Tizen/Artik IoT Lecture Chapter 5. OCF Standard & IoTivity Tutorial

Sungkyunkwan University

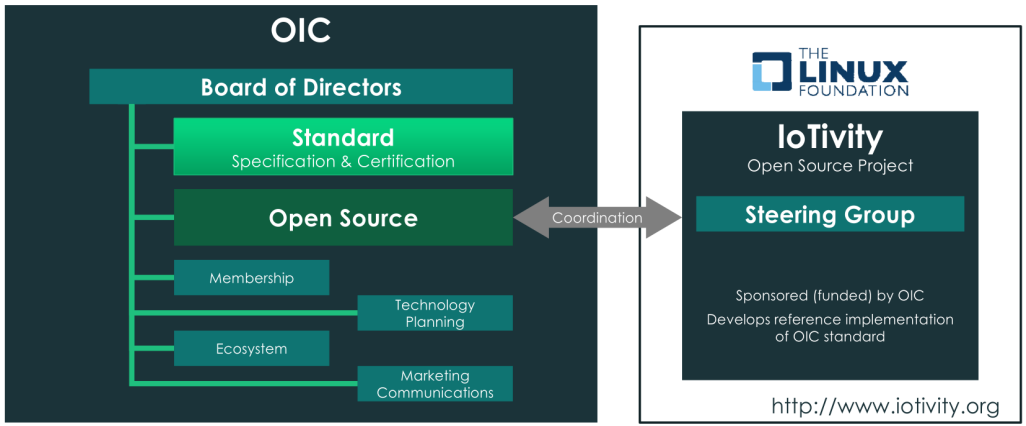
Contents

- **Part 1. OCF Standard Overview**
  - OCF Standard, OIC Roles & Abstractions
  - **RESTful Model**: URI & CRUDN
  - **RESTful Resource Model**
    - Resource, Resource Type, Link, Collection
    - Core Resources, Interfaces
    - Example
- **Part 2. IoTivity Overview**
  - IoTivity Structure: Base, Service
  - IoTivity Base Flow
  - IoTivity Source Tree

Part 1. OCF Standard Overview

IoTivity & OCF Standard

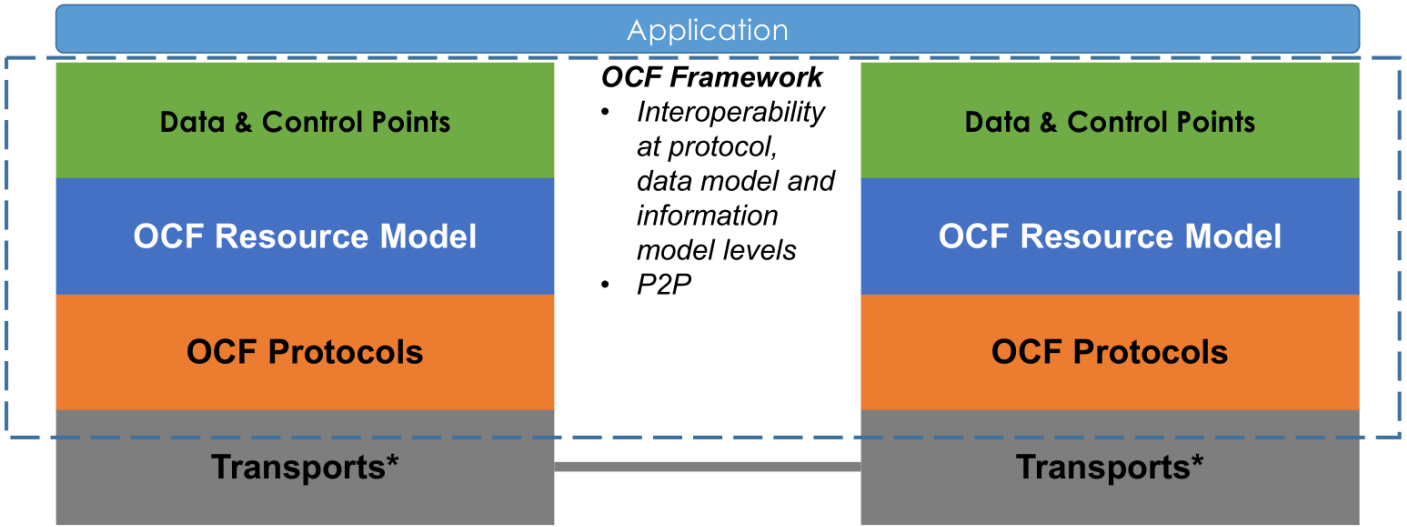
- **OCF Standard (Originally, OIC standard)**
  - IoT standard that supports service-level interoperability
- **IoTivity**
  - An open source software framework implementing OCF Standards



OCF Standard Specifications

- **OCF Core Spec.**
  - Spec. about OCF Framework
  - ex. OIC Core, Remote Access, Resource Type, Security
- **OCF Vertical Profiles Spec.**
  - Spec. about OCF profiles to enable IoT usages for different market segments
  - Based on Core Spec.
  - ex. Smart Home, Industrial, Healthcare and Automotive

Scope of OCF Standard



Data Models (RAML) (#)

URI + CRUDN
Encoding (CBOR/JSON)

