IoT Extensions for schema.org
Community Teleconference

January 16, 2020

Agenda

- Agenda review
- Recent developments – Project CHIP
- OneDM mapping to iotschema – Semantic Proxy
- Admin and community items - Planning
- AOB

Project CHIP

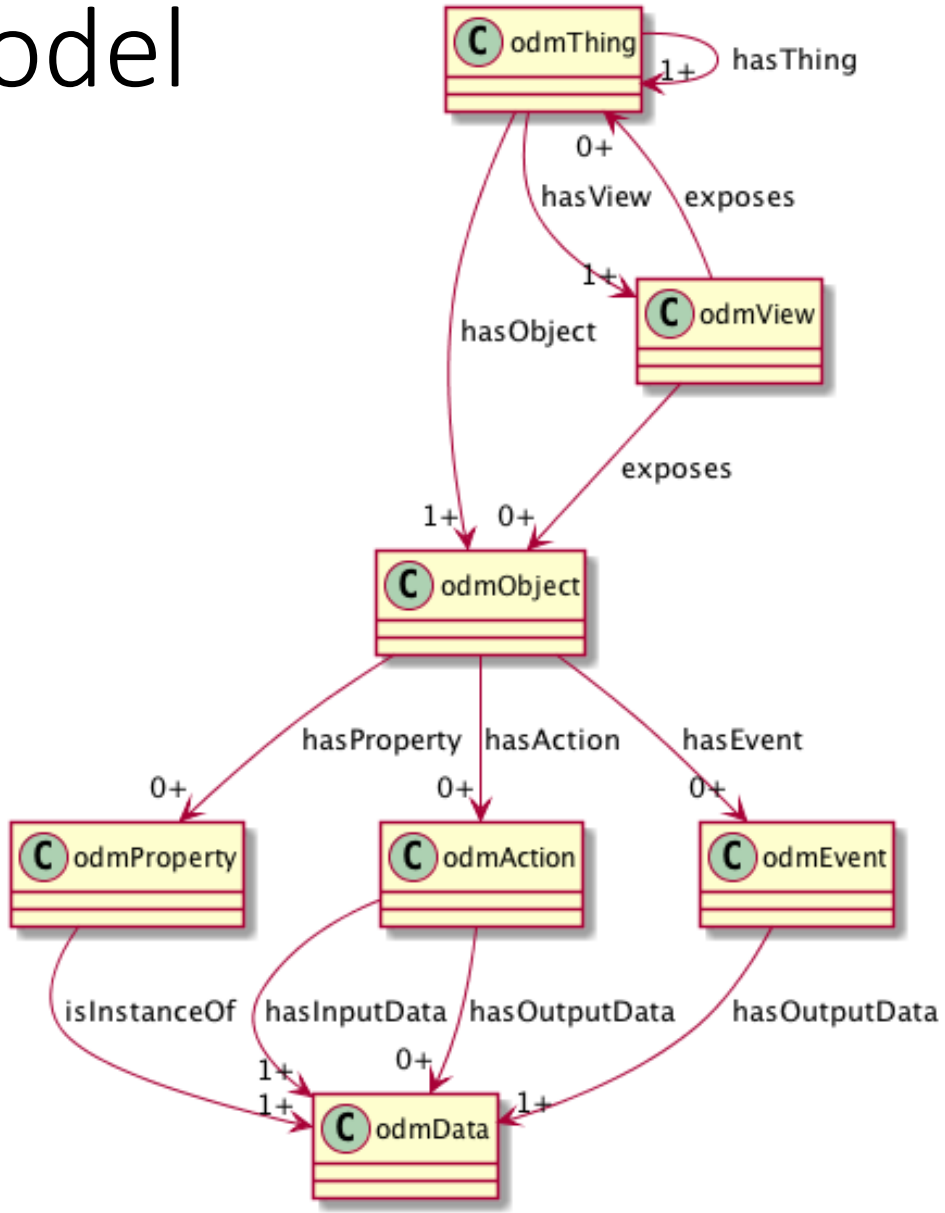
- Google and Apple joined Zigbee Alliance to create a new interoperable network standard for connected homes – Project Connected Home over IP
- Deliver a standard, open source reference stack, and certification program for interoperable devices
- What it means to iotschema - standardized data models for connected home devices + open source license

