identifier are used to indicate a specific resource model. The Internet media types can be defined to indicate the resource models, potentially with content-coding, such as "application/ipso+json", then assigned numeric Content-Format identifiers such as "123123" to minimize payload overhead for CoAP usage.

Second, CoAP Accept and Content-Format Option are used to exchange the Content-Format identifiers indicating the resource models which CoAP endpoints prefer or support. A client includes the CoAP Accept option to inform a server which resource model, potentially with content-encoding, is acceptable and the server returns the payload in the preferred resource model if available. The Content-Format Option indicates the resource model which the payload follows.

Annex A
(informative)

Operation Examples

A.1 Introduction

This section describes some example scenarios using sequence of operations between the entities involved. In all the examples below “Light” is a Server and “Smartphone” is a Client. In one of the scenario “Garage” additionally acts as a Server. All the examples are based on the following example resource definitions:

rt=oic.example.light with resource type definition as illustration in Table 27.

Table 27. oic.example.light resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
on-off	of	boolean			R, W	yes	On/Off Control: 0 = Off 1 = On
dim	dm	integer	0-255		R, W	yes	Resource which can take a range of values minimum being 0 and maximum being 255

rt=oic.example.garagedoor with resource type definition as illustration in Table 28.

Table 28. oic.example.garagedoor resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
open-close	oc	boolean			R, W	yes	Open/Close Control: 0 = Open 1 = Close

/oic/mnt (rt=oc.wk.mnt) used in below examples is defined in section 11.5.1.

A.2 When at home: From smartphone turn on a single light

This sequence highlights (Figure 37) the discovery and control of an OCF light resource from an OCF smartphone.

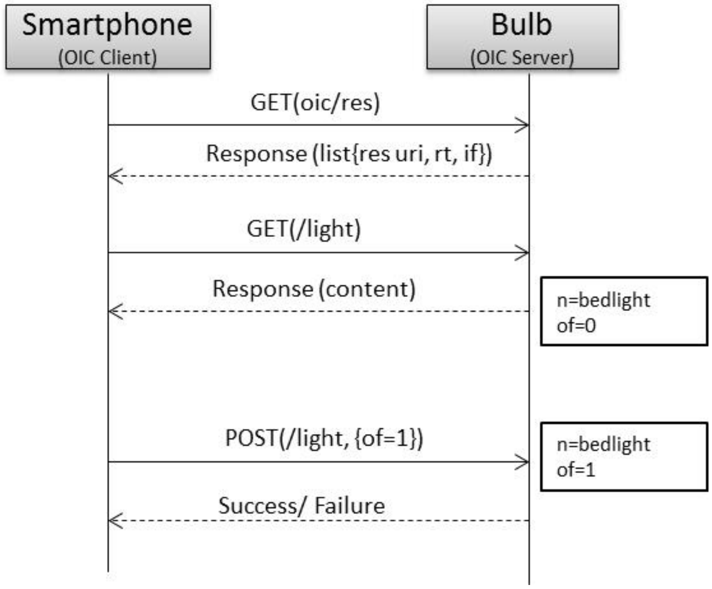

Figure 37. When at home: from smartphone turn on a single light

Discovery request can be sent to “All OCF Nodes” Multicast address FF0X::158 or can be sent directly to the IP address of device hosting the light resource.

1) Smartphone sends a GET request to /oic/res resource to discover all resources hosted on targeted end point
2) The end point (bulb) responds with the list of resource URI, resource type and interfaces supported on the end point (one of the resource is ‘/light’ whose rt=oic.example.light)
3) Smartphone sends a GET request to ’/light’ resource to know its current state
4) The end point responds with representation of light resource ({n=bedlight;of=0})
5) Smartphone changes the ‘of’ property of the light resource by sending a POST request to ‘/light’ resource ({of=1})
6) On Successful execution of the request, the end point responds with the changed resource representation. Else, error code is returned. Details of the error codes are defined in section 12.2.5.

A.3 GroupAction execution

This example will be added when groups feature is added in later version of specification

A.4 When garage door opens, turn on lights in hall; also notify smartphone

This example will be added when scripts feature is added in later version of specification

A.5 Device management

This sequence highlights (Figure 38) the device management function of maintenance.

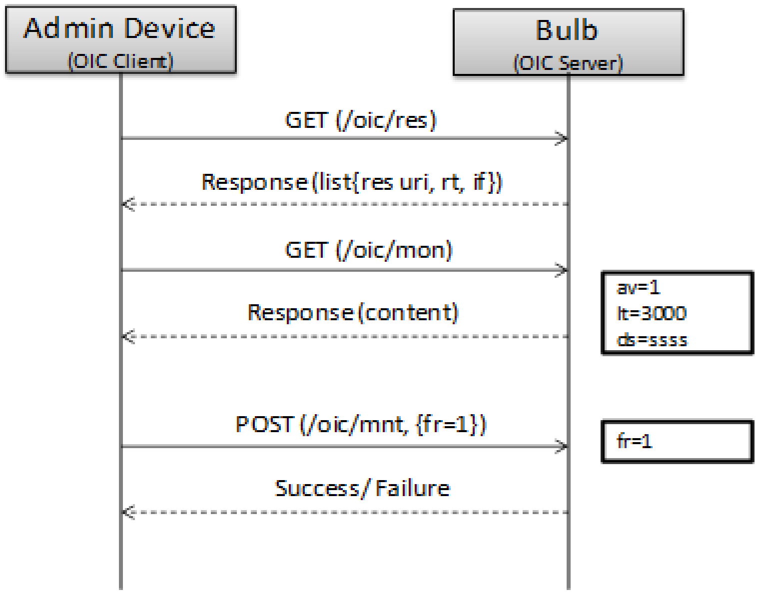

Figure 38. Device management (maintenance)

Pre-Condition: Admin device has different security permissions and hence can perform device management operations on the Device

1) Admin device sends a GET request to /oic/res resource to discover all resources hosted on a targeted end point (in this case Bulb)
2) The end point (bulb) responds with the list of resource URI, resource type and interfaces supported on the end point (one of the resources is /oic/mnt whose rt=oc.wk.mnt)
3) Admin Device changes the 'fr' property of the maintenance resource by sending a POST request to /oic/mnt resource ({fr=1}). This triggers a factory reset of the end point (bulb)
4) On successful execution of the request, the end point responds with the changed resource representation. Else, error code is returned. Details of the error codes are defined in section 12.2.5.

Annex B
(informative)

OCF interaction scenarios and deployment models

B.1 OCF interaction scenarios

A Client connects to one or multiple Servers in order to access the resources provided by those Servers. The following are scenarios representing possible interactions among Roles:

- Direct interaction between Client and Server (Figure 39). In this scenario the Client and the Server directly communicate without involvement of any other Device. A smartphone which controls an actuator directly uses this scenario.

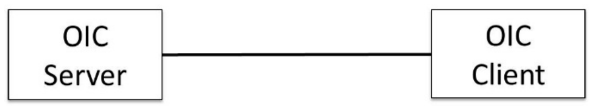

Figure 39. Direct interaction between Server and Client

- Interaction between Client and Server using another server (Figure 40). In this scenario, another Server provides the support needed for the Client to directly access the desired resource on a specific Server. This scenario is used for example, when a smartphone first accesses a discovery server to find the addressing information of a specific appliance, and then directly accesses the appliance to control it.

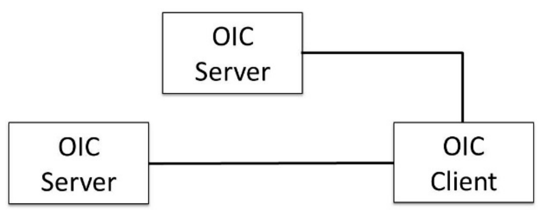

Figure 40. Interaction between Client and Server using another Server

- Interaction between Client and Server using Intermediary (Figure 41). In this scenario an Intermediary facilitates the interaction between the Client and the Server. A smartphone which controls appliances in a smart home via MQTT broker uses this scenario.

Figure 41. Interaction between Client and Server using Intermediary

- Interaction between Client and Server using support from multiple Servers and intermediary (Figure 42). In this scenario, both Server and Intermediary roles are present to facilitate the transaction between the Client and a specific Server. An example scenario is when a smartphone first accesses a Resource Directory (RD) server to find the address to a specific appliance, then utilizes MQTT broker to deliver a command message to the appliance. The smartphone can utilize the mechanisms defined in CoRE Resource Directory such as default location, anycast address or DHCP (IETF draft-ietf-core-resource-directory-02) to discover the Resource Directory information.

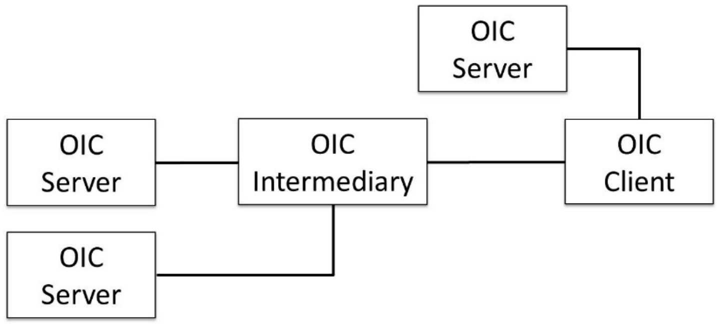

Figure 42. Interaction between Client and Server using support from multiple Servers and Intermediary

B.2 Deployment model

In deployment, Devices are deployed and interact via either wired or wireless connections. Devices are the physical entities that may host resources and play one or more Roles. There is no constraint on the structure of a deployment or number of Devices in it. Architecture is flexible and scalable and capable of addressing large number of devices with different device capabilities, including constrained devices which have limited memory and capabilities. Constrained devices are defined and categorized in [TCNN].