CoAP, DTLS/TLS,
UDP/TCP, IPv6, …

<L2 Connectivity>
Wi-fi, Wi-fi Direct,
Bluetooth, Zigbee, …

OIC Roles & Abstractions

<OIC Roles>

OIC Client

- Initiate an transaction
- Access an OIC server to get a service

OIC Server

- Host OIC resource
- Send a response and provide service

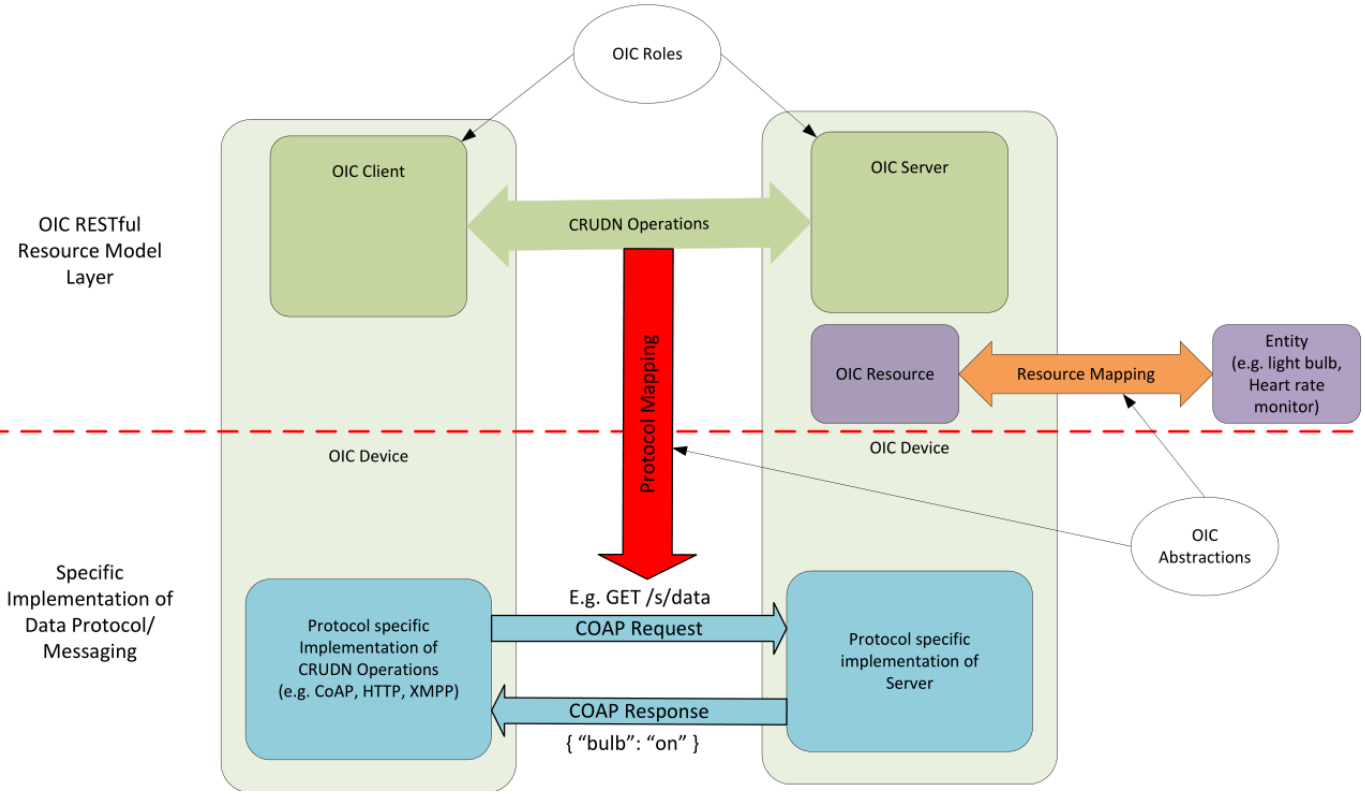
<OIC Abstractions>

Protocol Mapping

- CRUDN → CoAP/HTTP
- Implemented by IoTivity

Resource Mapping

- Entity → OIC Resource
- ex. Smart Home Spec.

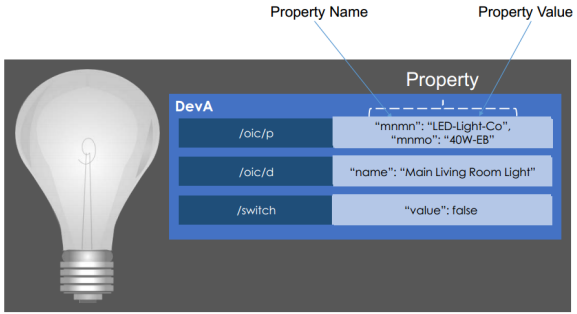


OIC Architecture

- **2 logical roles are defined in OIC architecture**
- **OIC Client**
  - Initiate an transaction
  - Access an OIC server to get a service
- **OIC Server**
  - Host OIC resource
  - Send a response and provide service
- **Based on RESTful Architecture**

RESTful Model: URI

- **Relative Reference**
  - Form: /<path>?<query>
  - ex. /oic/p?name=bulb;power=50
- **URI**
  - Form: oic://<reg-name>/<path>?<query>
  - ex. oic://<UUID>/oic/p
  - "oic": scheme for constructing and parsing the main parts
  - Path: segmented name
  - Query: Separator is ";", Order of query is important



RESTful Model: CRUDN

- **Operations performed by an OIC client on the resource contained in an OIC server**

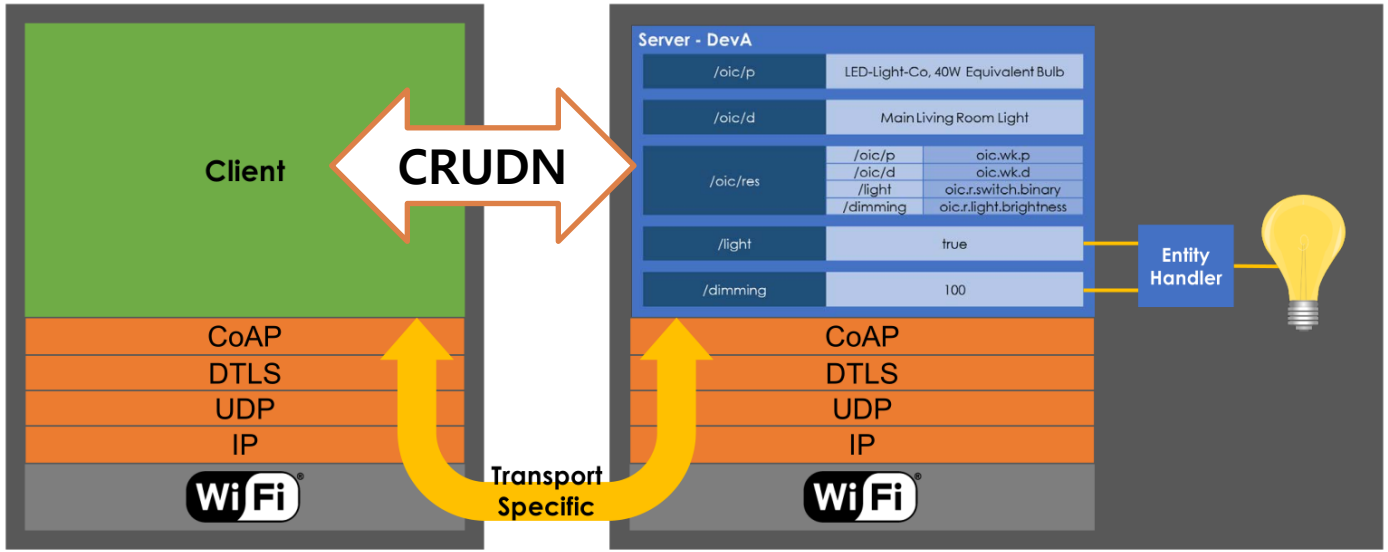
Operation	Description
CREATE	Create a new OIC resource by OIC server
RETRIEVE	Request the current state or representation of an OIC resource
UPDATE	Request a partial or complete replacing of the information in OIC resource
DELETE	Remove an OIC resource
NOTIFY	Request asynchronous notification of state changes

