OIC CORE SPECIFICATION V1.0.0

Part 1

Open Interconnect Consortium (OIC)
admin@openinterconnect.org

Legal Disclaimer

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

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

CONTENTS

Figures

Tables

1 Scope

The OIC specifications are divided into two sets of documents:

- Core Specification documents: The Core Specification documents specify the OIC Framework, i.e., the OIC core architecture, interfaces, protocols and services to enable OIC profiles implementation for Internet of Things (IoT) usages and ecosystems.
- Vertical Profiles Specification documents: The Vertical Profiles Specification documents specify the OIC 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 OIC core architecture defined in the Core Specification.

This document is the OIC Core specification which specifies the OIC 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 7230, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, June 2014.
https://tools.ietf.org/html/rfc7230.txt

IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, June 2014.
https://tools.ietf.org/html/rfc7231.txt

IETF RFC 7232, *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*, June 2014.
https://tools.ietf.org/html/rfc7232.txt

IETF RFC 7233, *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*, June 2014.
https://tools.ietf.org/html/rfc7233.txt

IETF RFC 7234, *Hypertext Transfer Protocol (HTTP/1.1): Caching*, June 2014.
https://tools.ietf.org/html/rfc7234.txt

IETF RFC 7235, *Hypertext Transfer Protocol (HTTP/1.1): Authentication*, June 2014.
https://tools.ietf.org/html/rfc7235.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 draft-ietf-6lo-btle-14, *IPv6 over BLUETOOTH(R) Low Energy*, June 25, 2015
http://www.ietf.org/id/draft-ietf-6lo-btle-14.txt

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

OIC Security, *Open Interconnect Consortium Security Capabilities*, Version 1.0,

UPnP AV CDS, *UPnP AV Content Directory Service, Version 4*
http://upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf

3 Terms, definitions, symbols and abbreviations

3.1 Terms and definitions

3.1.1
OIC Core Resources
those OIC Resources that are defined in this specification

3.1.2
Configuration Source

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

3.1.3
Entity

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

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

3.1.4
Observe
the act of monitoring an OIC Resource by sending a RETRIEVE request which is cached by the OIC Server hosting the OIC Resource and reprocessed on every change to that OIC Resource

3.1.5
OIC Client
a logical entity that accesses an OIC Resource on an OIC Server

3.1.6
OIC Collection
an OIC Resource that contains zero or more OIC Links

3.1.7
OIC Device
a logical entity that assumes one or more OIC roles (OIC Client, OIC Server)

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

3.1.8
OIC Functionality
the base/core functionality contained in any OIC Device

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

3.1.10
OIC Infrastructure Gateway
an OIC Platform that ensures interoperability between OIC Devices by including the following:

– Resource Directory

3.1.11
OIC Platform
a physical device containing one or more OIC Devices

3.1.12
OIC Links
extends typed web links as specified in IETF RFC 5988

3.1.13
OIC Resource
represents an artifact modelled and exposed by the OIC Framework

3.1.14
OIC Resource Interface
a qualification of the permitted requests on an OIC Resource

3.1.15
OIC Resource Property
a significant aspect or notion including metadata that is exposed through the OIC Resource

3.1.16
OIC Resource Type
a uniquely named definition of class of OIC Resource Properties and the interactions that is supported by that class

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

3.1.17
OIC Server
a logical entity with the role of providing resource state information and facilitating remote interaction with its resources

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

3.1.18
Non OIC Device
A device which does not comply to the OIC specifications

3.1.19
Notification
the mechanism to make an OIC Client aware of resource state changes in an OIC Resource

3.1.20
RAE Client
an OIC Client which supports XMPP functionality

3.1.21
Remote Access Endpoint (RAE) Server
an OIC Server which supports XMPP and it can publish its resource(s) to an XMPP server, thus becoming remotely addressable and accessible

Note 1 to entry: It also supports ICE/STUN/TURN if the application on the OIC Server requires it.

3.1.22
Resource Directory
an OIC Device that hosts descriptions of resources held on other OIC Servers, allowing lookups to be performed for those resources

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

3.1.23
Scene
a single value listed in Scene Values

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.

3.1.24
Scene Collection
an OIC Resource that contains an enumeration of possible Scene Values and the current Scene Value

Note 1 to entry: This resource is a collection resource with additional data, and the member values of the scene resource are scene members.

3.1.25
Scene Value
a Scene enumerator representing the state in which an OIC Resource can be

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

3.1.27
Rule
an OIC Resource that contains a condition which when evaluated as true will launch a Script in an OIC Server

3.1.28
Rule Condition
an expression describing how to evaluate a resource property against a value

Note 1 to entry: The Rule Condition is expressed in EBNF and uses references to a property in a resource on a specified OIC Server.

3.1.29
Rule Member
an OIC resource that contains the values of a property in the resource that are set when an associated Rule Condition is true.

3.1.30
Script
a set of Rule Members to be executed when a Rule Condition holds true

3.1.31
Default Interface
The first interface listed within the interface ('if') property for a specific resource in /oic/res is the Default Interface. If the resource is not exposed in /oic/res then the Default Interface is the interface as defined for that resource in an OIC Specification. When an interface is omitted in a request, the Default Interface will be the interface used to generate the response.

3.2 Symbols and abbreviations

3.2.1
BLE
Bluetooth Low Energy

3.2.2
CBOR
Concise Binary Object Representation

3.2.3
CoAP
Constrained Application Protocol

609 **3.2.4**
610 **EXI**
611 Efficient XML Interchange

612 **3.2.5**
613 **IRI**
614 Internationalized Resource Identifiers

615 **3.2.6**
616 **ISP**
617 Internet Service Provider

618 **3.2.7**
619 **JSON**
620 JavaScript Object Notation

621 **3.2.8**
622 **mDNS**
623 Multicast Domain Name Service

624 **3.2.9**
625 **MTU**
626 Maximum Transmission Unit

627 **3.2.10**
628 **NAT**
629 Network Address Translation

630 **3.2.11**
631 **OIC**
632 Open Interconnect Consortium

633 the organization that created this specification

634 **3.2.12**
635 **URI**
636 Uniform Resource Identifier

637 **3.2.13**
638 **URN**
639 Uniform Resource Name

640 **3.2.14**
641 **UTC**
642 Coordinated Universal Time

643 **3.2.15**
644 **UUID**
645 Universal Unique Identifier

646 **3.2.16**
647 **XML**
648 Extensible Markup Language

649 ## 3.3 Conventions

650 In this specification a number of terms, conditions, mechanisms, sequences, parameters, events,
651 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 an OIC Resource. The data types are derived from JSON values as defined in ECMA-4-4. However an OIC resource can overload a JSON defined value to specify a particular subset of the JSON value. These OIC specific data types are defined in Table 1. The OIC 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	Description
boolean	false true	Binary-value {0, 1}.
BSV	string	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	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	As defined in ISO 8601. The format is [yyyy]-[mm]-[dd].
datetime	string	As defined in ISO 8601. 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.
enum	enum	Enumerated type.
float	number	Signed IEEE 754 single precision float value.
integer	number	Signed 32 bit integer.
json	object/array	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	Character string shall not exceed a max length of 64 octets (bytes).
time	string	As defined in ISO 8601 but restricted to UTC with a trailing “Z”. The format is [hh]:[mm]:[ss]Z.
URI	string	A uniform resource identifier (URI) is a string of characters used to identify a resource. The URI value shall not exceed a max length of 256 octets (bytes).
UUID	string	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 OIC 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 OIC Core Architecture and should be implemented. Recommended features take advantage of the capabilities OIC 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 OIC 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 OIC architecture

5.1 Overview

The OIC architecture enables resource based interactions among IoT artefacts, i.e. physical devices or applications. The OIC 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 OIC 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 OIC 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 OIC operations. Section 5.3 defines the OIC functional block diagram and OIC Framework. Section 5.4 provides an example scenario with OIC Roles. Section 5.5 provides an example scenario of bridging to non-OIC ecosystem.

5.2 Principle

In the OIC 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 7.1.5) using operations that adhere to Representational State Transfer (REST) architectural style, i.e., RESTful interactions.

The OIC architecture defines the overall structure of the OIC Framework as an information system and the interrelationships of the Entities that make up OIC. Entities are exposed as OIC Resources, with their unique identifiers (URIs) and support interfaces that enable RESTful operations on the OIC Resources. Every RESTful operation has an initiator of the operation (the client) and a responder to the operation (the server). In the OIC Framework, the notion of the client and server is realized through OIC Roles (section 5.4). Any OIC Device can act as an OIC Client and initiate a RESTful operation on any OIC Device acting as an OIC Server. Likewise, any OIC Device that exposes Entities as OIC Resources acts as an OIC Server. Conformant to the REST architectural style, each RESTful operation in OIC 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 OIC Resources.

Figure 1 depicts the OIC architecture.

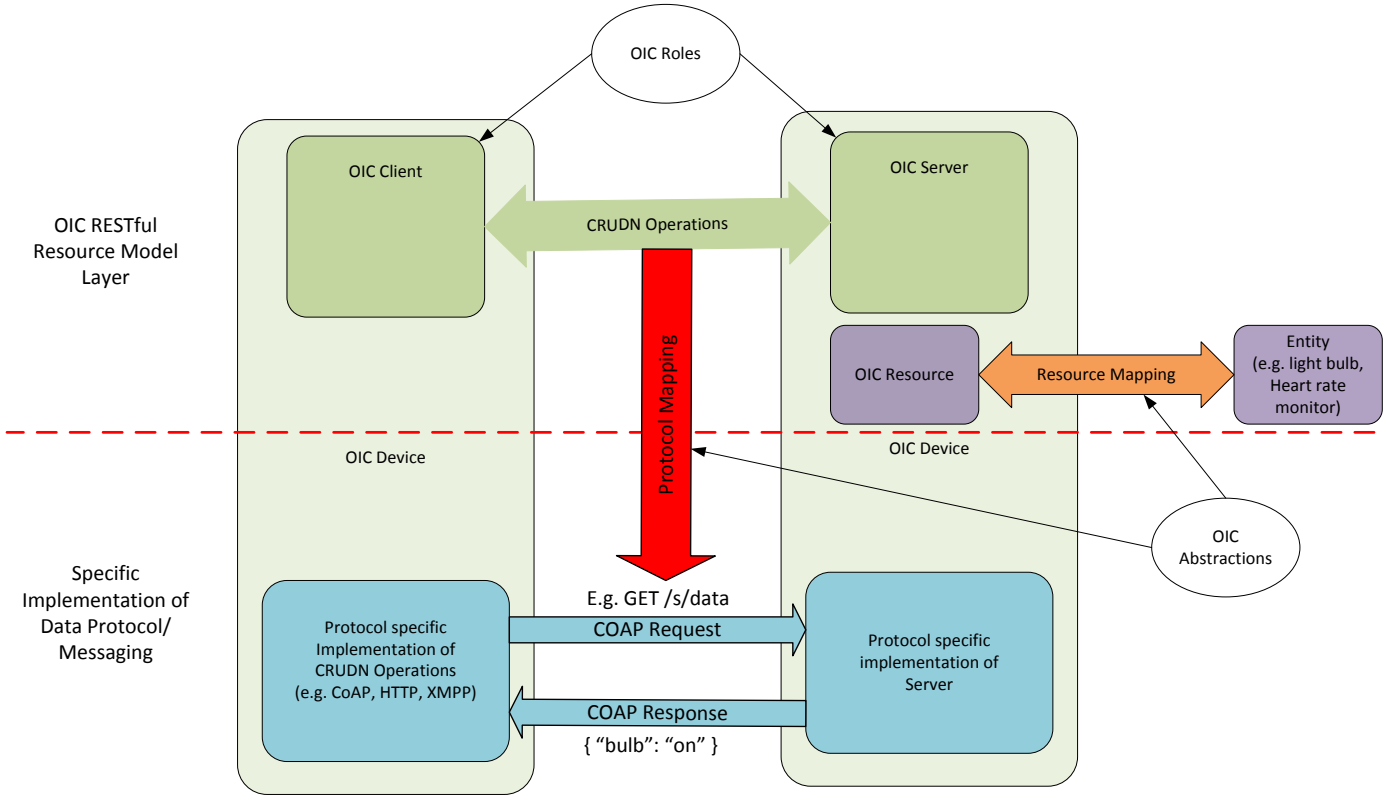


Figure 1: OIC architecture - concepts

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 resource model is agnostic to an application domain like smart home, industrial or automotive. For example, the resource model defines an OIC Resource which abstracts an Entity and the representation of an OIC 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 an OIC Resource in a protocol and technology agnostic way. The specific communication or messaging protocols are part of the protocol abstraction and mapping of OIC 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 an OIC 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 OIC Resources to Entities that are reached over protocols that are not natively supported by OIC.

5.3 OIC functional block diagram

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

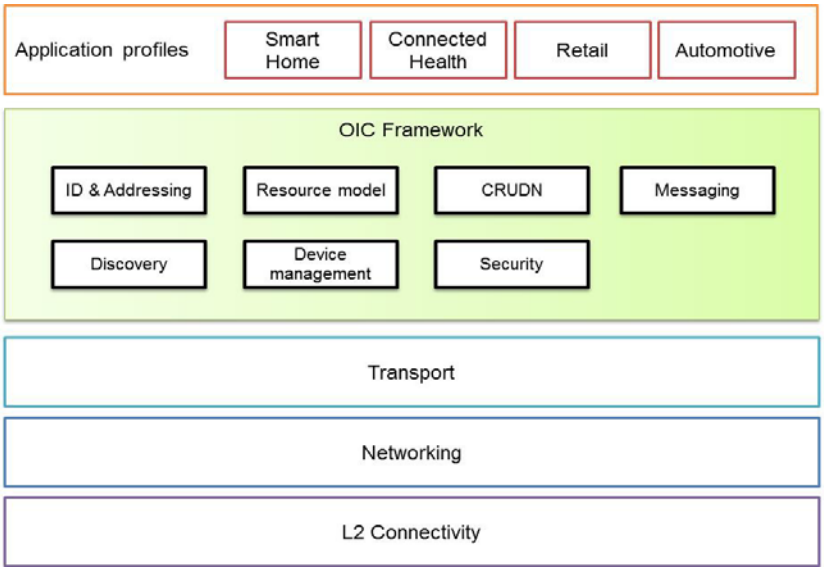


Figure 2: OIC functional block diagram

- **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 OIC 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).
- **OIC Framework**: Provides the OIC 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 OIC 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 OIC Devices communicate with each other, each functional block in an OIC Device interacts with its counterpart in the peer OIC Device as shown in Figure 3.

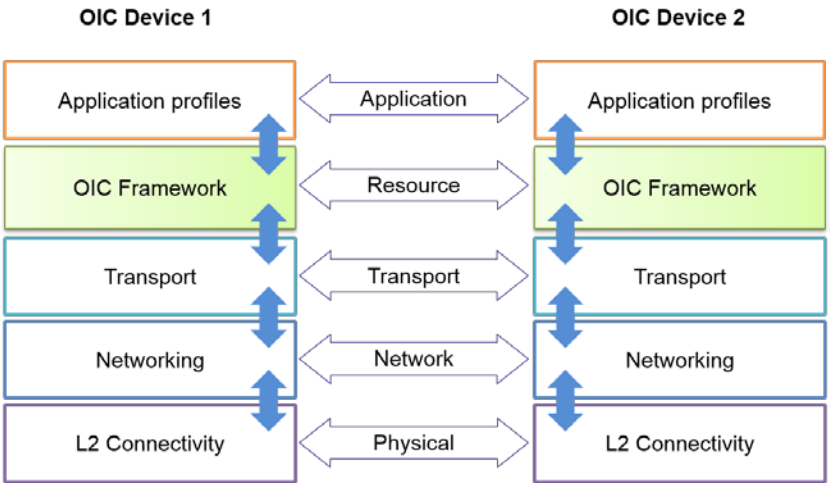


Figure 3: OIC communication layering model

5.3.1 OIC Framework

OIC Framework consists of functions which provide core functionalities for OIC 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) OIC Devices (Endpoint Discovery in section 10) and
   b) OIC 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 an OIC Client and OIC 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 an OIC 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 OIC Roles

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

Figure 4 illustrates an example of the OIC 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 an OIC Client, the gateway takes the role of an OIC Intermediary and the thermostat takes the role of an OIC Server.

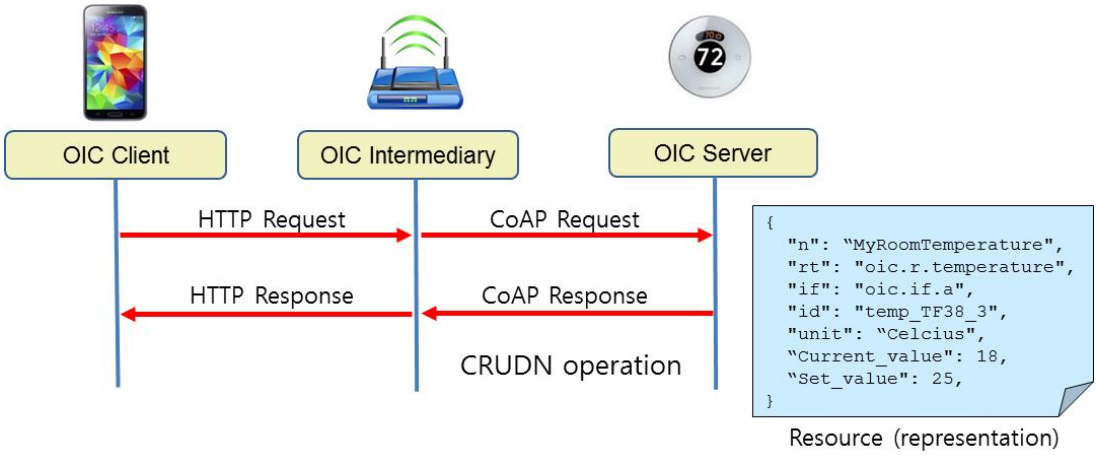


Figure 4: Example illustrating the OIC Roles

5.5 Example Scenario: Bridging to Non-OIC 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 OIC supported.

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

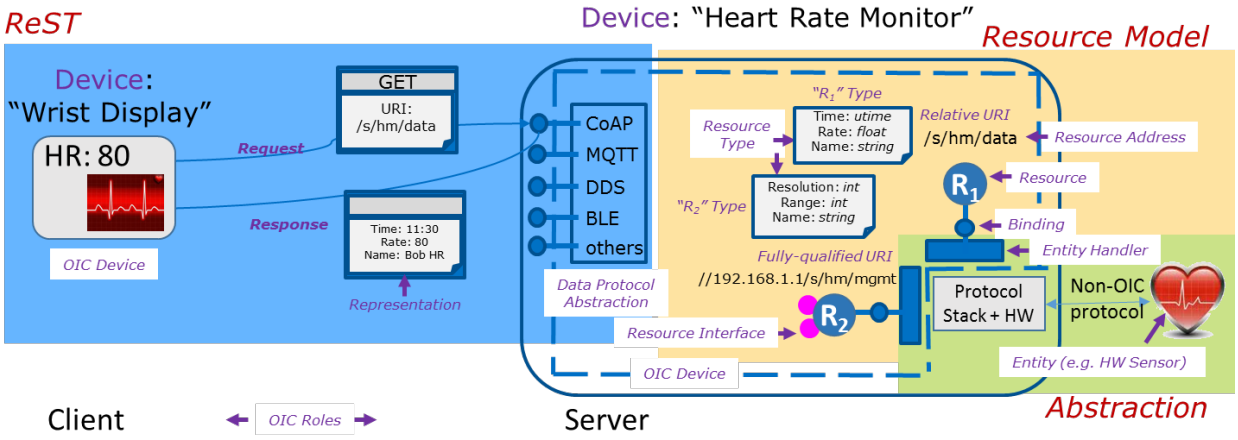


Figure 5: OIC Framework - Architecture Detail

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

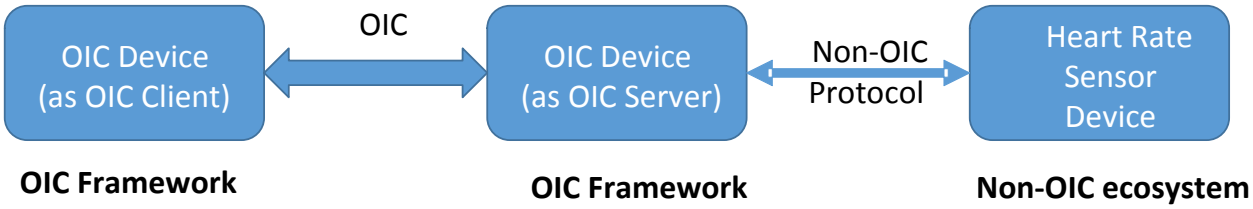


Figure 6: OIC Server bridging to Non-OIC device

On start-up the OIC Server runs the entity handlers which discover the non-OIC systems (e.g., Heart Rate Sensor Device) and create resources for each device or functionality discovered. The entity handler creates an OIC Resource for each discovered device or functionality and binds itself to that OIC Resource. These resources are made discoverable by the OIC 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 OIC Server are then interpreted by the entity handler and forwarded to the non-OIC device using the protocol supported by the non-OIC device. The returned information from the non-OIC 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 OIC 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 either one 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 and HTTP 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 eample 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 OIC URI is based on the general form of a URI as defined in IETF RFC 3986 as follows:

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

Specifically the OIC 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.

The *scheme* above is modified by replacing ‘oic’ with the indicator for the underlying transport protocol (e.g., ‘coap’, ‘http’) before sending over the network by the requestor. Similarly on the receiver side, the indicator for the underlying transport protocol (e.g., ‘coap’ or ‘http’) is replaced with ‘oic’ before handing over to resource model layer on receiver.

If the authority is the local OIC Device, then ‘oic’ shall 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 OIC 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 OIC Server. A *path* shall be preceded by a ‘/’ (slash). The *path* may have ‘/’ (slash)

separated segments for human readability reasons. In the OIC 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 OIC 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 to 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 OIC 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 OIC Device on which it is hosted. Alternatively, a URI may be generated by the OIC 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 OIC 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 path prefix /oic for relative URIs is reserved as a namespace for OIC defined resources (including discoverable resources). Resources not defined by OIC specifications shall not use prefix /oic for relative URIs.

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 an OIC 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 OIC resource model

7.1.1 Introduction

The OIC resource model provides core interoperability among OIC Devices. This section describes the concepts and mechanisms used to represent the primary details of a resource using the OIC Framework. These include OIC Resource, OIC Interface, OIC Resource Property, OIC Resource Type, resource representation, and resource structures including collections.

7.1.2 OIC Resource

In the OIC Framework, a physical or software artefact or concept that needs to be made visible and manipulated is represented as an OIC Resource. The OIC Resource encapsulates and represents the salient aspects of an Entity. An OIC Resource has an address as defined in section 6, and properties defined in section 7.1.4. In general, an OIC Resource can be assigned any URI and the URI doesn't encode or provide any information of the resource characteristics. However some core OIC Resources have fixed URI (e.g., /oic/res defined in section 11.3.4) and the fixed URI determines its characteristics.

An OIC Resource requires that the following properties are specified:

- **Fixed URI** (optional) – a fixed URI assigned to an OIC Resource for a Resource Type ID (e.g., /oic/res, /oic/d).
- **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.r.humidity). A 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 exact string matches – this allows other forms of naming like URNs where required for ecosystem interoperability.
- **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 2 and Table 3 provide an example description of foobar resource type and properties.