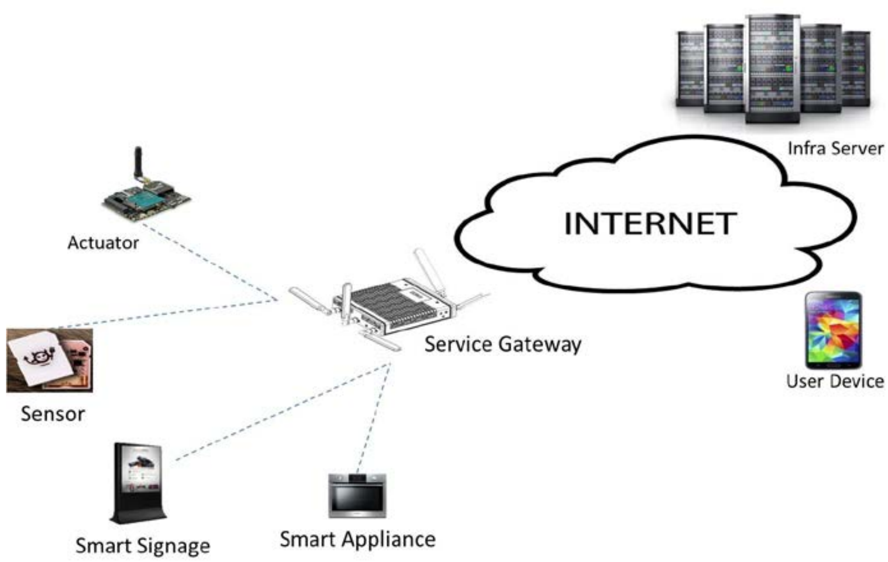

Figure 43. Example of Devices

Figure 43 depicts a typical deployment and set of Devices, which may be divided in the following categories:

- **Things**: Networked devices which are able to interface with physical environments. Things are the devices which are primarily controlled and monitored. Examples include smart appliances, sensors, and actuators. Things mostly take the role of Sever but they may also take the role of Client, for example in machine-to-machine communications.
- **User Devices**: Devices employed by the users enabling the users to access resources and services. Examples include smart phones, tablets, and wearable devices. User Devices mainly take the role of Client, but may also take the role of Server or Intermediary.

- **Service Gateways**: Network equipment which take the role of Intermediary. Examples are home gateways.
- **Infra Servers**: Data centers residing in cloud infrastructure, which facilitate the interaction among Devices by providing network services such as AAA, NAT traversal or discovery. It can also play the role of Client or Intermediary

Annex C
(informative)

Other Resource Models and OCF Mapping

C.1 Multiple resource models

RESTful interactions are defined dependent on the resource model; hence, Devices require a common understanding of the resource model for interoperability.

There are multiple resource models defined by different organizations including OCF, IPSO Alliance and oneM2M, and used in the industry, which may restrict interoperability among respective ecosystems. The main differences from Resource model are as follows:

- **Resource structure**: Resources may be defined to have properties (e.g., oneM2M defined resources), or may be defined as an atomic entity and not be decomposable into properties (e.g., IPSO alliance defined resources). For example, a smart light may be represented as a resource with an on-off property or a resource collection containing an on-off resource. In the former, on-off property doesn't have a URI of its own and can only be accessed indirectly via the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and can be directly manipulated.
- **Resource name & type**: Resources may be allowed to be named freely and have their characteristics indicated using a resource type property (e.g., as defined in oneM2M). Alternatively, the name of resources may be defined a priori in a way that the name by itself is indicative of its characteristic (e.g., as defined by IPSO alliance). For example, in oneM2M resource model, a smart light can be named with no restrictions, such as 'LivingRoomLight_1" but in IPSO alliance resource model it is required to have the fixed Object name with numerical Object ID of "IPSO Light Control (3311)". Consequently, it's likely that in the former case the data path in URI is freely defined and in the latter case it is predetermined.
- **Resource hierarchy**: Resources may be allowed to be organized in hierarchy where a resource contains another resource with a parent-child relationship (e.g., in oneM2M definition of resource model). Resources may also be required to have a flat structure and associate with other resources only by referencing their links.

In addition to the above, different organizations use different syntax and define different features (e.g., resource interface), which preclude interoperability.

C.2 OCF approach for support of multiple resource models

In order to expand the IoT ecosystem the Framework takes an inclusive approach for interworking with existing resource models. Specifically, the Framework defines a resource model while providing a mechanism to easily map to other models. By embracing existing resource models OCF is inclusive of existing ecosystems while allowing for the transition toward definition of a comprehensive resource model integrating all ecosystems.

The following OCF characteristics enable support of other resource models:

- **resource model is the superset of multiple models**: the resource model is defined as the superset of existing resource models. In other words, any existing resource model can be mapped to a subset of resource model concepts.
- **Framework may allow for resource model negotiation**: the Client and Server exchange the information about what resource model(s) each supports. Based on the exchanged information, the Client and Server choose a resource model to perform RESTful interactions or to perform translation. This feature is out of scope of the current version of this specification, however, the following is a high level description for resource model negotiation..

C.3 Resource model indication

The Client and server exchange the information about what resource model(s) each supports. Based on the exchanged information, the Client and Server choose a resource model to perform RESTful interactions or to perform translation. The exchange could be part of discovery and negotiation. Based on the exchange, the Client and Server follow a procedure to ensure interoperability among them. They may choose a common resource model or execute translation between resource models.

- **Resource model schema exchange**: The Client and Server may share the resource model information when they initiate a RESTful interaction. They may exchange the information about which resource model they support as part of session establishment procedures. Alternatively, each request or response message may carry the indication of which resource model it is using. For example, [COAP] defines “Content-Format option” to indicate the “representation format” such as “application/json”. It’s possible to extend the Content-Format Option to indicate the resource model used with the representation format such as “application/ipso-json”.
- **Ensuing procedures**: After the Client and Server exchange the resource model information, they perform a suitable procedure to ensure interoperability among them. The simplest way is to choose a resource model supported by both the Client and Server. In case there is no common resource model, the Client and Server may interact through a 3rd party.

In addition to translation which can be resource intensive, a method based on profiles can be used in which an OCF implementation can accommodate multiple profiles and hence multiple ecosystems.

- **Resource Model Profile**: the Framework defines resource model profiles and implementers or users choose the active profile. The chosen profile constraints the Device to strict rules in how resources are defined, instantiated and interacted with. This would allow for interoperation with devices from the ecosystem identified by the profile (e.g., IPSO, OneM2M etc.). Although this enables a Device to participate in and be part of any given ecosystem, this scheme does not allow for generic interoperability at runtime. While this approach may be suitable for resource constrained devices, more resource capable devices are expected to support more than one profile.

C.4 An Example Profile (IPSO profile)

IPSO defines smart objects that have specific resources and they take values determined by the data type of that resource. The smart object specification defines a category of such objects. Each resource represents a characteristic of the smart object being modelled.

While the terms may be different, there are equivalent concepts in OCF to represent these terms. This section provides the equivalent OCF terms and then frames the IPSO smart object in OCF terms.

The IPSO object Light Control defined in Section 16 of the IPSO Smart Objects 1.0 is used as the reference example.

C.4.1 Conceptual equivalence

The IPSO smart object definition is equivalent to an Resource Type definition which defines the relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent to a Property that like an IPSO Resource has a defined data type, enumeration of acceptable values, units, a general description and access modes (based on the Interface).

The general method for developing the equivalent Resource Type from an IPSO Smart Object definition is to ignore the Object ID and replace the Object URN with and OCF ‘.’ (dot) separated name that incorporates the IPSO object. Alternatively the Object URN can be used as the Resource

Type ID as is (as long as the URN does not contain any '.' (dots)) – using the same Object URN as the Resource Type ID allows for compatibility when interacting with an IPSO compliant device. The object URN based naming does not have any bearing for OCF to OCF interoperability and so the OCF format is preferred – for OCF to OCF interoperability only the data model consistency is required.

Two models are available to render IPSO objects into OCF.

1) One is where the IPSO Smart Object represents a Resource. In this case, the IP Smart Object is regarded as a resource with the Resource Type matching the description of the Smart Object. Furthermore, each resource in the IPSO definition is represented as an Property in the Resource Type (the IPSO Resource ID is replaced with a string representing the Property). This is the preferred approach when the IPSO Data Model is expressed in the Resource Model.
2) The other approach is to model an IPSO Smart Object as an Collection. Each IPSO Resource is then modelled as an Resource with an Resource Type that matches the definition of the IPSO Resource. Each of these resource instances are then bound to the Collection that represents this IPSO Smart Object.

Below is an example showing how an IPSO LightControl Object is modelled as a Resource.

Resource Type: Light Control

Description: This Object is used to control a light source, such as a LED or other light. It allows a light to be turned on or off and its dimmer setting to be controlled as a percentage value between 0 and 100. An optional colour setting enables a string to be used to indicate the desired colour. Table 29 and Table 30 define the resource type and its properties, respectively.