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.

Parameters of CRUDN Messages

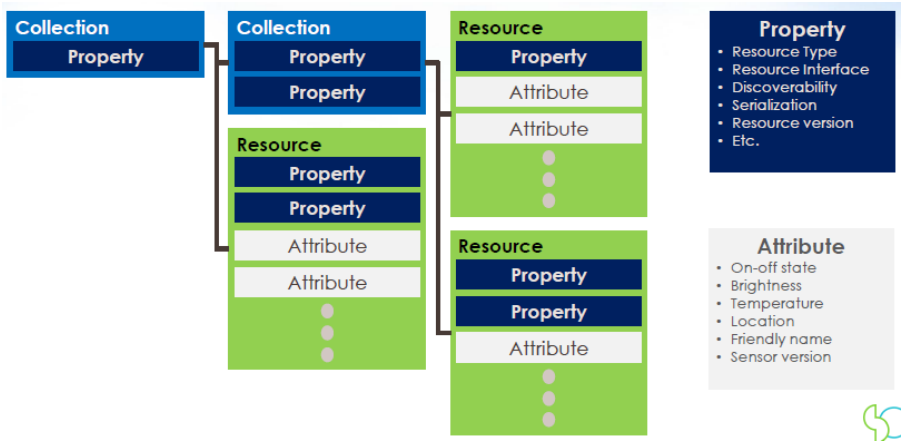
RESTful Resource Model

- **Role of RESTful Resource Model**
  - OIC devices can access the remote OIC devices independently to network protocols.



RESTful Resource Model

- **Describe Device's Capability in Abstract and Hierarchical Model**
  - Colletion – Resource/Property
  - Resource – Property/Attribute



RESTful Resource Model: Resource

- **A physical or software artifact or concept that needs to be made visible and manipulated**
  - Encapsulate & represent the salient aspects of an entity
  - Has address(URI) & properties

Property	Description
Fixed URI (optional)	A fixed URI assigned to an OIC resource for a Resource Type ID (ex. /oic/res, /oic/d)
Resource type title (optional)	Human friendly name to designate the resource type
Resource type ID	"rt" property (ex. oic.r.humidity)
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
Related Resource Types (optional)	Specification of other resource types that may be referred as part of the resource type, applicable to collections
MIME Types (optional)	Serialization methods (application/cbor, application/json, application/xml)

RESTful Resource Model: Resource Type

- **Resource Type**: Resource Template (#)
  - Composed of JSON schema and RAML file

```
"oic.r.switch.binary": {
   "type": "object",
   "properties": {
      "value": {
         "type": "boolean",
         "description": "Status of the switch"
      }
   }
}
```

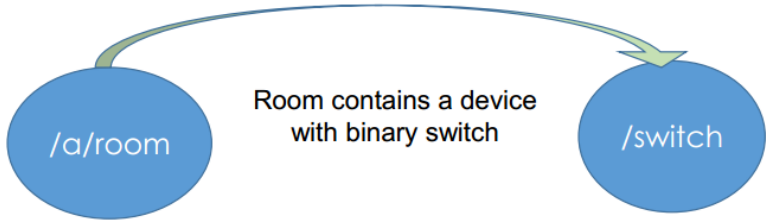
JSON Schema
→ Properties, Representation

```
get:
   responses :
     200:
       body:
          application/json:
             schema: |
               {
               }
```

RAML(RESTful API Modelling Language)
→ Request(CRUDN), Response(CoAP reseponse)

RESTful Resource Model: Link

- **Define the Connection between 2 resources**




```
{
   "href": "/switch",
   "rel": "contains",
   "anchor": "/a/room",
   "rt": "oic.r.switch.binary",
   "if": "oic.if.a",
   "bif": "oic.if.baseline"
}
```

