OCF Specification Overview
Core Technology Specification

OCF 2.0.1 Release

February 2019

Core Framework Topics Outline (1 of 2)

- Objectives

- RESTful Architecture

- OCF Roles

- Resources

- Basic Operations

- Organization of an OCF Device

- OCF Specification Features

- Protocol Stack

- Device Example

- Endpoint Overview

Core Framework Topics Outline (2 of 2)

- Resource Discovery (CoAP Discovery)

- Encoding Schemes

- Defining OCF Components

- Vendor Extensions

- Introspection

- Collection Resources

- Atomic Measurement Resources

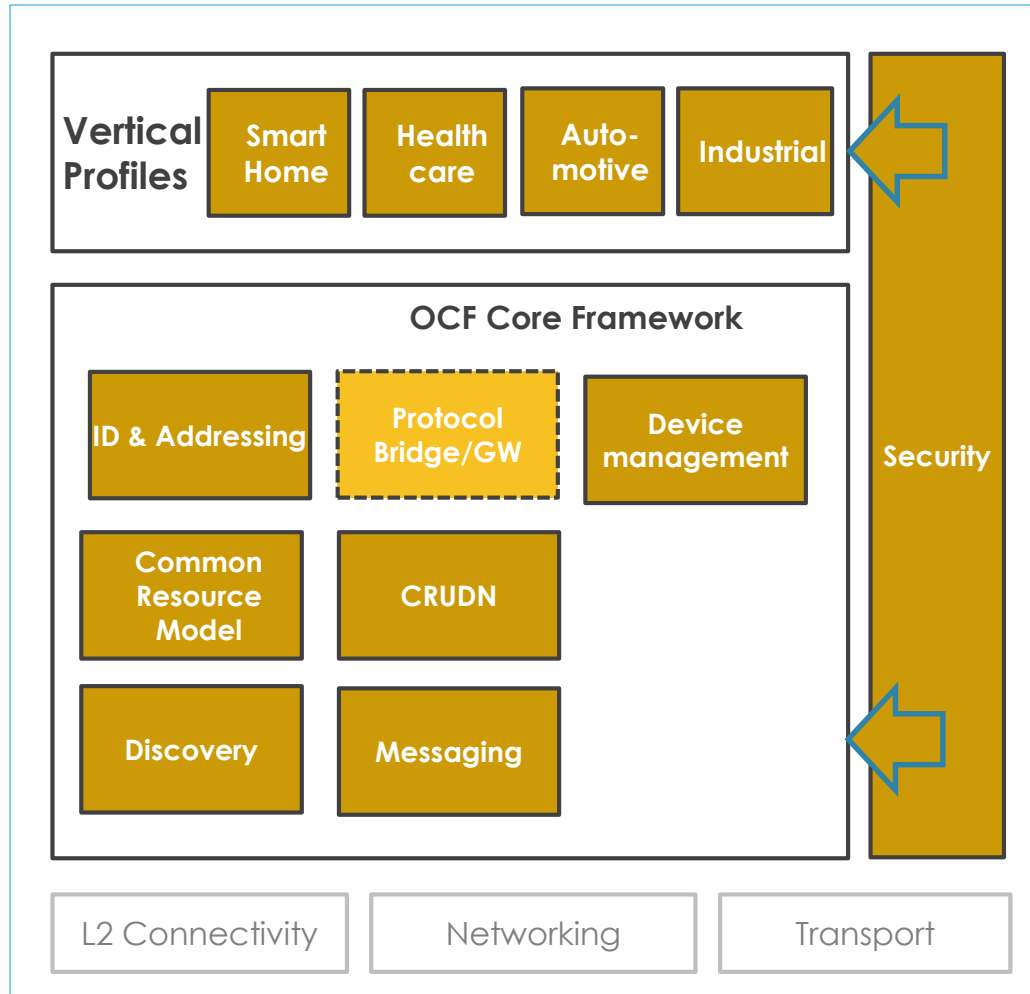- Resource Discovery (Resource Directory)

