OCF 1.0 Candidate Release

Summary, Analysis, and Update from OIC1.1

Michael McCool
Intel

Osaka, W3C Web of Things F2F, 12 May 2017

Outline

- OCF 1.0 Draft Candidate Specification now publically available
- Summary of Changes
  - Major Change 1: Introspection and Data Models
    - Using OpenAPI (Swagger 2.0)
  - Major Change 2: Enhanced Security
- Preliminary ER Model
  - Need to converge on common notation and tooling with oneM2M, IoTschema, WoT ontology work
  - Need to formalize and encode as an RDF model
  - Need to validate with OCF
- WoT/OCF Interoperability Demonstration/Test Case
  - Smart Home Demo

Summary of Changes from OIC 1.1

- Introspection
  - Swagger 2.0 (OpenAPI) available from /oic/res/introspection
  - Meant to augment, not replace, other introspection capabilities (eg /oic/res) and data models
- Enhanced security
  - Alignment with IETF ACE and AllJoyn
  - Better specification of uses of certificates
  - Better management of onboarding and *offboarding* processes
  - Mandatory access control
  - System management (eg firmware updates)

Major Change 1: Introspection and Data Models

See pages 132-134, Section 11.8 of OCF Core Specification

- The intended usage of the Introspection Device Data is to enable “dynamic clients”.
  - Dynamically generate a generic “browser” UI
  - Dynamically create translations of the hosted Resources to another eco-system.
- Other usages of Introspection
  - Generate client code.
- Designed to *augment* the existing data already “on the wire”.
  - Existing mechanisms (eg /oic/res) need to be used to get a full overview of what is implemented in the Device.
  - For example, the Introspection Device Data does not convey information about which properties are observable, since that is already conveyed with the “p” property on the links in “/oic/res”

RAML vs. OpenAPI/Swagger

- Both designed for Web APIs, not IoT.
  - Neither handles Observables (Events), for instance
- RAML is based on YAML (but CAN be encoded in JSON)
- Swagger uses JSON-Schema (but CAN also use YAML)
  - However… choice to use Swagger for OCF introspection seems to be driven by some technical issues with encoding certain types in YAML as CBOR
  - OCF 1.0 specifies Swagger 2.0 for introspection, but implies upgrade to Swagger 3.0 in later revision
- For detailed comparisons (in the context of Web APIs), see:
  - http://modeling-languages.com/modeling-web-api-comparing/
  - http://nordicapis.com/top-specification-formats-for-rest-apis/

RAML vs. OpenAPI/Swagger

RAML

```
get:
  description: ...
  queryParameters:
    units:
      displayName: Units
      enum: ["C","F","K"]
  responses:
    200:
      body:
        application/json:
          schema: Temperature
          example: |
            {
              "rt":          ["oic.r.temperature"],
              "id":          "unique_example_id",
              "temperature":  20.0,
              "units":       "C",
              "range":       [0.0,100.0]
            }
```

OpenAPI/Swagger

```
{
  "/res": {
    "get": {
      "description": "...",
      "produces": [
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "...",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/res"
            }
          }
        }
      }
    }
  }
}
```

Major Change 2: Enhanced Security

Details are here…

- Property access
- Mandatory device state
- Software update
- Off-boarding
- ACE Resource matching
- CSR Resource
- Certificate format
- Use Directory Name Roles
- Role Certificates
- Mandatory ACLs
- ACE Subject Matching
- Randomized Identifier Onboarding
- SVR Arrays CRUD Query Behavior