Table 29. Light control resource type definition

Resource Type	Resource Type ID	Multiple Instances	Description
Light Control	"oic.light.control" or "urn:oma:lwm2m:ext:3311"	Yes	Light control object with on/off and optional dimming and energy monitor

Table 30. Light control resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
On/Off	"on-off"	boolean			R, W	yes	On/Of Control: 0 = Off 1 = On
Dimmer	"dim"	integer		%	R, W	no	Proportional Control, integer value between 0 and 100 as percentage
Color	"color"	string	0 – 100	Defined by "units" property	R, W	no	String representing some value in color space
Units	"units"	string			R	no	Measurement Units Definition e.g., "Cel" for Temperature in Celsius.
On Time	"ontime"	integer		s	R, W	no	The time in seconds that the light has been on.

							Writing a value of 0 resets the counter
Cumulative active power	“cumap”	float		Wh	R	no	The cumulative active power since the last cumulative energy reset or device start
Power Factor	“powfact”	float			R	no	The power factor of the load

Annex D
(normative)

Resource Type definitions

D.1 List of resource type definitions

Table 31 contains the list of defined core resources in this specification.

Table 31. Alphabetized list of core resources

Friendly Name (informative)	Resource Type (rt)	Section
Collections	oic.wk.col	D.2
Configuration	oic.wk.con	D.3
Device	oic.wk.d	D.4
Discoverable Resources	oic.wk.res	D.8
Maintenance	oic.wk.mnt	D.5
Platform	oic.wk.p	D.6
Ping	oic.wk.ping	D.7
Resource Directory	oic.wk.rd	D.12
Scenes (Top Level)	oic.wk.sceneList	D.9
Scenes Collections	oic.wk.sceneCollection	D.10
Scenes Member	oic.wk.sceneMember	D.11

D.2 OCF Collection

D.2.1 Introduction

OCF Collection Resource Type contains properties and links. The oic.if.baseline interface exposes a representation of the links and the properties of the collection resource itself

D.2.2 Fixed URI

/CollectionBaselineInterfaceURI

D.2.3 Resource Type

The resource type (rt) is defined as: oic.wk.col.

D.2.4 RAML Definition

```
#%RAML 0.8
title: Collections
version: 1.0
traits:
 - interface-ll :
     queryParameters:
       if:
         enum: ["oic.if.ll"]
 - interface-b :
     queryParameters:
       if:
         enum: ["oic.if.b"]
 - interface-baseline :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/CollectionBaselineInterfaceURI:
  description: |
    OCF Collection Resource Type contains properties and links.
    The oic.if.baseline interface exposes a representation of
    the links and the properties of the collection resource itself

  is : ['interface-baseline']
  get:
    description: |
      Retrieve on Baseline Interface

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
schema.json#",
                  "title": "Collection",
                  "definitions": {
                      "uuid": {
                          "type":"string",
                          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
[a-fA-F0-9]{12}$"
                      },
                      "oic.collection.setoflinks": {
                          "description": "A set (array) of simple or individual OIC Links. In
addition to properties required for an OIC Link, the identifier for that link in this set is also
required",
                          "type": "array",
                          "items": {
                              "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
                          }
                      },
                      "oic.collection.tags": {
                          "type": "object",
                          "description": "The tags that can be used for tagging links in a
collection",
```

```
                        "properties": {
                            "n": {
                                "type": "string",
                                "description": "Used to name i.e. tag the set of links"
                            },
                            "id": {
                                "description": "Id for each set of links i.e. tag. Can be an value that is unique to the use context or a UUIDv4",
                                "anyOf": [
                                    {
                                        "type": "integer",
                                        "description": "A number that is unique to that collection; like an ordinal number that is not repeated"
                                    },
                                    {
                                        "type": "string",
                                        "description": "A unique string that could be a hash or similarly unique"
                                    },
                                    {
                                        "$ref": "#/definitions/uuid",
                                        "description": "A unique string that could be a UUIDv4"
                                    }
                                ]
                            },
                            "di": {
                                "$ref": "#/definitions/uuid",
                                "description": "The device ID which is an UUIDv4 string"
                            },
                            "base": {
                                "type": "string",
                                "description": "The base URI to be used if the links are relative URIs  (i.e. relative references); see base URI in Core spec for details",
                                "format": "uri"
                            }
                        },
                        "minProperties": 1
                    },
                    "oic.collection.tagged-setoflinks": {
                        "type": "array",
                        "description": "A tagged link is a set (array) of links that are tagged with one or more key-value pairs usually either an ID or Name or both",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tags"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.setof-tagged-setoflinks": {
                        "type": "array",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.alllinks": {
                        "description": "All forms of links in a collection",
                        "oneOf": [
                            {
                                "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            },
                            {
```

```
                                    "$ref": "#/definitions/oic.collection.setoflinks"
                                }
                            ]
                        },
                        "oic.collection": {
                            "type": "object",
                            "description": "A collection is a set (array) of tagged-link or set
(array) of simple links along with additional properties to describe the collection itself",
                            "properties": {
                                "n": {
                                    "type": "string",
                                    "description": "User friendly name of the
collection"                 },
                                "id": {
                                    "anyOf": [
                                        {
                                            "type": "integer",
                                            "description": "A number that is unique to that
collection; like an ordinal number that is not repeated"
                                        },
                                        {
                                            "type": "string",
                                            "description": "A unique string that could be a hash or
similarly unique"
                                        },
                                        {
                                            "$ref": "#/definitions/uuid",
                                            "description": "A unique string that could be a UUIDv4"
                                        }
                                    ],
                                    "description": "ID for the collection. Can be an value that is
unique to the use context or a UUIDv4"
                                },
                                "di": {
                                    "$ref": "#/definitions/uuid",
                                    "description": "The device ID which is an UUIDv4 string; used for
backward compatibility with Spec A defintion of /oic/res"
                                },
                                "rts": {
                                    "type": "string",
                                    "description": "Defines the list of allowable resource types (for
Target and anchors) in links included in the collection; new links being created can only be from
this list"                },
                                "drel": {
                                    "type": "string",
                                    "description": "When specified this is the default relationship
to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
                                },
                                "links": {
                                    "$ref": "#/definitions/oic.collection.alllinks"
                                }
                            }
                        }
                    },
                    "type": "object",
                    "allOf": [
                        {
                            "$ref": "#/definitions/oic.collection"
                        }
                    ]
                }

            example: |
              {
                "rt": ["oic.wk.col"],
                "id": "unique_example_id",
                "rts": [ "oic.r.switch.binary", "oic.r.airFlow" ],
                "links": [
                  {
                    "href": "switch",
```

```
                    "rt":   "oic.r.switch.binary",
                    "if":   "oic.if.a"
                  },
                  {
                    "href": "airFlow",
                    "rt":   "oic.r.airFlow",
                    "if":   "oic.if.a"
                  }
                ]
              }

  post:
    description: |
      Update on Baseline Interface

    body:
      application/json:
        schema: |
          {
              "$schema": "http://json-schema.org/draft-04/schema#",
              "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
              "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-schema.json#",
              "title": "Collection",
              "definitions": {
                  "uuid": {
                      "type":"string",
                      "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$"
                  },
                  "oic.collection.setoflinks": {
                      "description": "A set (array) of simple or individual OIC Links. In addition to properties required for an OIC Link, the identifier for that link in this set is also required",
                      "type": "array",
                      "items": {
                          "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
                      }
                  },
                  "oic.collection.tags": {
                      "type": "object",
                      "description": "The tags that can be used for tagging links in a collection",
                      "properties": {
                          "n": {
                              "type": "string",
                              "description": "Used to name i.e. tag the set of links"
                          },
                          "id": {
                              "description": "Id for each set of links i.e. tag. Can be an value that is unique to the use context or a UUIDv4",
                              "anyOf": [
                                  {
                                      "type": "integer",
                                      "description": "A number that is unique to that collection; like an ordinal number that is not repeated"
                                  },
                                  {
                                      "type": "string",
                                      "description": "A unique string that could be a hash or similarly unique"
                                  },
                                  {
                                      "$ref": "#/definitions/uuid",
                                      "description": "A unique string that could be a UUIDv4"
                                  }
                              ]
                          },
                          "di": {
```

```
                            "$ref": "#/definitions/uuid",
                            "description": "The device ID which is an UUIDv4 string"
                        },
                        "base": {
                            "type": "string",
                            "description": "The base URI to be used if the links are relative
URIs  (i.e. relative references); see base URI in Core spec for details",
                            "format": "uri"
                        }
                    },
                    "minProperties": 1
                },
                "oic.collection.tagged-setoflinks": {
                    "type": "array",
                    "description": "A tagged link is a set (array) of links that are tagged with
one or more key-value pairs usually either an ID or Name or both",
                    "items": [
                        {
                            "$ref": "#/definitions/oic.collection.tags"
                        },
                        {
                            "$ref": "#/definitions/oic.collection.setoflinks"
                        }
                    ],
                    "additionalItems": false
                },
                "oic.collection.setof-tagged-setoflinks": {
                    "type": "array",
                    "items": [
                        {
                            "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                        }
                    ],
                    "additionalItems": false
                },
                "oic.collection.alllinks": {
                    "description": "All forms of links in a collection",
                    "oneOf": [
                        {
                            "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
                        },
                        {
                            "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                        },
                        {
                            "$ref": "#/definitions/oic.collection.setoflinks"
                        }
                    ]
                },
                "oic.collection": {
                    "type": "object",
                    "description": "A collection is a set (array) of tagged-link or set (array)
of simple links along with additional properties to describe the collection itself",
                    "properties": {
                        "n": {
                            "type": "string",
                            "description": "User friendly name of the
collection"                },
                        "id": {
                            "anyOf": [
                                {
                                    "type": "integer",
                                    "description": "A number that is unique to that collection;
like an ordinal number that is not repeated"
                                },
                                {
                                    "type": "string",
                                    "description": "A unique string that could be a hash or
similarly unique"
                                },
                                {
```

```
                                        "$ref": "#/definitions/uuid",
                                        "description": "A unique string that could be a UUIDv4"
                                    }
                                ],
                                "description": "ID for the collection. Can be an value that is unique
to the use context or a UUIDv4"
                            },
                            "di": {
                                "$ref": "#/definitions/uuid",
                                "description": "The device ID which is an UUIDv4 string; used for
backward compatibility with Spec A defintion of /oic/res"
                            },
                            "rts": {
                                "type": "string",
                                "description": "Defines the list of allowable resource types (for
Target and anchors) in links included in the collection; new links being created can only be from
this list"              },
                            "drel": {
                                "type": "string",
                                "description": "When specified this is the default relationship to
use when an OIC Link does not specify an explicit relationship with *rel* parameter"
                            },
                            "links": {
                                "$ref": "#/definitions/oic.collection.alllinks"
                            }
                        }
                    }
                },
                "type": "object",
                "allOf": [
                    {
                        "$ref": "#/definitions/oic.collection"
                    }
                ]
            }

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
schema.json#",
                  "title": "Collection",
                  "definitions": {
                      "uuid": {
                          "type":"string",
                          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
[a-fA-F0-9]{12}$"
                      },
                      "oic.collection.setoflinks": {
                          "description": "A set (array) of simple or individual OIC Links. In
addition to properties required for an OIC Link, the identifier for that link in this set is also
required",
                          "type": "array",
                          "items": {
                              "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
                          }
                      },
                      "oic.collection.tags": {
                          "type": "object",
                          "description": "The tags that can be used for tagging links in a
collection",
                          "properties": {
```

```
                            "n": {
                                "type": "string",
                                "description": "Used to name i.e. tag the set of links"
                            },
                            "id": {
                                "description": "Id for each set of links i.e. tag. Can be an
value that is unique to the use context or a UUIDv4",
                                "anyOf": [
                                    {
                                        "type": "integer",
                                        "description": "A number that is unique to that
collection; like an ordinal number that is not repeated"
                                    },
                                    {
                                        "type": "string",
                                        "description": "A unique string that could be a hash or
similarly unique"
                                    },
                                    {
                                        "$ref": "#/definitions/uuid",
                                        "description": "A unique string that could be a UUIDv4"
                                    }
                                ]
                            },
                            "di": {
                                "$ref": "#/definitions/uuid",
                                "description": "The device ID which is an UUIDv4 string"
                            },
                            "base": {
                                "type": "string",
                                "description": "The base URI to be used if the links are relative
URIs  (i.e. relative references); see base URI in Core spec for details",
                                "format": "uri"
                            }
                        },
                        "minProperties": 1
                    },
                    "oic.collection.tagged-setoflinks": {
                        "type": "array",
                        "description": "A tagged link is a set (array) of links that are tagged
with one or more key-value pairs usually either an ID or Name or both",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tags"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.setof-tagged-setoflinks": {
                        "type": "array",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.alllinks": {
                        "description": "All forms of links in a collection",
                        "oneOf": [
                            {
                                "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.setoflinks"
```



```
                                }
                            ]
                        },
                        "oic.collection": {
                            "type": "object",
                            "description": "A collection is a set (array) of tagged-link or set
(array) of simple links along with additional properties to describe the collection itself",
                            "properties": {
                                "n": {
                                    "type": "string",
                                    "description": "User friendly name of the
collection"                  },
                                "id": {
                                    "anyOf": [
                                        {
                                            "type": "integer",
                                            "description": "A number that is unique to that
collection; like an ordinal number that is not repeated"
                                        },
                                        {
                                            "type": "string",
                                            "description": "A unique string that could be a hash or
similarly unique"
                                        },
                                        {
                                            "$ref": "#/definitions/uuid",
                                            "description": "A unique string that could be a UUIDv4"
                                        }
                                    ],
                                    "description": "ID for the collection. Can be an value that is
unique to the use context or a UUIDv4"
                                },
                                "di": {
                                    "$ref": "#/definitions/uuid",
                                    "description": "The device ID which is an UUIDv4 string; used for
backward compatibility with Spec A defintion of /oic/res"
                                },
                                "rts": {
                                    "type": "string",
                                    "description": "Defines the list of allowable resource types (for
Target and anchors) in links included in the collection; new links being created can only be from
this list"                },
                                "drel": {
                                    "type": "string",
                                    "description": "When specified this is the default relationship
to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
                                },
                                "links": {
                                    "$ref": "#/definitions/oic.collection.alllinks"
                                }
                            }
                        }
                    },
                    "type": "object",
                    "allOf": [
                        {
                            "$ref": "#/definitions/oic.collection"
                        }
                    ]
                }
```