Table 2. Example foobar Resource Type

fixed 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 3. Example foobar properties

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource type	**rt**	**string**			**R**	**yes**	**Resource type**
Interface	**if**	**string**			**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/schema",
  "type": "object",
  "properties": {
    "rt": {"type": "string"},
    "if": {"type": "string"},
    "value": {"type": "string"}
  },
  "required": ["rt", "if", "value"]
}
```

7.1.3 OIC Interface

An interface (term derived from IETF RFC 6690) is used to qualify and select the type of permitted requests on an OIC Resource. A resource interface identifies the specific context in which a request is to be evaluated; the same request when evaluated against different resource interfaces would elicit different responses.

7.1.4 OIC Resource Properties

7.1.4.1 Introduction

An OIC Resource Property, or Property in short, describes an aspect or notion that is exposed through the OIC Resource including meta-information related to that resource.

Properties can be consumed within the OIC Framework, e.g., a query may use specific Properties with desired values to select OIC Resources that meet those criteria. Properties may also be exposed to the applications.

The following items may be included when defining an OIC Property:

- **Property** – a “key=value” pair
- **Property Title** - a human friendly name to designate the property, usually not sent over the wire.
- **Property Name**– the key in "key=value” pair. The data type of the property name is “string”.
- **Property Value** – the value in "key=value” pair.
- **Value Type** –the data type that the value of the property may take as defined in section 3.4 (e.g., stringor boolean).
- **Value Rules** –the rules on the data value that the property may take. The rules could for example define a fixed range, a set of enumerated values, a pattern, conditional values, or dependencies on other values. The specification of rules could be used by a validator that validates these values in an implementation (e.g., "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rp", "oic.if.p"]).
- **Units** – specifies the units that the values for this property are referenced against.
- **Access Modes** – specifies whether the value may be read or written to. Updates are equivalent to a write. R is used for read and W for write – both can be specified (note: write does not automatically imply read).
- **Description** – is descriptive text on the purpose and expected use of this property.
- **Mandatory** – specifies if the OIC Property is mandatory or not for a given resource

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 OIC 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 7.1.4.2.

7.1.4.2 Common properties

7.1.4.2.1 Introduction

The common properties defined in this section may be specified for all OIC Resources. The following properties are defined as common properties “Resource Type”, “Resource Interface”, “Policy” and “Name”.

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", "p"). 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 OIC defined Resource Types. However, this may become cumbersome as the number of OIC Resource Types grow. To prevent such name conflicts in the future, OIC 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.

7.1.4.2.2 Resource type

A resource type indicates a class or category of resources. A resource type may explicitly mandate the existence of an optional common Property.

The resource type may be pre-defined. A resource type may be defined either by this specification (called a core resource type), OIC vertical specifications (called a vertical resource type), by a manufacturer, end user, or developer of OIC Devices (called a vendor defined resource type).

Note: Pre-defined types and details may be communicated out of band (like in documentation) while explicitly defined resource types may be downloaded and used by, for example, an API or application.

Every OIC Resource shall have a resource type at the time of its creation. Resource types may be explicitly discovered or knowledge of type may be implicitly shared between the user (e.g. OIC Client) and the host of the OIC Resource (e.g., OIC Server).

7.1.4.2.3 Interface

The Interface property lists the interfaces the resource supports. A resource interface may be defined either by OIC specifications (a “core resource interface”), OIC vertical specifications (a “vertical resource interface”), by a manufacturer, end user, or developer of OIC Devices (a “vendor defined resource interface”).

All resources shall have at least one Interface. The Default Interface of a resource is defined in section 3.1.31 and if discoverable shall be communicated to the client during resource discovery in the response to a RETRIEVE operation on ‘/oic/res’. The Default Interface shall be defined by an OIC specification.

In addition to any OIC specification defined interface, all OIC resources shall support the Baseline interface (oic.if.baseline) as defined in Table 8.

Selection of an interface may appear in the query parameter. If no query parameter is specified, then the System Default Interface is selected.

7.1.4.2.4 Policy

Policy defines how the resource is to be accessed or treated. Policy may be tied with security where required. Two policies are defined:

- Discoverable: A resource can be explicitly hidden or made discoverable using this policy. Only resources marked as discoverable shall be returned as part of a discovery through /oic/res.
- Observable: A resource not marked as observable does not allow OBSERVE operation. An error shall be returned on such requests

Discoverable and Observable are independent of each other and both can be present in the same OIC Resource.

7.1.4.2.5 Name

A human friendly name for the resource, i.e. a specific resource instance name (e.g., MyLivingRoomLight),

7.1.5 Resource representation

The externally visible and operable snapshot of OIC Resource Properties and their respective values taken at a point in time is known as the resource representation. Resource representation captures the state of an OIC Resource at a particular time. The resource representation is exchanged in the request and response interactions with an OIC 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).

7.1.6 Structure

7.1.6.1 Introduction

In many scenarios and contexts, the OIC 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 OIC Framework provides the means to model and map these structures and the relationships among OIC Resources. The primary building block for resource structures in OIC Framework is the collection. A collection represents a container, which is extensible to model complex structures.

7.1.6.2 OIC Link

An OIC Link embraces and extends typed web links as specified in IETF RFC 5988 as a means of expressing relationships between resources. An OIC Link consists of:

- a context IRI,
- a target IRI,
- a relation from the context IRI to the target IRI
- a set of parameters that provide metadata about the target IRI.

The target IRI is mandatory and the other items in an OIC Link are optional. Additional items in the OIC Link may be made mandatory based on the use of the links in different contexts (e.g. in collections, in discovery, in bridging etc.).

IETF RFC 5988 establishes a registry for link relations (section 4.1 of IETF RFC 5988) and also defines a base set of attributes (section 5.4 of IETF RFC 5988). Furthermore, IETF RFC 6690 utilizes typed web links to enable discovery of hosted resources within Constrained RESTful Environments (CoRE). It also adds a number of attributes (Section 3 of IETF RFC 6690) to allow typed web links to contain metadata that is useful for resource discovery.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://openinterconnect.org/rm/oic.oic-link.json",
  "type": "object",
  "properties": {
    "href": {
      "type": "string",
      "description": "RFC5988 style web-links described using JSON. This is the target URI",
      "format": "uri"
    },
    "rel": {
      "type": "string",
      "default": "null",
      "description": "The relation of the target URI referenced by the link to the context URI; the reserved value of 'null' is for no relationship"
    },
    "rt": {
      "type": "string",
      "description": "Resource Type - A standard OIC specified or vendor defined resource type of the resource referenced by the target URI"
    },
    "if": {
      "type": "string",
      "description": "Interface - The interfaces supported by the resource referenced by the target URI"
    },
    "obs": {
      "type": "boolean",
      "description": "Specifies if the resource referenced by the target URIis observable or not",
      "default": false
    },
    "title": {
      "type": "string",
      "description": "A title for the link relation. Can be used by the UI to provide a context"
```



```
    },
    "anchor": {
      "type": "string",
      "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",
          "description": "Any unique string including a URI"
        },
        {
          "type": "string",
          "format": "uuid",
          "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"
    },
    "mt": {
      "type": "array",
      "description": "A hint of the media type of the representation of the resource referenced by the target
URI",
      "items" : {
        "type": "string"
      },
      "minItems": 1,
      "default": "[ application/cbor ]"
    }
  },
  "required": [
    "href",
    "rt",
    "if"
  ]
```

Figure 7: JSON Schema for OIC Links

As shown in Figure 7 the relation between the context IRI and target IRI is specified using the “rel” JSON element and the value of this element specifies this relation. If a web link does not explicitly include the “rel” element, a value of “rel”=”hosts” shall be assumed for all web links returned in response to the “/oic/res” resource. 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]

The anchor parameter is used to change the context URI of an OIC 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 OIC Collections – a floor has rooms and rooms have lights – the lights can be defined in floor as OIC Links but the OIC 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 can also be turned on by using the room URI too).

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



```
/a/room1 {
  “links”: [
     {
       “href”: “/x/light1”,
       “rel”: “contains”
     }
   ]
}
```

Figure 8: Example of use of anchor in OIC Link

The OIC architecture utilizes typed web links as a mechanism for bootstrapping resource discovery through the known OIC Core Resource “/oic/res”. Additionally for this specification the IRIs are limited to URIs. A RETRIEVE operation on “/oic/res” returns (among other things) a serialized representation of typed web links to resources that are discoverable from that OIC Device. The serialization format should be negotiated using the underlying transport protocol i.e. using Accept and Content-Type headers in case of HTTP and CoAP. By default OIC uses CBOR as the payload. The payload (content) in CBOR for web links is described with the JSON Schema in Figure 7. Other serializations (e.g., XML/EXI) may be defined in future versions of this specification.

Figure 9 is a JSON Schema that specifies the representation of the response to “/oic/res”

```
{
  "$schema": "http://json-schema.org/draft-v4/schema#",
  "id": "http://openinterconnect.org/schemas/oic.res.json/"
  "definitions": {
    "oic.res-links.json": {
      "type": "object",
      "properties": {
        "di": {
          "description": "The device identifier as indicated by the /oic/d resource of the device",
          "type": "string",
          "format": "UUID"
        },
        "links": {
          "type": "array",
          "items": {
            "$ref": "oic.oic-link.json#"
          }
        }
      }
    }
  },
  "description": "The list of resources expressed as OIC Links",
  "type": "array",
  "items": {
    "$ref": "#/definitions/oic.res-links.json"
  }
}
```

Figure 9: JSON Schema for /oic/res

Figure 10 is an example of a RETRIEVE from “/oic/res”

```
[
  {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
.... "links":[
    {
      "href": "/door",
      "rt": "oic.r.door",
      "if": "oic.if.b oic.ll"
    },
    {
      "href": "/door/lock",
      "rt": "oic.r.lock",
```



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

Figure 10: Example /oic/res representation

The “type” element may be used to specify the various representations that are supported by a specific resource. The default type of application/cbor shall be used when “type” element is omitted. The example in Figure 10 shows that the “/door/lock” resource supports two representations (cbor and exi/xml); but is described in json for readability. Additional representations may be included as necessary. Once a client discovers this information for each resource, it may use one of the available representations in the Accept header of the request.

7.1.6.3 Collections

7.1.6.3.1 Overview

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

An OIC Collection shall have at least one resource type and at least one interface bound at all times during its lifetime. A resource type and interfaces shall be bound to a collection at the creation of the collection. These initial values may be overridden using mechanism used for overriding in the case of an OIC Resource. Additional resource types and interfaces may be bound to the collection at creation or later during the lifecycle of the OIC Collection.

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

An OIC Resource may be referenced in more than one OIC 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 OIC 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 OIC Link.

The list of OIC Links defined in a collection may be either a simple list of OIC Links (i.e. the value of the “links” property is a simple array of OIC Links) as illustrated in Figure 12 or may be a list of tagged sets of OIC Links where each tagged set shall have at least one tag (i.e. the value of the “links” property is an array where each element of the array is an object with an array of OIC Links and a set of one or more key-value pairs; the key-value pairs in that object are the tags for the array of OIC Links in that object; the key is the tag name and the value is the tag value) as illustrated in Figure 13.

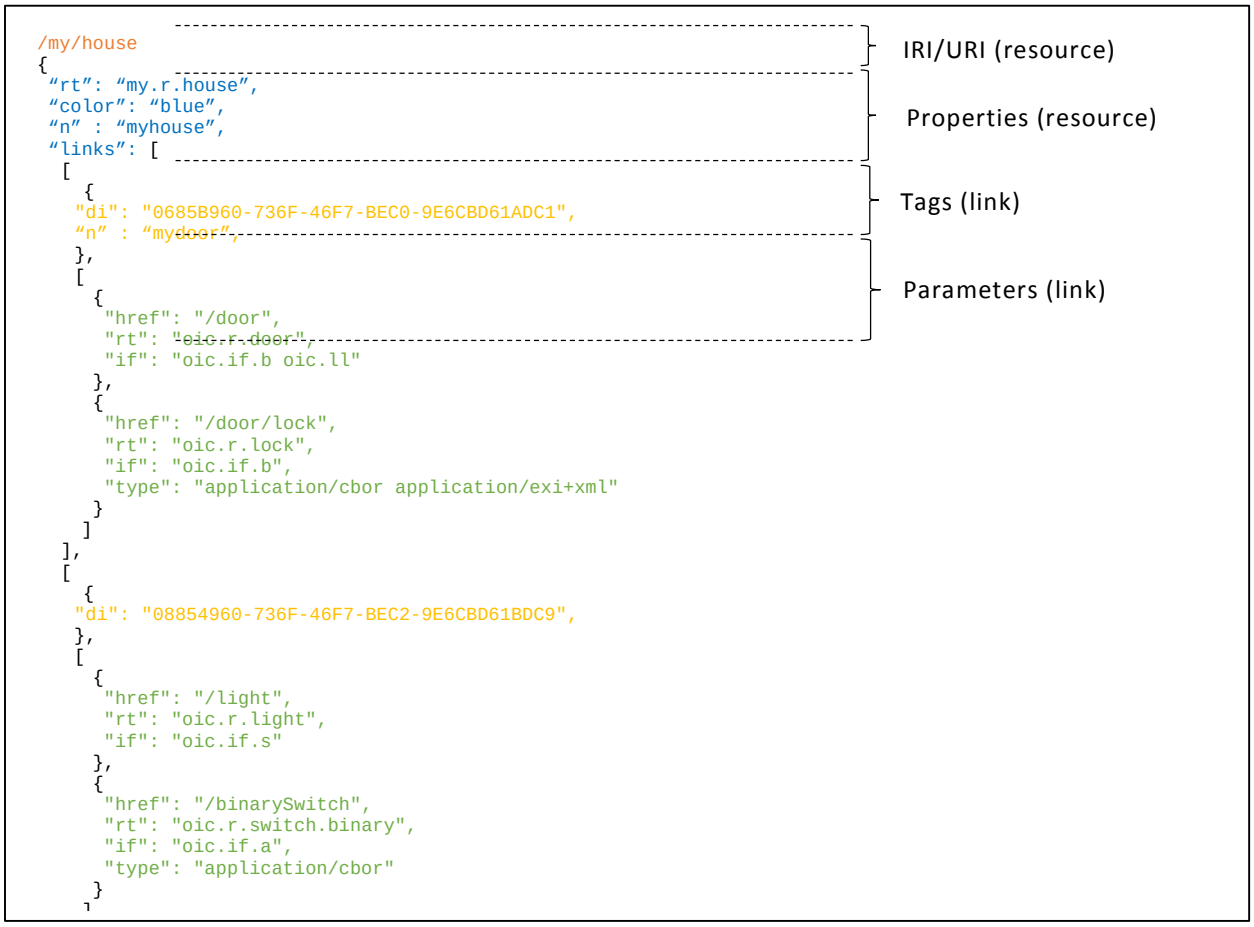


Figure 11: Example showing parts of OIC Collection and OIC Links

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



```
    }
  ]
}
```

Figure 12: Example OIC Collection with simple links (JSON)

```
{
 “links”: [
  [
   {
   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
   },
   [
    {
     "href": "/door",
     "rt": "oic.r.door",
     "if": "oic.if.b oic.ll"
    },
    {
     "href": "/door/lock",
     "rt": "oic.r.lock",
     "if": "oic.if.b",
     "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",
     "type": "application/cbor"
    }
   ]
  ]
 ]
}
```

Figure 13: Example OIC Collection with tagged OIC 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 OIC 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 OIC Device but other OIC Devices may access or update the collection
- A hosted collection where the collection is centralized but is managed by an authorized agent or party.

7.1.6.3.2 Collection properties

An OIC 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 4. This list of common properties are in addition to those defined for OIC Resources in section 7.2.1. When a property is repeated in Table 4 for OIC Collections, the conditions in this definition shall override those in the general list for OIC Resources.

Table 4: Common properties for OIC Collections (in addition to Common Properties defined in section 7.2.1)

Property	Description	Property name	Value Type	Mandatory
Links	The set of links in the collection	"links"	json Array of OIC 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"	CSV Format: Comma separated list of resource type names	No
Default relationship	Specifies the default relationship to use for OIC 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 OIC 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 be modified by using the UPDATE (POST) operation to the oic.wk.baseline interface with a jSON object having the list of properties and their updated values. In the case of a single property being updated, the query with property name could be used to select the property and the request payload has the value of that property (e.g. GET /<col URI>?property).

7.1.6.3.3 Interfaces for collections

The interfaces allows for modulation of the requests against collection for a given resource type. The primary interfaces defined by OIC are shown in Table 5.

The resource type determines which of these interfaces are valid for that collection type. For example: a factory collection (defined by a resource type that includes factory semantics) may support the "link batch" interface but a resource type for a simple collection may only define a "links list" interface.

1358 **Table 5: OIC-defined Interfaces for OIC Collection**

Interface	Name	Description
Baseline	oic.if.baseline	Interface provides access to the entire representation of the collection. Can be used to modify properties and links in the collection.
Links List	oic.if.ll	Interface provides access to only the links in the collection i.e. the value of the “links” property. In the case of tagged links, the tag of the link shall be provided in the query. A RETRIEVE using this interface retrieves the list of links A UPDATE on this interface can be used to add, modify or update the links in the collection. . When UPDATE is done using this interface, no checks are done if the links reference valid and available resources. This interface is used to simply “update” the links.
Batch	oic.if.b	Interface provides access to the resources/collections that are referenced by the links in the collection. A RETRIEVE using this interface shall return the representation of all the resources referenced by links in the collection. The response that have all the representation aggregated. A UPDATE on this interface shall lead to a UPDATE request to each of the referenced resources in the collection’s links; the representation in the payload for each of these implicit requests is the same as the request to the collection. Properties in the payload that are not understood by the target resources shall be ignored. [NOTE: Security implications of using this interface will be specified in the future when security interactions are defined beyond link layer security].
Link Batch	oic.if.lb	Interface provides means to create the resource that is reference in a link or set of links and add those links to the collection in one atomic operation. (In this instance, atomic implies that no new requests are processed against this collection until this request and response completes fully or fails). An UPDATE request on this interface will carry the link(s) that need to have resource created and then added to the collection. If the resource reference by the link is already present then the link is tested to be valid and available. [NOTE: This is a harder implication on the updates on links and should be used carefully and judiciously only when this behaviour is necessary. Simple operations on links can be done using update with the “links list” interface] A RETRIEVE request on this interface is not supported. [NOTE: Security implications of using this interface will be specified in the future when security interactions are defined beyond link layer security].

1359 #### 7.1.6.3.4 Default resource type

1360 A default resource type, oic.wk.col, shall be available for OIC Collections. This resource type shall
1361 be used only when another type has not been defined on the collection or when no resource type
1362 has been specified at the creation of the collection.

1363 The default resource type provides support for the common properties in Table 5 including the
1364 “links” property. For the default resource type, the value of “links” shall be a simple array of OIC
1365 Links and tagging of links shall not be supported.

1366 The default resource type shall support the ‘baseline’ and ‘links list’ interfaces.

7.1.6.3.5 Bootstrapping collections

OIC Devices that support creation of collections shall expose the standard bootstrap resource /oic/col with resource type oic.wk.col and supporting the link batch interface oic.if.lb. The bootstrap resource shall be discoverable and observable.

The resource /oic/col shall be discovered through /oic/res to indicate to other OIC Device that collection creation is supported.

Once the bootstrap resource has been discovered, an OIC Client may request creation of collections on that bootstrap resource. This is done by sending a CREATE (POST) operation to the oic.if.lb interface of /oic/col with a body having a representation defined by the default resource type oic.wk.col. The links in the body shall have the resource type of the collection or resources to be created. More than one collection may be requested and created in a single request. The policy for the created collection may be defined in the OIC Link for that collection (by default the policy for collections created using /oic/col is discoverable and observable). The target URIs in the links in the request shall have either a URI as a hint to the OIC Server or have the nul value to indicate that the server is to generate. The OIC Server may ignore the URI hint.

The result of the request is that the collections in the request are created and, if the policy "discoverable" does not disallow, will be published in /oic/res in one atomic operation. The links with the URIs of the created collection shall be returned in the response. The OIC Server shall also set the value of the "ins" parameter of the links returned in the response.

7.1.6.3.6 Creation of collection and resource using collections

An OIC Collection that exposes the link batch interface can be used as a factory for resources of the resource types that the collection permits (see "rts" property). The representation in the body of the request shall be defined by the resource type of the collection to which the request is sent.

The creation of the resource or collection and the information returned in the response is similar to the bootstrap. The OIC Links for the created collection or resource may be added to the OIC Collection that process the request.

The default policy for resource created through general collections (unlike bootstrap) is non-discoverable and non-observable.

7.1.6.3.7 Deletion of collections

To delete a collection a DELETE operation is sent to the URI of the collection to be deleted. Optionally, a policy shall be defined on the collection to disallow deletion if the collection has links. If the policy disallows deletion of a collection with links then an error shall be returned.

To delete a collection and all the collections and resource that are referenced from that collection then the DELETE operation shall be sent to the link batch interface of the collection (e.g., DELETE /<coll URI>?if=oic.if.lb). This is a dangerous operation and should be used with caution.

7.1.6.3.8 Managing collections

- Add links:

  To add links into the collection, a UPDATE (POST) operation to the collection URI shall use the oic.if.ll interface in the query. The body of the request shall have the "links" property and the value shall be an array of one or more OIC Links to add to the collection. (No checking is done on the OIC Server if the URIs in the OIC Link point to valid and active resources). For a add request the "ins" parameter in the links in the request payload shall be empty/null or the "ins" parameter shall be omitted.

  The OIC Server shall respond to the request by adding the links to the collection and returning in the body of the response the OIC Links with the "ins" parameter assigned to the appropriate

value. This value of the “ins” parameter shall uniquely identify this link within the collection over the entire lifetime of the collection.

The payload of the response shall also include the Time to Live for the links – a single time to live is support for all links in the request payload.

- Modify links

To modify the links in a collection, a UPDATE (POST) operation to the collection URI shall use the oic.if.ll interface in the query. The body of the request shall have the “links” property and the value shall be an array of one or more OIC Links to add to the collection. For a modify request, the “ins” parameter shall have the value that identifies the link to be modified. (No checking is done on the OIC Server if the URIs in the OIC Link point to valid and active resources).

The OIC Server shall replace the parameters of link in the collection that has the matching “ins” value with corresponding parameters in the link in the request payload. To replace the entire link all link parameters (except “ins”) in the collection have to have updated values. This method may be used to update only the Time to Live for the links.

- Delete links

To delete links in a collection, a DELETE (DELETE) operation is sent to collection URI with the instance to delete identified in the query of the request (e.g. DELETE /<coll URI>?if=oic.wk.ll;ins=”yyz”).

For multiple instance to be deleted, the “ins” parameter may be repeated for each of the link (interface oic.wk.ll is specified only once).

To delete all the links in a collection, the DELETE operation shall be sent to the link list interface of the collection (e.g. DELETE /<coll URI>?if=oic.wk.ll)

7.1.6.3.9 Example flows

Figure 14 and Figure 15 provides examples of message flows for managing collections

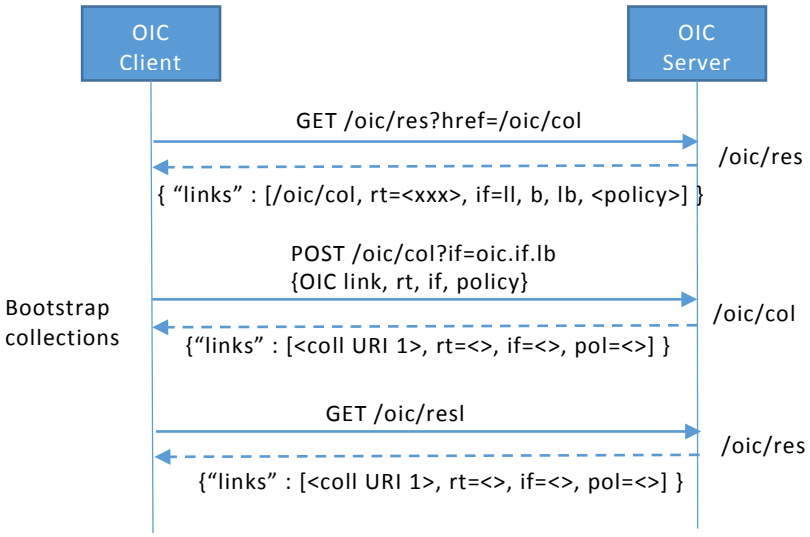


Figure 14. Bootstrap collections

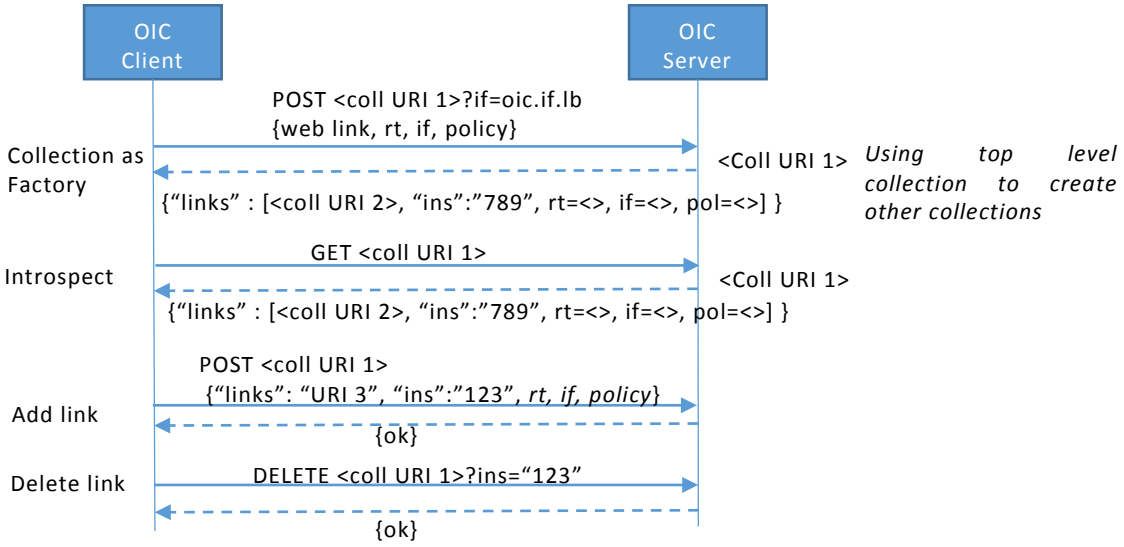


Figure 15. Collection as a factory

7.1.6.3.10 Selectors and filters (in query strings)

In many use cases, there is a need to return a handle (URI) to a collection based on the matching of the properties and parameters of the links in collection. For example: trying to get all rooms in a floor that have LED bulbs In these cases, a simple query is insufficient because the properties of a collection can also appear in the links in the collection – there is no clean way to disambiguate using the simple query mechanism. OIC introduces the filter keyword in the query to separate the part of the query – one on the properties of the collection itself and one on the properties in the links in the collection. The selector selects based on the properties themselves whereas the filter is based key-value pairs in the values of properties in the collection.

So GET /oic/res?rt=oic.wk.foo will match all collections in /oic/res that are of resource type oic.wk.foo. A GET /oic/res?filter="rt=oic.wk.foo" will match collections in /oic/res that *contain* collections that are of resource type oic.wk.foo. (Using an interface to separate selectors and filter is not possible since query strings don't have an ordering in the key value pairs).

A query in a request shall have two parts – a selector which is the primary query and a filter. The filter shall be designated with the keyword "filter". The value of the filter is also a query with the same key-value pairs in query without the filter keyword. A filter shall be applied to match the query in the filter against the links in the collection – if a match is found then the collection containing the match shall be used as the resulting URI of the filtered query. A query string not enclosed in a "filter" is a selector and shall be applied to match the properties-values of the collection itself.

7.2 Usage of OIC resource model

7.2.1 Values for common properties

All OIC Resources shall have the following common properties as defined in section 7.2.1.1 through section 7.2.1.4.

7.2.1.1 Resource type property definition

Table 6. Resource type property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource type	rt	string	The resource type ID		R	yes	The property name rt is as described in IETF RFC 6690

Resource type property name (Table 6) 'rt' shall be designated as 'rt=<resource type value>'. The value of a resource type property is a '.' (dot) separated string where each sub-string separated by a "." represents a namespace and the last sub-string represents the name of the resource type in that namespace hierarchy. The namespace 'oic' when used in the first sub-string, is reserved for OIC specifications and is used to namespace OIC Resource types. The resource type value may also be a reference to an authority similar to IANA that may be used to find the definition of a resource type.

Every OIC Resource shall have at least one resource type. A single resource may be assigned multiple resource types only when the assigned resource types don't conflict. The resource type is bound to a resource at the time of its creation.

Some examples follow:

- Light is resource type (one type to manage both toggle and dimmer); client may only read properties
  `/light/a, rt=Light, if= oic.if.r`
- *`ToggleLight`* is a resource type that defines the "powerstate" property which may have value "on"/"off"; may set and read on this resource
  `/light/b, rt=ToggleLight, if= oic.if.rw`
- Using this URI client may set and read the "dimmerstate" property (defined in resource type *`DimmerLight`*)
  `/light/c, rt=DimmerLight, if= oic.if.rw`
- Since the resource type "*`Light`*" supports read and set both powerstate and dimmerstate, the URI *'`/light/d`* ' may be used to turn on/off the light and also to dim it.
  `/light/d, rt=Light, if= oic.if.rw`

All the URI refer to the same light bulb. The property name 'rt' defines the resource type and 'if' defines the interface

7.2.1.2 Resource interface property definition

Table 7. Resource interface property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface	if	string	Dot separated string		R	yes	The property name if is as described in IETF RFC 6690

Resource interface property name (Table 7) **'if'** shall be designated as 'if=<resource interface value>'. The value of a resource interface property is a '.' (dot) separated string where each sub-string separated by a "." represents a namespace and the last sub-string represents the name of the resource interface in that namespace hierarchy. The namespace 'oic' when used in the first sub-string, is reserved and is used to namespace OIC Resource interfaces. The resource interface value may also be a reference to an authority similar to IANA that may be used to find the definition of a resource interface.