Core Framework Objectives

- Core Framework Specification Scope
  - Specifies the technical specification(s) comprising of the core architectural framework, messaging, interfaces and protocols based on approved use-case scenarios
  - Enables the development of vertical profiles (e.g. Smart Home) on top of the core while maintaining fundamental interoperability
- Architect a core framework that is scalable from resource constrained devices to resource rich devices
- Reuse open standards solutions (e.g. IETF) where they exist
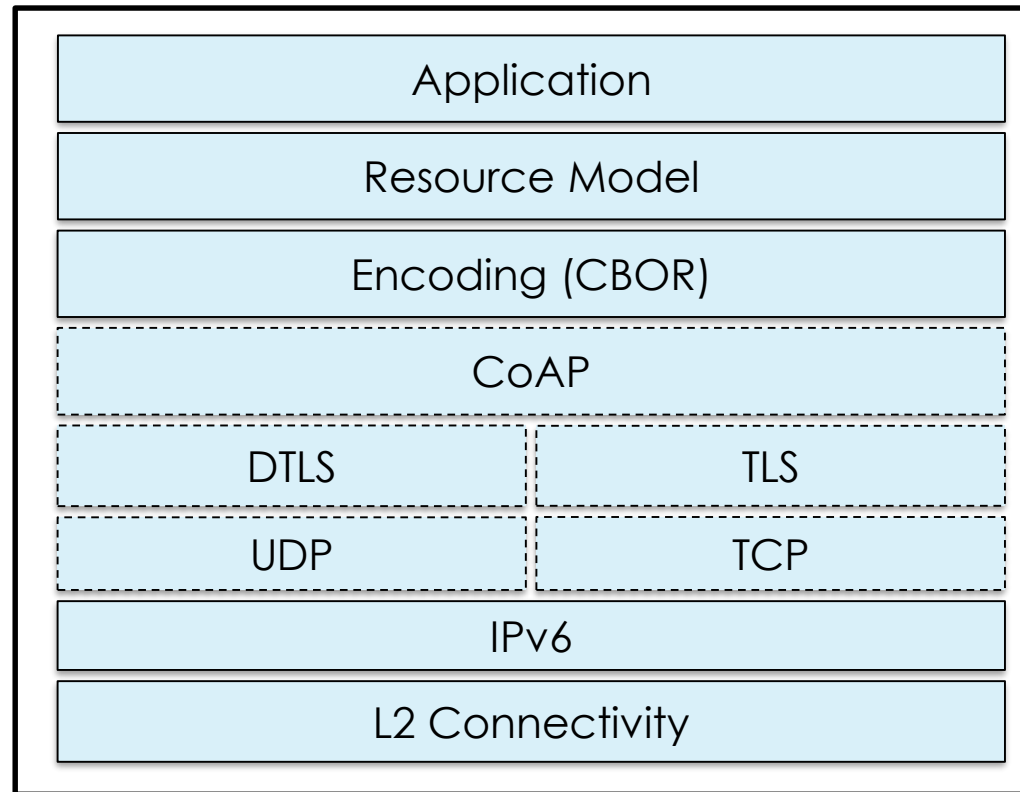- Ensure alignment with Iotivity open source releases

OCF Spec Features – Core Framework Spec

Vertical Profiles
- Smart Home
- Health care
- Auto-motive
- Industrial

OCF Core Framework
- ID & Addressing
- Protocol Bridge/GW
- Device management
- Common Resource Model
- CRUDN
- Discovery
- Messaging

Security

- L2 Connectivity
- Networking
- Transport

① **Discovery:** Common method for device discovery (IETF CoRE)

② **Messaging:** Constrained device support as default (IETF CoAP) as well as protocol translation via bridges

③ **Common Resource Model:** Real world entities defined as data models (resources)

④ **CRUDN:** Simple Request/Response mechanism with Create, Retrieve, Update, Delete and Notify commands

⑤ **ID & Addressing:** OCF IDs and addressing for OCF entities (Devices, Clients, Servers, Resources)

⑥ **Protocol Bridge/GW**: Handled by the Bridging Spec with some implications on the Core