D.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
href	string	yes	Read Write	This is the target URI, it can be specified as a Relative

				Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique.
rel	string		Read Write	
rt	array	yes	Read Write	
if	array	yes	Read Write	
di	string		Read Write	The Device ID on which the Relative Reference in href is to be resolved on. Base URI should be used in preference where possible
buri	string		Read Write	The base URI used to fully qualify a Relative Reference in the href parameter. Use the OCF Schema for URI
p			Read Write	Specifies the framework policies on the Resource referenced by the target URI
bm		yes	Read Write	Specifies the framework policies on the Resource referenced by the target URI for e.g. observable and discoverable
sec			Read Write	Specifies if security needs to be turned on when looking to interact with the Resource
port			Read Write	Secure port to be used for connection
bp	string		Read Write	Batch Parameters: Uri Parameters To Use With An Oic.If.B Batch

				Request Using This Link
anchor	string		Read Write	This is used to override the context URI e.g. override the URI of the containing collection
ins	object		Read Write	

D.2.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/CollectionBaselineInterfaceURI		get	post		

D.2.7 Referenced JSON schemas

D.2.8 oic.oic-link-schema.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
  "definitions": {
    "oic.oic-link": {
      "type": "object",
      "properties": {
        "href": {
          "type": "string",
          "maxLength": 256,
          "description": "This is the target URI, it can be specified as a Relative Reference or
fully-qualified URI. Relative Reference should be used along with the di parameter to make it
unique.",
          "format": "uri"
        },
        "rel": {
          "type": "string",
          "default": "hosts",
          "maxLength": 64,
          "description": "The relation of the target URI referenced by the link to the context URI"
        },
        "rt": {
          "type": "array",
          "items" : [
            {
              "type" : "string",
              "maxLength": 64
            }
          ],
          "minItems" : 1,
          "readOnly": true,
          "description": "Resource Type"
        },
        "if": {
          "type": "array",
          "items": [
            {
              "type" : "string",
              "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
"oic.if.a", "oic.if.s" ]
            }
          ],
          "minItems": 1,
          "readOnly": true,
          "description": "The interface set supported by this resource"
        },
        "di": {
           "type": "string",
           "description": "The Device ID on which the Relative Reference in href is to be resolved
```

```
on. Base URI should be used in preference where possible",
           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$"
        },
        "buri": {
            "type": "string",
          "description": "The base URI used to fully qualify a Relative Reference in the href
parameter. Use the OCF Schema for URI",
          "maxLength": 256,
          "format": "uri"
        },
        "p": {
          "readOnly": true,
          "description": "Specifies the framework policies on the Resource referenced by the target
URI",
          "type": "object",
          "properties": {
            "bm": {
              "readOnly": true,
              "description": "Specifies the framework policies on the Resource referenced by the
target URI for e.g. observable and discoverable",
              "type": "integer"
            },
            "sec": {
              "readOnly": true,
              "description": "Specifies if security needs to be turned on when looking to interact
with the Resource",
              "type": "boolean"
            },
            "port": {
              "readOnly": true,
              "description": "Secure port to be used for connection",
              "type": "integer"
            }
          },
          "required" : ["bm"]
        },
        "bp": {
          "type": "string",
          "description": " Batch Parameters: URI parameters to use with an oic.if.b batch request
using this link"
        },
        "title": {
          "type": "string",
          "maxLength": 64,
          "description": "A title for the link relation. Can be used by the UI to provide a
context"
        },
        "anchor": {
          "type": "string",
          "maxLength": 256,
          "description": "This is used to override the context URI e.g. override the URI of the
containing collection",
          "format": "uri"
        },
        "ins": {
          "oneOf": [
            {
              "type": "integer",
              "description": "An ordinal number that is not repeated - must be unique in the
collection context"
            },
            {
              "type": "string",
              "maxLength": 256,
              "format" : "uri",
              "description": "Any unique string including a URI"
            },
            {
              "type": "string",
              "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
```

```
9]{12}$",
              "description": "Use UUID for universal uniqueness - used in /oic/res to identify the
device"
            }
          ],
          "description": "The instance identifier for this web link in an array of web links - used
in collections"
        },
        "type": {
          "type": "array",
          "description": "A hint at the representation of the resource referenced by the target
URI. This represents the media types that are used for both accepting and emitting",
          "items" : [
            {
            "type": "string",
            "maxLength": 64
            }
          ],
          "minItems": 1,
          "default": "application/cbor"
        }
      },
      "required": [ "href", "rt", "if" ]
    }
  },
  "type": "object",
  "allOf": [
    { "$ref": "#/definitions/oic.oic-link" }
  ]
}
```

D.3 OIC Configuration

D.3.1 Introduction

Known resource that is hosted by every Server. Allows for device specific information to be configured.

D.3.2 Fixed URI

/oic/con

D.3.3 Resource Type

The resource type (rt) is defined as: oic.wk.con.

D.3.4 RAML Definition