1487

`if=oic.if.ll` – Link list interface may return links (reference to other resources).
`GET //oic/mycollection?if=oic.if.ll` (returns links which are fully qualified)

1488

1489 The OIC resource model defines the standard interfaces listed in Table 8.

1490 **Table 8. OIC standard interface**

Interface	Name	Applicable Methods	Description
Baseline	oic.if.baseline	RETRIEVE, UPDATE	The Baseline interface shall be available on all OIC resources from the time of their creation. The Baselinei interface is used when an OIC Client needs to retrieve all the properties of the resource. The OIC Client includes the `?if="oic.if.baseline"` in a RETRIEVE request. The OIC Server shall respond with a representation of all the properties it has at the time of the request. If the OIC Server is unable to send back the whole resource representation, it shall reply with an error message. The OIC Server shall not return a partial resource representation. The properties of a resource may be modified by using the UPDATE request to the oic.if.baseline interface with the list of properties and their updated values.
List Links	oic.if.ll	RETRIEVE	The List Links interface is used to list the references (links) to other OIC Resources contained in an OIC Resource. The concept borrows from IETF draft-ietf-core-interfaces-02 section 5.1 with the following semantics The serialization shall be in the format requested in the request. In response to a RETRIEVE request on List Links interface, the URIs of the referenced OIC Resources shall be returned as fully qualified URIs. If there are no links present in an OIC Resource then an empty list shall be returned.
Batch	oic.if.b	RETRIEVE, UPDATE	The concept borrows from IETF draft-ietf-core-interfaces-02 section 5.2 with the following semantics An OIC Resource with a Batch interface may have both local and remote references (fully qualified). For local resources, the semantics in the IETF spec applies with the exception that resources shall not use Batch interface as an extension of the Link List interface. For remote resources: A request to the batch interface of an OIC Resource is processed by the OIC Resources that are referenced within that resource. To select a subset of the referenced resources the other query parameters, if present shall be processed first. Then the original request is modified to create new requests targeting each referenced resource by substituting the Authority and Path part of the URI in the original request with the Authority and Path of the referenced resource. The payload in the original request is replicated as the payload for the new request. The response from all the referenced OIC Resources shall be aggregated as a single response to the original request and returned to the OIC Client. If

			the target OIC Resources cannot process the new request, an empty response shall be returned.
Read-Only	oic.if.r	RETRIEVE	The Read-Only interface limits the applicable methods that can be applied to an OIC Resource to RETRIEVE only. An attempt by an OIC Client to apply a method other than RETRIEVE to an OIC Resource when applying the oic.if.r interface shall be rejected with a response code of 4.05 (if using CoAP) or 405 (if using HTTP).
Read-Write	oic.if.rw	RETRIEVE, UPDATE	The Read-Write interface limits the applicable methods that can be applied to an OIC Resource to RETRIEVE and UPDATE only.
Actuator	oic.if.a	CREATE, RETRIEVE, UPDATE	The Actuator interface adheres to the definition of the Actuator interface defined by the IETF draft-ietf-core-interfaces-02 with the following normative changes: A RETRIEVE action on this interface only provides (subject to any queryParameters that may also exist) the properties identified as part of the resource representation; it does not provide the properties defined as common properties for the resource. An UPDATE action on this interface shall also provide a payload or body that contains the requested values for the target resource. A resource using this interface shall return a media type of CBOR as defined in IETF RFC 7049.
Sensor	oic.if.s	RETRIEVE	The Sensor interface adheres to the definition of the Sensor interface defined by the IETF draft-ietf-core-interfaces-02 with the following normative changes: A RETRIEVE action on this interface only provides (subject to any queryParameters that may also exist) the properties identified as part of the resource representation; it does not provide the properties defined as common properties for the resource. A resource using this interface shall return a media type of CBOR as defined in IETF RFC 7049.

1491

1492 The following is an example of the RETRIEVE operation on a batch interface

1. Retrieve with batch interface

REQUEST: GET oic://192.168.1.114:52383/a/room?if=oic.if.b

RESPONSE:
{"oic":[{"href":"/a/room"},{"href":"/a/light","rep":{"state":"0","color":"0"}},{"href":"/a/fan","rep":{"state":"1","speed":10}}]}

2. Update with batch interface

REQUEST: POST oic:// 192.168.1.114:52383/a/room?if=oic.if.b ; **PAYLOAD**:
{"state":"1","speed":"1"}

RESPONSE:
{"oic":[{"href":"/a/room"},{"href":"/a/light","rep":{"state":"1","color":"0"}},{"href":"/a/fan","rep":{"state":"1","speed":1}}]}

Note: Since both /a/light and /a/fan have a property "state" both these are set to true because of the batch interface.

7.2.1.3 Policy property definition

Table 9. Policy Property Definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Policy	p	string	The policy name		R	no	

Policy property name (Table 9) ‘p’ shall be designated as ‘p=<policy value>’. The value of a policy property is defined as {“bm”: “<bitmap>”} where <bitmap> is a bitmap of one octet.

The least significant bit (LSB) in the bitmap shall be set to 1 if the resource is discoverable and shall be set to 0 otherwise. The second LSB shall be set to 1 if the resource is observable and set to 0 otherwise. The default value of any of these specified bits is 0 (which implies that resources are non-discoverable unless made discoverable and resources are not observable until explicitly made observable). Not including the Policy property is equivalent of the value of the bitmap being set to 0.

The setting in the bitmap on the resource will lead to appropriate elements being defined in the discovery information of that resource and representation of the resource as required.

The bits in the bitmap that are not defined in this specification are reserved for future use of OIC specifications.

7.2.1.4 Name property definition

Table 10. 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)

7.2.2 OIC Core Resources

OIC 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 OIC Core Resources, ‘/oic/res’ and ‘oic/d’ shall be supported on all OIC Devices. OIC Devices may support other OIC Core Resources depending on the functional interactions they support. Additionally, a Resource Type ID for all well-known resources is provided for consistency, but is not transmitted by an OIC Device.

8 CRUDN

8.1 Overview

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

CRUDN operations utilize a set of parameters that are carried in the messages and are defined in Table 11. An OIC Device shall use CBOR as the default payload (content) encoding scheme for resource representations included in CRUDN operations and operation responses; an OIC Device may negotiate a different payload encoding scheme (e.g, see in section 12.2.3 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 11. 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 OIC 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 OIC Resources on the OIC Server. The CREATE operation is initiated by the OIC Client and consists of three steps, as depicted in Figure 16 and described below.

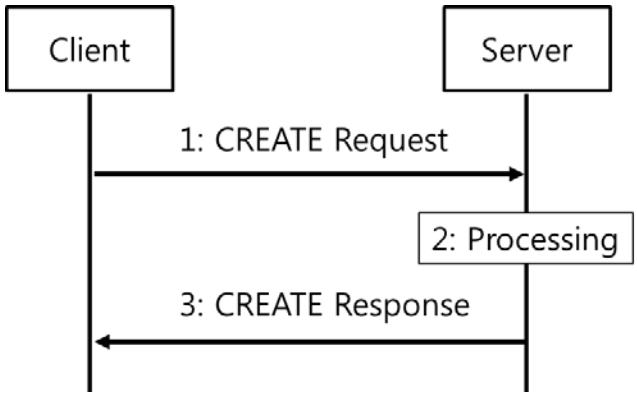


Figure 16. CREATE operation

8.2.1 CREATE request

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

- *fr*: Unique identifier of the OIC 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 OIC 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 OIC Server

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

8.2.3 CREATE response

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

- *fr*: Unique identifier of the OIC Server
- *to*: Unique identifier of the OIC Client
- *ri*: Identifier included in the CREATE request
- *cn*: Information of the resource as created by the OIC 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 an OIC Resource. The RETRIEVE operation is initiated by the OIC Client and consists of three steps, as depicted in Figure 17 and described below.

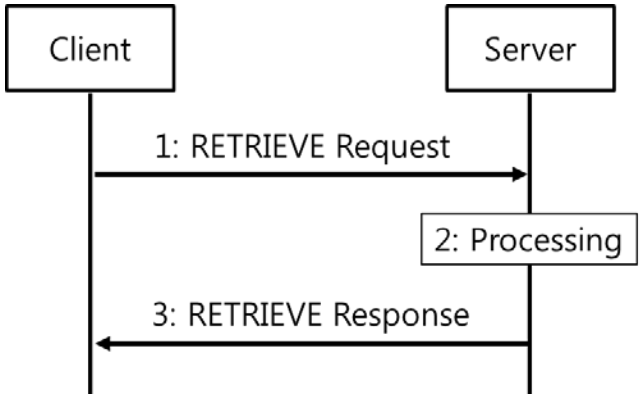


Figure 17. RETRIEVE operation

8.3.1 RETRIEVE request

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

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

8.3.2 Processing by the OIC Server

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

8.3.3 RETRIEVE response

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

- *fr*: Unique identifier of the OIC Server
- *to*: Unique identifier of the OIC Client
- *ri*: Identifier included in the RETRIEVE request
- *cn*: Information of the resource as requested by the OIC 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 used to request a partial or complete replacement of the information in an OIC Resource. The UPDATE operation is initiated by the OIC Client and consists of three steps, as depicted in Figure 18 and described below.

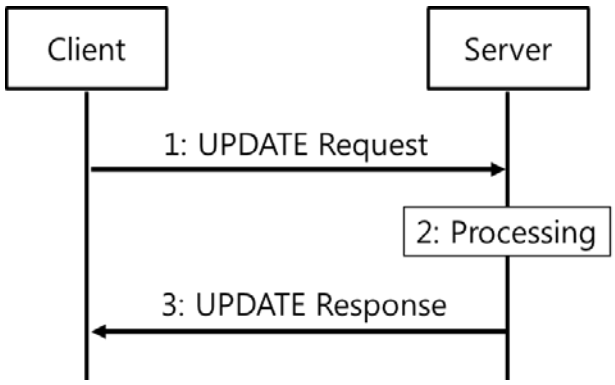


Figure 18. UPDATE operation

8.4.1 UPDATE request

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

- *fr*: Unique identifier of the OIC 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 OIC Server

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

8.4.3 UPDATE response

The OIC Server shall send an UPDATE response message in response to an UPDATE request message from an OIC Client. The UPDATE response message will include the following parameters.

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

8.5 DELETE

The DELETE operation is used to request the removal of an OIC Resource. The DELETE operation is initiated by the OIC Client and consists of three steps, as depicted in Figure 19 and described below.

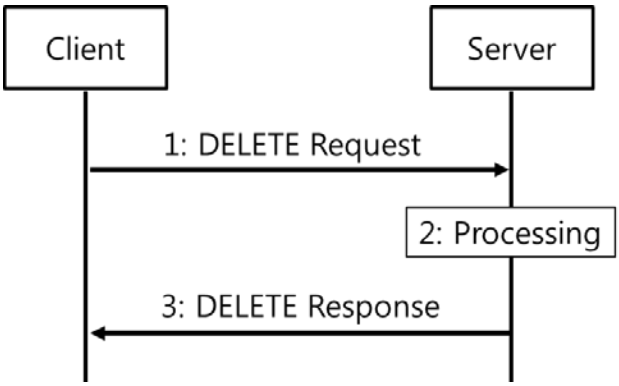


Figure 19. DELETE operation

8.5.1 DELETE request

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

- *fr*: Unique identifier of the OIC 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 OIC Server

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

8.5.3 DELETE response

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

- *fr*: Unique identifier of the OIC Server
- *to*: Unique identifier of the OIC 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 OIC Server to notify the URLs identified by the OIC Client of a state change. The NOTIFICATION response message carries the following parameters.

- *fr*: Unique identifier of the OIC Server
- *to*: URI of the OIC 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 OIC 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 PHY/DLL layers, 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, Wi-Fi, or 802.15.4) and concentrates on the network layer and the associated DLL 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's (NAT's). 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 OIC 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 20.

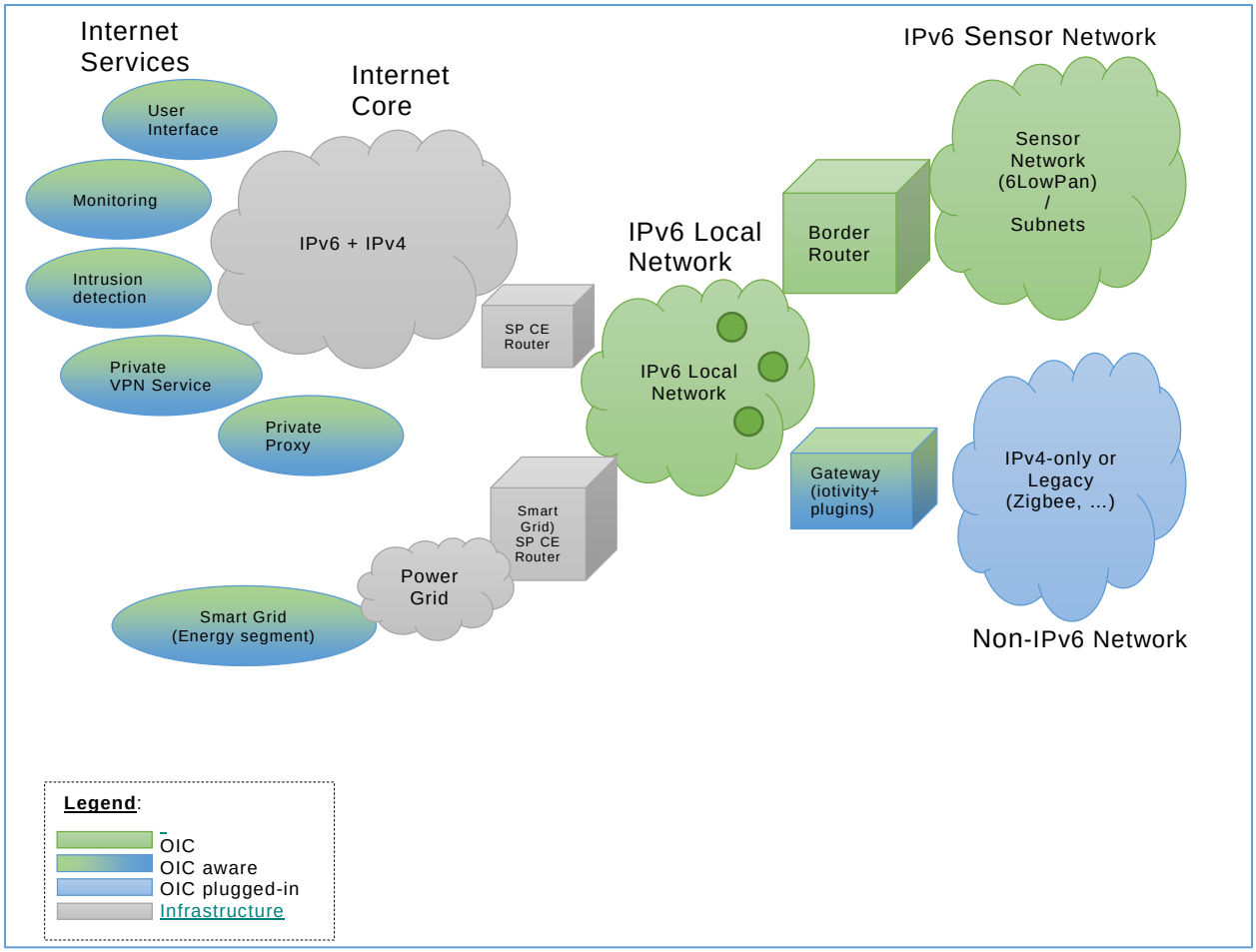


Figure 20. High Level Network & Connectivity Architecture

Devices depicted in Figure 20 assume one of several roles:

- IPv6 Node, IPv6 Router and IPv6 Host as defined in IETF RFC 6434
- CE Router (Customer Edge Router): As defined in IETF RFC 7084.
- 6LN (6LoWPAN Node), 6LR (6LoWPAN Router), 6LBR (6LoWPAN Border Router) as defined in IETF RFC 6775.
- IPv6 Translator: A device which translates and routes messaging between IPv6 and non-IPv6 networks. An example of a translator is the gateway in Figure 20.
- Constrained Nodes: a node that due to the constrained environment (limited processing power, memory, non-volatile storage and transmission capacity) requires special adaptation layers under the IP Network layer and requires dedicated routing protocols. Examples include devices transmitting over low power PHY's like IEEE 802.14.5, ITU G9959, BLUETOOTH Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),

9.3 IPv6 network layer requirements

9.3.1 Introduction

Projections indicate that many 10s of billions of new IoT endpoints and related services will be brought online in the next few years. These endpoint's capabilities will span from battery powered nodes with limited compute, storage, and bandwidth to more richly resourced devices operating over Ethernet and 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. The OIC 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. OIC recommends IPv6 as the common network layer protocol to ensure interoperability across all OIC devices. More capable devices may also include additional protocols creating multiple-stack devices. The remaining 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 CoAP Nodes” address FF0X::FD joined on an interface.

9.3.3 IPv6 router

An IPv6 router shall support all node requirements defined in section 9.3.2.

9.3.4 IPv6 host

An Ipv6 host shall support all node requirements defined in section 9.3.2.

9.3.5 IPv6 constrained nodes

9.3.5.1 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 in sections in order to support IPv6.

9.3.5.2 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)”.

9.3.5.3 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 draft-ietf-6lo-btle-14 and related updates.

10 Endpoint discovery

10.1 Introduction

This section describes how an OIC Endpoint is discovered by another OIC Endpoint in a network. An OIC Endpoint shall support CoAP discovery. HTTP discovery is optional for an OIC Endpoint. For CoAP transport protocol the Endpoint discovery uses mechanisms as defined in IETF RFC 7252. If HTTP discovery is supported, then Endpoint discovery uses mDNS as defined in IETF RFC 6762. Endpoint discovery is a Multicast discovery which is not required when the Endpoint is already known.

10.2 CoAP based Endpoint discovery

The following is the description of CoAP based Endpoint discovery:

a) All advertising or publishing OIC Devices shall join the ‘All CoAP Nodes’ multicast group, i.e. FF0X::FD for IPv6 and listen on the port “5683“.

b) The OIC Client that wants to discover resources shall first join the ‘All CoAP Nodes’ multicast group.

c) Then the OIC Client shall send a discovery request (GET request) to the multicast group ‘All CoAP Nodes’ and port 5683 where the URI in the request shall be /oic/res.

d) If the OIC Client is in the process of discovering a particular resource type, then it shall use the Query mechanism (section 6.2.1) with key “rt” and the wanted target as value of rt

e) When “rt” Query key is omitted all OIC Devices shall respond to that request

f) The considerations for handling multicast requests shall be as described in section 8 of IETF RFC 7252 and section 4.1 in IETF RFC 6690.

g) The OIC Devices that receive this request shall respond using CBOR as the payload (content) encoding. An OIC Device shall indicate support for CBOR as an additional supported payload (content) encoding for multicast discovery as described in section 12.2.3. An OIC Device shall respond to a received multicast discovery that indicates support for CBOR using CBOR as the payload (content) encoding. In later versions of the specification other formats could be included (e.g., JSON, XML/EXI).

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

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

Request

```
GET /oic/res
```

Response

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

To search for oic.r.light resources on the network an OIC Client can issue:

Request

```
GET /oic/res?rt=”oic.r.light”
```

Response

```
[
  {
   "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
    "links":[
    {
      "href": "/light",
      "rt": "oic.r.light",
      "if": "oic.if.s"
    }
    ]
```



```
  }
]
```

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

Request

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

Response

```
[
  {
   "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
    "links":[
    {
      "href": "/binarySwitch",
      "rt": "oic.r.switch.binary",
      "if": "oic.if.a",
      "type": "application/cbor"
    }
    ]
  }
]
```

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 an OIC Client and an OIC 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 12. More details about these resources are provided later in this section.

Table 12. List of OIC Core Resources

Fixed URI	**Resource Type Title**	Related **Functional Interaction**	**Requirement (M/CR/O)**
/oic/res	Default	Discovery	M
/oic/p	Platform	Discovery	M
/oic/d	Device	Discovery	M
/oic/rts	Resource Type	Discovery	CR
/oic/ifs	Interface	Discovery	CR
/oic/con	Configuration	Device Management	CR
/oic/mon	Monitoring	Device Management	CR
/oic/mnt	Maintenance	Device Management	CR

11.2 Provisioning

Provisioning in OIC Framework includes two distinct processes: On-Boarding and Configuration.

On-Boarding is the process which delivers required information to an OIC Device for joining the OIC network. When On-Boarding process is completed, the OIC Device has necessary information and is able to join the OIC network (State #1 in Figure 21). Further details about provisioning can be found in OIC Security specification (Owner PSK).

Configuration is the process which delivers required information to a device for accessing OIC services. At the end of the configuration process, the OIC Device has all the necessary information and is able to access OIC services (State #2 in Figure 21).

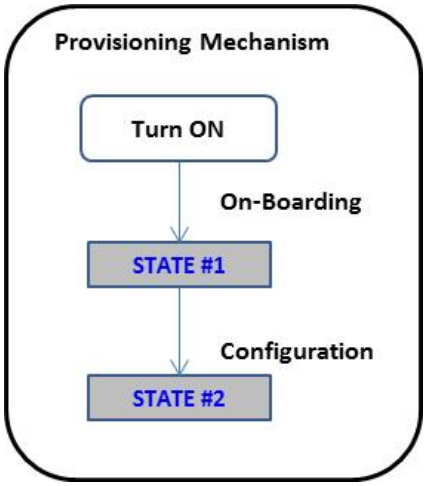


Figure 21. Provisioning State Changes

#1 On-Boarding

OIC 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., presence sensors. Additionally, the OIC Devices may support different communication and connectivity technologies, e.g., Bluetooth, Wi-Fi, etc. Different communication and connectivity technologies provide different on-boarding 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 On-Boarding, instead, specifies the state of the OIC Device upon completion of the OnBboarding process.

As part of the On-Boarding 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 On-Boarding 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 On-Barding across different communication and connectivity technologies.

#2 Configuration

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

The configuration information is also in OIC core resource '/oic/con'. Upon completion of the On-Boarding process and as soon as the OIC 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 OIC 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 OIC Device retrieves them from the Configuration Source. During the lifetime of an OIC Device an OIC Client may retrieve or update the configuration state of the OIC 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 22 depicts the interactions triggered by an OIC Device to retrieve its configuration information from the Configuration Source which may be located on a remote OIC Device or locally. These interactions occur instantly following completion of On-Boarding process; the OIC Device may retrieve its configuration at any time during its lifetime.

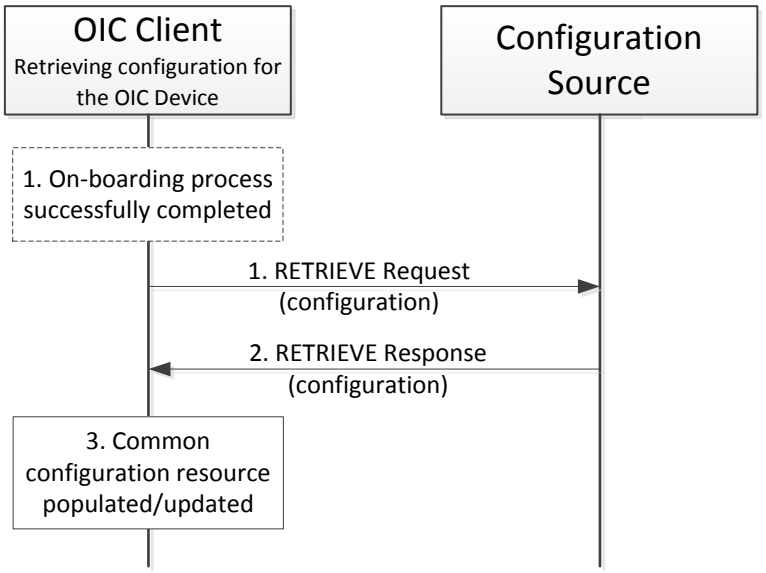


Figure 22. Interactions initiated by the OIC Device to retrieve its configuration from a configuration source

Figure 23 depicts the interactions when the retrieve of configuration information is done by an OIC Client.

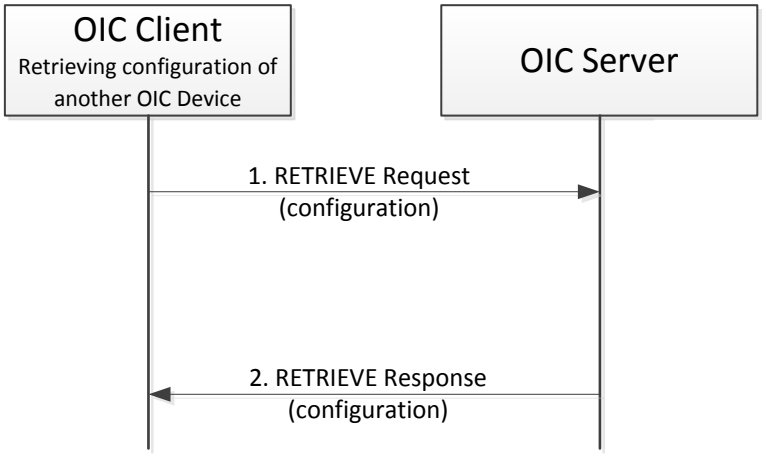


Figure 23. Interactions for retrieving the configuration state of an OIC Device.

Figure 24 depicts the interactions when the configuration information of an OIC Device is updated by an OIC Client, e.g., the Configuration Source.

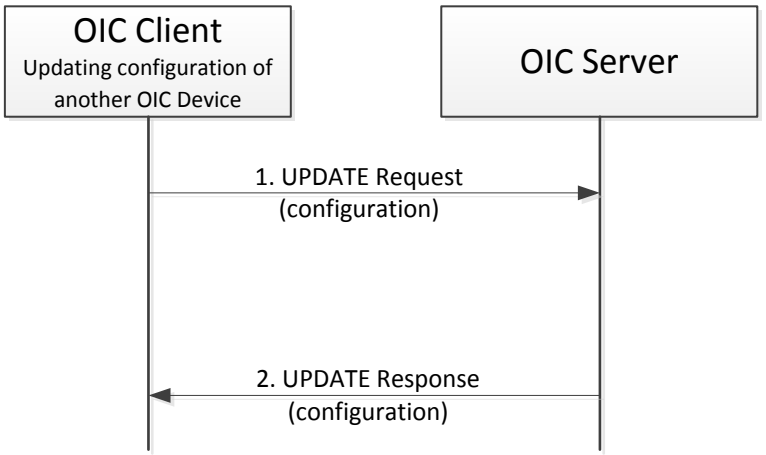


Figure 24. Update of and OIC Device configuration

If Configuration is supported by an OIC Device, i.e., the configuration information may be dynamically updated, the OIC Core Resource /oic/con shall be supported as the designated configuration resource as described in Table 15.

Configuration Resource
An OIC Device or an OIC Platform may be initially configured from information that is set or provisioned at bootstrap. In addition, the OIC Device and OIC 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 collectivly 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 13. Configuration Resources

Fixed 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 OIC Device is exposed. The **resource properties** exposed by /oic/con are listed in Table 14.	Configuration

Table 14 defines the oic.wk.con resource type.