Target

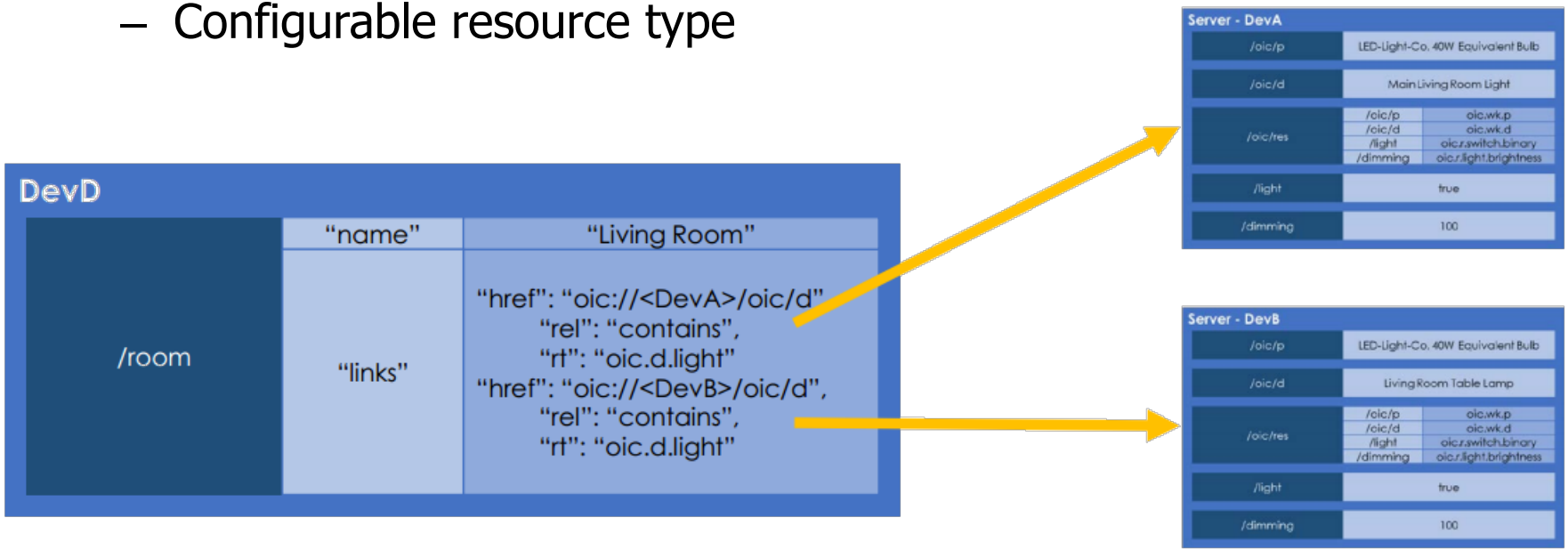
Relation

Context

Parameters

RESTful Resource Model: Collection

- **A Resource that also has links**
  - "links" property
  - Configurable resource type



RESTful Resource Model: Core Resources

UUID & Collection of other resources & Messaging protocol

Manufacturer, Model, OS information

UUID, Spec ver., Data model ver.

List of Resource Types

List of Interfaces

Device Name, Location, Currency, Region

Availability, LastedActedTime, DeviceStatistics (rx, tx traffic)

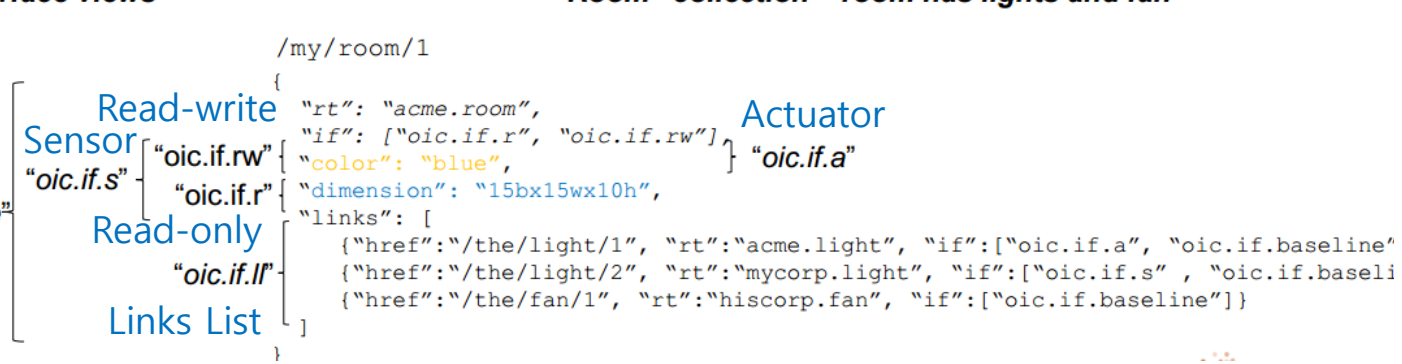
FactoryReset, Reboot, StartStatCollection

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

RESTful Resource Model: Interfaces

- **A view into a resource or collection**
  - Defined either by OCF core spec. or by vertical spec. or by vendor spec.
  - Resource requires at least one interface.
  - Determines which methods are allowed and what is the method's functionalities.
- **Predefined Interfaces in OCF standard**
  - Baseline, List Links, Batch, Read-only, Read-write, Actuator, Sensor

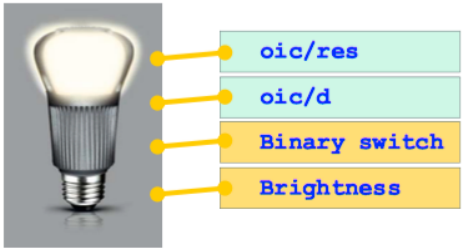
ex. Links List
RETRIEVE: return all the links
UPDATE: add, modify, delete link