```
#%RAML 0.8

title: OIC Configuration
version: v1-20160622

traits:
 - interface :
     queryParameters:

       if:
         enum: ["oic.if.rw", "oic.if.baseline"]


/oic/con:

  description: |
    Known resource that is hosted by every Server.
    Allows for device specific information to be configured.

  is : ['interface']

  get:

    description: |
```



```
        Retrieves the current configuration settings

      responses :
        200:
          body:
            application/json:
              schema: |
                {
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-schema.json#",
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
                  "definitions": {
                    "oic.wk.con": {
                      "type": "object",
                      "properties": {
                        "n": {
                          "type": "string",
                          "maxLength": 64,
                          "description": "Human friendly name"
                        },
                        "loc": {
                          "type": "string",
                          "description": "Location information"
                        },
                        "locn": {
                          "type": "string",
                          "maxLength": 64,
                          "description": "Human Friendly Name"
                        },
                        "c": {
                          "type": "string",
                          "maxLength": 64,
                          "description": "Currency"
                        },
                        "r": {
                          "type": "string",
                          "maxLength": 64,
                          "description": "Region"
                        }
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    { "$ref": "#/definitions/oic.wk.con" }
                  ],
                  "required": [ "n" ]
                }

              example: |
                {
                  "rt":   ["oic.wk.con"],
                  "n":    "My Friendly Device Name",
                  "loc":  "My Location Information",
                  "locn": "My Location Name",
                  "c":    "USD",
                  "r":    "MyRegion"
                }

    post:
      description: |
        Update the information about the Device

```

```
    body:
      application/json:
        schema: |
          {
            "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-schema.json#",
            "$schema": "http://json-schema.org/draft-04/schema#",
            "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
            "definitions": {
              "oic.wk.con": {
                "type": "object",
                "properties": {
                  "n": {
                    "type": "string",
                    "maxLength": 64,
                    "description": "Human friendly name"
                  },
                  "loc": {
                    "type": "string",
                    "description": "Location information"
                  },
                  "locn": {
                    "type": "string",
                    "maxLength": 64,
                    "description": "Human Friendly Name"
                  },
                  "c": {
                    "type": "string",
                    "maxLength": 64,
                    "description": "Currency"
                  },
                  "r": {
                    "type": "string",
                    "maxLength": 64,
                    "description": "Region"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "#/definitions/oic.wk.con" }
            ],
            "required": [ "n" ]
          }

        example: |
          {
              "n":  "My Friendly Device Name"
          }

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-schema.json#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
                "definitions": {
                  "oic.wk.con": {
                    "type": "object",
                    "properties": {
```

```
                "n": {
                  "type": "string",
                  "maxLength": 64,
                  "description": "Human friendly name"
                },
                "loc": {
                  "type": "string",
                  "description": "Location information"
                },
                "locn": {
                  "type": "string",
                  "maxLength": 64,
                  "description": "Human Friendly Name"
                },
                "c": {
                  "type": "string",
                  "maxLength": 64,
                  "description": "Currency"
                },
                "r": {
                  "type": "string",
                  "maxLength": 64,
                  "description": "Region"
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "#/definitions/oic.wk.con" }
          ],
          "required": [ "n" ]
        }

    example: |
        {
            "n":  "My Friendly Device Name"
        }
```

D.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.
n	string	yes	Read Write	Human friendly name
loc	string		Read Write	Location information
locn	string		Read Write	Human Friendly Name
c	string		Read Write	Currency
r	string		Read Write	Region

D.3.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/con		get	post		

D.4 Device

D.4.1 Introduction

Known resource that is hosted by every Server. Allows for logical device specific information to be discovered.

D.4.2 Fixed URI

/oic/d

D.4.3 Resource Type

The resource type (rt) is defined as: oic.wk.d.

D.4.4 RAML Definition

```
#%RAML 0.8
title: OIC Root Device
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.r", "oic.if.baseline"]


/oic/d:
  description: |
    Known resource that is hosted by every Server.
    Allows for logical device specific information to be discovered.

  is : ['interface']
  get:
    description: |
      Retrieve the information about the Device

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.d-
schema.json#",
                "definitions": {
                  "uuid": {
                      "type":"string",
                      "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
fA-F0-9]{12}$"
                  },
                  "oic.wk.d": {
                    "type": "object",
                    "properties": {
                      "n": {
                        "type": "string",
                        "maxLength": 64,
                        "readOnly": true,
                        "description": "Human friendly name"
                      },
                      "di": {
```



```
                  "$ref": "#/definitions/uuid",
                  "readOnly": true,
                  "description": "Unique identifier for device (UUID)"
                },
                "icv": {
                  "type": "string",
                  "maxLength": 64,
                  "readOnly": true,
                  "description": "The version of the OIC Server"
                },
                "dmv": {
                  "type": "string",
                  "maxLength": 64,
                  "readOnly": true,
                  "description": "The spec version of the vertical and/or resource
specification"
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "#/definitions/oic.wk.d" }
          ],
          "required": [ "n", "di", "icv", "dmv" ]
        }

      example: |
        {
          "n":    "Device 1",
          "rt":   ["oic.wk.d"],
          "di":   "54919CA5-4101-4AE4-595B-353C51AA983C",
          "icv":  "core.1.1.0",
          "dmv":  "res.1.1.0"
        }
```

D.4.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string	yes	Read Only	
di		yes	Read Only	Unique identifier for device (UUID)
icv	string	yes	Read Only	
dmv	string	yes	Read Only	

D.4.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/d		get			

D.5 Maintenance

D.5.1 Introduction

The resource through which an Device is maintained and can be used for diagnostic purposes. fr (Factory Reset) is a boolean. The value 0 means No action (Default), the value 1 means Start Factory Reset After factory reset, this value shall be changed back to the default value rb (Reboot) is a boolean. The value 0 means No action (Default), the value 1 means Start Reboot After Reboot, this value shall be changed back to the default value

D.5.2 Fixed URI

/oic/mnt

D.5.3 Resource Type

The resource type (rt) is defined as: oic.wk.mnt.

D.5.4 RAML Definition

```
#%RAML 0.8
title: Maintenance
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.r", "oic.if.baseline"]


/oic/mnt:
  description: |
    The resource through which an Device is maintained and can be used for diagnostic purposes.
    fr (Factory Reset) is a boolean.
      The value 0 means No action (Default), the value 1 means Start Factory Reset
    After factory reset, this value shall be changed back to the default value
    rb (Reboot) is a boolean.
      The value 0 means No action (Default), the value 1 means Start Reboot
    After Reboot, this value shall be changed back to the default value

  is : ['interface']
  get:
    description: |
      Retrieve the maintenance action status

    queryParameters:
      if:
        enum: oic.if.r
    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
schema.json#",
                "definitions": {
                  "oic.wk.mnt": {
                    "type": "object",
                    "properties": {
                      "n": {
                        "type" : "string",
                        "maxLength" : 64,
                        "description": "Name"
                      },
                      "fr":{
                        "type": "boolean",
                        "description": "Factory Reset"
                      },
                      "rb": {
                        "type": "boolean",
                        "description": "Reboot Action"
                      }
                    }
```

```
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "#/definitions/oic.wk.mnt" }
            ],
            "required": ["fr"]
          }

        example: |
          {
            "rt":   ["oic.wk.mnt"],
            "n":    "My Maintenance Actions",
            "fr":   false,
            "rb":   false
          }

  post:
    description: |
      Set the maintenance action(s)

    queryParameters:
      if:
        enum: oic.if.rw
    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schemas.org/draft-04/schema#",
            "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
            "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-schema.json#",
            "definitions": {
              "oic.wk.mnt": {
                "type": "object",
                "properties": {
                  "n": {
                    "type" : "string",
                    "maxLength" : 64,
                    "description": "Name"
                  },
                  "fr":{
                    "type": "boolean",
                    "description": "Factory Reset"
                  },
                  "rb": {
                    "type": "boolean",
                    "description": "Reboot Action"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "#/definitions/oic.wk.mnt" }
            ],
            "required": ["fr"]
          }

        example: |
          {
            "n":    "My Maintenance Actions",
            "fr":   false,
            "rb":   false
```

```
          }

      responses :
        200:
          body:
            application/json:
              schema: |
                {
                  "$schema": "http://json-schemas.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-schema.json#",
                  "definitions": {
                    "oic.wk.mnt": {
                      "type": "object",
                      "properties": {
                        "n": {
                          "type" : "string",
                          "maxLength" : 64,
                          "description": "Name"
                        },
                        "fr":{
                          "type": "boolean",
                          "description": "Factory Reset"
                        },
                        "rb": {
                          "type": "boolean",
                          "description": "Reboot Action"
                        }
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    { "$ref": "#/definitions/oic.wk.mnt" }
                  ],
                  "required": ["fr"]
                }

              example: |
                {
                  "n":    "My Maintenance Actions",
                  "fr":   false,
                  "rb":   false
                }
```

D.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
n	string		Read Write	Name
fr	boolean	yes	Read Write	Factory Reset
rb	boolean		Read Write	Reboot Action

D.5.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/mnt		get	post		

D.6 Platform

D.6.1 Introduction

Known resource that is defines the platform on which a Server is hosted. Allows for platform specific information to be discovered.

D.6.2 Fixed URI

/oic/p

D.6.3 Resource Type

The resource type (rt) is defined as: oic.wk.p.

D.6.4 RAML Definition

```
#%RAML 0.8
title: Platform
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.r", "oic.if.baseline"]