Security is fundamental to the OCF ecosystem and applies to all elements

Protocol Stack

OCF Stack

The diagram shows the OCF protocol stack with the following layers from top to bottom:
- Application
- Resource Model
- Encoding (CBOR)
- CoAP
- DTLS | TLS
- UDP | TCP
- IPv6
- L2 Connectivity

Resources

- An OCF Server contains one or more Resources to describe a real world entity

- Each Resource contains Properties that describes an aspect that is exposed through a Resource including meta-information related to that Resource

- Each Resource contains Interface(s) that provides first a view into the Resource and then defines the requests and responses permissible on that view of the Resource

Collection Resources (Optional)

- An OCF Resource that contains one or more references (specified as OCF Links) to other OCF Resources, where each Link is individually addressable, is an OCF Collection

- An OCF Link embraces and extends typed "web links" as specified in RFC 5988

- The primary example of a collection is /oic/res (Discovery Resource).

  - A small number of Resources in the Resource Model are also collections

Atomic Measurement Resources (Optional)

- An OCF Resource that ensures a Client can only access the Properties of linked Resources (specified as OCF Links) atomically, as a whole, and read-only, using the "batch" interface

  - Atomically, meaning the value of all properties of the Atomic Measurement are sampled at the same time

  - As a whole, meaning that the values of all properties of the Atomic Measurement will be returned, or no value will be returned

  - Read-only, meaning that the properties of the Atomic Measurement can only be read, not written, using the batch interface. Any attempt to write to any property of the Atomic Measurement will result in an error.

- An OCF Link embraces and extends typed "web links" as specified in RFC 5988

- The primary example of Atomic Measurement Resources are with Healthcare vertical defined OCF Resources (e.g blood pressure measurement)

Encoding Schemes – CBOR

- Everything in OCF is a Resource.
- All Resources are specified using Swagger (OpenAPI2.0) to define the associated API
- OCF has mandated CBOR as the default encoding scheme on the wire

	CBOR	JSON	XML/EXI
Description	**- Concise binary object representation based on JSON data model**	- Lightweight, text-based, language-independent data interchange format	- Binary compression standard for XML
		If needed in future revisions	
Standard	**IETF RFC 7049**	IETF RFC 7159	W3C Efficient XML Interchange Format 1.0
Content Type	**/application/vnd.ocf+cbor**	/application/json	/application/exi
OCF M/O	**Mandatory**	Can be supported	Can be supported

Resource Discovery (CoAP Discovery)

- OCF devices make use of CoAP Discovery using IANA defined OCF Service Address (not the default CoAP address).

- Multicast RETRIEVE (CoAP GET) sent to well known URI /oic/res

- Response is an array of links; each link represents a Resource hosted by the responding server

- Links provide:

  - href

  - Relationship (self link, hosted link, bridged link)

  - Endpoint binds

  - Supported interfaces

  - Observability of the Resource

Resource Discovery via Resource Directory (Optional)

- OCF Devices may use Resource Directory to find the Resources hosted in the 3rd party Devices.
  - Publishing Devices register the Resources (i.e. Links) to a Resource Directory, to which a Client subsequently makes an inquiry to discover those Resources.
- Resource Directory
  - An OCF Device facilitating indirect discovery by exposing 3rd party Resources (i.e. Links) with the following features
  - RD discovery
    - Discover an RD and select one with oic.wk.rd
  - Resource publish
    - publish/update/delete Resource (i.e. Links) in /oic/res of an RD
  - Resource exposure
    - Expose published Resources via /oic/res of RD, which is aligned with CoAP discovery.

Endpoint Overview

- Definition
    - An (OCF) Endpoint is defined as the source or destination of a request and response messages for a given Transport Protocol Suites (e.g. CoAP over UDP over IPv6). The specific definition of an Endpoint depends on the Transport Protocol Suites being used.
        - (e.g.) For CoAP/UDP/IPv6, Endpoint is identified as IP address + port number.
- Endpoint characteristics for OCF Device
    - Each OCF Device shall associate with at least one Endpoint with which it can exchange Request & Response messages.
        - When a message is sent to an Endpoint, it shall be delivered to the OCF Device which is associated with the Endpoint. When a Request message is delivered to an Endpoint, path component is enough to locate the target Resource.
    - OCF Device can be associated with multiple Endpoints.
        - E.g. OCF Device may support both CoAP & HTTP
    - An endpoint can be shared among multiple OCF Devices, only when there is a way to clearly indicate the target Resource with Request URI.