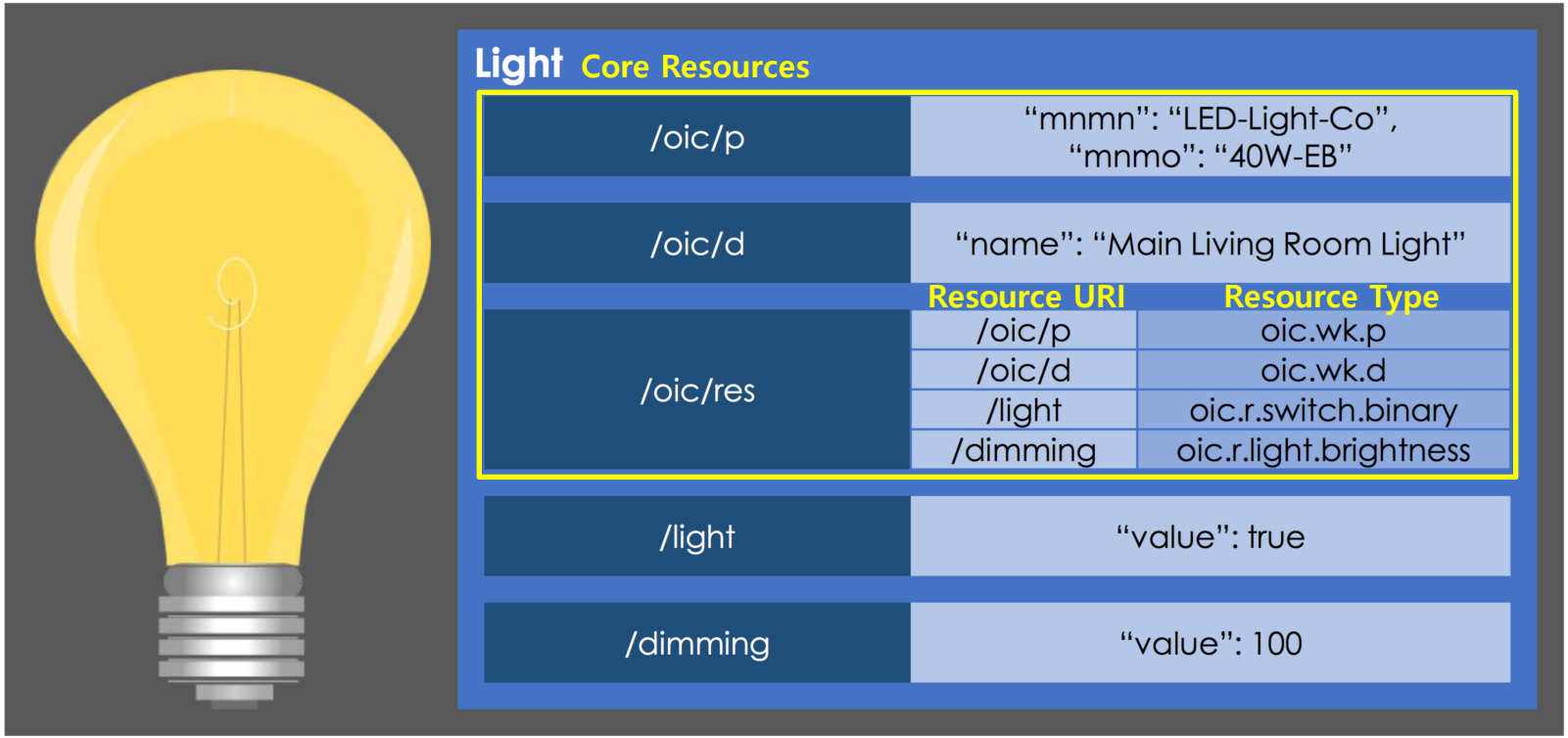
RESTful Resource Model: Example

- **Example Overview**
  - Smart light device with i) binary switch & ii) brightness resource
- **Device Type: Light Device (oic.d.light)**
- **Associated Resources**
  - Core resources: i) oic/res ii) oic/d
  - Device specific resources: iii) binary switch (oic.r.switch.binary)
  - Other optional resources can be exposed, in this example iv) brightness resource (oic.r.light.brightness)

Device Title	Device Type	Associated Resource Type	M/O
Light	oic.d.light	oic/res (oic.wk.core)	M
		oic/d (oic.d.light)	M
		Binary switch (oic.r.switch.binary)	M
		Brightness (oic.r.light.brightness)	O



RESTful Resource Model: Example



Platform Info.

Device Info.

Resource List of This Device

Light Resource

Dimming Resource

Part 2. IoTivity Overview

IoTivity: Main Features

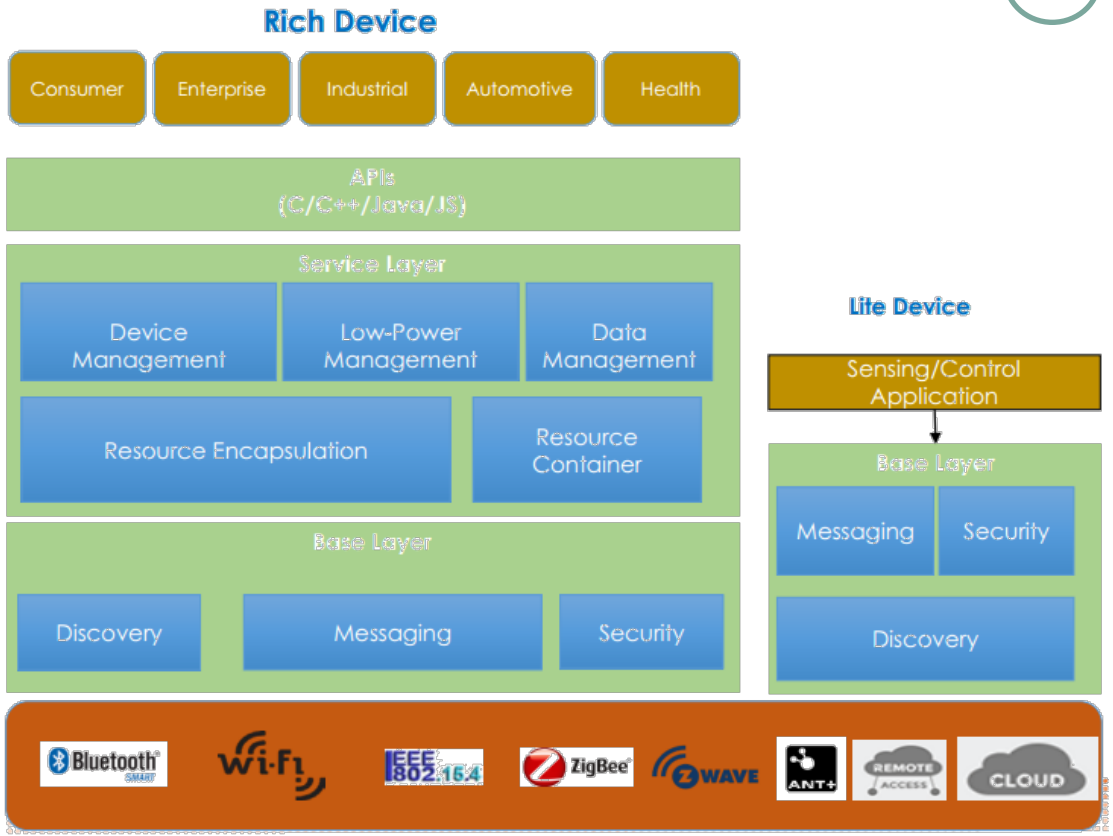
- **Cross-platform**
  - Supported: Tizen, iOS, Android, Ubuntu, OS X, Windows, etc.
  - **scons**: cross-platform build tool
- **Support Constrained(Lite) Devices**
  - IETF RFC 7228 Class 1 & 2
  - ex. Arduino
- **Various API Language Bindings**
  - C, C++, Java(Android), etc.
  - JavaScript: Node.js, IoT.js (Planned)

Class	Data Size (SRAM)	Code Size (Flash)
0	<< 10KiB	<< 100KiB
1	**~10KiB**	**~ 100KiB**
2	**~50KiB**	**~250KiB**