/oic/p:
  description: |
    Known resource that is defines the platform on which an Server is hosted.
    Allows for platform specific information to be discovered.

  is : ['interface']
  get:
    description: |
      Retrieve the information about the Platform

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.p-
schema.json#",
                "definitions": {
                  "uuid": {
                      "type":"string",
                      "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
fA-F0-9]{12}$"
                  },
                  "oic.wk.p": {
                    "type": "object",
                    "properties": {
                      "pi": {
                        "$ref": "#/definitions/uuid",
                        "readOnly": true,
                        "description": "Platform Identifier as a UUID"
                      },
                      "mnmn": {
                        "type": "string",
```

```
          "readOnly": true,
          "description": "Manufacturer Name",
          "maxLength": 64
        },
        "mnml": {
          "type": "string",
          "readOnly": true,
          "description": "Manufacturer's URL",
          "maxLength": 256,
          "format": "uri"
        },
        "mnmo": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Model number as designated by manufacturer"
        },
        "mndt": {
          "type": "string",
          "readOnly": true,
          "description": "Manufacturing Date as defined in ISO 8601, where the format is [yyyy]-[mm]-[dd].",
          "pattern": "^([0-9]{4})-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$"
        },
        "mnpv": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Platform Version"
        },
        "mnos": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Platform Resident OS Version"
        },
        "mnhw": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Platform Hardware Version"
        },
        "mnfv": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Manufacturer's firmware version"
        },
        "mnsl": {
          "type": "string",
          "readOnly": true,
          "description": "Manufacturer's Support Information URL",
          "maxLength": 256,
          "format": "uri"
        },
        "st": {
          "type": "string",
          "readOnly": true,
          "description": "Reference time for the device as defined in ISO 8601, where concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.",
          "format": "date-time"
        },
        "vid": {
          "type": "string",
          "maxLength": 64,
          "readOnly": true,
          "description": "Manufacturer's defined string for the platform. The string is freeform and up to the manufacturer on what text to populate it"
        }
      }
```



```
        }
      },
      "type": "object",
      "allOf": [
        { "$ref": "#/definitions/oic.wk.p" }
      ],
      "required": [ "pi", "mnmn" ]
    }

  example: |
    {
      "pi":   "54919CA5-4101-4AE4-595B-353C51AA983C",
      "rt":   ["oic.wk.p"],
      "mnmn": "Acme, Inc"
    }
```

D.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
pi		yes	Read Only	Platform Identifier as a UUID
mnmn	string	yes	Read Only	Manufacturer Name
mnml	string		Read Only	Manufacturer's URL
mnmo	string		Read Only	
mndt	string		Read Only	Manufacturing Date as defined in ISO 8601, where the format is [yyyy]-[mm]-[dd].
mnpv	string		Read Only	
mnos	string		Read Only	
mnhw	string		Read Only	
mnfv	string		Read Only	
mnsl	string		Read Only	Manufacturer's Support Information URL
st	string		Read Only	Reference time for the device as defined in ISO 8601, where concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.
vid	string		Read Only	

D.6.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/p		get			

D.7 Ping

D.7.1 Introduction

The resource using which an Client keeps its Connection with an Server active.

D.7.2 Fixed URI

/oic/ping

D.7.3 Resource Type

The resource type (rt) is defined as: oic.wk.ping.

D.7.4 RAML Definition

```
#%RAML 0.8
title: Ping
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.rw", "oic.if.baseline"]


/oic/ping:
  description: |
    The resource using which an Client keeps its Connection with an Server active.

  is : ['interface']
  get:
    description: |
      Retrieve the ping information

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.ping-
schema.json#",
                "definitions": {
                  "oic.wk.ping": {
                    "type": "object",
                    "properties": {
                      "in": {
                        "type": "integer",
                        "description": "ReadWrite, Indicates the interval for which connection
shall be kept alive"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  { "$ref": "#/definitions/oic.wk.ping"}
                ],
                "required": [
                  "in"
                ]
```

```
            }

          example: |
            {
              "rt":   ["oic.wk.ping"],
              "n": "Ping Information",
              "in":   16
            }
```

D.7.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
in	integer		Read Write	ReadWrite, Indicates the interval for which connection shall be kept alive
in			Read Write	

D.7.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/ping		get			

D.8 Discoverable Resources

D.8.1 Introduction

The resource through which the corresponding Server is discovered and introspected for available resources.

D.8.2 Fixed URI

/oic/res

D.8.3 Resource Type

The resource type (rt) is defined as: oic.wk.res.

D.8.4 RAML Definition

```
#%RAML 0.8
title: Discoverable Resources
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.ll", "oic.if.baseline"]


/oic/res:
  description: |
    The resource through which the corresponding Server is discovered and introspected for
available resources.

  is : ['interface']
  get:
    description: |
      Retrieve the discoverable resource set

    responses :
```



```
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-v4/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-schema.json#",
                "definitions": {
                  "uuid": {
                      "type":"string",
                      "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$"
                  },
                  "oic.res-links.json": {
                    "type": "object",
                    "properties": {
                      "n": {
                        "type": "string",
                        "maxLength": 64,
                        "readOnly": true,
                        "description": "Human friendly name"
                      },
                      "di": {
                        "$ref": "#/definitions/uuid",
                        "readOnly": true,
                        "description": "Unique identifier for device (UUID) as indicated by the /oic/d resource of the device"
                      },
                      "mpro": {
                        "readOnly": true,
                        "description": "Supported messaging protocols",
                        "type": "string",
                        "maxLength": 64
                      },
                      "links": {
                        "type": "array",
                        "items": {
                          "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
                        }
                      }
                    },
                    "required": ["di", "links"]
                  }
                },
                "description": "The list of resources expressed as OIC links",
                "type": "array",
                "items": {
                  "$ref": "#/definitions/oic.res-links.json"
                }
              }

            example: |
              [
                {
                "rt": ["oic.wk.res"],
                "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
                "links":
                  [
                    {
                      "href": "/res",
                      "rel":  "self",
                      "rt":   ["oic.r.collection"],
                      "if":   ["oic.if.ll"]
                    },
                    {
```

```
                        "href": "/smartDevice",
                        "rel":  "contained",
                        "rt":   ["oic.d.smartDevice"],
                        "if":   ["oic.if.a"]
                      }
                    ]
                  }
                ]
```

D.8.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string		Read Only	
di		yes	Read Only	Unique identifier for device (UUID) as indicated by the /oic/d resource of the device
mpro			Read Write	Supported messaging protocols
links	array	yes	Read Write	

D.8.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/res		get			

D.9 Scenes (Top level)

D.9.1 Introduction

Toplevel Scene resource. This resource is a generic collection resource. The rts value shall contain oic.sceneCollection resource types.

D.9.2 Fixed URI

/SceneListResURI

D.9.3 Resource Type

The resource type (rt) is defined as: oic.wk.sceneList.

D.9.4 RAML Definition

```
#%RAML 0.8
title: Scene
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]


/SceneListResURI:
  description: |
    Toplevel Scene resource.
    This resource is a generic collection resource.
    The rts value shall contain oic.sceneCollection resource types.

```



```
  get:
    description: |
      Provides the current list of web links pointing to scenes

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
schema.json#",
                  "title": "Collection",
                  "definitions": {
                      "uuid": {
                          "type":"string",
                          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
[a-fA-F0-9]{12}$"
                      },
                      "oic.collection.setoflinks": {
                          "description": "A set (array) of simple or individual OIC Links. In
addition to properties required for an OIC Link, the identifier for that link in this set is also
required",
                          "type": "array",
                          "items": {
                              "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
                          }
                      },
                      "oic.collection.tags": {
                          "type": "object",
                          "description": "The tags that can be used for tagging links in a
collection",
                          "properties": {
                              "n": {
                                  "type": "string",
                                  "description": "Used to name i.e. tag the set of links"
                              },
                              "id": {
                                  "description": "Id for each set of links i.e. tag. Can be an
value that is unique to the use context or a UUIDv4",
                                  "anyOf": [
                                      {
                                          "type": "integer",
                                          "description": "A number that is unique to that
collection; like an ordinal number that is not repeated"
                                      },
                                      {
                                          "type": "string",
                                          "description": "A unique string that could be a hash or
similarly unique"
                                      },
                                      {
                                          "$ref": "#/definitions/uuid",
                                          "description": "A unique string that could be a UUIDv4"
                                      }
                                  ]
                              },
                              "di": {
                                  "$ref": "#/definitions/uuid",
                                  "description": "The device ID which is an UUIDv4 string"
                              },
                              "base": {
                                  "type": "string",
                                  "description": "The base URI to be used if the links are relative
URIs  (i.e. relative references); see base URI in Core spec for details",
```

```
                                "format": "uri"
                            }
                        },
                        "minProperties": 1
                    },
                    "oic.collection.tagged-setoflinks": {
                        "type": "array",
                        "description": "A tagged link is a set (array) of links that are tagged
with one or more key-value pairs usually either an ID or Name or both",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tags"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.setof-tagged-setoflinks": {
                        "type": "array",
                        "items": [
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            }
                        ],
                        "additionalItems": false
                    },
                    "oic.collection.alllinks": {
                        "description": "All forms of links in a collection",
                        "oneOf": [
                            {
                                "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.tagged-setoflinks"
                            },
                            {
                                "$ref": "#/definitions/oic.collection.setoflinks"
                            }
                        ]
                    },
                    "oic.collection": {
                        "type": "object",
                        "description": "A collection is a set (array) of tagged-link or set
(array) of simple links along with additional properties to describe the collection itself",
                        "properties": {
                            "n": {
                                "type": "string",
                                "description": "User friendly name of the
collection"                },
                            "id": {
                                "anyOf": [
                                    {
                                        "type": "integer",
                                        "description": "A number that is unique to that
collection; like an ordinal number that is not repeated"
                                    },
                                    {
                                        "type": "string",
                                        "description": "A unique string that could be a hash or
similarly unique"
                                    },
                                    {
                                        "$ref": "#/definitions/uuid",
                                        "description": "A unique string that could be a UUIDv4"
                                    }
                                ],
                                "description": "ID for the collection. Can be an value that is
unique to the use context or a UUIDv4"
                            },
```

