OCF Semantic Interoperability Status
June 2017
OCF Core Framework

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
OCF Standard Resource Types

• Total of 74 Resource Types defined as of OCF 1.0
  • All OCF Resource Type IDs are IANA registered:
    http://www.iana.org/assignments/core-parameters/core-parameters.xhtml

• Each resource type definition contains:
  • Unique identifier (rt)
  • List of mandatory and optional properties per resource
  • Identification of the default interface and other supported interfaces
  • List of supported methods (create, read, update, …)
  • List per method the JSON schema defining the supported payload

Resources are specified in RESTful API Modelling Language (RAML) and Swagger2.0
Vendor extensions

• Vendor is allowed to:
  • Create own defined (non-OCF standardized) resources
  • Create own defined (non-OCF standardized) device types
  • Extend existing devices with additional (not mandated) resources
    • With standardized resource types
    • With vendor defined resource types

• All vendor extensions follow an OCF-defined naming scheme
OCF Bridge – Definition

- An OCF Bridge is a device that represents one or more non-OCF devices (bridged devices) as virtual OCF devices on the OCF network.
- The bridged devices themselves are out of the scope of OCF.
- The only difference between a ‘regular’ OCF device and a virtual OCF device is that the latter is encapsulated in an OCF Bridge device.
- An OCF Bridge device itself is indicated on the network with an “rt” of “oic.d.bridge”.

Diagram:
- OCF device (client)
- OCF Bridge device
- OCF Light device
- OCF Fan device
- Non OCF communication
- OCF communication
Bridging Concept – Bidirectional Operation

OCF Ecosystem

Partner Ecosystem

OCF Server

Partner Ecosystem Client

Bridge

OCF Client

Partner Ecosystem Server
OCF Bridging Specification

• Specifies a framework for bi-directional translation between devices in OCF and non-OCF ecosystems.

• Specifies general requirements for translation between OCF and non-OCF ecosystems
  • Requirements for resource discovery, message translation, security, and handling of multiple bridges.

• Specifies specific requirements for translation between OCF and AllJoyn ecosystems (other projects in progress include UPnP, OneM2M, etc.)
  • Requirements for mapping core resources, propagating errors, and algorithmically translating custom resource types.
  • Refers to OCF to AllJoyn Mapping specification for translating well-known resource types.
Bridging Security

• OCF Bridge needs to be a trusted entity as it translates message payloads.

• OCF Bridge itself and all virtual devices that it exposes must be onboarded and provisioned for secure operation.

• Each virtual device exposed by the OCF Bridge must implement the security requirements of the ecosystem that it is connected to.

• Bridging specifies mechanisms to selectively block communications between the OCF Bridge and OCF devices and between the OCF Bridge and bridged devices. This fine-grained control enables an administrator to control communications across ecosystems that may not have similar security capabilities.
Bridging Concept – Data Model

OCF Resource Model

Derived Model

Partner Ecosystem Resource Model

Resource Spec
Device Spec

Bridging Spec
Mapping Spec

Other Ecosystem Spec
Derived Models: e.g., AllJoyn

- Models the interworking between OCF and AllJoyn

- Defines the mapping in terms of:
  - Device Type equivalency
  - Resource <-> Interface equivalency
  - Detailed Property by Property mapping on a per Interface Basis (Derived Models)

- Syntax currently assumes OCF is superset model; so any Device Types and Resource Types (as equivalents to AllJoyn interfaces) that were missing from OCF were defined in the equivalent OCF Specifications.
• Derived models use standard JSON schema syntax. Fundamentally, derived models provide a conversion mapping between OCF data models and the data models in AllJoyn.

```json
"asa.environment.targethumidity": {
  "type": "object",
  "properties": {
    "targetvalue": {
      "type": "number",
      "description": "Measured value",
      "x-ocf-conversion": {
        "x-ocf-alias": "oic.r.humidity,oic.r.selectablelevels",
        "x-to-ocf": [
          "if minvalue != maxvalue, ocf.desiredhumidity = targetvalue;ocf.targetlevel = selectablehumiditylevels[0].",
          "if minvalue == maxvalue, ocf.targetlevel = targetvalue."
        ],
        "x-from-ocf": [
          "if x-ocf-alias == oic.r.humidity, targetvalue = desiredhumidity.",
          "if x-ocf-alias == oic.r.selectablelevels, targetvalue = targetlevel."
        ]
      }
    }
  }
}
```
OneIoTa Tool

- Web based (see: http://oneiota.org) development tool
- Supports RAML, JSON, and Swagger2.0 syntax
- Populated to date with all OCF Resources, Swagger2.0 versions of all such Resources, and OCF-AllJoyn derived models.
- Supports multiple organizations
  - Each submitting organization defines their own license terms
Introspection

• Why
  • On par with existing AllJoyn framework

• What
  • Device description is available on the network
  • Device description:
    • List all end points
    • Per end point
      – Which method are implemented
        » Query parameters per method
        » Payloads definitions (request and response)

• How
  • Put the data described in RAML and JSON on wire as a CBOR encoded Swagger2.0 document.
    • Describes the payload on JSON level
      – Property names
      – Type
      – range
Thank You

**OCF Specifications:**

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

**Resource Type Definitions**

- Core Resources: https://github.com/openconnectivityfoundation

- Vertical Resources and Derived Models: https://oneiota.org/documents?filter%5Bmedia_type%5D=application%2Framl%2Byaml