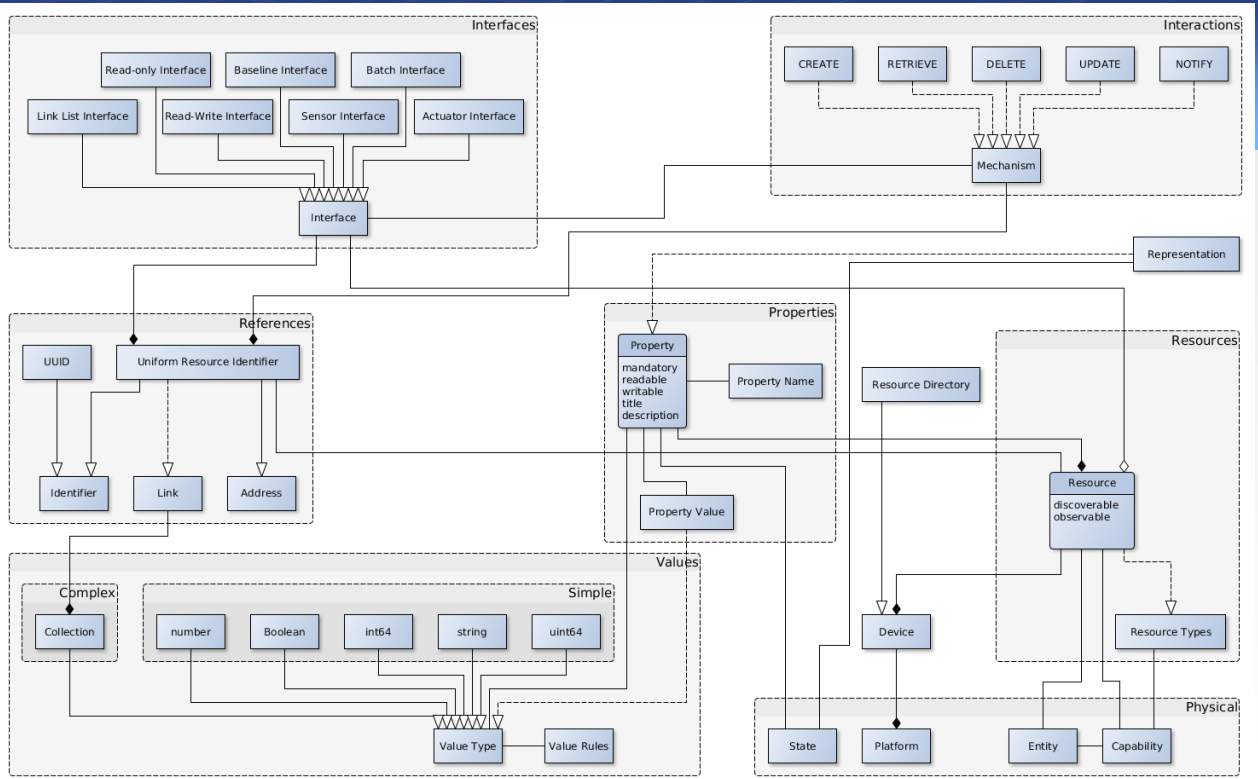
Other Changes

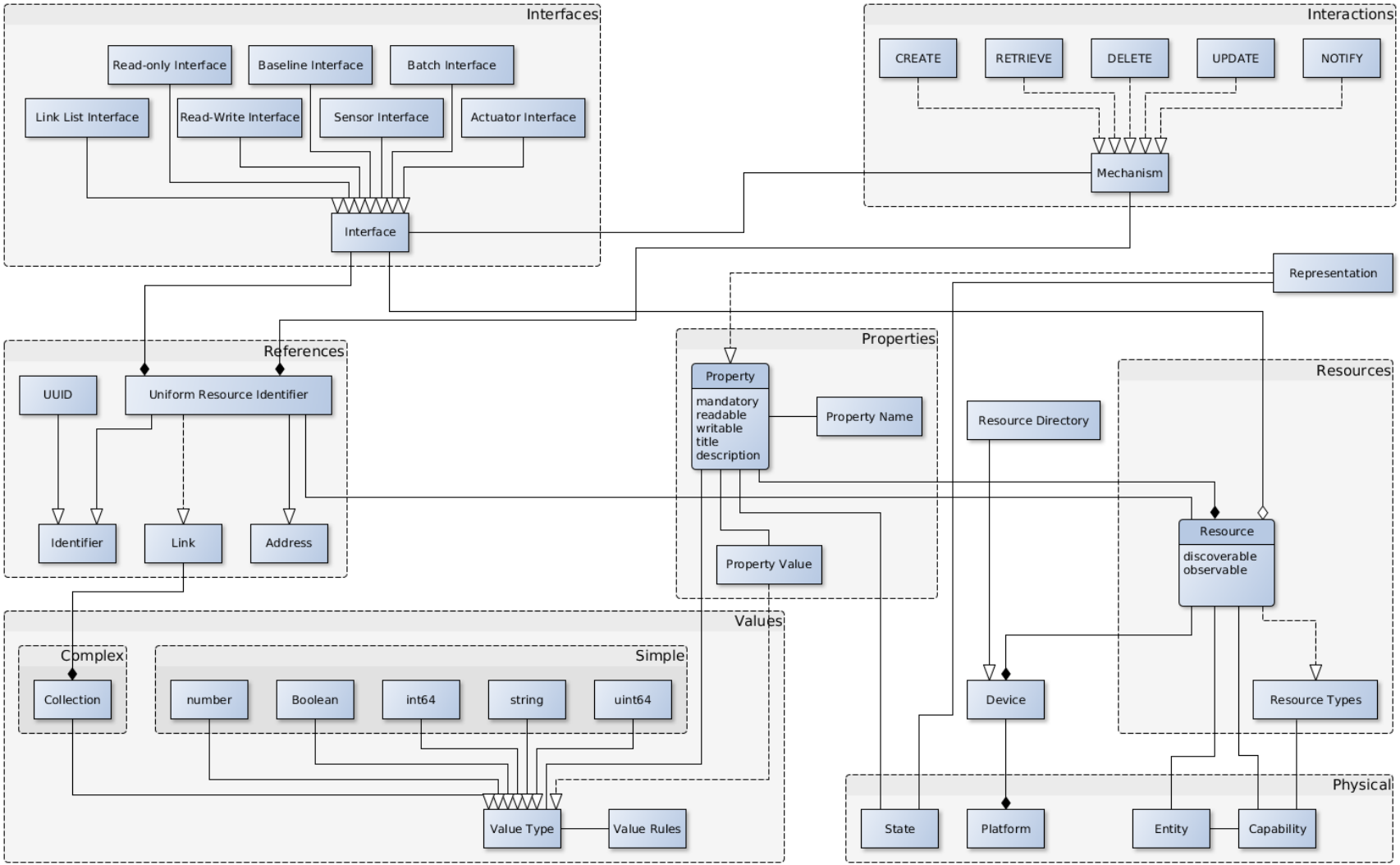
- AllJoyn Bridge
  - How to map to legacy AllJoyn devices (mappings of ASR resources)
- Smart Home Device Specification
  - Set of conventions and data models especially for “Smart Home” devices

OCF ER Model

Omitted/Incomplete/Wrong:

- Mappings from abstract mechanisms to concrete mechanisms
- Collections, links, scenes
- Introspection
  - New in OCF1.0, introspection resource is available to retrieve OpenAPI data model