Table 14. 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 For example, "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, an OIC Client may find appropriate information about other OIC peers. This information could be instances of resources, resource types or any other information represented in the resource model that an OIC peer would want another OIC 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 a) specified and commonly known a-priori or at bootstrap (e.g., specified in vertical specification) or b) discovered (e.g., using out of band methods)
3) An OIC Device for hosting the resource to enable discovery shall be identified.
4) A mechanism and process to publish the information that needs to be discovered with the resource to enable discovery.
5) 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.
6) A scope for the publication
7) A scope for the access.
8) A policy for visibility of the information.

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

- Direct discovery, where the OIC Resources are published locally at the OIC Device hosting the resources and are discovered through peer inquiry.
- Indirect discovery, where OIC 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.
- Presence, where the resource to enable discovery is hosted local to the initiator of the discovery inquiry but remote to the OIC Devices that are publishing discovery information.

An OIC Device shall support direct discovery.

11.3.2.2 Direct discovery

In direct discovery,

1) The OIC Device that is providing the information shall host the resource to enable discovery.
2) The OIC Device publishes the information available for discovery with the local resource to enable discovery (i.e. local scope).
3) OIC Clients interested in discovering information about this OIC 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” OIC Device that receives the request shall send a response with the discovered information directly back to the requesting “client” OIC 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 OIC Device.

11.3.2.3 Indirect discovery of OIC Resources (resource directory based discovery)

In indirect discovery the information about the resource to be discovered is hosted on an OIC 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 an OIC Device that is neither the client initiating the discovery nor the OIC Device that is providing or publishing the information to be discovered. This OIC 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 OIC Device to be discovered or with information to discover, publishes that information with resource to be discovered on a different OIC Device. The policies on the information shared including the lifetime/validity are specified by the publishing OIC Device. The publishing OIC Device may modify these policies as required.
c) The client doing the discovery may send a unicast discovery request to the OIC Device hosting the discovery information or send a multicast request that shall be monitored and responded to by the OIC Device. In both cases, the OIC Device hosting the discovery information is acting on behalf of the publishing OIC Device.
d) The discovery policies may be set by the OIC 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 OIC Device that is initiating the discovery request (client). The resource to enable discovery may be an OIC 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 OIC Device with information to be discovered shall publish the appropriate information to the resource that enables discovery.
d) The publishing OIC Device is responsible for the published information. The publishing OIC 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 OIC 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 25. The OIC 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 OIC Device which hosts the resource that enables discovery. The OIC Device hosting the resource to enable discovery adds/updates the resource to enable discovery with the provided information and then responds to the OIC Device which has requested the publication of the resource with an UPDATE response.

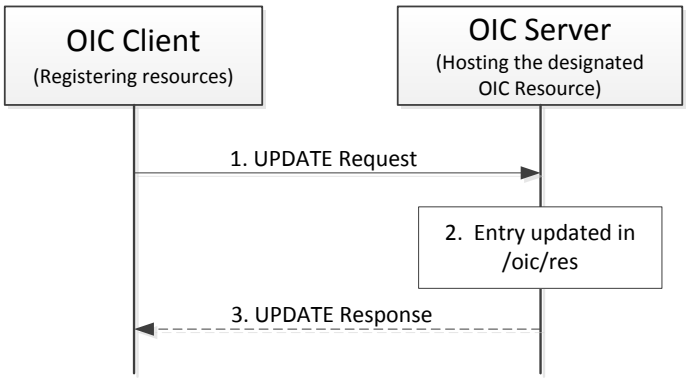


Figure 25. Resource based discovery: Information publication process

11.3.4 Resource based discovery: Finding information

The discovery process (Figure 26) is initiated as a RETRIEVE request to the resource to enable discovery. The request may be sent to a single OIC Device (as in a Unicast) or to multiple OIC 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.

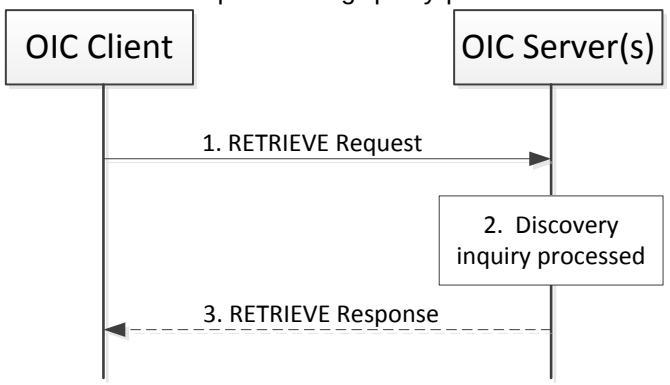


Figure 26. Resource based discovery: Finding information

Discovery Resources

Some of the OIC Core Resources (section 7.2.2) shall be implemented on all OIC Devices to support discovery. The OIC 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 OIC Core Resources are described in Table 15

Platform resource –

The OIC recognizes that more than one instance of OIC 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 OIC Device on the same OIC Platform shall have the same values of any properties exposed (i.e. an OIC 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 OIC Devices on that OIC Platform)

Device resource

The device resource shall have the fixed URI /oic/d. The resource /oic/d exposes the properties pertaining to an OIC Device as defined in Table 15. The properties exposed are determined by the specific instance of OIC Device and defined by the resource type(s) of /oic/d on that OIC 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 15.)

Protocol indication

An OIC 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, an OIC Device uses the protocol indication to indicate the transport protocols they support and can communicate over.

Table 15. Mandatory discovery OIC Core Resources

Fixed 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 OIC Server is discovered and introspected for available resources. The semantics for /oic/res shall be same as a collection that supports the link-list interface and whose referenced resources is the list of resources that are discoverable on that OIC Server. The resource properties exposed by /oic/res are listed in Table 16.	Discovery
/oic/p	Platform	oic.wk.p	oic.if.r	The resource through which platform specific information is discovered. The **resource properties** exposed by /oic/p are listed in Table 18	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 OIC Device instance. The **resource properties** exposed by /oic/d are listed in Table 18 /oic/d may have one or more resource types that are specific to OIC 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 OIC 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

2116 Table 16 defines oic.wk.res resource type.

2117 **Table 16. oic.wk.res resource type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Device Identifier	di	UUID			R	yes	The device identifier as indicated by the /oic/d resource of the device
Links	json	string	See 7.1.6.2		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 17 For example, “1 and 3” indicates that the OIC device supports coap and http as messaging protocols.

2118 An OIC Device which wants to indicate its messaging protocol capabilities may add the property
2119 ‘mpro’ in response to a request on /oic/res. An OIC Device shall support CoAP based discovery as
2120 the baseline discovery mechanism (see section 10.2). An OIC Client which sees this property in a
2121 discovery response can choose any of the supported messaging protocols for communicating with
2122 the OIC Server for further messages. For example, if an OIC Device supporting multiple protocols
2123 indicates it supports a value of “1 3” for the ‘mpro’ property in the discovery response, then it
2124 cannot be assumed that there is an implied ordering or priority. But a vertical service specification
2125 may choose to specify an implied ordering or priority. If the ‘mpro’ property is not present in the
2126 response, An OIC Client shall use the default messaging protocol as specified in the vertical
2127 specification for further communication. Table 17 provides an OIC registry for protocol schemes.

2128 **Table 17. Protocol scheme registry**

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

2129 Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by an OIC Client
2130 to form requests in a different messaging protocol other than discovery is out of scope.

2131

2132 The following applies to the use of /oic/d as defined above:

2133 • /oic/d is a well-known URI for the Device resource
2134 • The Device resource has a base resource type ID of oic.wk.d
2135 • The base resource type oic.wk.d shall have mandatory properties as defined in Table 3

- A vertical may choose to expose its Device Class (e.g., refrigerator or A/C) by overriding the Resource Type ID set associated with /oic/d.
    - For example; rt of /oic/d becomes ‘oic.d.<thing>’
- A vertical may also choose to expose its Device Class (e.g., refrigerator or A/C) by adding to the Resource Type ID set associated with /oic/d.
    - For example; rt of /oic/d becomes ‘oic.wk.d,oic.d.<thing>’
- The properties exposed for the Device Class specified resource type ID are by default the mandatory properties in Table 3
- A vertical may choose to extend the properties exposed via /oic/d with the use of a Device Class specific resource type ID.
    - However the mandatory properties defined in Table 3 shall always be present
- /oic/d is exposed in /oic/res as an entry in the set of web-links
- Should there be more than one resource type ID listed; then 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.
- The mandatory properties need to be in the device specific class type only if that device class type can be made a default. If a vertical defines more than one Device class resource type then all of them don’t need to carry these base mandatory properties.

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

Table 18. oic.wk.d resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R	yes	Human friendly name For example, “Bob’s Thermostat”
Spec Version	lcv	string			R	yes	Spec version of the core specification this device is implemented to, The syntax is "core.major.minor”]
Device ID	di	UUID			R	yes	Unique identifier for OIC Device (UUID)
Data Model Version	dmv	CSV			R	yes	Spec version(s) of the vertical specifications this device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor”]. <vertical> is the name of the vertical (i.e. sh for Smart Home)

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 19. 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	Platform Identifier
Manufacturer Name	mnmn	string			R	yes	Name of manufacturer (not to exceed 16 characters)
Manufacturer Details Link (URL)	mnml	URL			R	no	URL to manufacturer (not to exceed 32 characters)
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 URL	mnsl	URL			R	no	URL that points to support information from manufacturer
SystemTime	st	datetime			R	no	Reference time for the device

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 OIC server as a composition. One example method would be to have OIC Platform which represents the composite device to have more than one instance of an OIC Device on the OIC Platform. Each OIC Device instance represents one of the distinct devices in the composition. Each instance of OIC 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 OIC Server.

Thus, for the above refrigerator example if modeled as a single OIC 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.

Additional Discovery Resources

The OIC Core Resources that may be implemented to support additional discovery are:

- '/oic/rts' for discovery of resource types
- '/oic/ifs' for discovery of interfaces
- '/oic/ad' for presence discovery

Details for these optional OIC Core Resources are described in Table 20

Table 20. Optional discovery OIC Core Resources

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	**Interfaces**	**Description**	Related **Functional Interaction**
/oic/rts	Resource Type	oic.wk.rts	oic.if.r	The resource through which the supported resource types on an OIC Device are identified. The resource types can be pre-loaded or pre-configured at build time or may be downloaded at run time. This resource is read-only which implies that resource types cannot be pushed to the OIC Device (this does not preclude the OIC Device from "downloading" new types from a resource type registry/directory and then exposing those as supported). Resource types discovered through /oic/rts is used for specifying the resource types for resources that are CREATED. This resource is modelled as a simple resource. The **resource properties** exposed by /oic/rts are listed in Table 21.	Discovery
/oic/ifs	Interface	oic.wk.ifs	oic.if.r	The resource through which the supported interfaces are identified. This is interface information per-resource returned as part of resource discovery. This resource announces the supported interfaces that may be used for creating resources on this OIC Server. The **resource properties** exposed by /oic/ifs are listed in Table 22.	Discovery
/oic/ad	Advertisement	oic.wk.ad	oic.if.rw	The resource through which the OIC Devices advertise their presence. This allows "advertisement" to be enabled as a mode of discovery at the resource model level. Advertisements could be for resources, presence, state changes, etc. The advertisement is done as a CREATE or UPDATE request to this resource. This resource is modelled as a collection. The resource type of resources i.e. oic.wk.ad exposed by /oic/ad are listed in Table 23.	Discovery

Table 21 defines oic.wk.rts resource type.

Table 21. oic.wk.rts supported resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Type list	tl	BSV	0 to many resource type IDs		R	yes	List of the resource types supported by the OIC Device.

Table 22 defines oic.wk.ifs resource type.

Table 22. oic.wk.ifs resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface list	il	BSV	0 to many resource interfaces		R	yes	List of the resource interfaces supported by the OIC Device.

Table 23 defines oic.wk.ad resource type. This is the type of the resource that is posted (advertised) to the /oic/ad resource to announce the presence of a resource. The resource /oic/ad is a collection with each resource of type oic.wk.ad. To announce the presence of a device the fully qualified resource URI (from /oic/d) is published along with resource type as oid.wk.d.

Table 23. oic.wk.ad resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Time to Live	ttl	integer	Seconds		R, W	Yes	Time to live for this information expressed in seconds.
Nonce	non	integer	A random number or monotonically increasing from 1. If it is just monotonically increasing it will roll over causing duplicates over some period of time.		R, W	Yes	The nonce is used to make sure that the current representation is different from another representation that the resource may have been updated with before. This allows OIC Clients that monitor this to know when the different representation is available or if they do an RETRIVE

							operation on the resource then they can compare with a previous RETRIEVE operation to make sure this is a new representation.
Announcement Trigger	trg	enum	create, change, delete		R, W	Yes	Reason for this announcement can be creation or deletion of the resource or change, to one or more properties in a resource.
Resource URI	href	URI			R	No	Fully qualified URI of the resource whose presence is being announced.
Resource type	rt	BSV	1 to many (as space separated)		R	Yes	Resource type

11.3.5 Resource discovery using '/oic/res'

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

a) Every OIC Device updates its local '/oic/res' with the resources that are discoverable (see section 7.1.4.2.4). Every time a new resource is instantiated on the OIC Device and if that resource is discoverable by a remote OIC Device then that resource is published with the '/oic/res' resource that is local to the OIC Device (as the instantiated resource).

b) An OIC Device wanting to discover resources or resource types on one or more remote OIC Devices makes a RETRIEVE request to the /oic/res on the remote OIC 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 OIC 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 OIC 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 (specifically 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 OIC 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 Figure 7 and Figure 9 in section 7.1.6.2. 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, OIC Clients may discover additional details about OIC Server by performing discovery using /oic/p, /oic/rts etc. If an OIC Client already knows about OIC 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.

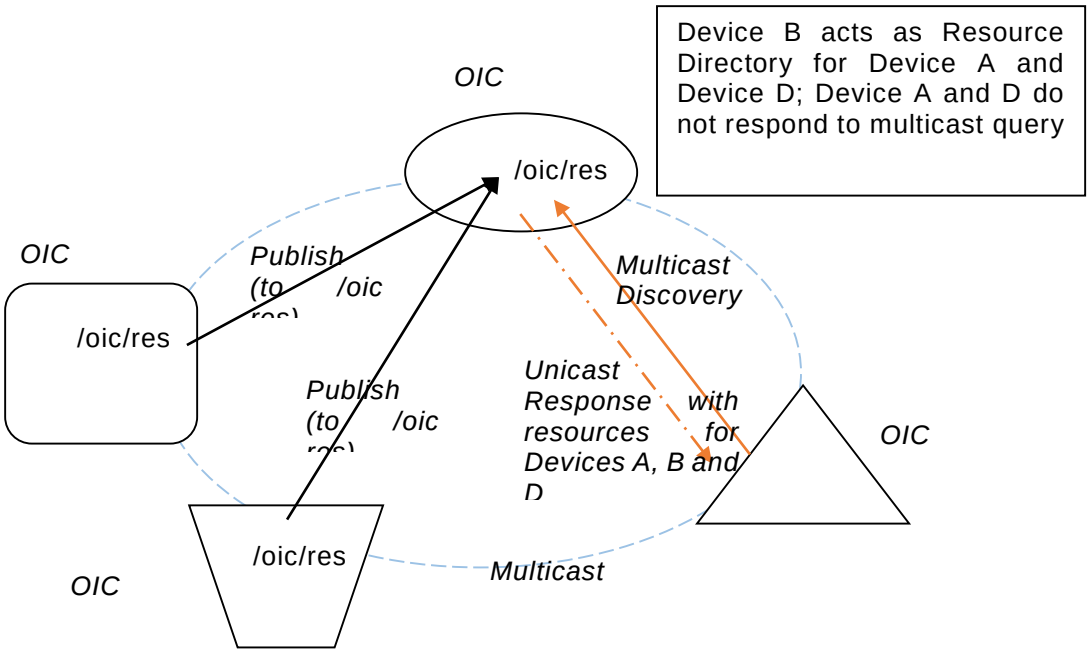


Figure 27. Indirect discovery of resource by resource directory

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 OIC Device that hosts that resource.

11.3.6.1.2 Resource directory

A resource directory (RD) is an OIC Device that assists with indirect discovery. A RD can be queried at its /oic/res resource to find resources hosted on other OIC Devices. These OIC Devices can be sleepy nodes or any other device that cannot or may not respond to discovery requests. OIC 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 OIC Device (for example: when an OIC Device may go to sleep). For general resource discovery, the RD behaves like any other OIC Server in responding to requests to /oic/res.

Any OIC Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The OIC 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 weighting 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 OIC Device shall select one RD (per context) to publish its resources. A context is any set of circumstances that would one set of OIC Clients from another for the purpose of resource discovery. A context is usually determined at deployment and from application requirements. An example of a context could be a multicast group- an OIC 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 OIC Device may decide to choose other RDs during its lifetime but at no instance shall the OIC Device publish its resource information to more than one RD per context. Devices such as TV, network router, desktop will have higher weightage or bias factor compared to mobile phone device.

This remainder of this section is divided into two parts first part covers discovering of the RD and publishing, updating and deleting of resources for the constrained/sleepy device. Second part covers where RD replies to queries from devices looking to discover resources.

11.3.6.2 Resource directory discovery

11.3.6.2.1 Discovering a resource directory

A RD in the OIC 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.

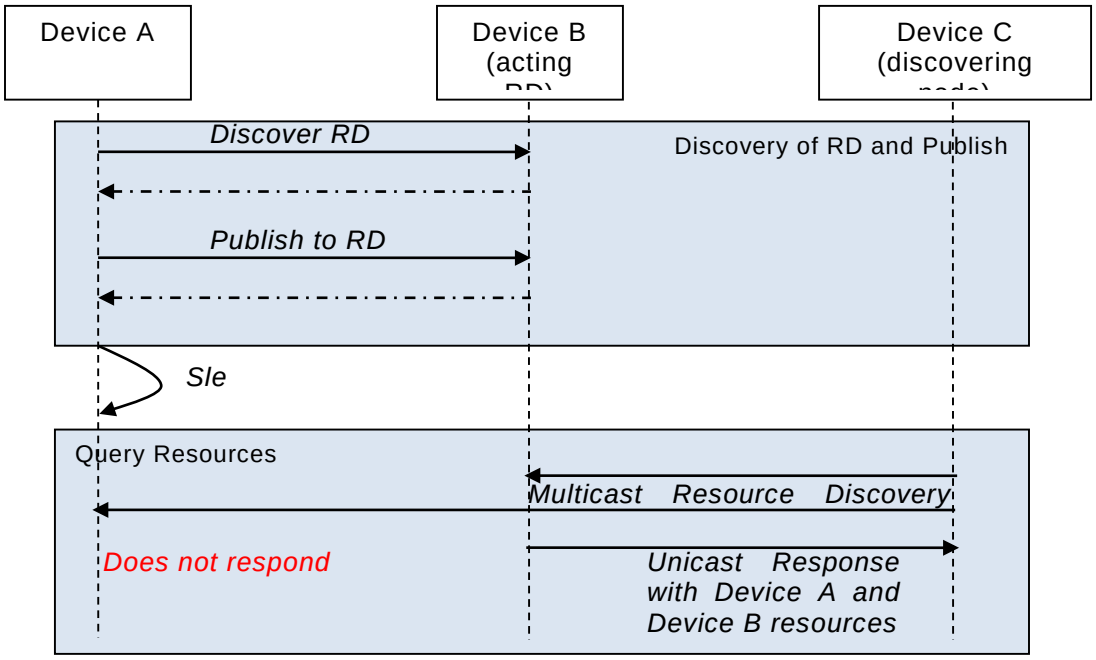


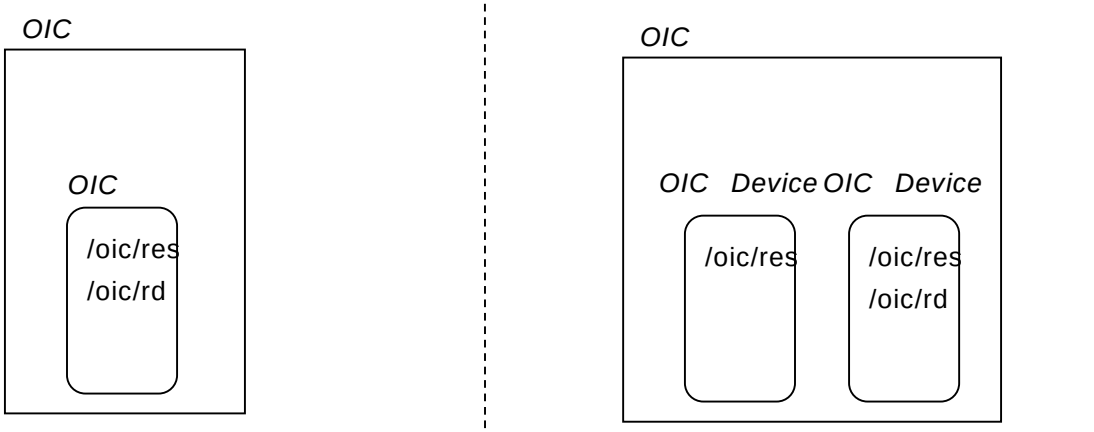
Figure 28. RD discovery and RD supported query of resources support

As shown in Figure 28, the OIC 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 OIC 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 OIC Device acting as RD on an OIC Platform.

Discovering of an RD could 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. An OIC Client that performs resource discovery uses an RD just like it may use any other OIC 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.



OIC Device serving as Resource Directory

OIC Platform with dedicated Resource Directory

Figure 29. 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 on-boarding 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: An OIC 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 and all devices that can be a RD shall host the /oic/rd and 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 is a conditionally mandatory core resource that shall be instantiated on all OIC Devices only when offering or acting as resource directories.

```
{
       "$schema": "http://json-schema.org/draft-04/schema#",
       "id": "http://openinterconnect.org/schemas/oic.rd.selection.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": {
```



```
          "type": "string",
          "description": "A unique identifier for the Resource Directory",
          "format": "uuid"
        },
        "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."
                },
                "cpu": {
                  "type": "integer",
                  "description": "Memory available at request in MHz units"
                },
                "memory": {
                  "type": "integer",
                  "description": "A processing capacity of the CPU specified in MB
units"
                },
                "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 10 minute window, in the last hour and hourly average over the RD
current lifetime"
                }
              }
            },
            {
              "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"
            }
          ]
        }
      },
      "required": [ "sel" ]
    }
  },
```



```
  "type": "object",
  "$ref": "#/definitions/oic.rd.attributes"
}
```

Figure 30. Information in a response to a query to /oic/rd

- Advertisement/Presence: An RD may advertise about its presence to devices. It is a combination of presence and advertisement packet. The devices that are already publishing to a RD may use this as a presence or 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. An OIC 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 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 weighting or 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 on-boarding or subsequent configuration of the RD or may be based on a formula determined by the implementation of the RD. (OIC 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 24 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 or point of presence.

Parameters: Optionally, parameters defined in Table 24 (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 24: 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 publishing device uses advertisement and query mechanism to find about a RD. There are four scenarios based on how selection process will work:

1) A single or multiple RDs are already present in the network when a new device boots up.
2) No RD is present in the network and a new device boots up.
3) Another device boots up that has the capability of working as a RD.
4) Two RD boots at the same time, like after power failure.

For first scenario where RD is already present, device listens to advertisement packet or queries for RDs on the multicast address, as described above. If a single or multiple responses are received publishing device uses the weightage information from the query response to select the RD to publish resources to.

In the second scenario, device will listen to the advertisement. Once RD advertisement packet is received it can receive the weightage information or retrieve it and if the bias meets its criteria, it can register its resources on the RD.

In the third scenario, if the publishing device is publishing to an existing RD and it discovers the new RD, the publishing device may choose to move to the RD if the relative bias factors favour the new RD. If the decision is made to select the new RD, the then publishing device shall delete its resource information from the previous RD and then publish the information to the new RD. (In the transition period, once the delete request has been sent, the publishing device shall respond to resource discovery requests).

In the last scenario, each RD starts and advertises about their presence. Publishing device based on the weightage criteria selects appropriate a RD for publishing its resource information.