```
                              "di": {
                                  "$ref": "#/definitions/uuid",
                                  "description": "The device ID which is an UUIDv4 string; used for
backward compatibility with Spec A defintion of /oic/res"
                              },
                              "rts": {
                                  "type": "string",
                                  "description": "Defines the list of allowable resource types (for
Target and anchors) in links included in the collection; new links being created can only be from
this list"                },
                              "drel": {
                                  "type": "string",
                                  "description": "When specified this is the default relationship
to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
                              },
                              "links": {
                                  "$ref": "#/definitions/oic.collection.alllinks"
                              }
                          }
                      }
                  },
                  "type": "object",
                  "allOf": [
                      {
                          "$ref": "#/definitions/oic.collection"
                      }
                  ]
              }

            example: |
              {
                  "rt":       "oic.wk.sceneList",
                  "n":        "list of scene Collections",
                  "rts":      "oic.wk.sceneCollection",
                  "links": [
                   ]
              }
```

D.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string		Read Write	Used to name i.e. tag the set of links
id			Read Write	
di			Read Write	The device ID which is an UUIDv4 string
base	string		Read Write	The base URI to be used if the links are relative URIs (i.e. relative references); see base URI in Core spec for details
n	string		Read Write	User friendly name of the collection
id			Read Write	

di			Read Write	The device ID which is an UUIDv4 string; used for backward compatibility with Spec A defintion of /oic/res
rts	string		Read Write	Defines the list of allowable resource types (for Target and anchors) in links included in the collection; new links being created can only be from this list
drel	string		Read Write	When specified this is the default relationship to use when an OIC Link does not specify an explicit relationship with *rel* parameter
links			Read Write	

D.9.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/SceneListResURI		get			

D.10 Scene Collections

D.10.1 Introduction

Collection that models a set of Scenes. This resource is a generic collection resource with additional parameters. The rts value shall contain oic.sceneMember resource types. The additional parameters are lastScene, this is the scene value last set by any OIC Client sceneValueList, this is the list of available scenes lastScene shall be listed in sceneValueList.

D.10.2 Fixed URI

/SceneCollectionResURI

D.10.3 Resource Type

The resource type (rt) is defined as: oic.wk.sceneCollection.

D.10.4 RAML Definition

```
#%RAML 0.8

title: Scene
version: v1-20160622

traits:
 - interface :
     queryParameters:

       if:
         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
```



```
/SceneCollectionResURI:
  description: |
    Collection that models a set of Scenes.
    This resource is a generic collection resource with additional parameters.
    The rts value shall contain oic.sceneMember resource types.
    The additional parameters are
      lastScene, this is the scene value last set by any OIC Client
      sceneValueList, this is the list of available scenes
      lastScene shall be listed in sceneValueList.

  get:
    description: |
      Provides the current list of web links pointing to scenes

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
schema.json#",
                "title" : "Scene Collection",
                "definitions": {
                  "oic.sceneCollection": {
                    "type": "object",
                    "properties": {
                      "lastScene": {
                        "type": "string",
                        "description": "Last selected Scene, shall be part of sceneValues",
                        "format": "UTF8"
                      },
                      "sceneValues": {
                        "type": "string",
                        "readOnly": true,
                        "description": "All available scene values",
                        "format": "CSV"
                      },
                      "n": {
                        "type": "string",
                        "description": "Used to name the Scene collection",
                        "format": "UTF8"
                      },
                      "id": {
                          "type": "string",
                                      "description" : "A unique string that could be a hash or
similarly unique"
                      },
                      "rts": {
                        "type": "string",
                        "readOnly": true,
                        "description": "Defines the list of allowable resource types in links
included in the collection; new links being created can only be from this list",
                        "format": "UTF8"
                      },
                      "links": {
                        "type": "array",
                        "description": "Array of OIC web links that are reference from this
collection",
                        "items" : {
                          "allOf": [
                            { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
```

```
                                { "required" : [ "ins" ] }
                              ]
                            }
                          }
                        },
                        "required": [ "lastScene","sceneValues","rts","id" ]
                      }
                    },

                    "type": "object",
                    "allOf" : [
                      { "$ref": "#/definitions/oic.sceneCollection" }
                    ]
                  }

                example: |
                  {
                      "lastScene": "off",
                      "sceneValues": "off,Reading,TVWatching",
                      "rt":       "oic.wk.sceneCollection",
                      "n":        "My Scenes for my living room",
                      "id":       "0685B960-736F-46F7-BEC0-9E6CBD671ADC1",
                      "rts":      "oic.wk.sceneMember",
                      "links": [
                       ]
                  }

  put:
    description: |
      Provides the action to change the last settted scene selection.
      Calling this method shall update of all sceneMembers to the prescribed membervalue.
      When this method is called with the same value as the current lastScene value
      then all sceneMembers shall be updated.

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
            "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
schema.json#",
            "title" : "Scene Collection",
            "definitions": {
              "oic.sceneCollection": {
                "type": "object",
                "properties": {
                  "lastScene": {
                    "type": "string",
                    "description": "Last selected Scene, shall be part of sceneValues",
                    "format": "UTF8"
                  },
                  "sceneValues": {
                    "type": "string",
                    "readOnly": true,
                    "description": "All available scene values",
                    "format": "CSV"
                  },
                  "n": {
                    "type": "string",
                    "description": "Used to name the Scene collection",
                    "format": "UTF8"
                  },
                  "id": {
                      "type": "string",
                              "description" : "A unique string that could be a hash or
```

```
similarly unique"
                  },
                  "rts": {
                    "type": "string",
                    "readOnly": true,
                    "description": "Defines the list of allowable resource types in links included
in the collection; new links being created can only be from this list",
                    "format": "UTF8"
                  },
                  "links": {
                    "type": "array",
                    "description": "Array of OIC web links that are reference from this
collection",
                    "items" : {
                      "allOf": [
                        { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
                        { "required" : [ "ins" ] }
                      ]
                    }
                  }
                },
                "required": [ "lastScene" ]
              }
            },

            "type": "object",
            "allOf" : [
              { "$ref": "#/definitions/oic.sceneCollection" }
            ]
          }

        example: |
          {
             "lastScene": "Reading"
          }

    responses :
      200:
        description: |
          Indicates that the value is changed.
          The changed properties are provided in the response.

        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
schema.json#",
                "title" : "Scene Collection",
                "definitions": {
                  "oic.sceneCollection": {
                    "type": "object",
                    "properties": {
                      "lastScene": {
                        "type": "string",
                        "description": "Last selected Scene, shall be part of sceneValues",
                        "format": "UTF8"
                      },
                      "sceneValues": {
                        "type": "string",
                        "readOnly": true,
                        "description": "All available scene values",
                        "format": "CSV"
                      },
```

```
                      "n": {
                        "type": "string",
                        "description": "Used to name the Scene collection",
                        "format": "UTF8"
                      },
                      "id": {
                          "type": "string",
                                      "description" : "A unique string that could be a hash or
similarly unique"
                      },
                      "rts": {
                        "type": "string",
                        "readOnly": true,
                        "description": "Defines the list of allowable resource types in links
included in the collection; new links being created can only be from this list",
                        "format": "UTF8"
                      },
                      "links": {
                        "type": "array",
                        "description": "Array of OIC web links that are reference from this
collection",
                        "items" : {
                          "allOf": [
                            { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
                            { "required" : [ "ins" ] }
                          ]
                        }
                      }
                    },
                    "required": [ "lastScene" ]
                  }
                },

                "type": "object",
                "allOf" : [
                  { "$ref": "#/definitions/oic.sceneCollection" }
                ]
              }

            example: |
              {
                  "lastScene": "Reading"
              }
```

D.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id		yes	Read Write	
lastScene	string	yes	Read Write	Last selected Scene, shall be part of sceneValues
sceneValues	string	yes	Read Only	All available scene values
n	string		Read Write	Used to name the Scene collection
id	string	yes	Read Write	A unique string that could be a hash or similarly unique
rts	string	yes	Read Only	Defines the list of allowable resource types in

				links included in the collection; new links being created can only be from this list
links	array		Read Write	Array of OIC web links that are reference from this collection

D.10.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/SceneCollectionResURI	put	get			

D.11 Scene Member

D.11.1 Introduction

Collection that models a sceneMember.

D.11.2 Fixed URI

/SceneMemberResURI

D.11.3 Resource Type

The resource type (rt) is defined as: oic.r.switch.binary.

D.11.4 RAML Definition

```
#%RAML 0.8
title: Scene
version: v1-20160622
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]

/SceneMemberResURI:
  description: |
    Collection that models a sceneMember.

  get:
    description: |
      Provides the scene member

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneMember-
schema.json#",
                "title" : "Scene Member",
                "definitions": {
                  "oic.sceneMember": {
```