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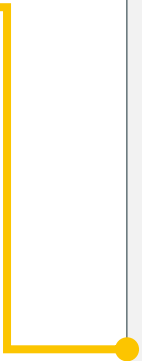
```
[
  { "href": "/oic/res",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989/oic/res",
    "rel": "self",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[fe80::b1d6]:44444"}] },
  { "href": "/oic/p",
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",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

/oic/res
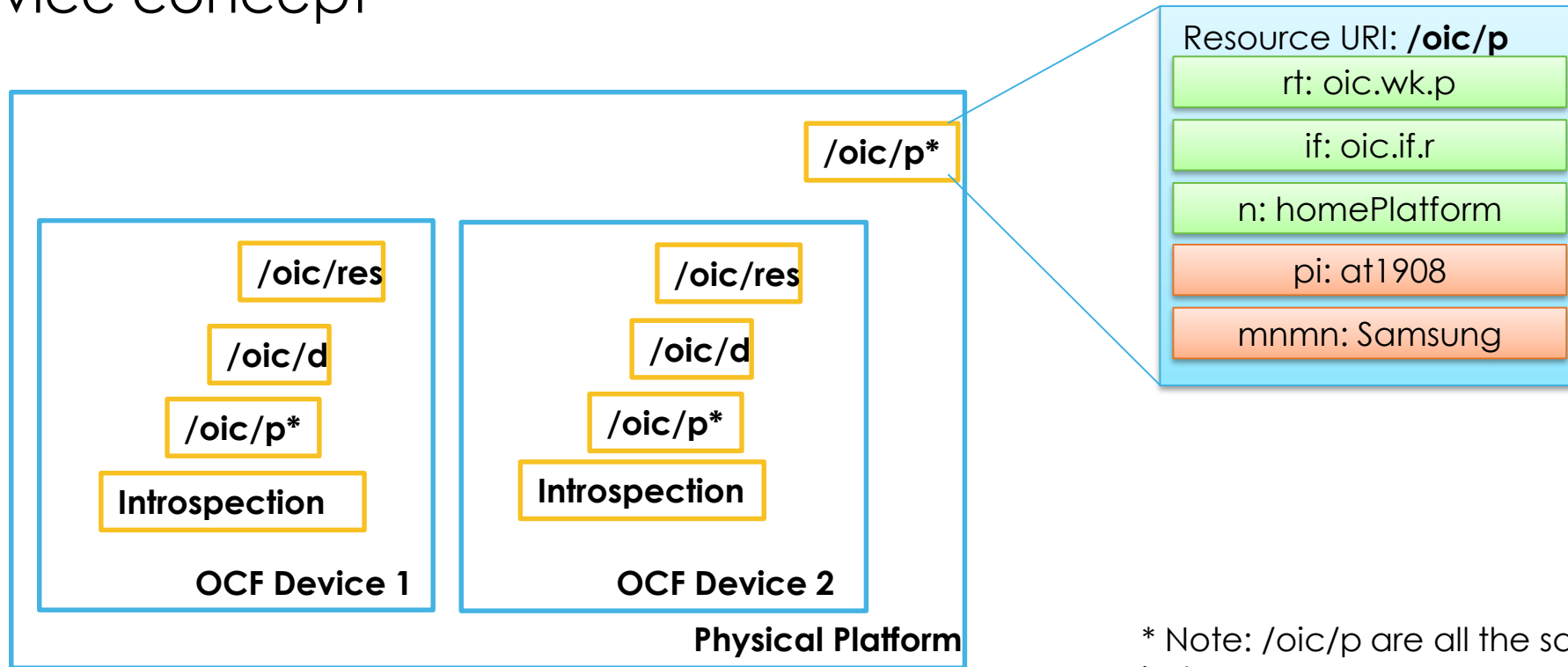
Endpoint for each target resource.

Introspection

- What
  - Device description is available on the network
  - Device description:
    - List all end points
    - Per end point
      - Which methods are implemented
        - Query parameters per method
        - Payloads definitions (request and response)
- How
  - Put the complete set of Resources exposed by a Device on the wire as a CBOR encoded Swagger2.0 document.
    - Describes the payload on JSON level
      - Property names
      - Type
      - range

Organization of an OCF Device

- OCF Device concept

Resource URI: **/oic/p**
- rt: oic.wk.p
- if: oic.if.r
- n: homePlatform
- pi: at1908
- mnmn: Samsung

/oic/p*

OCF Device 1
- /oic/res
- /oic/d
- /oic/p*