11.3.6.3 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.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://openinterconnect.org/schemas/oic.rd.publish.json#",
  "title": "RD Publish & Update",
  "definitions": {
    "oic.rd.publish": {
      "type": "object",
      "description": "Publishes resources as OIC Links into the resource
directory",
```



```
      "properties": {
        "n": {
          "type": "string",
          "description": "Readonly, Human friendly name of the publishing OIC
Device"
        },
        "id": {
          "type": "string",
          "description": "ReadOnly, Unique identifier (UUID) for device that is
publishing",
          "format": "uuid"
        },
        "$ref":
"oic.collection.json#/definitions/oic.collection.setoflinks/properties",
        "lt": {
          "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"
        }
      },
      "dependencies": {
        "links": [ "lt" ]
      }
    }
  },

  "type": "object",
  "$ref": "#/definitions/oic.rd.publish",
  "required": [ "di" ]
}
```

Figure 31. Publish – push resource information

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. OIC 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).

11.3.6.4.2 Security considerations

Resource directory should support DTLS. Communication between device and resource directory should be based on the DTLS where ever possible. The two ends, device and resource directory should be authenticated using PSK, certificates and raw public key where ever possible.

The device should communicate with resource directory using the UUID after registration i.e. retrieval, update and delete should use UUID as the IP address can change where the resource are located and will confuse the resource directory.

Access control list, should be used for publish and lookup purpose as they might differ. Lookup should check where the request originates from, network or resource level.

Resource directory when a wild card lookup is requested should only be returned where routing is possible to check to avoid DDoS attacks. As with UDP it will not be possible to check routing, wild card lookup should be supported only over DTLS or TCP or TLS.

11.4 Notification

11.4.1 Overview

An OIC Server shall support NOTIFY operation to enable an OIC Client to request and be notified of desired states of one or more OIC 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 OIC Client utilizes the RETRIEVE operation to require the OIC Server for updates in case of OIC 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 7.1.4.2.4).

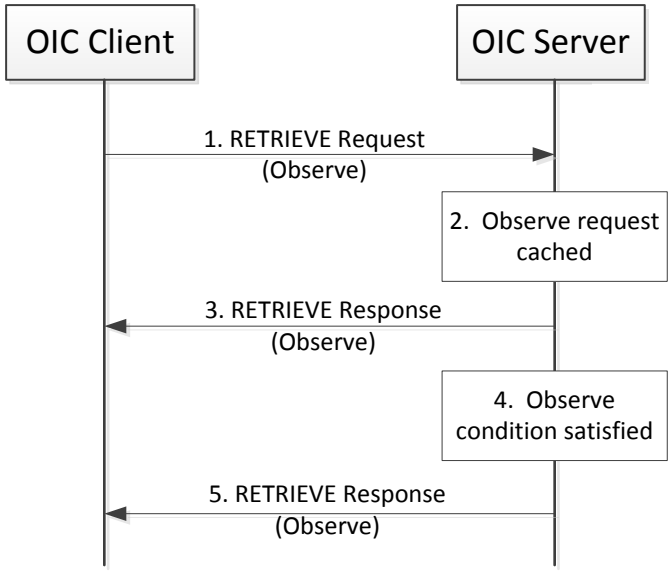


Figure 32. Observe Mechanism

11.4.2.1 RETRIEVE request with observe indication

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

- *fr*: Unique identifier of the OIC Client
- *to*: Resource that the OIC 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 OIC Server

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

11.4.2.3 RETRIEVE response with observe indication

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

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

The OIC Server shall monitor the state the OIC Resource identified in the observe request from the OIC Client. Anytime there is a change in the state of the observed resource, the OIC Server sends another RETRIEVE response with the observe indication.

11.4.2.5 Additional RETRIEVE responses with observe indication

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

11.4.2.6 Cancelling Observe

The OIC Client which does not want to receive any more responses shall not confirm the received response in which case OIC Server assumes that the OIC Client is no more interested in the response and cancels Observe for that OIC Client. Additionally, the OIC Client can explicitly cancel observe by sending a RETRIEVE request without observe field to the same resource on OIC Server which it was observing.

11.5 Device management

The OIC Device Management includes the following functions:

- Provisioning (On-Boarding and Configuration)
- Monitoring
- 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 Monitoring

Monitoring in OIC Framework is used for monitoring the current state of the OIC Devices, typically to check their functionality state and to ensure the OIC device is operating as expected. Monitoring includes periodic device availability check and/or device health check.

If Monitoring is supported by an OIC Device, the OIC Core Resource /oic/mon shall be supported as the designated monitoring resource as described in Table 25.

Table 25. Optional monitoring device management OIC Core Resources

Fixed URI	Resource Type Title	Resource Type ID (“rt” value)	**Interfaces**	**Description**	Related **Functional Interaction**
/oic/mon	Monitoring	oic.wk.mon	oic.if.r	The resource through which the OIC Device is monitored. The resource exposes properties relevant to aspects that may be monitored. The **resource properties** exposed by /oic/mon are listed in Table 26.	Device Management

Table 26 defines oic.wk.mon resource type.

Table 26. oic.wk.mon resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Availability	av	boolean	0 (not available) 1 (available)		R	yes	Indicates if the device is available or not to provide service even though it is discoverable on the network
LastedActedTime	lat	integer		sec	R	yes	Indicates the elapsed time in seconds after the device was invoked or acted upon
DeviceStatistics	ds	CSV			R	no	Contains Device Statistics Info as a CSV of integers in that order (no. of received packets, no. of sent packets)

An OIC Client may monitor an OIC Device by retrieving resource representation of the designated monitoring resource hosted by an OIC Server (see Figure 33) or by retrieving information related to a particular aspect from the designated monitoring resource (see Figure 34). Alternatively, the OBSERVE mechanism as specified in section 11.4.2 may be used.

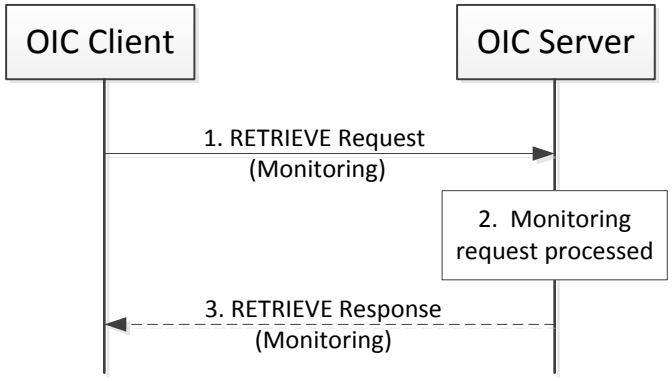


Figure 33. Retrieving all the monitoring information in a single request

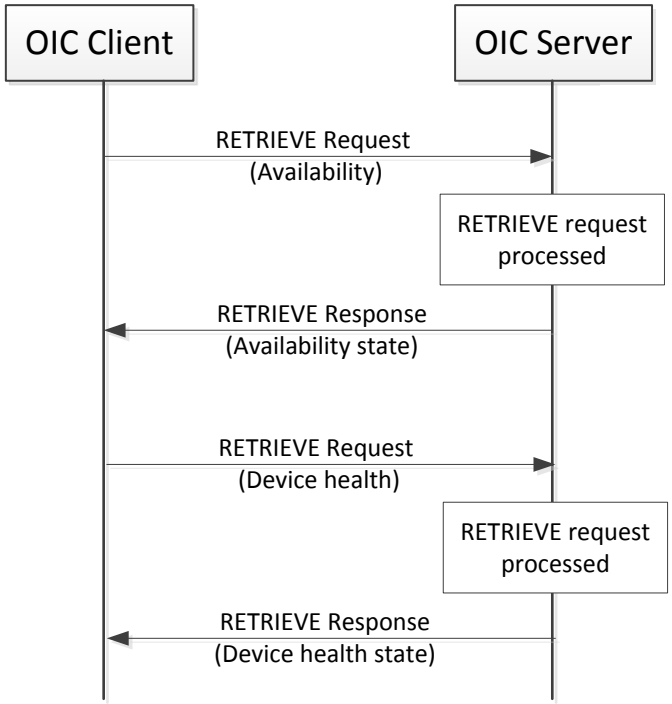


Figure 34. Retrieving specific Monitoring information in multiple requests

11.5.2 Diagnostics and maintenance

The Diagnostics and Maintenance function in OIC Framework is intended for use by the administrators to resolve issues encountered with the OIC Devices while operating in the field. If diagnostics and maintenance is supported by an OIC Device, the OIC Core Resource '/oic/mnt' shall be supported as described in Table 27.

Table 27. Optional diagnostics and maintenance device management OIC Core Resources

Fixed 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 28.	Device Management

Table 28 defines oic.wk.mnt resource type.

Table 28. 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	yes	0 – No action (Default*) 1 – Start Factory Reset

							After factory reset, this value shall be changed back to the default value After factory reset all configuration and state data will be lost
Reboot	rb	boolean			R, W	yes	0 – No action (Default) 1 – Start Reboot After Reboot, this value shall be changed back to the default value The reboot shall be finished within 60 seconds
StartStatCollection	ssc	boolean			R, W	Yes	0 – No collection of statistics 1 – Starts collecting statistics Toggles between collecting and not collecting any device statistics (ds property in /oic/mon) depending on the value being 0 or 1

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

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

- Factory_Reset: Updates the device configuration to its original (default) state (factory state and equivalent to hard reboot)
- Start_Collection: Triggers collection and logging information for diagnostic purposes
- Reboot: Triggers a soft reboot of an OIC Device maintaining most of the configurations intact

Execution of these commands may result in a change in the configuration state of an OIC Device. The configuration information in the configuration resource is expected to be updated following execution of these commands by the OIC Device, if needed. An OIC Client invokes operations on the OIC Server for executing the Diagnostic functions by sending an UPDATE message to the OIC Server. Figure 35 depicts the example of interactions between the OIC Client and the OIC Server for executing Factory Reset command.

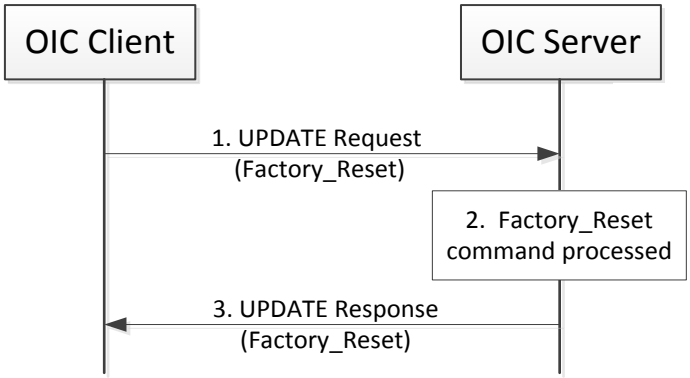


Figure 35. Factory_Reset command

11.5.3 Security considerations for device management

Device management operations have security implications on devices. Appropriate level of security needs to be applied to all device management operations.

11.6 Scenes, Rules and Scripts

11.6.1 Introduction

Scenes, rules, and scripts are mechanisms 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 OIC resources that may be hosted by multiple separate OIC servers. Scenes, once set up, can be used by multiple OIC clients to recall a setup.

A rule is a logical “if then” statement. It consists of a rule condition and a Rule Member (a script). The rule condition is an evaluation criterion which can include evaluation of the value of a sensor on an OIC Server. When the evaluation criterion is evaluated true then the Rule Members are set to a specific determined value. A rule condition is evaluated when one of the observed resources in the rule condition changes.

A script is a programmatic element that can be used to incorporate conditionals, delays, loops and other programmatic devices, including reading and writing scenes. Scripts can consist of a set of steps that are executed either upon meeting the conditions of a rule or as part of another script, in order to automate tasks. Scripts can also be used to set a scene to a specific value. For the purposes of this specification a Script is realized as the set of Rule Members that are executed when a rule condition holds true.

Rules, scripts and scenes can be grouped and reused:

- A group of scenes is also a scene.
- A group of scripts is also a script. A script can call other scripts (similar to subroutines) and read and set scenes.
- A group of rules is useful for setting up scenarios or modes. For example, a vacation scenario may include a random lighting rule, and day rule, a feed the dog rule, etc.

In short:

- Scenes are bundled user settings
- Scripts are automated background tasks
- Rules are conditional statements that execute scripts when the condition is true

11.6.2 Scenes

11.6.2.1 Introduction

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

The scene functionality is created by multiple resources and has the structure depicted in Figure 36. 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.

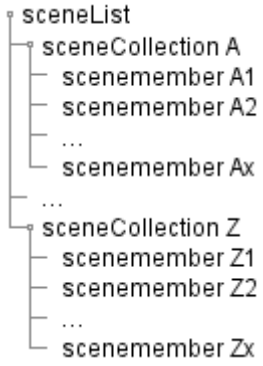


Figure 36 Generic scene resource structure

11.6.2.2 Scene creation

An OIC Client that wants to create scenes needs to verify that an OIC server supports the scene feature; the sceneMembers of a scene do not have to be co-located on the OIC 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 37. The sceneCollection can be created by an end user using a capable OIC Client. This will entail defining the scene with an applicable list of scene values and the mappings for each OIC 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 OIC Resource indicated by the link.

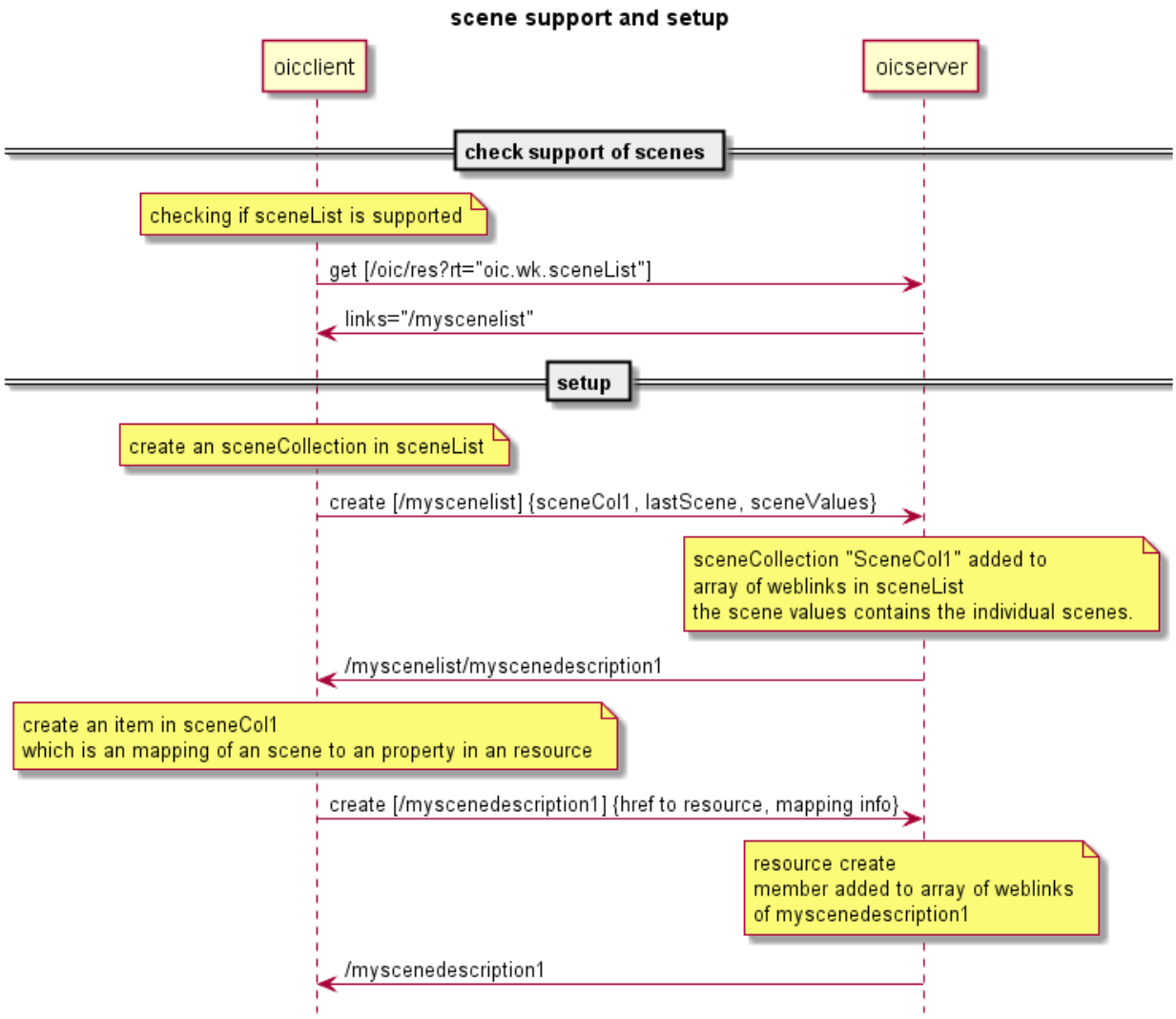


Figure 37 Interactions to check Scene support and setup of specific scenes

11.6.2.3 Interacting with Scenes

All capable OIC Clients can interact with scenes. The allowed scene values and the last applied scene value can be retrieved from the OIC 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 38. 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 OIC Client can change just one resource being part of the scene without having feedback that the state of the scene is changed.

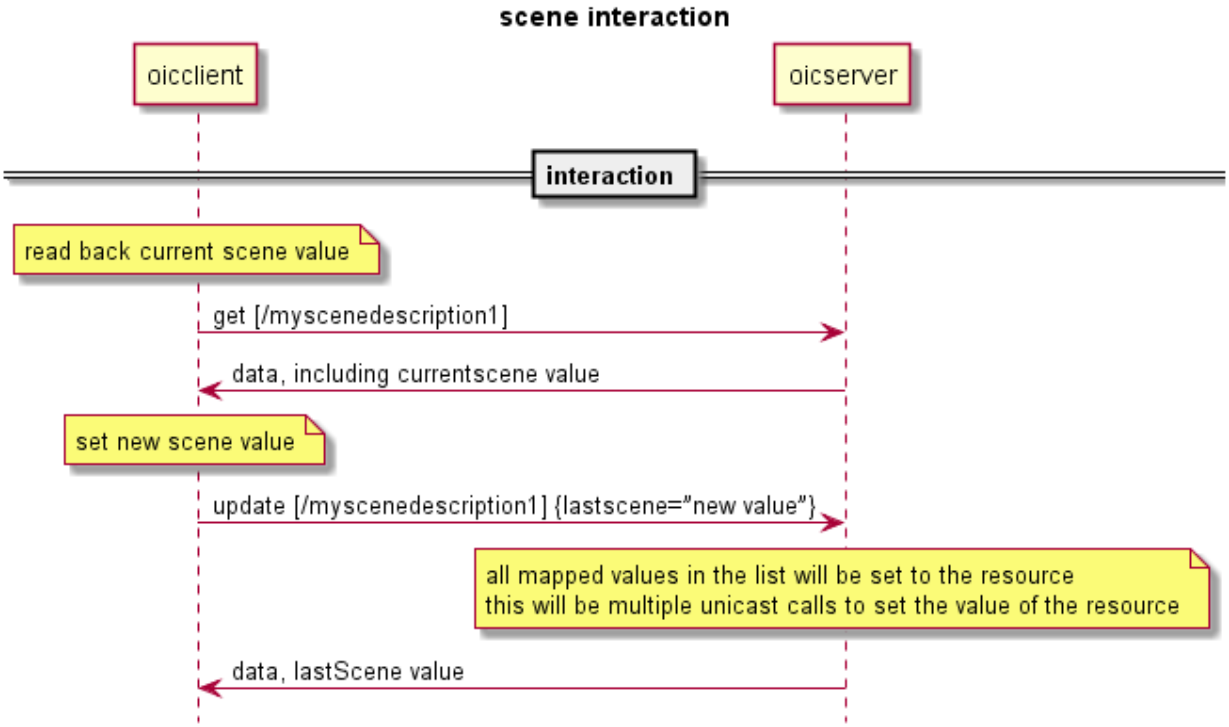


Figure 38 Client interactions on a specific scene

As described previously, a scene can reference one or more resources that are present on one or more OIC Servers. The scene members are re-evaluated each time a scene change takes place. This evaluation is triggered by an OIC Client that is either embedded as part of the OIC 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 OIC Server. This behaviour is depicted in Figure 39.

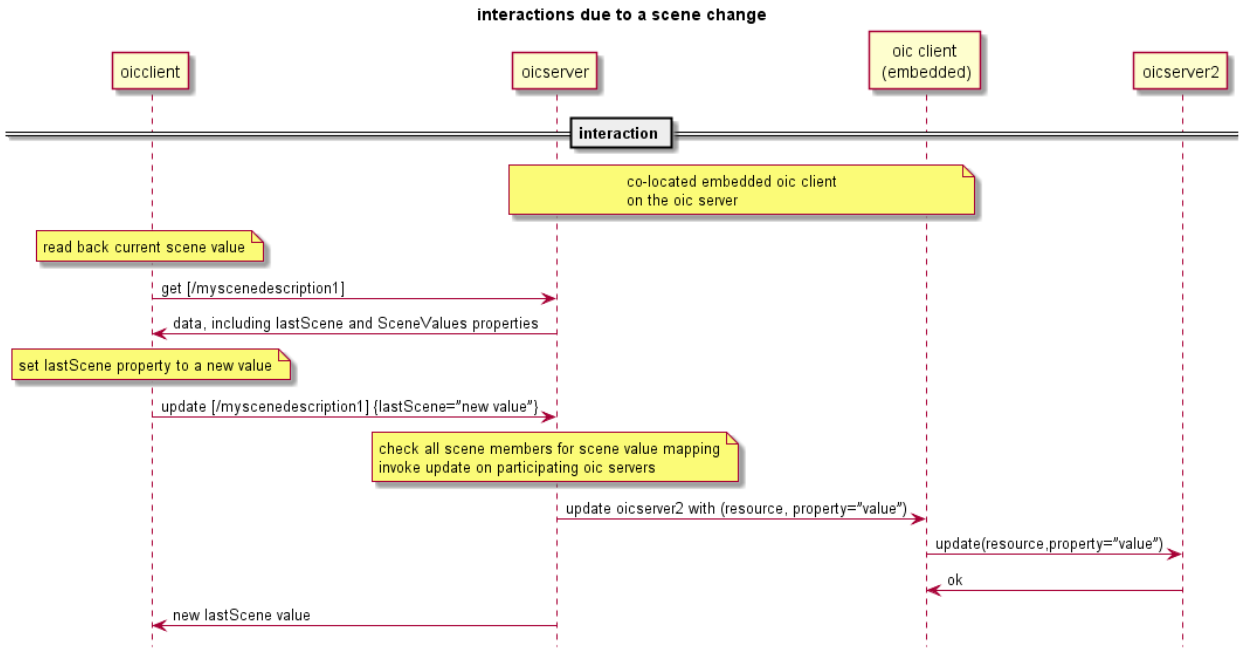


Figure 39 Interaction overview due to a Scene change

11.6.2.4 Deletion of a Scene

When the sceneCollection is not needed anymore the end user can remove the sceneCollection from the OIC Server. As the sceneCollection is a specialization of an OIC Collection as defined in section 7.1.6.3, then it is deleted by using the mechanism defined for deletion of collections in section 7.1.6.3.7. Note that this also deletes all contained sceneMember resources. The deletion is depicted in Figure 40.

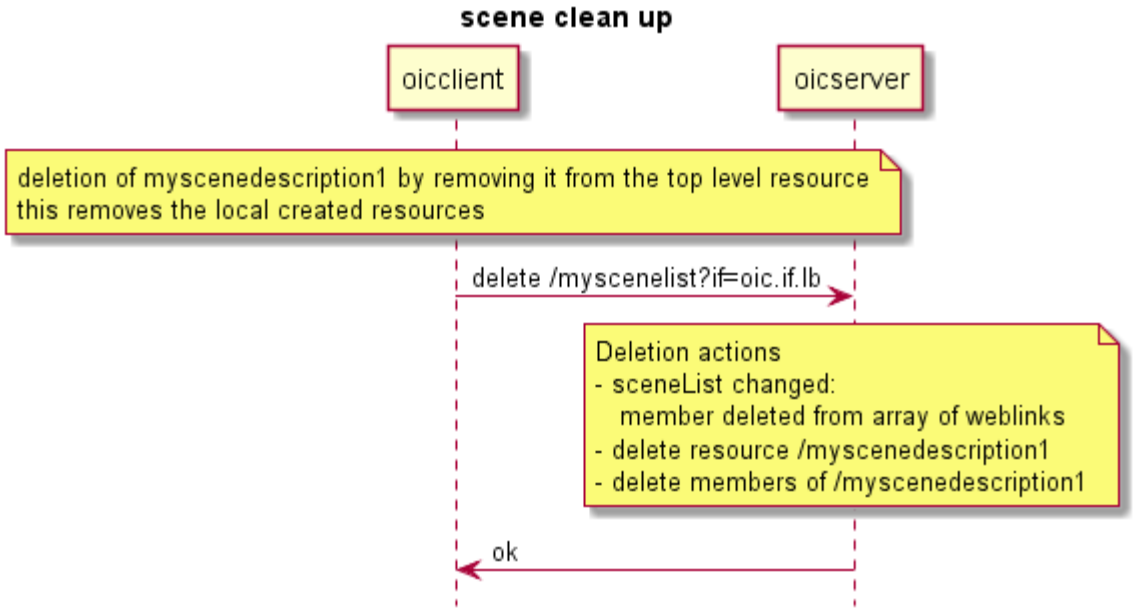


Figure 40 Clean up of Scene resource structure

11.6.2.5 Summary of resource types defined for Scene functionality

Table 29 summarizes the list of resource types that are part of Scenes.

Table 29 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 Rules

11.6.3.1 Introduction

Rules functionality (or conditional scripting) is described in OIC by means of resources. The rule resources are hosted by an OIC Server and the top level resource is listed in /oic/res. This means that an OIC Client can determine if the rule functionality is hosted on an OIC Server via a RETRIEVE on /oic/res or via resource discovery. The setup of rules is done by OIC Client interactions. This includes creating new rule resources and mappings of OIC Server resource properties. A rule resource consist of a rule condition that evaluates and when the evaluation is true, executing the mappings that are in the script part of the resource (in essence the set of Rule Mmembers)

The Rule functionality is created by multiple resources and has the structure depicted in Figure 41. The ruleList and rule resources are overloaded collection resources.

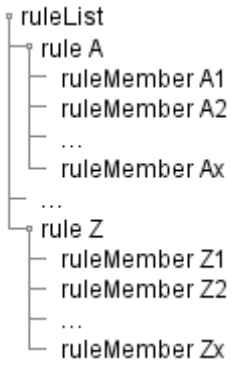


Figure 41 Generic rule resource structure

11.6.3.2 Rule creation

An OIC Client that wants to create rule resources needs to verify that an OIC server supports the rule feature. This can be done by checking if /oic/res contains the rt of the ruleList resource. This is depicted in first steps of Figure 42. The rule resources can be created by an end user using a capable OIC Client. This will entail creating a resource that defines a rule. The script (Rule Members) will be executed when the rule condition evaluates to true. The rule condition is defined by the EBNF defined in 5.6.16.1 of UPnP AV CDS. The property in the EBNF indicates a variable which will be mapped to a property in a resource on an OIC Server. The syntax to indicate the reference is defined as: <deviceurn>:<resourceid>:<resource properties>.

An example of rule condition that references resources is:

(uuid:mybinaryid:value = true) and (uuid:myid:temperature > 30)

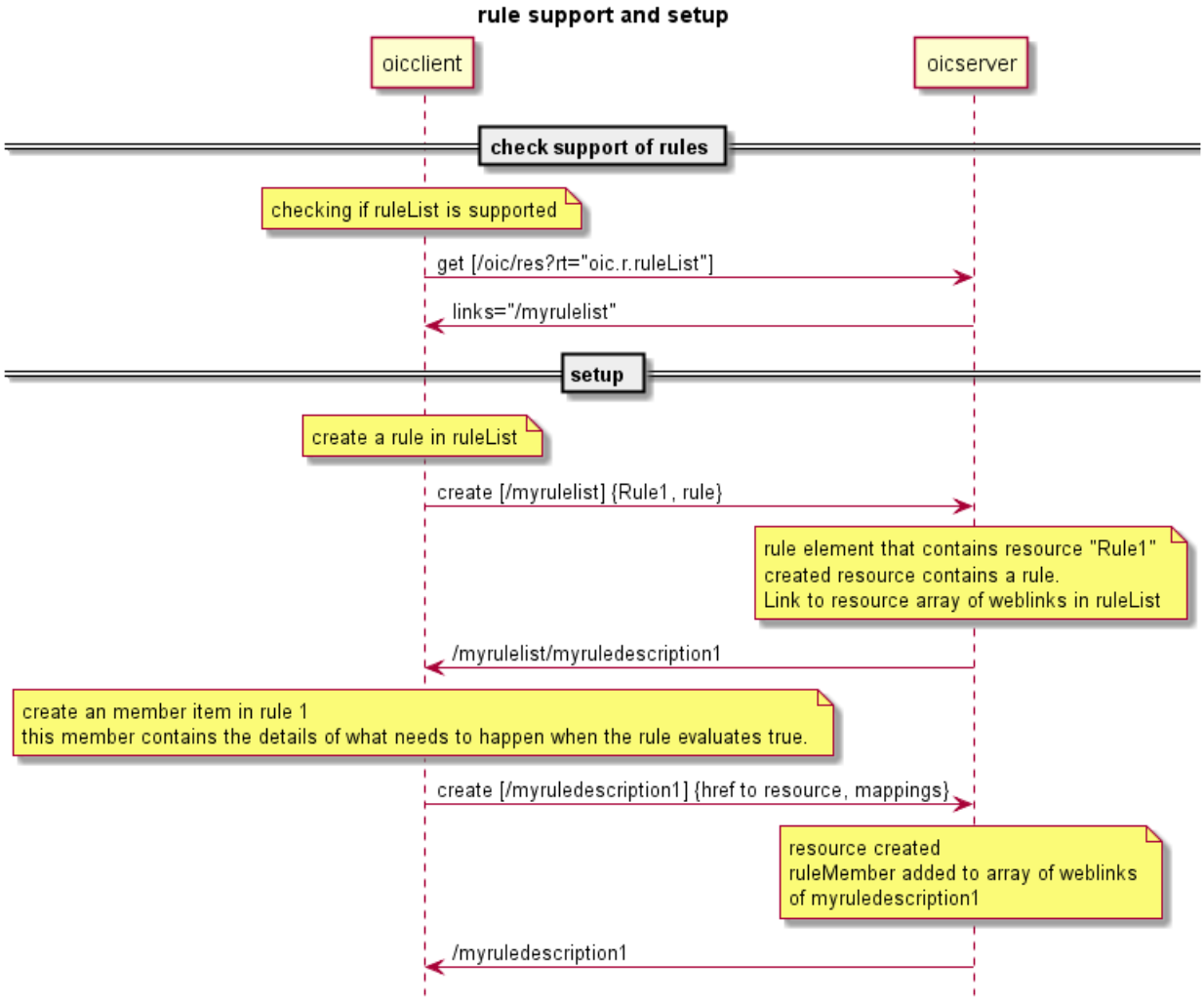


Figure 42 Interactions to check Rule support and setup of specific rules

11.6.3.3 Interacting with Rules

All capable OIC Clients can interact with rules. However there are only two interactions for OIC Clients since the rules are set up as a background process. The two interactions are: enabling/disabling the rule and activating the test mode of a rule.

An OIC Client that wishes to enable a rule shall do so by initiating an UPDATE operation with a payload that sets the currentStatus property of the rule resource (section 8.4) to ‘enabled’. An OIC Client that wishes to disable a rule shall do so by initiating an UPDATE operation with a payload that sets the currentStatus property of the rule resource (section 8.4) to ‘disabled’.

An OIC Client that wishes to run the test mode of a rule shall do so by initiating an UPDATE operation with a payload that sets the test property of the rule resource (section 8.4) to ‘true’. On completion of the test mode the rule on the OIC Server shall reset the test property to ‘false’.

These steps are depicted in Figure 43.

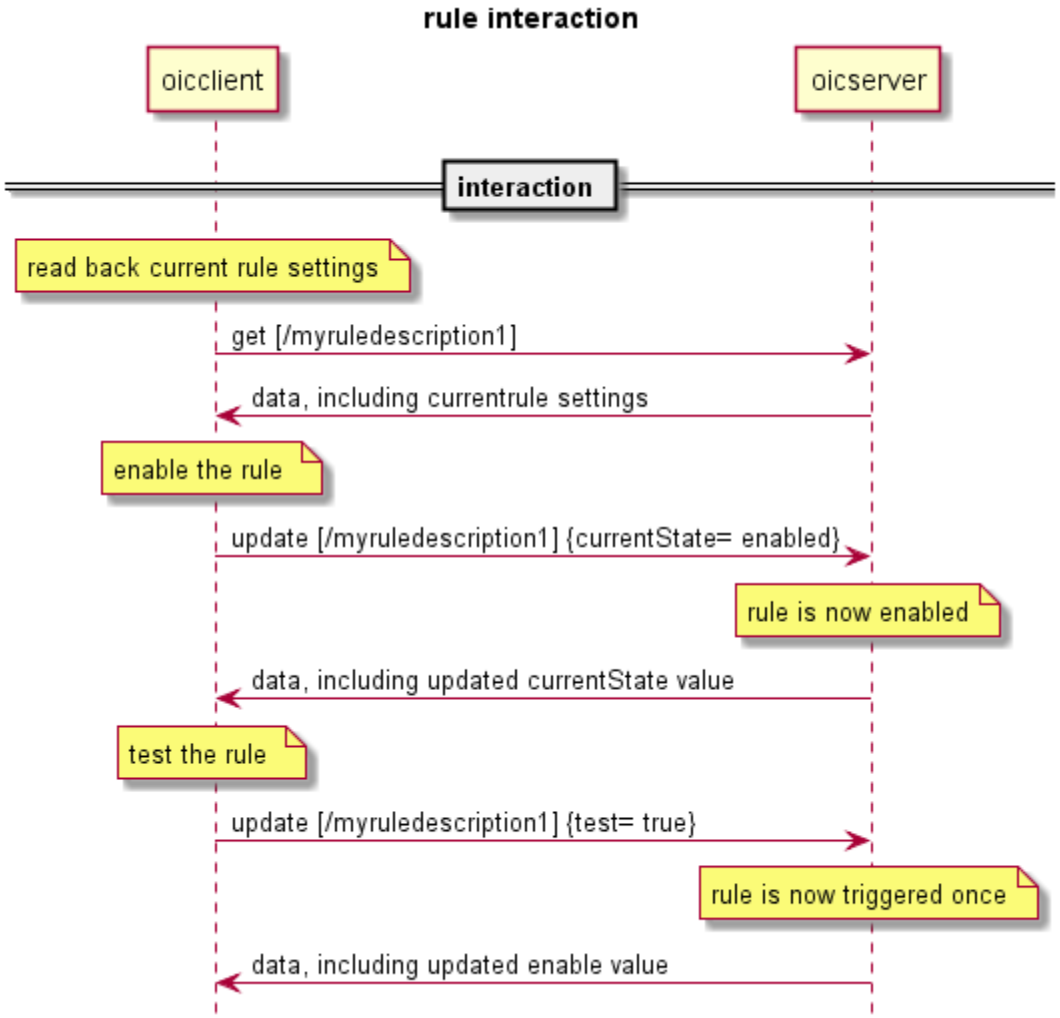


Figure 43 Client interactions on rules

As described a rule condition can reference one or more resources that are present on one or more OIC Servers. The rule condition is re-evaluated each time the status of a resource that is part of the condition changes. This evaluation is triggered by an OIC Client that is either embedded as part of the OIC Server hosting the rule, or separate to the server having knowledge of the rule via a RETRIEVE action, observing the referenced resources using the mechanism described in Section [Ref]. If the rule condition evaluates to true then the Rule Members that are part of the script are executed. This behaviour assuming an embedded client is depicted in Figure 44.

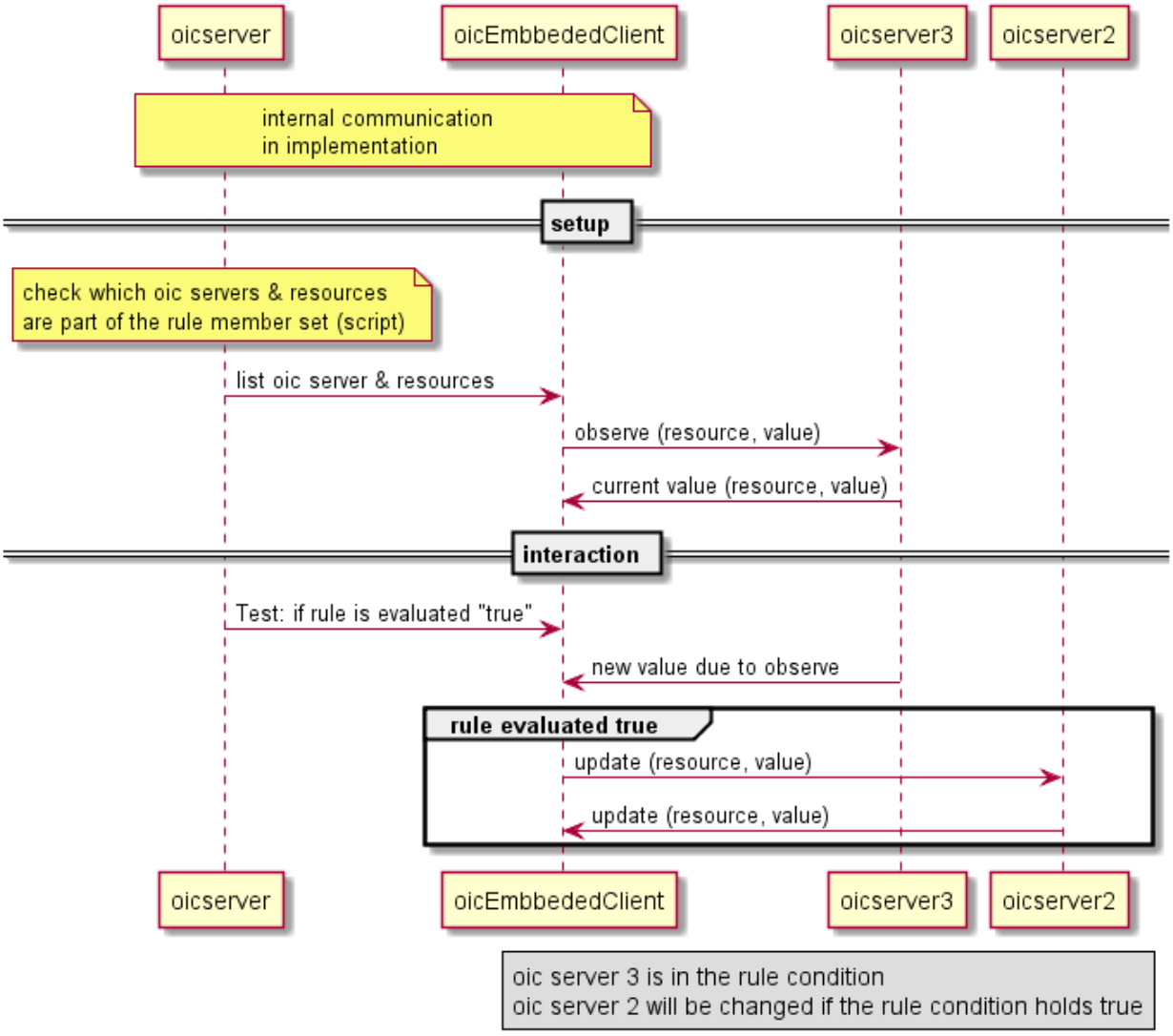


Figure 44 Interaction overview due to a rule condition evaluating to true

11.6.3.4 Deletion of a Rule

When the rule is not needed anymore the end user can remove the rule from the OIC Server. As the rule is a specialization of a Collection as defined in section 7.1.6.3 then it is deleted by using the mechanism defined for deletion of collections in section 7.1.6.3.7. Note that this also deletes all contained ruleMember resources. The deletion is depicted in Figure 45.

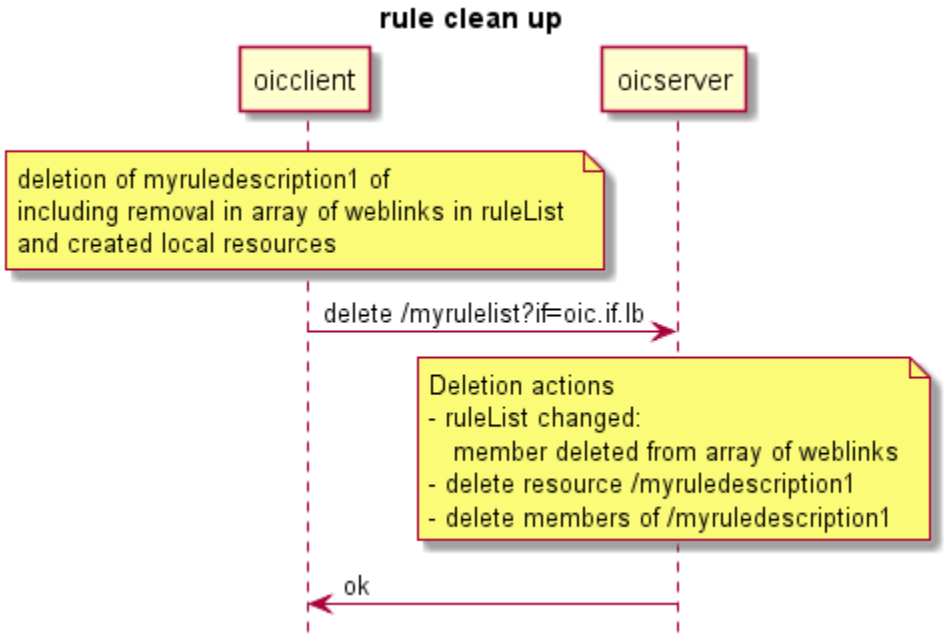


Figure 45 Clean-up of rule resource structure

11.6.3.5 Rules in error state

The resources that are part of a rule condition or are referenced as a Rule Member may, for whatever reason, become unavailable or inaccessible. When it is detected that a resource (either as a result of observe or some other means) becomes unavailable the OIC Client that handles the evaluation of the rule condition (embedded or separate) shall set the currentState of the rule on the OIC Server to 'Error' by issuing an UPDATE operation.

If on execution of test mode of a rule one or more of the UPDATE operations on the Rule Members returns a non-success response code, then the OIC Client that handles the execution of the rule (embedded or separate) shall set the currentState of the rule on the OIC Server to 'Error' by issuing an UPDATE operation.

11.6.3.6 Summary of resource types defined for Rule functionality

Table 30 summarizes the list of resource types that are part of Rules.

Table 30 List of resource types part of Rules

Friendly Name (informative)	Resource Type (rt)	Short Description	Section
ruleList	oic.wk.ruleList	Top Level collection containing rules	
rule	oic.wk.rule	Description of a rule scenario containing 1 condition and a script (set of Rule Members)	
ruleMember	oic.wk.ruleMember	Description specific resources and their values which must be set when the rule evaluates true	

11.6.4 Security considerations

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

Reference [Ref 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 OIC Client to access the rule or scene member resource(s) on the OIC 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, HTTP, etc.). 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, HTTP) 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

An OIC Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in section 12.2.2. An OIC 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 12.2.5..

12.2.2 Request methods

Every request has a CoAP method that qualifies the request. The primary methods and their meanings are shown in Table 31, which presents the way to utilize GET/PUT/POST/DELETE methods for CREATE, RETRIEVE, UPDATE, DELETE operations. The descriptions are generic behaviours using these methods but resource interfaces may modify these generic semantics (the Default Interface reflects the generic semantics).

Table 31. CoAP request methods

CRUDN	Method	General semantics (see individual use for any exceptions)
RETRIEVE	GET	RETRIEVE operation is performed with the GET method as in [RFC7252]. The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request URI. GET is a safe method and is idempotent. GET is also used for introspection of a resource
CREATE **UPDATE**	PUT	PUT shall be used to completely replace or overwrite the entire representation of a resource. The resource representation in the payload

		of the PUT request shall be the complete representation (partial representation with only a few of the properties defined is not part of PUT semantics – see POST). CREATE with PUT uses a new target URI for the new resource to be created. A client sends a PUT request to a server with the target URI for the new resource and the new resource representation in the payload. The server creates the new resource with the target URI provided in the PUT request and sends back a response. UPDATE with PUT i) uses an existing target URI for the resource to be updated and ii) carries the whole replacement as the payload. A client sends a PUT request to a server with the target URI for the resource to be updated and the representation of the total replacement in the payload. The server replaces the existing resource identified by the request URI with the representation enclosed in the PUT request and sends back a response. If the resource previously exists and does not have a resource representation compatible with the PUT representation in the payload (i.e. the "overwrite" semantic cannot be honoured) then an OIC Client error shall be returned. PUT is an unsafe method but it is idempotent which implies that when a PUT request is repeated the outcome shall be the same each time. (PUT should be a supported method on a resource or resource interaction only when idempotent behaviour can be preserved).