```
                    "type": "object",
                    "properties": {
                      "n": {
                        "type": "string",
                        "description": "Used to name the Scene collection",
                        "format": "UTF8"
                      },
                      "id": {
                        "type": "string",
                        "description": "Can be an value that is unique to the use context or a
UUIDv4"
                      },
                      "SceneMappings" : {
                        "type": "array",
                         "description": "array of mappings per scene, can be 1",
                        "items": [
                          {
                            "type": "object",
                            "properties": {
                              "scene": {
                                "type": "string",
                                "description": "Specifies a scene value that will acted upon"
                              },
                              "memberProperty": {
                                "type":  "string",
                                "readOnly": true,
                                "description": "property name that will be mapped"
                              },
                                "memberValue": {
                                "type": "string",
                                "readOnly": true,
                                "description": "value of the Member Property"
                              }
                            },
                            "required": [ "scene", "memberProperty", "memberValue" ]
                          }
                        ]
                      },

                      "link": {
                        "type": "string",
                        "description": "web link that points at a resource",
                        "$ref": "oic.oic-link-schema.json#"
                      }
                    },
                    "required": [ "link" ]
                  }
                },

                "type": "object",
                "allOf" : [
                  { "$ref": "#/definitions/oic.sceneMember" }
                ]
              }

            example: |
              {
                "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
                "n": "my binary switch (for light bulb) mappings",
                "link": { "href":"coap://mydevice/mybinaryswitch",
                          "if": "oic.if.a",
                          "rt": "oic.r.switch.binary" },
                "sceneMappings": [
                 {
                   "scene":           "off",
                   "memberProperty":  "value",
                   "memberValue":     true
                 },
                 {
                   "scene":           "Reading",
```



```
                    "memberProperty":  "value",
                    "memberValue":     false
                  },
                  {
                    "scene":           "TVWatching",
                    "memberProperty":  "value",
                    "memberValue":     true
                  }
                ]
              }
```


D.11.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
n	string		Read Write	Used to name the Scene collection
id	string		Read Write	Can be an value that is unique to the use context or a UUIDv4
SceneMappings	array		Read Write	Array Of Mappings Per Scene, Can Be 1
scene	string	yes	Read Write	Specifies a scene value that will acted upon
memberProperty	string	yes	Read Only	Property Name That Will Be Mapped
memberValue	string	yes	Read Only	Value Of The Member Property
link	string	yes	Read Write	Web Link That Points At A Resource

D.11.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/SceneMemberResURI		get			

D.12 Resource directory resource

D.12.1 Introduction

Resource to be exposed by any Device that can act as a Resource Directory

D.12.2 Fixed URI

/oic/rd

D.12.3 Resource Type

The resource type (rt) is defined as: oic.wk.rd.

D.12.4 RAML Definition

```
#%RAML 0.8

title: Resource Directory
version: v1-20160622

traits:
 - rddefinterface :
```

```
      queryParameters:
        if:
          description: Interface is optional since there is only one interface supported for the
Resource Type
Both for RD selectin and for publish

          type: string
          enum: ["oic.if.baseline"]
          default: oic.if.baseline


/oic/rd:
  description: |
    Resource to be exposed by any Device that can act as a Resource Directory

  get:
    description: |
      Get the attributes of the Resource Directory for selection purposes.

    queryParameters:
      rt:
        enum: oic.wk.rd
        type: string
        description: Only one Resource Type is used for GET; RT is optional

        required: false
        example: GET /oic/rd?rt=oic.wk.rd

    responses :
      200:
        description: |
          Respond with the selector criteria - either the set of attributes or the bias factor

        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.selection-
schema.json#",
                "title" : "RD Selection",
                "definitions": {
                  "oic.rd.attributes": {
                    "type": "object",
                    "properties": {
                      "n": {
                        "type": "string",
                        "description": "A human friendly name for the Resource Directory",
                        "format": "UTF8"
                      },
                      "di": {
                        "$ref": "oic.types-schema.json#/definitions/uuid",
                        "description": "A unique identifier for the Resource Directory - the same
as the device ID of the RD"

                      },
                      "sel": {
                        "description": "Selection criteria that a device wanting to publish to any
RD can use to choose this Resource Directory over others that are discovered",
                        "oneOf": [
```

```
                              {
                                "type": "object",
                                "properties": {
                                  "pwr": {
                                    "type": "string",
                                    "enum": [ "ac", "batt", "safe" ],
                                    "description": "A hint about how the RD is powered. If AC then this is stronger than battery powered. If source is reliable (safe) then appropriate mechanism for managing power failure exists"
                                  },
                                  "conn": {
                                    "type": "string",
                                    "enum": [ "wrd", "wrls" ],
                                    "description": "A hint about the networking connectivity of the RD. *wrd* if wired connected and *wrls* if wireless connected."
                                  },
                                  "bw": {
                                    "type": "string",
                                    "description": "Qualitative bandwidth of the connection",
                                    "enum": [ "high", "low", "lossy" ]
                                  },
                                  "mf": {
                                    "type": "integer",
                                    "description": "Memory factor - Ratio of available memory to total memory expressed as a percentage"
                                  },
                                  "load": {
                                    "type": "array",
                                    "items": {
                                      "type": "number"
                                    },
                                    "minitems": 3,
                                    "maxitems": 3,
                                    "description": "Current load capacity of the RD. Expressed as a load factor 3-tuple (upto two decimal points each). Load factor is based on request processed in a 1 minute, 5 minute window and 15 minute window"
                                  }
                                }
                              },
                              {
                                "type": "integer",
                                "minimum": 0,
                                "maximum": 100,
                                "description": "A bias factor calculated by the Resource directory - the value is in the range of 0 to 100 - 0 implies that RD is not to be selected. Client chooses RD with highest bias factor or randomly between RDs that have same bias factor"
                              }
                            ]
                          }
                        }
                      }
                    },
                    "type": "object",
                    "allOf": [ {"$ref": "#/definitions/oic.rd.attributes"}],
                    "required": ["sel"]
                  }

              example: |

                  {
                    "rt": "oic.wk.rd",
                    "sel": 50
                  }

      post:

        description: |
          Publish the resource information
          Appropriates parts of the information posted will be discovered through /oic/res
```

```
      queryParameters:
        rt:
          enum: oic.wk.rdpub
          type: string
          description: Only one Resource Type is used for GET; RT is optional

          required: false
          example: GET /oic/rd?rt=oic.wk.rdpub

      body:
        application/json:
          schema: |
            {
              "$schema": "http://json-schema.org/draft-04/schema#",
              "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
              "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
schema.json#",
              "title": "RD Publish & Update",
              "definitions": {
                "oic.rd.publish": {
                  "description": "Publishes resources as OIC Links into the resource directory",
                  "properties": {
                    "linkSet": {
                      "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-tagged-
setoflinks"
                    },
                    "ttl": {
                      "type": "integer",
                      "description": "Time to indicate a RD, how long to keep this published item.
After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
publishing device updates the ttl using the update schema"
                    }
                  }
                }
              },
              "type": "object",
              "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
              "required": [ "links" ],
              "dependencies": {
                "links": [ "ttl" ]
              }
            }

      responses :
        200:
          description: |
            Respond with the same schema as publish but with the links have the "ins" parameter set
to the appropriate instance value.
            This value is used by the receiver to manage that OIC Link instance.

          body:
            application/json:
              schema: |
                {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
                  "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
schema.json#",
                  "title": "RD Publish & Update",
                  "definitions": {
                    "oic.rd.publish": {
                      "description": "Publishes resources as OIC Links into the resource directory",
```

```
                    "properties": {
                      "linkSet": {
                        "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-
tagged-setoflinks"
                      },
                      "ttl": {
                        "type": "integer",
                        "description": "Time to indicate a RD, how long to keep this published
item. After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
publishing device updates the ttl using the update schema"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
                "required": [ "links" ],
                "dependencies": {
                  "links": [ "ttl" ]
                }
              }

            example: |
              {
                "links": [
                  {
                    "href": "coap://someAuthority:1000/somePath",
                    "rt":   "oic.r.someResource",
                    "if":   "oic.if.a",
                    "ins":  12345
                  },
                  {
                    "href": "coap://someAuthority:1000/somePath",
                    "rt":   "oic.r.someOtherResource",
                    "if":   "oic.if.baseline",
                    "ins":  54321
                  }
                ],
                "ttl": 600
              }

  delete:
    description: |
      Delete a particular OIC Link - the link may be a simple link or a link in a tagged set.

    queryParameters:
      di:
        type: string
        description: This is used to determine which set of links to operata on. (Need
authentication to ensure that there is no spoofing). If instance is ommitted then the entire set of
links from this device ID is deleted

        required: true
        example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"

      ins:
        type: string
        description: Instance of the link to delete
Value of parameter is a string where instance to be deleted are comma separated

        required: false
        example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1";ins="20"
```



```
    responses :
      200:
        description: |
          The delete succeeded
```

D.12.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
n	string		Read Write	A human friendly name for the Resource Directory
di			Read Write	A unique identifier for the Resource Directory - the same as the device ID of the RD
sel		yes	Read Write	
pwr	string		Read Write	A hint about how the RD is powered. If AC then this is stronger than battery powered. If source is reliable (safe) then appropriate mechanism for managing power failure exists
conn	string		Read Write	A hint about the networking connectivity of the RD. *wrd* if wired connected and *wrls* if wireless connected.
bw	string		Read Write	Qualitative bandwidth of the connection
mf	integer		Read Write	Memory factor - Ratio of available memory to total memory expressed as a percentage
load	array		Read Write	

D.12.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/rd		get	post	delete	