- Introspection

OCF Device 2
- /oic/res
- /oic/d
- /oic/p*
- Introspection
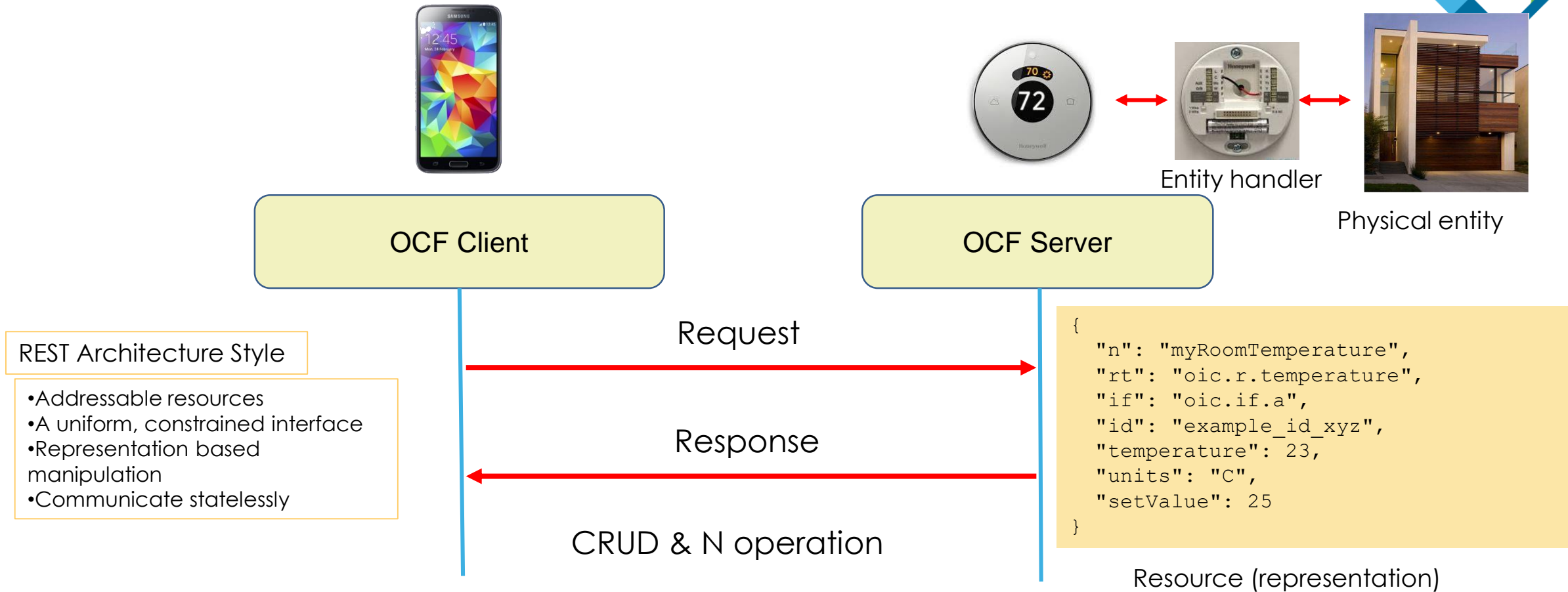
Physical Platform

* Note: /oic/p are all the same instance

Defining OCF Components (on top of CORE)

- OCF Servers
  - Defined by *device identifier*: standardized name of the device
  - List of mandatory OCF Resource Types per device
  - Note that OCF Clients are implicitly specified as "opposite" side of an OCF Server.
    - Currently OCF does not impose interaction sequences.
    - All instances of a Resource Type are allowed to talk to/from any OCF Client at any point in time
- OCF Resource Type
  - Defined by *resource identifier*: standardized name of the resource
  - List of mandatory properties per Resource Type
  - List of allowed actions (read/readwrite/..) per Resource Type
  - All OCF Resource Type IDs are IANA registered: http://www.iana.org/assignments/core-parameters/core-parameters.xhtml

RESTful Architecture

OCF Client

OCF Server

Entity handler

Physical entity

REST Architecture Style

- Addressable resources
- A uniform, constrained interface
- Representation based manipulation
- Communicate statelessly

Request

Response

CRUD & N operation