CREATE **UPDATE**	POST	POST may be used only in situations where the URI in the target of the request is valid – i.e. is the URI of an existing resource on the server that is processing the request. If the resource is not present, a "Resource not found error" shall be returned. CREATE with POST i) uses an existing target URI for the resource responsible for the creation and ii) the URI of the new resource is determined by the server, then forwarded to the client. A client sends a POST request to a server with the target URI for the resource which is responsible for creating the new resource and the representation of the new resource in the payload. Take notice that the target URI in the request is not for the new resource. The server creates the new resource with a URI, then returns the response with the new URI for the newly created resource to the requesting client. UPDATE with POST i) uses an existing target URI for the resource to be updated and ii) carries a partial modification as the payload. A client sends a POST request to a server with the target URI for the resource to be updated and the representation of the partial update in the payload. The server incorporates the representation enclosed in the POST request into the existing resource identified by the request URI and sends back a response. POST is unsafe and is the supported method when idempotent behaviour cannot be expected or guaranteed.
DELETE	DELETE	DELETE operation is performed with the DELETE method as in [RFC7252]. The DELETE method requests that the resource identified by the request URI be deleted. DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

12.2.3 Content Type negotiation

The OIC 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 OIC Client.

Content types supported are as shown in Table 32.

Table 32. 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 OIC vertical can mandate a specific content type.

OIC Server and OIC 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 32.

12.2.4 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.

12.2.5 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-187 shall be used by all OIC 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-187 shall be supported by all OIC 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).

An OIC Client may support both the block1 (as descriptive) and block2 (as control) options as described by IETF draft-ietf-core-block-187. An OIC Server may support both the block1 (as control) and block2 (as descriptive) options as described by IETF draft-ietf-core-block-187.

12.2.6 CoAP serialization over TCP

12.2.6.1 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 an OIC Client and the OIC Server is located at the user's home. The OIC 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. An OIC Device implementing CoAP Serialization over TCP shall conform to IETF draft-tschofenig-core-coap-tcp-tls-04.

12.2.6.2 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, an OIC 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.

12.2.6.3 Message type and header

The message type transported between OIC Client and OIC 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 5 in IETF draft-tschofenig-core-coap-tcp-tls-04 shall be used for messages transmitted between an OIC Client and an OIC Server. An OIC Device shall use "Alternative L3" as defined in IETF draft-tschofenig-core-coap-tcp-tls-04.

12.2.6.4 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.

12.2.6.5 KeepAlive

12.2.6.5.1 Overview

In order to ensure that the connection between an OIC Device is maintained, when using CoAP serialization over TCP, an OIC 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)

12.2.6.5.2 KeepAlive Mechanism

OIC Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. An OIC Server shall support a resource of type oic.wk.ping as defined in Table 33.

Table 33. Ping resource

Fixed 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 an OIC Client keeps its Connection with an OIC Server active. The resource properties exposed by /oic/ping are listed in Table 34.	KeepAlive

Table 34 defines oic.wk.ping resource type.

Table 34. 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 an OIC Client and OIC Server:

1) An OIC Client which wants to keep the connection with an OIC Server alive shall send a PUT request to /oic/ping resource on the OIC 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 OIC Server receiving this ping request shall respond within 1 minute.
3) If an OIC Client does not receive the response within 1 minute, it shall terminate the connection.
4) If an OIC Server does not receive a PUT request to ping resource within the specified "interval" time, the OIC Server shall terminate the connection.

An example of the KeepAlive mechanism is as follows:

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

12.3 Mapping of CRUDN to HTTP

12.3.1 Supported HTTP features

This section defines the HTTP transport between an OIC Client and an OIC Server. The implementation of HTTP on an OIC Client and OIC Device shall conform to IETF RFC 2616]. The optional HTTP/1.1 features required by OIC are listed in this section.

12.3.2 Supported HTTP methods

HTTP/1.1 optional methods PUT, POST and DELETE methods should be supported by an OIC Client and OIC Server.

- The OIC Server shall respond with a 405 (method Not Allowed) status code when other than HEAD, GET, PUT, POST and DELETE methods are used.

12.3.3 Supported HTTP header fields

HTTP header fields (section IETF RFC 2616) are components of the message header of request and response messages. These HTTP headers define the operating parameters of an HTTP

transaction. All mandated HTTP headers in IETF RFC 2616 shall be supported by the OIC Client and OIC Server. Table 35 contains the OIC supported list of HTTP headers to request a response for particular body payload contents. Table 35 contains the OIC supported list of HTTP headers to signal the body content in a request and response method.

The discouraged in Table 35 means that however the header is allowed it is recommended not to use this field to conserve code space and or bandwidth.

Table 35. HTTP header fields usage in OIC

Header Name	Used in message type	RFC Allowed Mandatory	OIC usage	Allowed values
Accept	Request	Allowed	Mandatory (if more than one allowed value is specified)	application/cbor application/json text/xml
Accept-Charset	Request	Allowed	Mandatory	UTF-8
Accept-Encoding	Request	Allowed	Allowed	chunked
Accept-Language	Request	Allowed	Discouraged	
Accept-Ranges	Request Response	Allowed	Discouraged	
Age	Response	Allowed (required for cache)	Discouraged	
Allow	Response	Mandatory (in an error response)	Mandatory(in an error response)	
Authorization	Request Response	Allowed	Allowed	
Cache-Control	Request Response	Allowed	Discouraged	
Connection	Request	Conditional Mandatory	Conditional Mandatory	
Content-Encoding	Response	Conditional Mandatory	Mandatory	

Content-Language	Response	Allowed	Discouraged	
Content-Length	Request Response	Conditional Mandatory	Conditional Mandatory	
Content-Location	Response	Allowed	Discouraged	
Content-MD5	Response	Allowed	Discouraged	
Content-Range	Response	Allowed	Allowed	
Content-Type	Request Response	Mandatory	Mandatory	application/cbor application/json text/xml
Date	Request Response	Mandatory	Mandatory	
Etag	Response	Allowed	Allowed	
Expect	Request	Allowed	Discouraged	
Expires	Response	Allowed	Discouraged	
From	Request	Allowed	Discouraged	
Host	Request	Mandatory	Mandatory	
If-Match	Request	Allowed	Discouraged	
If-Modified-Since	Request	Allowed	Discouraged	
If-None_Match	Request	Allowed	Discouraged	
If-Range	Request	Allowed	Discouraged	
If-Unmodified-Since	Request	Allowed	Discouraged	
Last-Modified	Response	Allowed	Discouraged	
Location	Response	Allowed	Allowed	

Max-Forwards	Request	Allowed	Discouraged	
Pragma	Request Response	Allowed	Discouraged	
Proxy-Authenticate	Response	Allowed		
Proxy-Authorization	Request	Allowed		
Range	Request	Allowed	Allowed	
Referrer	Request	Allowed	Discouraged	
Retry-After	Response	Allowed	Allowed	
Server	Response	Allowed	Discouraged	
TE	Request	Allowed but discouraged	Discouraged	
Trailer	Response	Allowed but discouraged	Discouraged	
Transfer-Encoding	Response	Allowed	Allowed	
Upgrade	Request	Allowed	Allowed	
User-Agent	Request	Allowed	Discouraged	
Vary	Response	Allowed but discouraged	Discouraged	
Via	Request Response	Allowed	Discouraged	
Warning	Request Response	Allowed but discouraged	Discouraged	
WWW-Authenticate	Response	Allowed	Discouraged	

12.3.4 Content Type negotiation

The OIC 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-encoding header shall be used to indicate which content encodings are requested by the OIC Client.

Note that an OIC vertical can mandate a specific content type.

12.3.5 HTTP response codes

OIC Devices with HTTP capabilities shall support a subset of HTTP response codes as defined in HTTP/1.1 specification IETF RFC 2616 and listed in Table 36.

Table 36. HTTP response codes

HTTP Response Code	Description
200 (“OK”)	This response code is sent to indicate a successful transaction. This response code is often used in response to a successful GET request, with the entity-body containing a resource representation of the requested resource or object instance. Use of this response code in response to PUT, POST, or DELETE requests is discouraged, to avoid the potentially unnecessary traffic generated by returning the resource representation in the entity-body.
201 (“Created”)	This response code is sent to indicate a new object or resource has been created, at the client’s request with a PUT or POST. The Location header is used in conjunction with this response code to indicate the URI of the newly created resource. The inclusion of a resource representation of the newly created resource in the entity-body of the response is discouraged, to conserve bandwidth.
204 (“No Content”)	This response code is sent to indicate a successful transaction, but one where the response does not include an entity-body. This response code is often used in response to a successful POST request, where the resource is modified, not created. This response code is also sent in response to a successful DELETE request. This response code is also sent in response to a successful GET request, where the resource exists but has an empty representation.
400 (“Bad Request”)	This response code is used to indicate a client-side error and is used when no other 4xx response code is appropriate. Often, this response code indicates that the resource representation sent by a client with a PUT or POST is not appropriate or is malformed.
4xx	All other HTTP/1.1 defined responses in the 400 range.
500 (“Internal Server Error”)	This response code is used to indicate that the server has an internal problem and is a generic response.
5xx	All other HTTP/1.1 defined responses in the 500 range.

12.3.6 Method mapping

The resource manipulation methods defined in this specification (CRUDN) can be matched with HTTP’s GET, POST, PUT and DELETE methods. The HTTP request methods shall be as defined in 12.2.2.

13 Security

The details for handling security and privacy are specified in [OIC 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 OIC all use CoAP and IP/ UDP, [oneM2M-TS0008], [IPSO], [OIC] 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 [OIC], [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 OIC model and vice versa.

14.1.2 Different resource models

OIC 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 OIC, 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 OIC 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.

14.2.2 New Content-Formats (Internet Media Type) for OIC resource model

OIC is currently developing resource model for IoT services (e.g. smart home) and strives to align with other standard bodies such as oneM2M or IPSO, so that OIC devices can interoperate with the devices following different standards. For this purpose, OIC prepares a scheme to exchange resource model information as describes in the previous sections.

The resource mode indication scheme requires new Internet media types and CoAP Content-Format identifiers which can indicate OIC resource models with content encoding. Hence OIC requests from IANA the new Internet media types as below.

- **application/oic:** indicates the payload follows OIC resource model and used in CoAP Accept Option or Content-Format Option.
- **application/oic+json:** indicates the payload follows OIC resource model in json encoding and used in CoAP Accept Option or Content-Format Option.
- **application/oic+cbor:** indicates the payload follows OIC resource model in cbor coding and used in CoAP Accept Option or Content-Format Option.

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 an OIC Server and “Smartphone” is an OIC Client. In one of the scenario “Garage” additionally acts as an OIC Server. All the examples are based on the following example resource definitions:

rt=oic.example.light with resource type definition as illustration in Table 37.

Table 37. 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 38.

Table 38. 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/mon (rt=oc.wk.mon) and

/oic/mnt (rt=oc.wk.mnt) used in below examples are defined in section 7 (Resource model).

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

This sequence highlights (Figure 46) the discovery and control of an OIC light resource from an OIC smartphone.

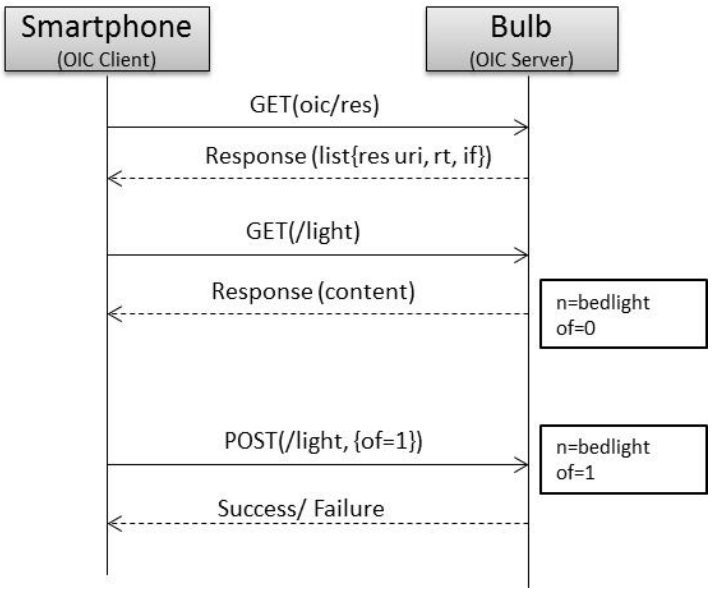


Figure 46. When at home: from smartphone turn on a single light

Discovery request can be sent to CoAP Multicast address 224.0.1.187 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 12.2.4.

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 47) the device management functions of monitoring and maintenance.

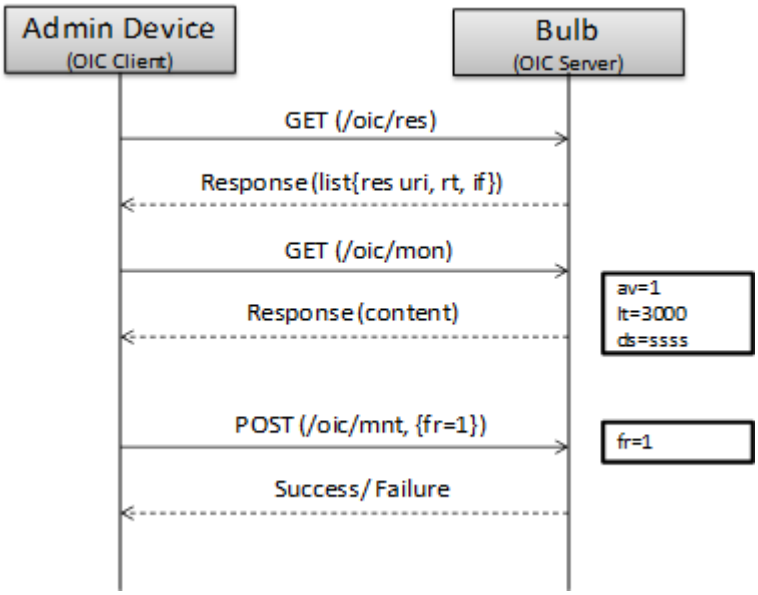


Figure 47. Device management (monitoring and maintenance)

Pre-Condition: Admin device has different security permissions and hence can perform device management operations on the OIC 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 resource is '/oic/mon' whose rt=oc.wk.mon and another resource is '/oic/mnt' whose rt=oc.wk.mnt)
3) Admin device sends a GET request to '/oic/mon' resource to know its current monitoring state
4) The end point responds with representation of monitoring resource ({av=1;lt=3000;ds=sss…})
5) After seeing the device statistics, 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)
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 12.2.4.

Annex B
(informative)

OIC interaction scenarios and deployment models

B.1 OIC interaction scenarios

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

- Direct interaction between OIC Client and OIC Server (Figure 48). In this scenario the OIC Client and the OIC Server directly communicate without involvement of any other OIC Device. A smartphone which controls an actuator directly uses this scenario.

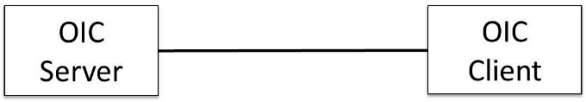


Figure 48. Direct interaction between OIC Server and OIC Client

- Interaction between OIC Client and OIC Server using another OIC server (Figure 49). In this scenario, another OIC Server provides the support needed for the OIC Client to directly access the desired resource on a specific OIC 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.

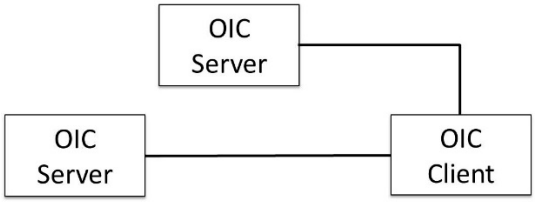


Figure 49. Interaction between OIC Client and OIC Server using another OIC Server

- Interaction between OIC Client and OIC Server using OIC Intermediary (Figure 50). In this scenario an OIC Intermediary facilitates the interaction between the OIC Client and the OIC Server. A smartphone which controls appliances in a smart home via MQTT broker uses this scenario.

Figure 50. Interaction between OIC Client and OIC Server using OIC Intermediary

- Interaction between OIC Client and OIC server using support from multiple OIC Servers and OIC intermediary (Figure 51). In this scenario, both OIC Server and OIC Intermediary roles are present to facilitate the transaction between the OIC Client and a specific OIC 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.

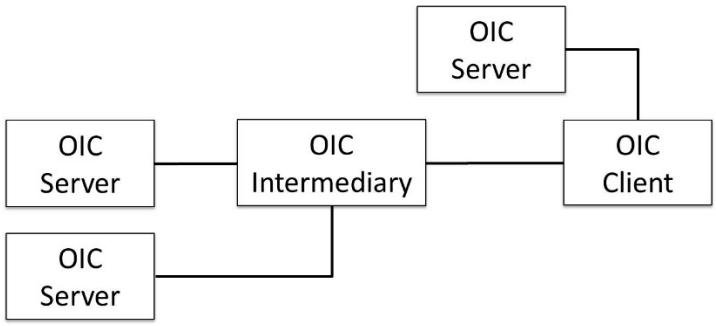


Figure 51. Interaction between OIC Client and OIC Server using support from multiple OIC Servers and OIC Intermediary

B.2 OIC deployment model

In OIC Deployment, OIC Devices are deployed and interact via either wired or wireless connections. OIC Devices are the physical entities that may host resources and play one or more OIC Roles. There is no constraint on the structure of a deployment or number of OIC Devices in it. OIC 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].

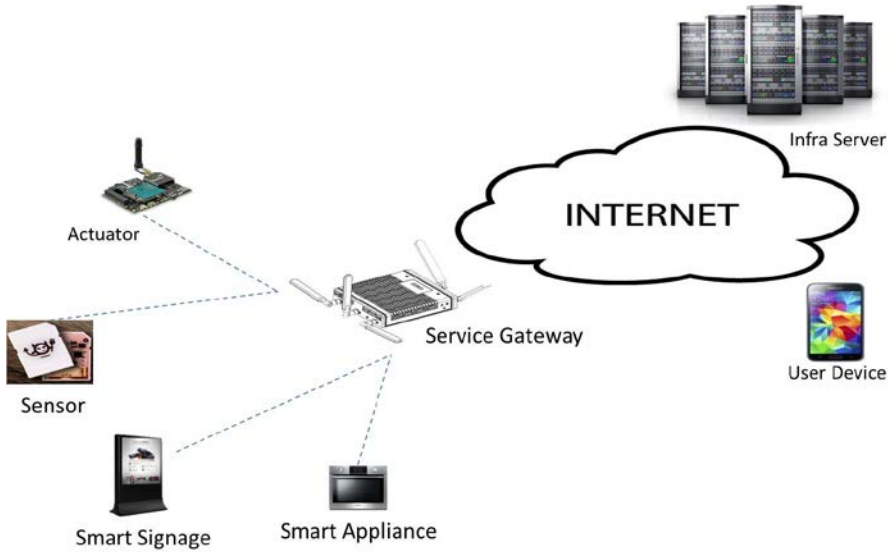


Figure 52. Example of OIC Devices

Figure 52 depicts a typical OIC deployment and set of OIC 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 OIC Sever but they may also take the role of OIC 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 OIC Client, but may also take the role of OIC Server or OIC Intermediary.

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

Annex C
(informative)

Other Resource Models and OIC Mapping

C.1 Multiple resource models

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

There are multiple resource models defined by different organizations including OIC, IPSO Alliance and oneM2M, and used in the industry, which may restrict interoperability among respective ecosystems. The main differences from OIC 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 OIC approach for support of multiple resource models

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

The following OIC characteristics enable support of other resource models:

- **OIC resource model is the superset of multiple models**: the OIC 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 OIC resource model concepts.
- **OIC Framework may allow for resource model negotiation**: the OIC Client and Server exchange the information about what resource model(s) each supports. Based on the exchanged information, the OIC 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 OIC client and server exchange the information about what resource model(s) each supports. Based on the exchanged information, the OIC 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 OIC 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 OIC 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 OIC 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 OIC Client and Server. In case there is no common resource model, the OIC 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 OIC implementation can accommodate multiple profiles and hence multiple ecosystems.

- **Resource Model Profile**: the OIC Framework defines resource model profiles and implementers or users choose the active profile. The chosen profile constraints the OIC 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 an OIC 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 OIC to represent these terms. This section provides the equivalent OIC terms and then frames the IPSO smart object in OIC 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 OIC Resource Type definition which defines the relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent to an OIC 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 OIC Interface).

The general method for developing the equivalent OIC Resource Type from an IPSO Smart Object definition is to ignore the Object ID and replace the Object URN with and OIC '.' (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 OIC to OIC interoperability and so the OIC format is preferred – for OIC to OIC interoperability only the data model consistency is required.

Two models are available to render IPSO objects into OIC.

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

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

OIC 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 39 and Table 40 define the resource type and its properties, respectively.

Table 39. Light control resource type definition

OIC 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 40. 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

3269

3270

Annex D
(informative)

Resource type definitions

D.1 List of resource type definitions

D.2 OIC Configuration

D.2.1 Introduction

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

D.2.2 Wellknown URI

/oic/con

D.2.3 Resource Type

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

D.2.4 RAML Definition