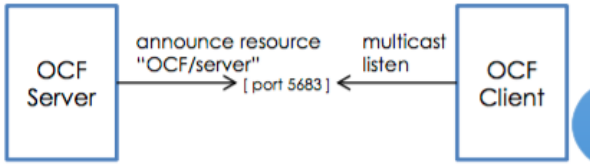
IETF RFC 7228 Constrained Device

IoTivity Structure: Base Layer

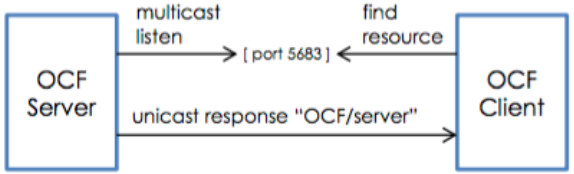
- **Discovery**
  - Finding a Resource
- **Messaging**
  - Connectivity Abstraction
  - Remote Access (XMPP)
  - Message Switching (Multi-hop)
- **Security**
  - Onboarding
  - Ownership Transfer
  - Provisioning
  - Access Control



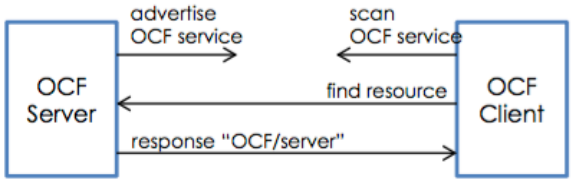
Discovery Subsystem



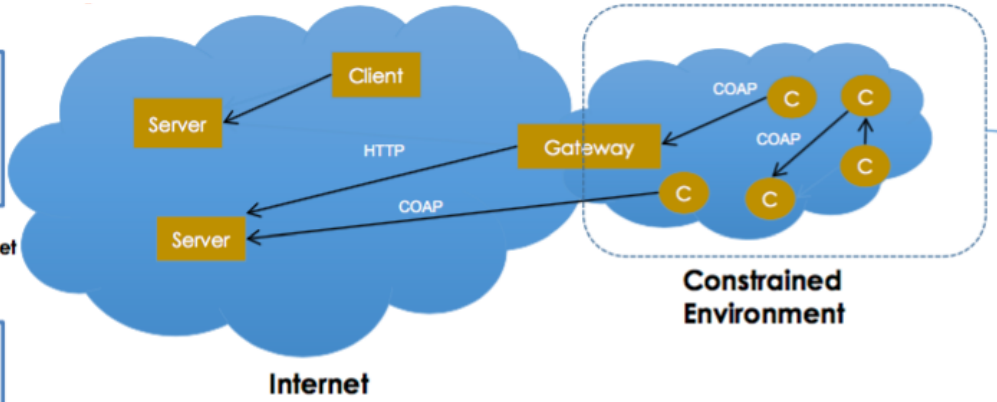
[Figure 1] Multicast announcement over Wi-Fi / Ethernet



[Figure 2] Multicast/Unicast over WiFi / Ethernet



[Figure 3] Advertise/Scan over BLE/BT



FOUNDATION™

Connectivity	Discovery Mechanism	Description
WiFi & Ethernet (over IP)	IP Multicast	CoAP Multicast Port: 5683 (Assigned by IANA) CoAP Secure Port: 5684
	IP Unicast over UDP	Precondition: OIC Server Address & Port are known
Bluetooth (EDR & BLE)	Using Scan & Advertise	OCF Specific Service UUID

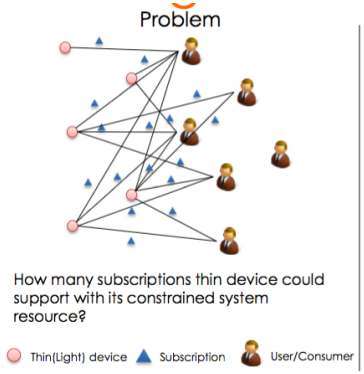
CoAP

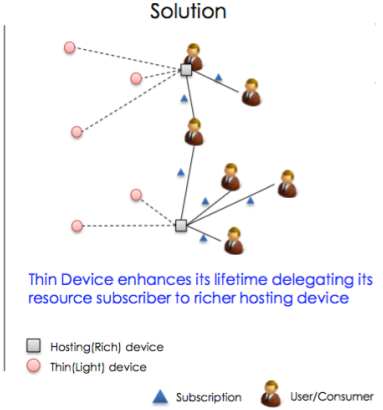
- Open IETF Standard (RFC 7252)
- Compact 4 Byte Header
- UDP (Default), SMS, TCP Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-In Discovery

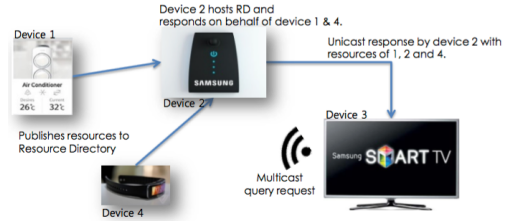
CoAP: Constrained Application Protocol
IANA: Internet Assigned Numbers Authority

IoTivity Structure: Service Layer (1/2)

- **Device Management**
  - Thing Manager
    - Manage groups of things
  - Scene Manager
    - Create a scenario for controlling multiple devices
- **Low-Power Management**
  - Resource Hosting
    - Offloads request/data handling from remote clients
  - Resource Directory
    - Constrained device that needs to sleep and cannot respond to multicast discovery queries