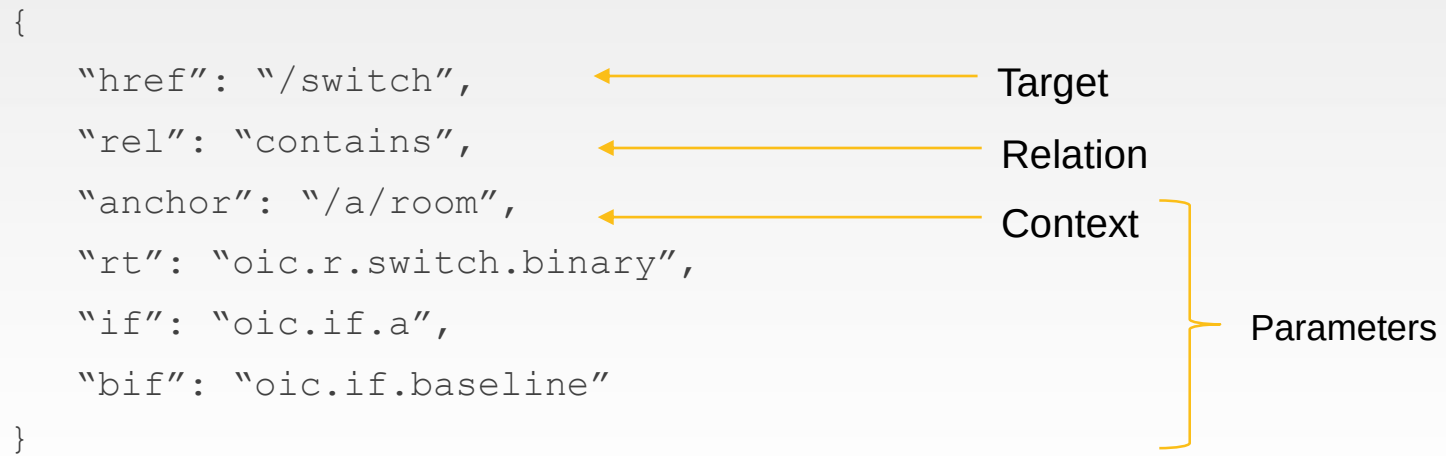



Interfaces
Read-only Interface
Baseline Interface
Batch Interface
Link List Interface
Read-Write Interface
Sensor Interface
Actuator Interface
Interface
Interactions
CREATE
RETRIEVE
DELETE
UPDATE
NOTIFY
Mechanism
Representation
References
UUID
Uniform Resource Identifier
Identifier
Link
Address
Properties
Property
mandatory
readable
writable
title
description
Property Name
Property Value
Resource Directory
Resources
Resource
discoverable
observable
Values
Complex
Collection
Simple
number
Boolean
int64
string
uint64
Value Type
Value Rules
Device
Resource Types
Physical
State
Platform
Entity
Capability

Issues with OCF ER Model

- Aggregate links should use 0..1 notation etc. rather than aggregation diamonds
  - Both for consistency and because it is easier to understand and lay out
- Relationships need to be labelled and categorized
- Links are incorrectly modelled right now
  - Actually have several additional fields besides URL href in OCF links: anchor, relationship, etc.
  - These are also currently not captured in the WoT ontology (which only has an href and a mediatype, and the latter is not given in an OCF link)
- Certain other aspects not modelled yet or not modelled well
  - Relationships between abstract CRUD-N mechanisms and concrete protocols (protocol bindings)
  - Client-Server “roles”
  - Scenes
  - Interfaces
- OCF model is actually based on CoRE
  - What are extensions specific to OCF, what are derived from CoRE? Should a version of the model also be upstreamed to CoRE?

OCF Links


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

WoT Links

```
{
  "href" : "coap://mytemp.example.com:5683/temp",
  "mediaType": "application/json"
}
```

Target

Media Type

Next Steps with OCF Model

- Converge Notation with oneM2M, IoTschema, etc.
  - UML-like notation seems to be common
  - Is there a formal definition anywhere?
- Formalize using RDF and define OCF ontology
  - Same notation, but with RDF behind the scenes defining an ontology
- Validate with OCF
  - Get feedback from OCF on accuracy of model
  - Perhaps even upstream and make it part of OCF specification…
  - Perhaps do something similar (validation, upstreaming) with a model for CoRE
- Mirror work done with oneM2M
  - Match OCF concepts with those in WoT ontology and define mappings from one to the other