```
#%RAML 0.8
title: OIC Configuration
version: v1-20150807
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.rw"]

/oic/con:
  description: |
    Known resource that is hosted by every OIC 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": "http://openinterconnect.org/oic.wk.con#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "definitions": {
                  "oic.wk.con": {
                    "type": "object",
                    "properties": {
                      "n": {
```



```
                        "type": "string",
                        "description": "Human friendly name"
                      },
                      "loc": {
                        "type": "string",
                        "description": "Location information",
                        "format": "json"
                      },
                      "locn": {
                        "type": "string",
                        "description": "Human Friendly Name"
                      },
                      "c": {
                        "type": "string",
                        "description": "Currency"
                      },
                      "r": {
                        "type": "string",
                        "description": "Region"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "oic.core.json#/definitions/oic.core"},
                  { "$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 OIC device

    body:
      application/json

        schema: |

          {
            "id": "http://openinterconnect.org/oic.wk.con#",
            "$schema": "http://json-schema.org/draft-04/schema#",
            "definitions": {
              "oic.wk.con": {
                "type": "object",
                "properties": {
                  "n": {
                    "type": "string",
                    "description": "Human friendly name"
                  },
                  "loc": {
                    "type": "string",
                    "description": "Location information",
                    "format": "json"
                  },
                  "locn": {
                    "type": "string",
                    "description": "Human Friendly Name"
```



```
                    },
                    "c": {
                      "type": "string",
                      "description": "Currency"
                    },
                    "r": {
                      "type": "string",
                      "description": "Region"
                    }
                  }
                }
              },
              "type": "object",
              "allOf": [
                {"$ref": "oic.core.json#/definitions/oic.core"},
                { "$ref": "#/definitions/oic.wk.con" }
              ],
              "required": [ "n" ]
            }

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

      responses:
        200:
          body:
            application/json:
              schema: |
                {
                  "id": "http://openinterconnect.org/oic.wk.con#",
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "definitions": {
                    "oic.wk.con": {
                      "type": "object",
                      "properties": {
                        "n": {
                          "type": "string",
                          "description": "Human friendly name"
                        },
                        "loc": {
                          "type": "string",
                          "description": "Location information",
                          "format": "json"
                        },
                        "locn": {
                          "type": "string",
                          "description": "Human Friendly Name"
                        },
                        "c": {
                          "type": "string",
                          "description": "Currency"
                        },
                        "r": {
                          "type": "string",
                          "description": "Region"
                        }
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    {"$ref": "oic.core.json#/definitions/oic.core"},
                    { "$ref": "#/definitions/oic.wk.con" }
                  ],
                  "required": [ "n" ]
```



```
            }

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

D.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description

D.2.6 CRUDN behavior

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

D.3 OIC Logical Device

D.3.1 Introduction

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

D.3.2 Wellknown URI

/oic/d

D.3.3 Resource Type

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

D.3.4 RAML Definition

```
#%RAML 0.8
title: OIC Root Device
version: v1-20150811
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.r"]


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

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.d#",
                "definitions": {
```



```
                "oic.wk.d": {
                  "type": "object",
                  "properties": {
                    "n": {
                      "type": "string",
                      "description": "Readonly, Human friendly name"
                    },
                    "di": {
                      "type": "string",
                      "description": "ReadOnly, Unique identifier for device (UUID)",
                      "format": "uuid"
                    },
                    "icv": {
                      "type": "string",
                      "description": "ReadOnly, The version of the OIC Server"
                    },
                    "dmv": {
                      "type": "string",
                      "description": "ReadOnly, The spec version of the vertical specification",
                      "format": "csv"
                    }
                  }
                }
              },
              "type": "object",
              "allOf": [
                { "$ref": "#/definitions/oic.wk.d" }
              ],
              "required": [ "n", "di", "icv" ]
            }

          example: |
            {
              "n":    "Device 1",
              "rt":   "oic.wk.d",
              "di":   "54919CA5-4101-4AE4-595B-353C51AA983C",
              "icv":  "OIC 1.0"
            }
```

D.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string	yes	Read Write	Readonly, Human friendly name
di	string	yes	Read Only	Unique Identifier For Device (Uuid)
icv	string	yes	Read Only	The Version Of The Oic Server
dmv	string		Read Only	The Spec Version Of The Vertical Specification

D.3.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/d		get			

D.4 OIC Inteface Types

D.4.1 Introduction

List of resource interfaces that are supported by this OIC Server

D.4.2 Wellknown URI

/oic/ifs

D.4.3 Resource Type

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

D.4.4 RAML Definition

```
#%RAML 0.8

title: OIC Interface Types
version: v1-20150811

traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.r"]


/oic/ifs:
  description: |
    List of resource interfaces that are supported by this OIC Server

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.ifs#",
                "definitions": {
                  "oic.wk.ifs": {
                    "type": "object",
                    "properties": {
                      "il": {
                        "type": "string",
                        "description": "Readonly, list of interface names",
                        "format": "bsv"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  { "$ref": "#/definitions/oic.wk.ifs" }
                ],
                "required": ["il"]
              }

            example: |
              {
                "rt": "oic.wk.ifs",
                "il": "oic.if.ll oic.if.bat oic.if.r"
              }
```

D.4.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
il	string	yes	Read Write	Readonly, list of interface names

D.4.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify

/oic/ifs		get			

D.5 OIC Maintenance

D.5.1 Introduction

The resource through which an OIC 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 ssc (Start Stat Collection) is a boolean. The value 0 means No collection of statistics, the value 1 means Starts collecting statistics

D.5.2 Wellknown 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: OIC Maintenance
version: v1-20150811
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.r"]

/oic/mnt:
  description: |
    The resource through which an OIC 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
    ssc (Start Stat Collection) is a boolean.
      The value 0 means No collection of statistics, the value 1 means Starts collecting statistics

  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#",
                "id": "http://openinterconnect.org/oic.wk.mnt#",
                "definitions": {
```



```
                    "oic.wk.mnt": {
                      "type": "object",
                        "fr":{
                          "type": "boolean",
                          "description": "Factory Reset"
                        },
                        "rb": {
                          "type": "boolean",
                          "description": "Reboot Action"
                        },
                        "ssc": {
                          "type": "boolean",
                          "description": "Start Stat Collection Action Toggle"
                        }
                      }
                  },
                  "type": "object",
                  "allOf": [
                    {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},
                    {"$ref": "#/definitions/oic.wk.mnt" }
                  ],
                  "required": ["fr", "rb", "ssc"]
                }

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

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

    queryParameters:
      if:
        enum: oic.if.rw
    body:
      application/json
        schema: |
          {
            "$schema": "http://json-schemas.org/draft-04/schema#",
            "id": "http://openinterconnect.org/oic.wk.mnt#",
            "definitions": {
              "oic.wk.mnt": {
                "type": "object",
                  "fr":{
                    "type": "boolean",
                    "description": "Factory Reset"
                  },
                  "rb": {
                    "type": "boolean",
                    "description": "Reboot Action"
                  },
                  "ssc": {
                    "type": "boolean",
                    "description": "Start Stat Collection Action Toggle"
                  }
                }
            },
            "type": "object",
            "allOf": [
              {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},
```



```
              {"$ref": "#/definitions/oic.wk.mnt" }
            ],
            "required": ["fr", "rb", "ssc"]
          }

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.mnt#",
                "definitions": {
                  "oic.wk.mnt": {
                    "type": "object",
                      "fr":{
                        "type": "boolean",
                        "description": "Factory Reset"
                      },
                      "rb": {
                        "type": "boolean",
                        "description": "Reboot Action"
                      },
                      "ssc": {
                        "type": "boolean",
                        "description": "Start Stat Collection Action Toggle"
                      }
                    }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.wk.mnt" }
                ],
                "required": ["fr", "rb", "ssc"]
              }

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

D.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
fr	boolean	yes	Read Write	Factory Reset
rb	boolean	yes	Read Write	Reboot Action
ssc	boolean	yes	Read Write	Start Stat Collection Action Toggle

D.5.6 CRUDN behavior

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

D.6 OIC Monitoring

D.6.1 Introduction

The resource through which an OIC Device is monitored.

D.6.2 Wellknown URI

/oic/mon

D.6.3 Resource Type

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

D.6.4 RAML Definition

```
#%RAML 0.8
title: OIC Monitoring
version: v1-20150401
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.r"]


/oic/mon:
  description: |
    The resource through which an OIC Device is monitored.

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.mon#",
                "definitions": {
                  "oic.wk.mon": {
                    "type": "object",
                    "properties": {
                      "av":{
                        "type": "boolean",
                        "description": "ReadOnly, Indicates if the device is available or not on
the network (like ping)"
                      },
                      "lat": {
                        "type": "integer",
                        "description": "ReadOnly, Indicates the elapsed time in seconds after the
device was invoked or acted upon"
                      },
                      "ds": {
                        "type": "string",
```



```
                        "description": "ReadOnly, Contains Device Statistics Info (no. of received
packets, no. of sent packets, time to respond etc.)",
                        "format": "csv"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "oic.core.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.wk.mon"}
                ],
                "required": ["av", "lat"]
              }

            example: |
              {
                "rt":   "oic.wk.mon",
                "name": "My Monitor Information",
                "av":   true,
                "lat":  50,
                "ds":   "1500, 2750, 0"
              }
```

D.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
av	boolean	yes	Read Only	Indicates If The Device Is Available Or Not On The Network (Like Ping)
lat	integer	yes	Read Only	Indicates The Elapsed Time In Seconds After The Device Was Invoked Or Acted Upon
ds	string		Read Only	Contains Device Statistics Info (No. Of Received Packets, No. Of Sent Packets, Time To Respond Etc.)

D.6.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/mon		get			

D.7 OIC Base Platform

D.7.1 Introduction

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

D.7.2 Wellknown URI

/oic/p

D.7.3 Resource Type

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

D.7.4 RAML Definition

```
#%RAML 0.8
title: OIC Base Platform
version: v1-20150804
traits:
 - interface
     queryParameters:
```



```
        if:
          enum: ["oic.if.r"]

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

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.p#",
                "definitions": {
                  "oic.wk.p": {
                    "type": "object",
                    "properties": {
                      "pi": {
                        "type": "string",
                        "description": "ReadOnly, Platform Identifier"
                      },
                      "mnmn": {
                        "type": "string",
                        "description": "ReadOnly, Manufacturer Name",
                        "maxLength": 16
                      },
                      "mnml": {
                        "type": "string",
                        "description": "ReadOnly, Manufacturer's URL",
                        "maxLength": 32,
                        "format": "uri"
                      },
                      "mnmo": {
                        "type": "string",
                        "description": "ReadOnly, Model number as designated by manufacturer"
                      },
                      "mndt": {
                        "type": "string",
                        "description": "ReadOnly, Manufacturing Date",
                        "format": "date-time"
                      },
                      "mnpv": {
                        "type": "string",
                        "description": "ReadOnly, Platform Version"
                      },
                      "mnos": {
                        "type": "string",
                        "description": "Readonly, Platform Resident OS Version"
                      },
                      "mnhw": {
                        "type": "string",
                        "description": "Readonly, Platform Hardware Version"
                      },
                      "mnfv": {
                        "type": "string",
                        "description": "ReadOnly, Manufacturer's firmware version"
                      },
```



```
                    "mnsl": {
                      "type": "string",
                      "description": "ReadOnly, Manufacturer's Support Information URL",
                      "format": "uri"
                    },
                    "st": {
                      "type": "string",
                      "description": "ReadOnly, Reference time for the device",
                      "format": "date-time"
                    }
                  }
                }
              },
              "type": "object",
              "allOf": [
                { "$ref": "#/definitions/oic.wk.p" }
              ],
              "required": [ "pi", "mnmn" ]
            }

          example: |
            {
              "pi":   "my-platform-identfier",
              "rt":   "oic.wk.p",
              "mnmn": "Acme, Inc"
            }
```

D.7.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
pi	string	yes	Read Only	Platform Identifier
mnmn	string	yes	Read Only	Manufacturer Name
maxLength				
mnml	string		Read Only	Manufacturer'S Url
maxLength				
mnmo	string		Read Only	Model Number As Designated By Manufacturer
mndt	string		Read Only	Manufacturing Date
mnpv	string		Read Only	Platform Version
mnos	string		Read Write	Readonly, Platform Resident OS Version
mnhw	string		Read Write	Readonly, Platform Hardware Version
mnfv	string		Read Only	Manufacturer'S Firmware Version
mnsl	string		Read Only	Manufacturer'S Support Information Url
st	string		Read Only	Reference Time For The Device

D.7.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/p		get			

D.8 OIC Ping

D.8.1 Introduction

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

D.8.2 Wellknown URI

/oic/ping

D.8.3 Resource Type

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

D.8.4 RAML Definition

```
#%RAML 0.8
title: OIC Ping
version: v1-20150814
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.rw"]

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

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.ping#",
                "definitions": {
                  "oic.wk.ping": {
                    "type": "object",
                    "properties": {
                      "in": {
                        "type": "integer",
                        "description": "ReadWrite, Indicates the interval for which connection
shall be kept alive"
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    {
                      "$ref": "oic.core.json#/definitions/oic.core"
                    },
                    {
                      "$ref": "#/definitions/oic.wk.ping"
                    }
                  ],
                  "required": [
                    "in"
                  ]
                }

            example: |
              {
                "rt":   "oic.wk.ping",
                "name": "Ping Information",
```



```
            "in":   16
          }
```

D.8.5 Property Definition

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

D.8.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/ping		get			

D.9 OIC Discoverable Resources

D.9.1 Introduction

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

D.9.2 Wellknown URI

/oic/res

D.9.3 Resource Type

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

D.9.4 RAML Definition

```
#%RAML 0.8
title: OIC Discoverable Resources
version: v1-20150807
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.ll"]


/oic/res:
  description: |
    The resource through which the corresponding OIC 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#",
                "id": "http://openinterconnect.org/schemas/oic.wk.res.json/",
                "definitions": {
```



```
                    "oic.res-links.json": {
                      "type": "object",
                      "properties": {
                        "n": {
                          "type": "string",
                          "description": "ReadOnly, Human friendly name"
                        },
                        "di": {
                          "description": "The device identifier as indicated by the /oic/d resource
of the device",
                          "type": "string",
                          "format": "UUID"
                        },
                        "mpro": {
                          "description": "ReadOnly, Supported messaging protocols",
                          "type": "string"
                        },
                        "links": {
                          "type": "array",
                          "items": {
                            "$ref": "oic.web-link.json#"
                          }
                        }
                      }
                    }
                  },
                  "description": "The list of resources expressed as web links",
                  "type": "array",
                  "items": {
                    "$ref": "#/definitions/oic.res-links.json"
                  },
                  "required": ["di", "links"]
                }

              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.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Only	Human Friendly Name
di		yes		
mpro				
links	array	yes		
items				
items				

D.9.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify

/oic/res		get			

D.10 OIC Resource Types

D.10.1 Introduction

List of resource types that are supported by this OIC Server

D.10.2 Wellknown URI

/oic/rts

D.10.3 Resource Type

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

D.10.4 RAML Definition

```
#%RAML 0.8
title: OIC Resource Types
version: v1-20150811
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.r"]

/oic/rts:
  description: |
    List of resource types that are supported by this OIC Server

  is : ['interface']
  get:
    description: |
      Retrieve the resource type list

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schemas.org/draft-04/schema#",
                "id": "http://openinterconnect.org/oic.wk.rts#",
                "definitions": {
                  "oic.wk.rts": {
                    "type": "object",
                    "properties": {
                      "tl": {
                        "type": "string",
                        "description": "Readonly, list of resource type names",
                        "format": "bsv"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  { "$ref": "#/definitions/oic.wk.rts" }
                ],
                "required": ["tl"]
              }
```



```
        example: |
          {
            "rt": "oic.wk.rts",
            "tl": "oic.r.example oic.r.other-example"
          }
```

D.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
tl	string	yes	Read Write	Readonly, list of resource type names

D.10.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/rts		get			

D.11 Scenes (Top level)

D.11.1 Introduction

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

D.11.2 Wellknown URI

/SceneListResURI

D.11.3 Resource Type

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

D.11.4 RAML Definition

```
#%RAML 0.8
title: OICScene
version: v1.0-20150630
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]

/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#",
               "id": "http://openinterconnect.org/schemas/oic.collection.json#",
```



```
                "title" : "Collection",
                "definitions": {
                        "oic.collection": {
                                "type": "object",
                                "properties": {
                                        "n": {
                                                "type": "string",
                                                "description": "Used to name the collection",
                                                "format": "UTF8"
                                        },
                                        "id": {
                                   "oneOf" : [
                                                        { "type": "number",  "description": "if id
property is an number" },
                                                        { "type": "string",  "description":  "if id
property is an number" }
                                                ]
                                        },
                                        "rts": {
                                                "type": "string",
                                                "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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "links" ]
                        }
                },

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

             example: |
               {
                   "rt":       "oic.wk.sceneList",
                   "n":        "list of scene Collections",
                   "id":       "my_sceneList_1",
                   "rts":      "oic.wk.sceneCollection",
                   "links": [
                    ]
               }

    post:
      description: |
        Provides the action to create a new sceneCollection in the SceneList resource
        The only resource type that is allowed to be created "oic.wk.sceneCollection".
        The example contains 3 scene values off, Reading and TVWatching.