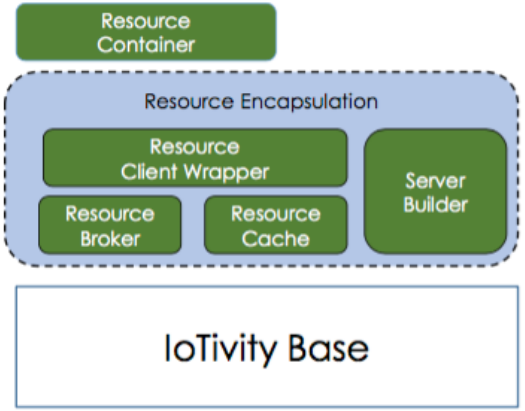


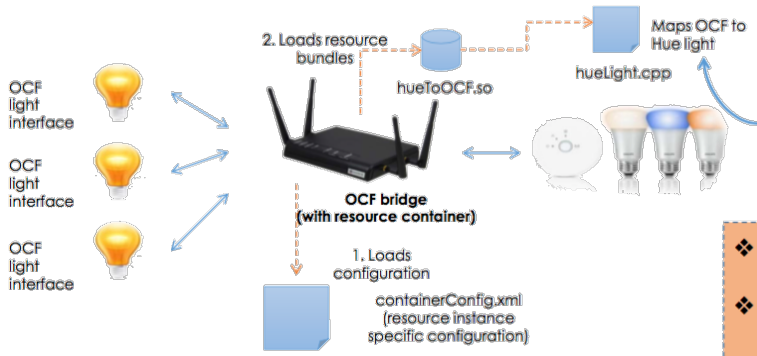




IoTivity Structure: Service Layer (2/2)

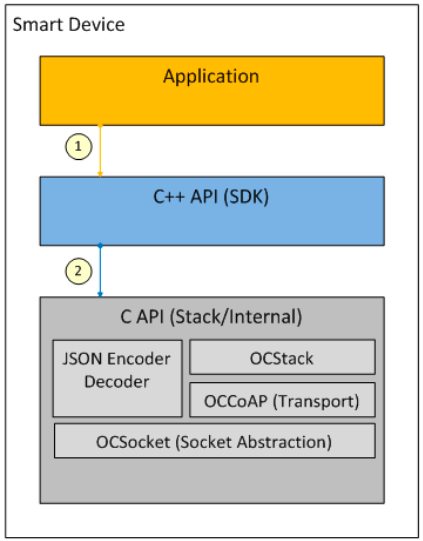
- **Resource Encapsulation**
  - Resource Broker
    - Remote resource presence check
    - Consistent reachability management for discovered resource of interest
  - Resource Cache
    - Maintain last information of remote resource
- **Resource Container**
  - Protocol Bridge: Integrates non-OCF resources

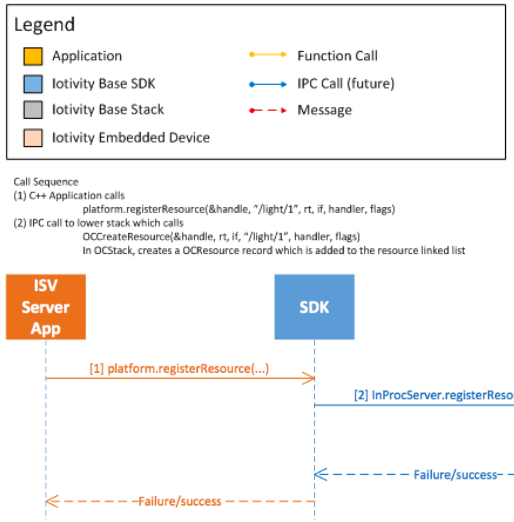




IoTivity Base Flow (1/5)

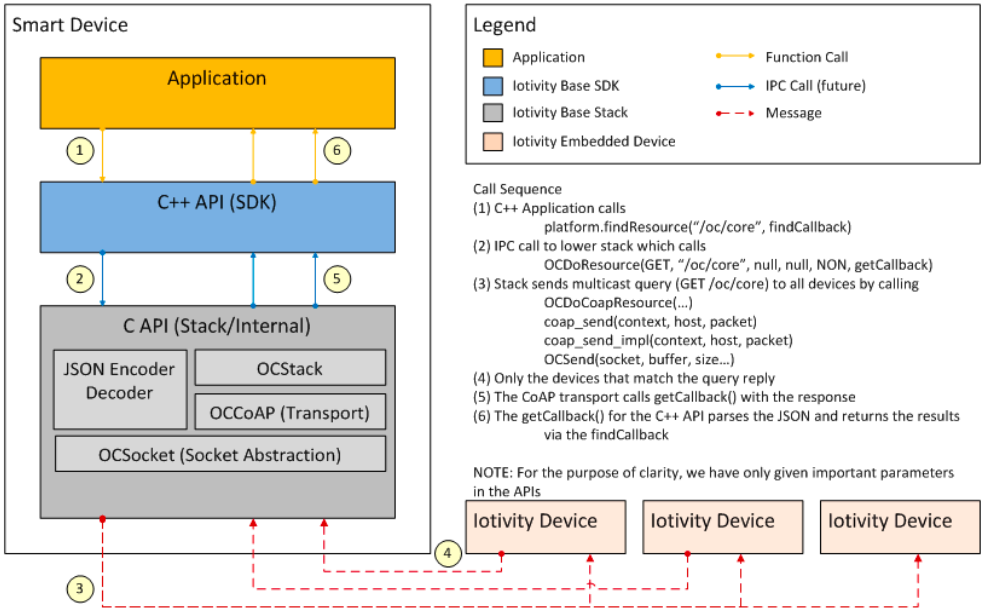
- **Registering a Resource**
  - Given a service running on port 5683 in a device at IP address 192.168.1.1,
  - If the application registers a resource with a URI path "/light/1",
  - The resulting fully qualified URI "oc://192.168.1.1:5683/light/1"