OCF/WoT Interop Demonstrator Planning

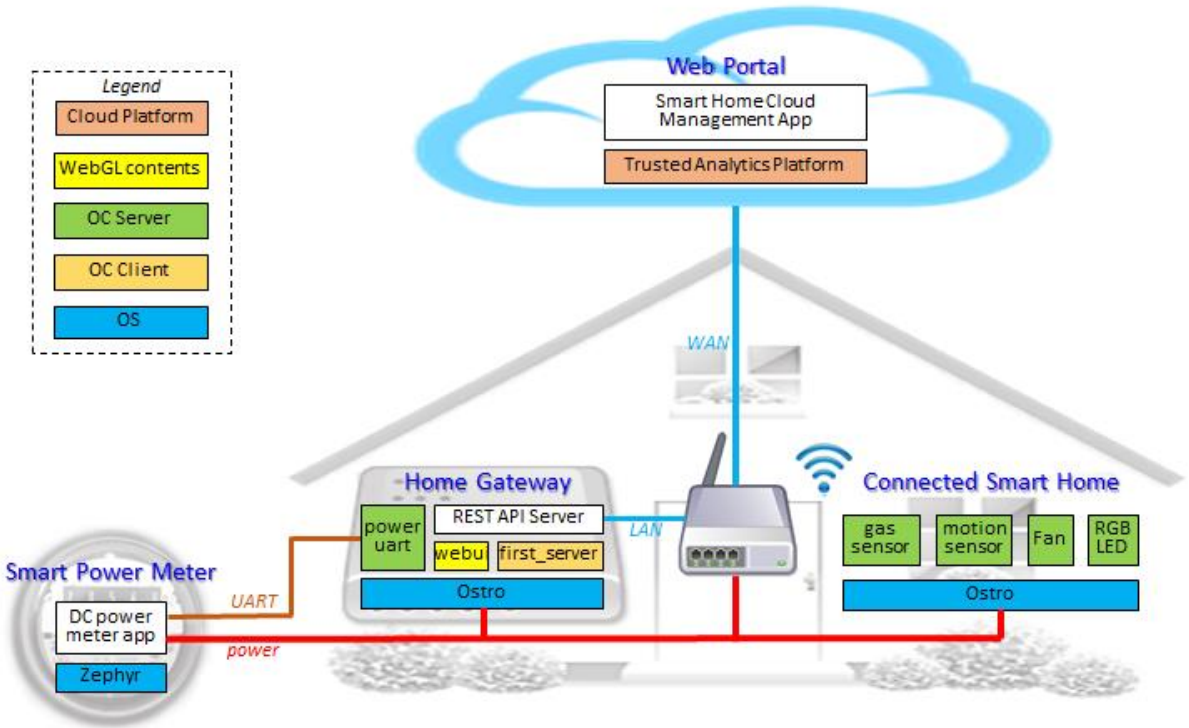
Need to demonstrate WoT system interoperating with OCF devices

- Select set of Simple OCF Devices to Use as a Test case
  - Smart Home demo good start, but…
  - Need something even simpler that can run with or without specialized hardware
  - Does not test certain things that are important, for example Collections, Links, Scenes, etc.
- Generate a Thing Description for the OCF Device(s)
  - First round: Manual generation
  - Second round: Automatic generation (if possible) from more specific Device models (eg from RAML/Swagger)
- Demonstrate Interoperability
  - Requires implementing some kind of protocol binding in a concrete implementation
  - Easiest place to start is with node-Iotivity and wot-node

OCF Smart Home

- Demonstrates multiple aspects and implementations of OCF: Iotivity-node, Iotivity-constrained, etc.
- Requires special hardware to run
- *Should* however be possible to convert to SW emulation (using QEMU for Zephyr component)

https://github.com/01org/Smart Home-Demo



Smart Home Demo Enhancements

- Convert demo to run in SW emulation
  - Give option for sensors and actuators to be replaced with socket data sources
  - Create “sensor emulations” to drive sensors and “actuator displays” to display actuator state
    - Eg Node.js process that presents a web interface
  - Upstream to OCF… enhances testing and demo capabilities
- Tweak demo to test things needed for WoT, trim extras
  - Add set up that supports multiple lights that can be treated as a collection, used with Scenes, etc.
  - Remove or make optional non-essential components (eg graphical UI)