```
{
    "n": "myRoomTemperature",
    "rt": "oic.r.temperature",
    "if": "oic.if.a",
    "id": "example_id_xyz",
    "temperature": 23,
    "units": "C",
    "setValue": 25
}
```

Resource (representation)

🔵 RESTful Architecture (Representational State Transfer)

▪ Resource based operation

  ▪ Real world 'entity' is represented as 'Resource'

▪ Resource manipulation via Request/ Response: CRUDN

Device example: light device (oic.d.light)

- Example overview
  - Smart light device with i) binary switch & ii) brightness resource
- Device type: Light device (oic.d.light) [Defined by the domain]
- Associated resources
  - Mandatory Core resources: oic/res, oic/p, oic/d
  - Mandatory Security Resources (not shown in the diagram)
  - Device specific resources: Binary switch (oic.r.switch.binary),
  - Other optional resources can be exposed, in this example Brightness resource (oic.r.light.brightness)

Example: Smart light device

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

oic/res
oic/p
oic/d
Binary switch
Brightness
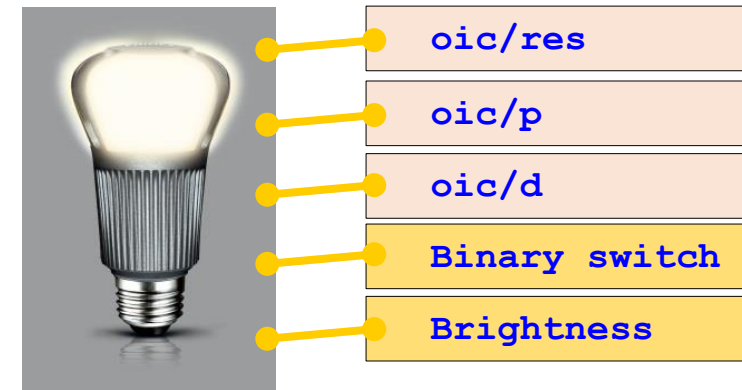
Vendor Extensions

- Vendor is allowed to:
  - Create their own defined (non-OCF standardized) Resource Types
  - Create their own defined (non-OCF standardized) Device Types
  - Extend existing devices with additional (not mandated) Resource Types
    - With standardized resource types
    - With vendor defined resource types
- All vendor extensions follow an OCF defined naming scheme
  - x.<vendor owned domain reversed>.<resourcename>
  - e.g. x.com.acme.proprietaryresource

Thank you!

- Access the OCF specifications
  https://openconnectivity.org/resources/specifications

- Contact OCF at admin@openconnectivity.org

References

Specification Location

Where can I find the specifications and Resource Type definitions?

OCF Specifications:

- https://openconnectivity.org/developer/specifications

Resource Type Definitions

- Core Resources: https://github.com/openconnectivityfoundation/core
- Core Extension Resources: https://github.com/openconnectivityfoundation/core-extensions
- Bridging Resources: https://github.com/openconnectivityfoundation/bridging
- Security Resources: https://github.com/openconnectivityfoundation/security-models
- Vertical Resources and Derived Models: https://oneiota.org/documents?filter%5Bmedia_type%5D=application%2Framl%2Byaml