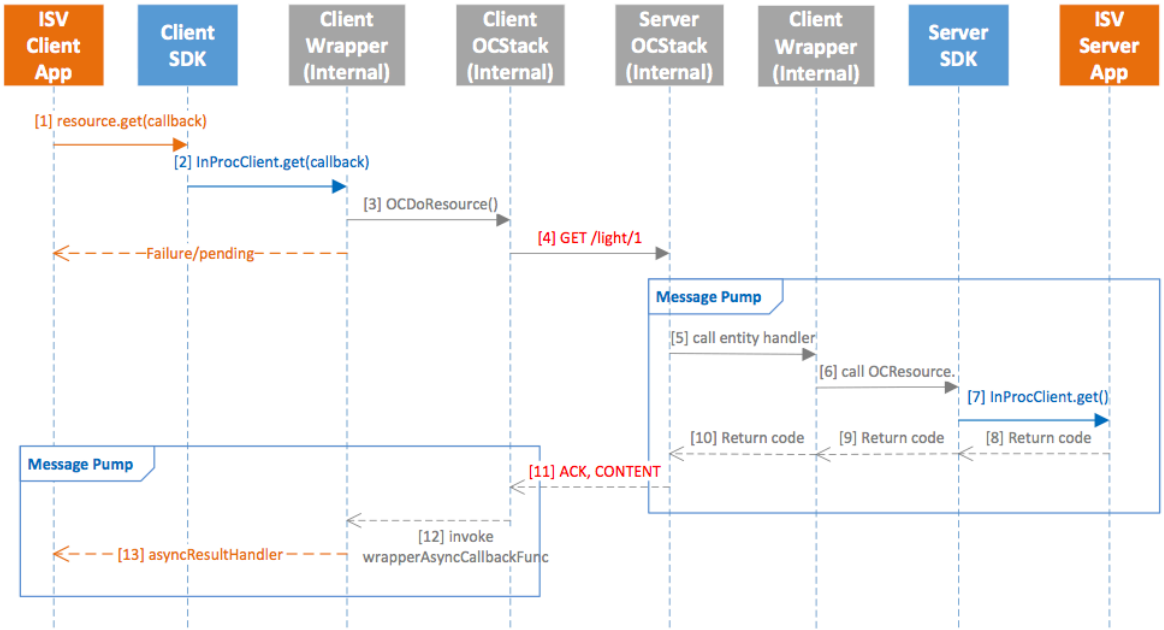
IoTivity Base Flow (2/5)

- **Finding a resource**
  - returns all resources of given type on the network service



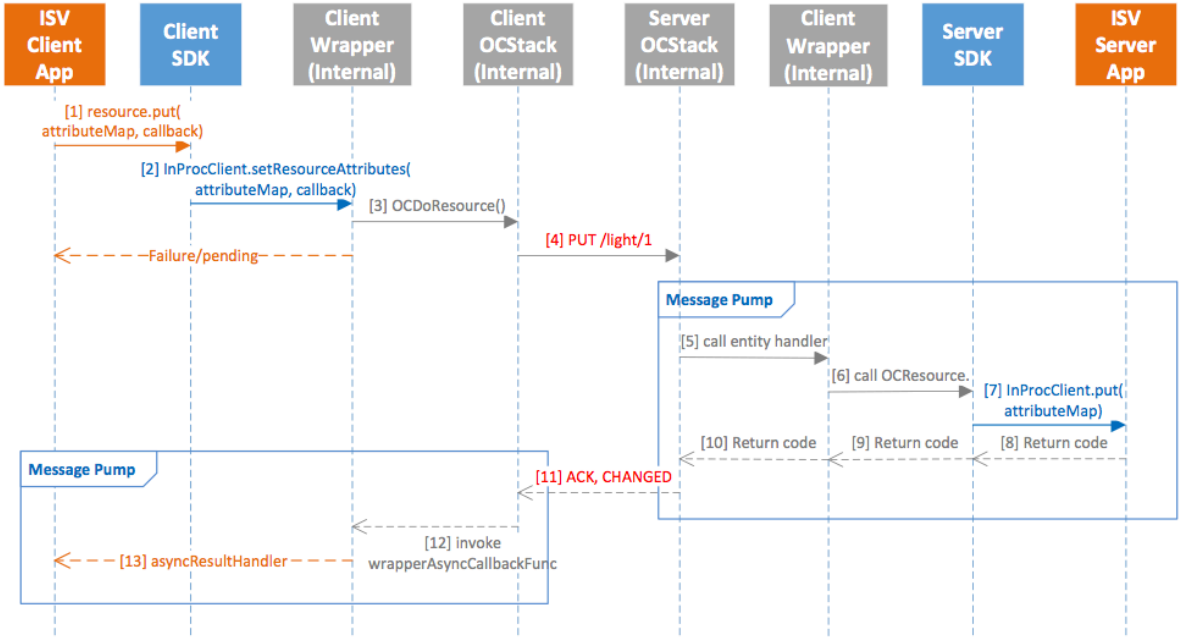
IoTivity Base Flow (3/5)

- **Querying resource state (GET)**



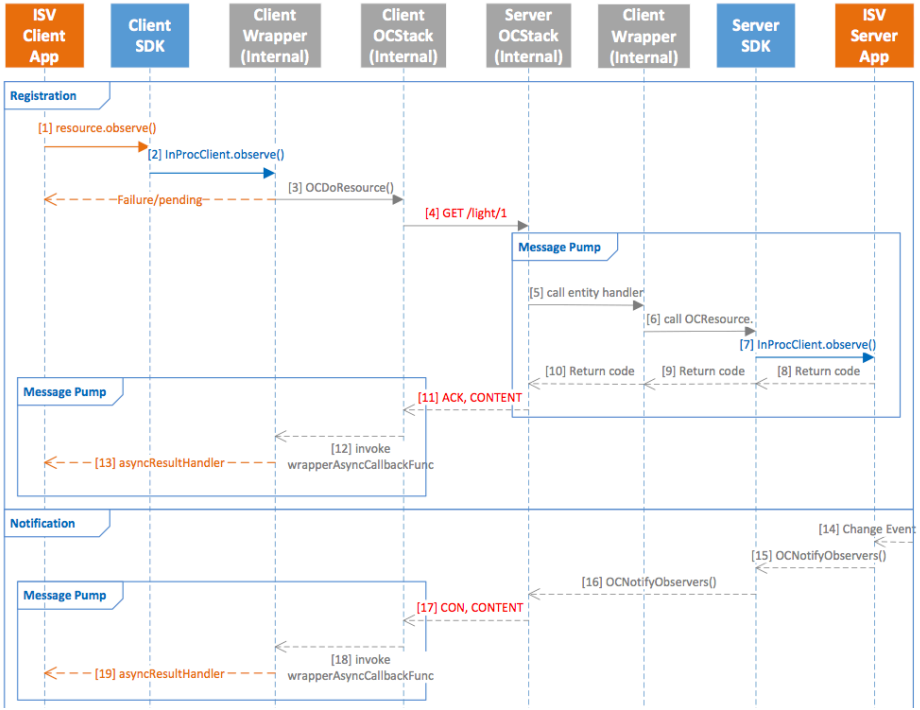
IoTivity Base Flow (4/5)

- **Setting resource state (PUT)**



IoTivity Base Flow (5/5)

- **Observing resource state**



IoTivity Source Tree