      body:
        application/json
          schema: |
```



```
            {
                 "$schema": "http://json-schema.org/draft-04/schema#",
                 "id": "http://openinterconnect.org/schemas/oic.sceneCollection.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",
                                                    "description": "ReadOnly, 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",
                                                    "description": "ReadOnly, 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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                  { "required" : [ "ins" ] }
                                                           ]
                                                    }
                                             }
                               },
                               "required": [ "lastScene","sceneValues","rts","id" ]
                        }
                 },

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

          example: |
            {
               "scenevalue": "off",
               "sceneValues": "off,Reading,TVWatching",
               "lastScene": "off",
               "rt":       "oic.wk.sceneCollection",
               "n":        "my first scene",
               "id":       "my_scene1",
               "rts":      "oic.r.sceneMember"
```



```
4356            }
4357
4358        responses:
4359          200:
4360            description: |
4361              Indicates that the target resource was created.
4362              The created resource attributes are provided in the response,
4363              including the server generated identifier.
4364
4365            body:
4366              application/json:
4367                schema: |
4368                  {
4369                   "$schema": "http://json-schema.org/draft-04/schema#",
4370                   "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4371                   "title" : "Scene Collection",
4372                   "definitions": {
4373                          "oic.sceneCollection": {
4374                                 "type": "object",
4375                                 "properties": {
4376                           "lastScene": {
4377                                                "type": "string",
4378                                                "description": "Last selected Scene, shall be part of
4379    sceneValues",
4380                                                "format": "UTF8"
4381                                         },
4382                           "sceneValues": {
4383                                                "type": "string",
4384                                                "description": "ReadOnly, All available scene
4385    values",
4386                                                "format": "CSV"
4387                                         },
4388                                         "n": {
4389                                                "type": "string",
4390                                                "description": "Used to name the Scene collection",
4391                                                "format": "UTF8"
4392                                         },
4393                                         "id": {
4394                                                        "type": "string",
4395                                                "description" : "A unique string that could be a hash or
4396    similarly unique"
4397                                         },
4398                                         "rts": {
4399                                                "type": "string",
4400                                                "description": "ReadOnly, Defines the list of
4401    allowable resource types in links included in the collection; new links being created can only be
4402    from this list",
4403                                                "format": "UTF8"
4404                                         },
4405                                         "links": {
4406                                                "type": "array",
4407                                                "description": "Array of OIC web links that are
4408    reference from this collection",
4409                                                "items" : {
4410                                                       "allOf": [
4411                                                              { "$ref":
4412    "http://openinterconnect.org/schemas/oic.web-link.json#" },
4413                                                              { "required" : [ "ins" ] }
4414                                                       ]
4415                                                }
4416                                         }
4417                                 },
4418                                 "required": [ "lastScene","sceneValues","rts","id" ]
4419                          }
4420                  },
4421
4422                  "type": "object",
4423                  "allOf" : [
```

```
                        { "$ref": "oic.core.json#/definitions/oic.core" },
                        { "$ref": "#/definitions/oic.sceneCollection" }
                    ]
                  }

              example: |
                  {
                    "scenevalue": "off",
                    "sceneValues": "off,Reading,TVWatching",
                    "lastScene": "off",
                    "rt":       "oic.r.sceneCollection",
                    "n":        "mymembername",
                    "link":     "coap://newscene",
                    "id":       "0685B960-736F-46F7-BEC0-9E6CBD671ADC1",
                    "rts":      "oic.r.sceneMember"
                   }

    delete:

      description: |
        No change from collection.
        When delete is used with the URI of the collection without and query parameters then the
entire collection is deleted
        When the delete uses the "ins" parameter with the value of a specific link then only that
link is deleted

      queryParameters:

        ins:
          type: string

          description: Delete the Web link identified by the string - could be a UUID.

          required: false

          example: DELETE /mycollection?ins="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"

      responses:

        200:

          description: |
            The web link instance or the the entire collection has been successfully deleted

        400:

          description: |
            The request is invalid
```

D.11.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the collection
id	object			
rts	string		Read Write	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	yes	Read Write	Array of OIC web links that are reference from this collection
items				

D.11.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify

/SceneListResURI		get	post	delete	

D.12 Scene Collections

D.12.1 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.12.2 Wellknown URI

/SceneCollectionResURI

D.12.3 Resource Type

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

D.12.4 RAML Definition

```
#%RAML 0.8
title: OICScene
version: v1.0-20150630
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]

/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#",
               "id": "http://openinterconnect.org/schemas/oic.sceneCollection.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",
                                                "description": "ReadOnly, 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",
                                                "description": "ReadOnly, 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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "lastScene","sceneValues","rts","id" ]
                        }
                },

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

    post:
      description: |
        Provides the action to create a new sceneMember in the SceneCollection resource
        The only resource type that is allowed to be created is "oic.wk.sceneMember".
        The id of the resource will be generated by the implementation.
        As example the mappings of the 3 scenes are mapped to different states of an binary switch

      body:
        application/json
```



```
        schema: |
          {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "http://openinterconnect.org/schemas/oic.sceneMember.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",
                               "description": "ReadOnly, property name that will be mapped"
                             },
                               "memberValue": {
                               "type": "string",
                               "description": "ReadOnly, value of the Member Property"
                             }
                           },
                           "required": [ "scene", "memberProperty", "memberValue" ]
                         }
                       ]
                     },

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

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

        example: |
          {
            "link": { "href":"coap://mydevice/mybinaryswitch",
                      "if": "oic.if.a",
                      "rt": "oic.r.switch.binary" },
            "n": "my binary switch (for light bulb) mappings",
            "sceneMappings": [
                {
```



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

    responses:
      200:
        description: |
          Indicates that the target resource was created.
          The new resource attributes are provided in the response.

        body:
          application/json:
            schema: |
              {
               "$schema": "http://json-schema.org/draft-04/schema#",
               "id": "http://openinterconnect.org/schemas/oic.sceneMember.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",
                                    "description": "ReadOnly, property name that will be mapped"
                                  },
                                    "memberValue": {
                                    "type": "string",
                                    "description": "ReadOnly, value of the Member Property"
                                  }
                                },
                                "required": [ "scene", "memberProperty", "memberValue" ]
                              }
                            ]
                          },
```



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

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

    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#",
                  "id": "http://openinterconnect.org/schemas/oic.sceneCollection.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",
                                                "description": "ReadOnly, 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",
                                                "description": "ReadOnly, 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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "lastScene" ]
                        }
                },

                "type": "object",
                "allOf" : [
                        { "$ref": "oic.core.json#/definitions/oic.core" },
                        { "$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#",
                 "id": "http://openinterconnect.org/schemas/oic.sceneCollection.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",
                                                "description": "ReadOnly, 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",
                                                "description": "ReadOnly, 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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "lastScene" ]
                        }
                },

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

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

    delete:

      description: |
        No change from collection.
        When delete is used with the URI of the collection without and query parameters then the
entire collection is deleted
        When the delete uses the "ins" parameter with the value of a specific link then only that
link is deleted
```



```
    queryParameters:
      ins:
        type: string
        description: Delete the Web link identified by the string - could be a UUID.

        required: false
        example: DELETE /mycollection?ins="0685B960-FFFF-46F7-BEC0-9E6234671ADC1"

    responses:
      200:
        description: |
          The web link instance or the the entire collection has been successfully deleted

      400:
        description: |
          The request is invalid
```

D.12.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
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
items				

D.12.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/SceneCollectionResURI	put	get	post	delete	

D.13 Scene Member

D.13.1 Introduction

Collection that models a sceneMember.

D.13.2 Wellknown URI

/SceneMemberResURI

D.13.3 Resource Type

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

D.13.4 RAML Definition

```
#%RAML 0.8

title: OICScene
version: v1.0-20150630
```



```
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]

/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#",
               "id": "http://openinterconnect.org/schemas/oic.sceneMember.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",
                                 "description": "ReadOnly, property name that will be mapped"
                               },
                                 "memberValue": {
                                 "type": "string",
                                 "description": "ReadOnly, value of the Member Property"
                               }
                             },
                             "required": [ "scene", "memberProperty", "memberValue" ]
                           }
                         ]
                       },

                       "link": {
                                            "type": "string",
                                            "description": "web link that points at an resource",
```



```
                                        "$ref": "oic.web-link.json#"
                                }
                        },
                        "required": [ "link" ]
                }
        },

        "type": "object",
        "allOf" : [
                { "$ref": "oic.core.json#/definitions/oic.core" },
                { "$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.13.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
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
items				
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 an resource

D.13.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/SceneMemberResURI		get			

D.14 Rules (Top level)

D.14.1 Introduction

Toplevel Rule resource. This resource is a generic collection resource The rts value shall contain oic.wk.rule resource types

D.14.2 Wellknown URI

/RuleListResURI

D.14.3 Resource Type

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

D.14.4 RAML Definition

```
#%RAML 0.8
title: OICRules
version: v1.0-20150810
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]

/RuleListResURI:
  description: |
    Toplevel Rule resource.
    This resource is a generic collection resource
    The rts value shall contain oic.wk.rule resource types

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

    responses:
      200:
        body:
          application/json:
            schema: |
              {
               "$schema": "http://json-schema.org/draft-04/schema#",
               "id": "http://openinterconnect.org/schemas/oic.collection.json#",
               "title" : "Collection",
               "definitions": {
                      "oic.collection": {
                             "type": "object",
                             "properties": {
                                    "n": {
                                           "type": "string",
                                           "description": "Used to name the collection",
                                           "format": "UTF8"
                                    },
                                    "id": {
                                  "oneOf" : [
                                                  { "type": "number",  "description": "if id
property is an number" },
                                                  { "type": "string",  "description":  "if id
property is an number" }
                                           ]
                                    },
                                    "rts": {
                                           "type": "string",
```



```
                                                "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":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "links" ]
                        }
                },

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

            example: |

              {
                  "rt":       "oic.wk.ruleList",
                  "n":        "list of rules",
                  "id":       "my_ruleList_1",
                  "rts":      "oic.wk.rule",
                  "links": [
                   ]
              }

  post:

    description: |
      Provides the action to create a new rule in the ruleList resource
      The only resource type that is allowed to be created "oic.wk.rule".
      The example contains a condition, currentStatus and test.

    body:
      application/json

        schema: |

          {
               "$schema": "http://json-schema.org/draft-04/schema#",
               "id": "http://openinterconnect.org/schemas/oic.rule.json#",
               "title" : "Rule",
               "definitions": {
                       "oic.rule": {
                               "type": "object",
                               "properties": {
                                       "condition": {
                                               "type": "string",
                                               "description": "condition of the rule",
                                               "format": "UTF8"
                                       },
                                  "currentStatus": {
                                               "type": "string",
                                               "description": "ReadOnly, the current state, can be
one of: enabled, disabled, error"
                                       },
                                       "n": {
```



```
                                                "type": "string",
                                                "description": "Used to name the Rule collection",
                                                "format": "UTF8"
                                        },
                                        "test": {
                                                "type": "boolean",
                                                "description": "Inidcates initiation of test mode for
the rule"
                                        },
                                        "id": {
                                                "type": "string",
                                                "description": "Can be an value that is unique to the
use context or a UUIDv4"
                                        },
                                        "rts": {
                                                "type": "string",
                                                "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
                                        },
                                "links": {
                                                "type": "array",
                                                "description": "Array of OIC web links that are the
rule members, this is the script",
                                                "items" : {
                                                        "allOf": [
                                                                { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                },
                                "required": [ "links", "condition", "currentStatus", "test", "id",
"rts" ]
                        }
                },

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


          example: |

            {
               "condition":    "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value = true)
and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFFF:tempid:temperature > 30)",
               "currentStatus": "off",
               "test":         false,
               "rt":           "oic.wk.rule",
               "n":            "my first rule",
               "id":           "my_rule1",
               "rts":          "oic.r.ruleMember",
               "links": [
                       ]
            }


      responses:

        200:

          description: |
            Indicates that the target resource was created.
            The created resource attributes are provided in the response,
            including the server generated identifier.


          body:
            application/json:
```



```
              schema: |
                {
                 "$schema": "http://json-schema.org/draft-04/schema#",
                 "id": "http://openinterconnect.org/schemas/oic.rule.json#",
                 "title" : "Rule",
                 "definitions": {
                        "oic.rule": {
                               "type": "object",
                               "properties": {
                                      "condition": {
                                             "type": "string",
                                             "description": "condition of the rule",
                                             "format": "UTF8"
                                      },
                                "currentStatus": {
                                             "type": "string",
                                             "description": "ReadOnly, the current state, can be
one of: enabled, disabled, error"
                                      },
                                      "n": {
                                             "type": "string",
                                             "description": "Used to name the Rule collection",
                                             "format": "UTF8"
                                      },
                                      "test": {
                                             "type": "boolean",
                                             "description": "Inidcates initiation of test mode for
the rule"
                                      },
                                      "id": {
                                             "type": "string",
                                             "description": "Can be an value that is unique to the
use context or a UUIDv4"
                                      },
                                      "rts": {
                                             "type": "string",
                                             "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
                                      },
                                "links": {
                                             "type": "array",
                                             "description": "Array of OIC web links that are the
rule members, this is the script",
                                             "items" : {
                                                    "allOf": [
                                                           { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                           { "required" : [ "ins" ] }
                                                    ]
                                             }
                                      }
                                },
                                "required": [ "links", "condition", "currentStatus", "test", "id",
"rts" ]
                        }
                 },

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

              example: |
                {
                  "condition": "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value == true)
and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFFF:tempid:temperature > 30)",
```



```
                "currentStatus": "off",
                "test":         false,
                "rt":       "oic.wk.rule",
                "n":        "my first rule",
                "id":       "FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1" ,
                "rts":          "oic.r.ruleMember",
                "links": [
                     ]
               }

  delete:

    description: |
      No change from collection.
      When delete is used with the URI of the collection without and query parameters then the
entire collection is deleted
      When the delete uses the "ins" parameter with the value of a specific link then only that
link is deleted

    queryParameters:

      ins:
        type: string

        description: Delete the Web link identified by the string - could be a UUID.

        required: false

        example: DELETE /mycollection?ins="FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1"

    responses:

      200:

        description: |
          The web link instance or the the entire collection has been successfully deleted

      400:

        description: |
          The request is invalid
```

D.14.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the collection
id	object			
rts	string		Read Write	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	yes	Read Write	Array of OIC web links that are reference from this collection
items				

D.14.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/RuleListResURI		get	post	delete	

D.15 Rule

D.15.1 Introduction

Collection that models a rule. This resource is an generic collection resource with additional parameters. The rts value shall contain oic.wk.ruleMember resource types. The additional

parameters are condition, this is the rule that will be evaluated currentStatus, the current state of the rule, can be "enabled, disabled, error" test an trigger once activation of the rule

D.15.2 Wellknown URI

/RuleResURI

D.15.3 Resource Type

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

D.15.4 RAML Definition

```
#%RAML 0.8
title: OICRules
version: v1.0-20150810
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]

/RuleResURI:
  description: |
    Collection that models a rule.
    This resource is an generic collection resource with additional parameters.
    The rts value shall contain oic.wk.ruleMember resource types.
    The additional parameters are
    condition, this is the rule that will be evaluated
    currentStatus, the current state of the rule, can be "enabled, disabled, error"
    test  an trigger once activation of the rule

  get:
    description: |
      Provides the current rule and list of web links to the rule members

    responses:
      200:
        body:
          application/json:
            schema: |
              {
               "$schema": "http://json-schema.org/draft-04/schema#",
               "id": "http://openinterconnect.org/schemas/oic.rule.json#",
               "title" : "Rule",
               "definitions": {
                       "oic.rule": {
                               "type": "object",
                               "properties": {
                                       "condition": {
                                               "type": "string",
                                               "description": "condition of the rule",
                                               "format": "UTF8"
                                       },
                                  "currentStatus": {
                                               "type": "string",
                                               "description": "ReadOnly, the current state, can be
one of: enabled, disabled, error"
                                       },
                                       "n": {
                                               "type": "string",
                                               "description": "Used to name the Rule collection",
                                               "format": "UTF8"
                                       },
```



```
                                        "test": {
                                                "type": "boolean",
                                                "description": "Inidcates initiation of test mode for
the rule"
                                        },
                                        "id": {
                                                "type": "string",
                                                "description": "Can be an value that is unique to the
use context or a UUIDv4"
                                        },
                                        "rts": {
                                                "type": "string",
                                                "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
                                        },
                                "links": {
                                                "type": "array",
                                                "description": "Array of OIC web links that are the
rule members, this is the script",
                                                "items" : {
                                                        "allOf": [
                                                                { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                                { "required" : [ "ins" ] }
                                                        ]
                                                }
                                        }
                                 },
                                 "required": [ "links", "condition", "currentStatus", "test", "id",
"rts" ]
                        }
                },

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

             example: |

                {
                   "condition": "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value == true)
and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFFF:tempid:temperature > 30)",
                   "currentStatus": "disabled",
                   "test":     false,
                   "rt":       "oic.wk.rule",
                   "n":        "my first rule",
                   "id":       "FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1",
                   "rts":          "oic.r.ruleMember",
                   "links": [
                      ]
               }


  post:

    description: |
      Provides the action to create a new ruleMember in the rule resource
      The only resource type that is allowed to be created is "oic.wk.ruleMember".
      The id of the resource will be generated by the implementation.


    body:
      application/json

        schema: |

          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",
```



```
              "title" : "Rule Member",
              "definitions": {
                "oic.ruleMember": {
                  "type": "object",
                  "properties": {
                    "n": {
                      "type": "string",
                      "description": "Used to name the Rule member",
                      "format": "UTF8"
                    },
                    "id": {
                      "type": "string",
                      "description": "Can be an value that is unique to the use context or a UUIDv4"
                    },
                    "memberProperty": {
                      "type":  "string",
                      "description": "ReadOnly, property name that will be mapped"
                    },
                    "memberValue": {
                        "oneOf" : [
                                                              { "type": "number",  "description": "if
member property is an number" },
                                                              { "type": "string",  "description":  "if
member property is an number" },
                                                              { "type": "boolean", "description": "if
member property is an boolean" }
                                                    ],
                      "description": "ReadOnly, value of the Member Property"
                    },
                    "link": {
                      "type": "string",
                      "description": "web link that points at a resource",
                      "$ref": "oic.web-link.json#"
                    }
                  },
                  "required": [ "id", "link", "memberProperty", "memberValue" ],
                  "additionalProperties": false
                }
              },

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

          example: |
            {
              "link": { "href":"coap://mydevice/mybinaryswitch",
                        "if": "oic.if.a",
                        "rt": "oic.r.switch.binary" },
              "id":       "",
              "n": "my binary switch (for light bulb) mappings",
              "memberProperty":  "value",
              "memberValue":     true
            }

      responses:
        200:
          description: |
            Indicates that the target resource was created.
            The new resource attributes are provided in the response.

          body:
            application/json:
              schema: |
```



```
                {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",
                  "title" : "Rule Member",
                  "definitions": {
                    "oic.ruleMember": {
                      "type": "object",
                      "properties": {
                        "n": {
                          "type": "string",
                          "description": "Used to name the Rule member",
                          "format": "UTF8"
                        },
                        "id": {
                          "type": "string",
                          "description": "Can be an value that is unique to the use context or a
UUIDv4"
                        },
                        "memberProperty": {
                          "type":  "string",
                          "description": "ReadOnly, property name that will be mapped"
                        },
                        "memberValue": {
                            "oneOf" : [
                                                              { "type": "number",  "description": "if
member property is an number" },
                                                              { "type": "string",  "description":  "if
member property is an number" },
                                                              { "type": "boolean", "description": "if
member property is an boolean" }
                                                          ],
                          "description": "ReadOnly, value of the Member Property"
                        },
                        "link": {
                          "type": "string",
                          "description": "web link that points at a resource",
                          "$ref": "oic.web-link.json#"
                        }
                      },
                      "required": [ "id", "link", "memberProperty", "memberValue" ],
                      "additionalProperties": false
                    }
                  },

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

              example: |
                {
                  "id": "FFFFB960-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" },
                  "memberProperty":  "value",
                  "memberValue":     true
                }

    put:

      description: |
        Provides the action to enable/disable the rule and an test mode for the rule.
        Calling this method with test = true will update of all ruleMembers to the prescribed
membervalue.
```



```
    body:
      application/json
        schema: |
          {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "http://openinterconnect.org/schemas/oic.rule.json#",
                "title" : "Rule",
                "definitions": {
                       "oic.rule": {
                              "type": "object",
                              "properties": {
                                     "condition": {
                                            "type": "string",
                                            "description": "condition of the rule",
                                            "format": "UTF8"
                                     },
                                "currentStatus": {
                                            "type": "string",
                                            "description": "ReadOnly, the current state, can be
one of: enabled, disabled, error"
                                     },
                                     "n": {
                                            "type": "string",
                                            "description": "Used to name the Rule collection",
                                            "format": "UTF8"
                                     },
                                     "test": {
                                            "type": "boolean",
                                            "description": "Inidcates initiation of test mode for
the rule"
                                     },
                                     "id": {
                                            "type": "string",
                                            "description": "Can be an value that is unique to the
use context or a UUIDv4"
                                     },
                                     "rts": {
                                            "type": "string",
                                            "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
                                     },
                                "links": {
                                            "type": "array",
                                            "description": "Array of OIC web links that are the
rule members, this is the script",
                                            "items" : {
                                                   "allOf": [
                                                          { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
                                                          { "required" : [ "ins" ] }
                                                   ]
                                            }
                                     }
                              },
                              "required": [ "currentStatus" ]
                       }
                },

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

        example: |
          {
             "currentStatus": "enabled"
```



```
5696            }
5697
5698        responses:
5699          200:
5700            description: |
5701              Indicates that the value is changed.
5702              The changed properties are provided in the response.
5703
5704            body:
5705              application/json:
5706                schema: |
5707                  {
5708                   "$schema": "http://json-schema.org/draft-04/schema#",
5709                   "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5710                   "title" : "Rule",
5711                   "definitions": {
5712                          "oic.rule": {
5713                                 "type": "object",
5714                                 "properties": {
5715                                        "condition": {
5716                                               "type": "string",
5717                                               "description": "condition of the rule",
5718                                               "format": "UTF8"
5719                                        },
5720                                  "currentStatus": {
5721                                               "type": "string",
5722                                               "description": "ReadOnly, the current state, can be
5723    one of: enabled, disabled, error"
5724                                        },
5725                                        "n": {
5726                                               "type": "string",
5727                                               "description": "Used to name the Rule collection",
5728                                               "format": "UTF8"
5729                                        },
5730                                        "test": {
5731                                               "type": "boolean",
5732                                               "description": "Inidcates initiation of test mode for
5733    the rule"
5734                                        },
5735                                        "id": {
5736                                               "type": "string",
5737                                               "description": "Can be an value that is unique to the
5738    use context or a UUIDv4"
5739                                        },
5740                                        "rts": {
5741                                               "type": "string",
5742                                               "description": "ReadOnly, Defines the list of
5743    allowable resource types in links included in the collection; new links being created can only be
5744    from this list"
5745                                        },
5746                                  "links": {
5747                                               "type": "array",
5748                                               "description": "Array of OIC web links that are the
5749    rule members, this is the script",
5750                                               "items" : {
5751                                                      "allOf": [
5752                                                             { "$ref":
5753    "http://openinterconnect.org/schemas/oic.web-link.json#" },
5754                                                             { "required" : [ "ins" ] }
5755                                                      ]
5756                                               }
5757                                        }
5758                                  },
5759                                  "required": [ "currentStatus" ]
5760                          }
5761                   },
5762
5763                   "type": "object",
```



```
                "allOf" : [
                       { "$ref": "oic.core.json#/definitions/oic.core" },
                       { "$ref": "#/definitions/oic.rule" }
                ]
               }

             example: |
               {
                   "currentStatus": "enabled"
               }

  delete:
    description: |
      No change from collection.
      When delete is used with the URI of the collection without and query parameters then the
entire collection is deleted
      When the delete uses the "ins" parameter with the value of a specific link then only that
link is deleted

    queryParameters:
      ins:
        type: string
        description: Delete the Web link identified by the string - could be a UUID.

        required: false
        example: DELETE /mycollection?ins="FFFFB960-FFFF-46F7-BEC0-9E6234671ADC1"

    responses:
      200:
        description: |
          The web link instance or the the entire collection has been successfully deleted

      400:
        description: |
          The request is invalid
```

D.15.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
condition	string	yes	Read Write	condition of the rule
currentStatus	string	yes	Read Only	The Current State, Can Be One Of: Enabled, Disabled, Error
n	string		Read Write	Used to name the Rule collection
test	boolean	yes	Read Write	Inidcates initiation of test mode for the rule
id	string	yes	Read Write	Can be an value that is unique to the use context or a UUIDv4
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	yes	Read Write	Array of OIC web links that are the rule members, this is the script
items				

D.15.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/RuleResURI	put	get	post	delete	

D.16 Rule Member

D.16.1 Introduction

Rule member resource. This resource is assignment statement of an property in a resource indicated by an URI

D.16.2 Wellknown URI

/RuleMemberResURI

D.16.3 Resource Type

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

D.16.4 RAML Definition

```
#%RAML 0.8
title: OICRules
version: v1.0-20150810
traits:
 - interface
     queryParameters:
       if:
         enum: ["oic.if.a", "oic.if.ll"]


/RuleMemberResURI:
  description: |
    Rule member resource.
    This resource is assignment statement of an property in a resource indicated by an URI

  get:
    description: |
      Provides the rule mappings

    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",
                "title" : "Rule Member",
                "definitions": {
                  "oic.ruleMember": {
                    "type": "object",
                    "properties": {
                      "n": {
                        "type": "string",
                        "description": "Used to name the Rule member",
                        "format": "UTF8"
                      },
                      "id": {
                        "type": "string",
                        "description": "Can be an value that is unique to the use context or a
UUIDv4"
                      },
```



```
                        "memberProperty": {
                          "type":  "string",
                          "description": "ReadOnly, property name that will be mapped"
                        },
                        "memberValue": {
                            "oneOf" : [
                                                              { "type": "number",  "description": "if
member property is an number" },
                                                              { "type": "string",  "description":  "if
member property is an number" },
                                                              { "type": "boolean", "description": "if
member property is an boolean" }
                                                          ],
                          "description": "ReadOnly, value of the Member Property"
                        },
                        "link": {
                          "type": "string",
                          "description": "web link that points at a resource",
                          "$ref": "oic.web-link.json#"
                        }
                      },
                      "required": [ "id", "link", "memberProperty", "memberValue" ],
                      "additionalProperties": false
                    }
                  },

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

              example: |
                  {
                    "id": "FFFFB960-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" },
                    "memberProperty":  "value",
                    "memberValue":     true
                  }
```

D.16.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the Rule member
id	string	yes	Read Write	Can be an value that is unique to the use context or a UUIDv4
memberProperty	string	yes	Read Only	Property Name That Will Be Mapped
memberValue	object	yes		
link	string	yes	Read Write	web link that points at a resource

D.16.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/RuleMemberResURI		get			