Semantic Proxy

- A Proxy for Semantic Interoperability
  - Proxy to enable translation from device protocol to application protocol
  - Provides for many-to-many mapping of application protocols to device protocols
  - many-to-one and one-to-many through a common semantic model
  - Could implement a "universal" IoT gateway

ODM Meta-Model

- Thing Class to compose Objects
- View (Interface) Class to virtualize affordances
- Reusable Objects
  - Property, Action, and Event Affordances
- Reusable Data Types



Meta Operations of Abstract Affordances

- Property
  - value = Read(), Write(value)
- Action
  - status response(s) = Invoke(parameters)
- Event
  - event occurrence responses = Subscribe()

Semantic Proxy – Protocol Binding

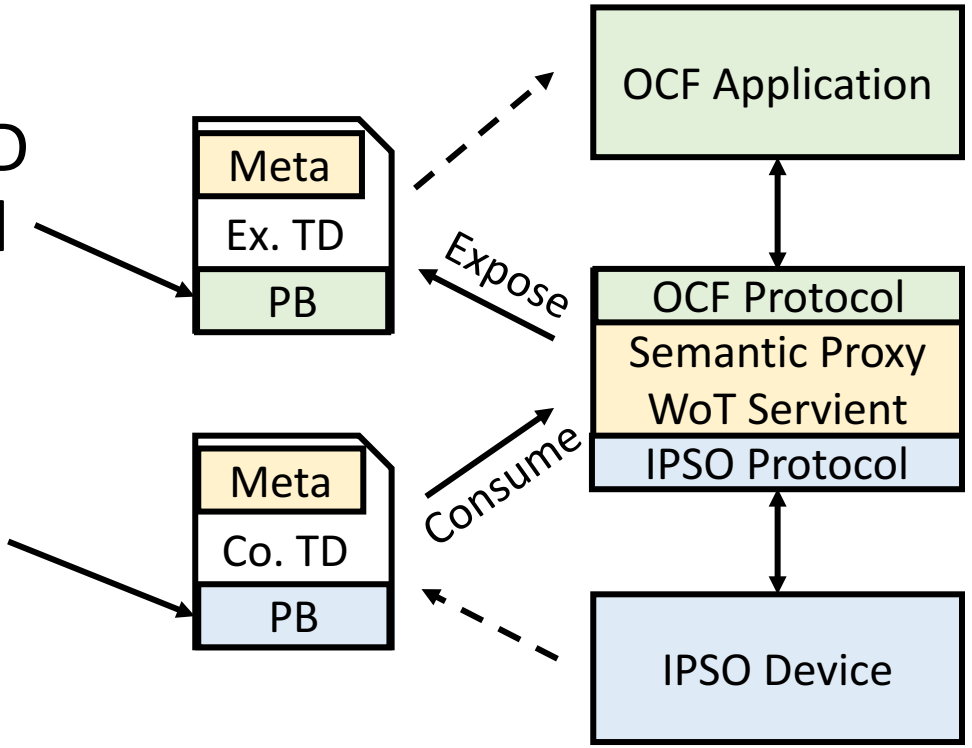
- Uses a common semantic model to connect applications to things over diverse network protocols and communication patterns
- Proxy maps the meta-model operations to network messages in the target protocol using protocol bindings
- Flavors of REST, Pub/Sub, RPC messages
- Example using W3C Web of Things Architecture

Semantic Proxy – W3C Web of Things Integration

- "Thing Description" associates semantic identifiers for Properties, Actions, and Events with affordance descriptions consisting of data schemas and protocol bindings
- Protocol bindings associate network operations with meta-operations in the semantic model
- "Incoming" Consumed TD and "Outgoing" Exposed TD have the same affordances in the semantic model, and customized data schemas and protocol bindings

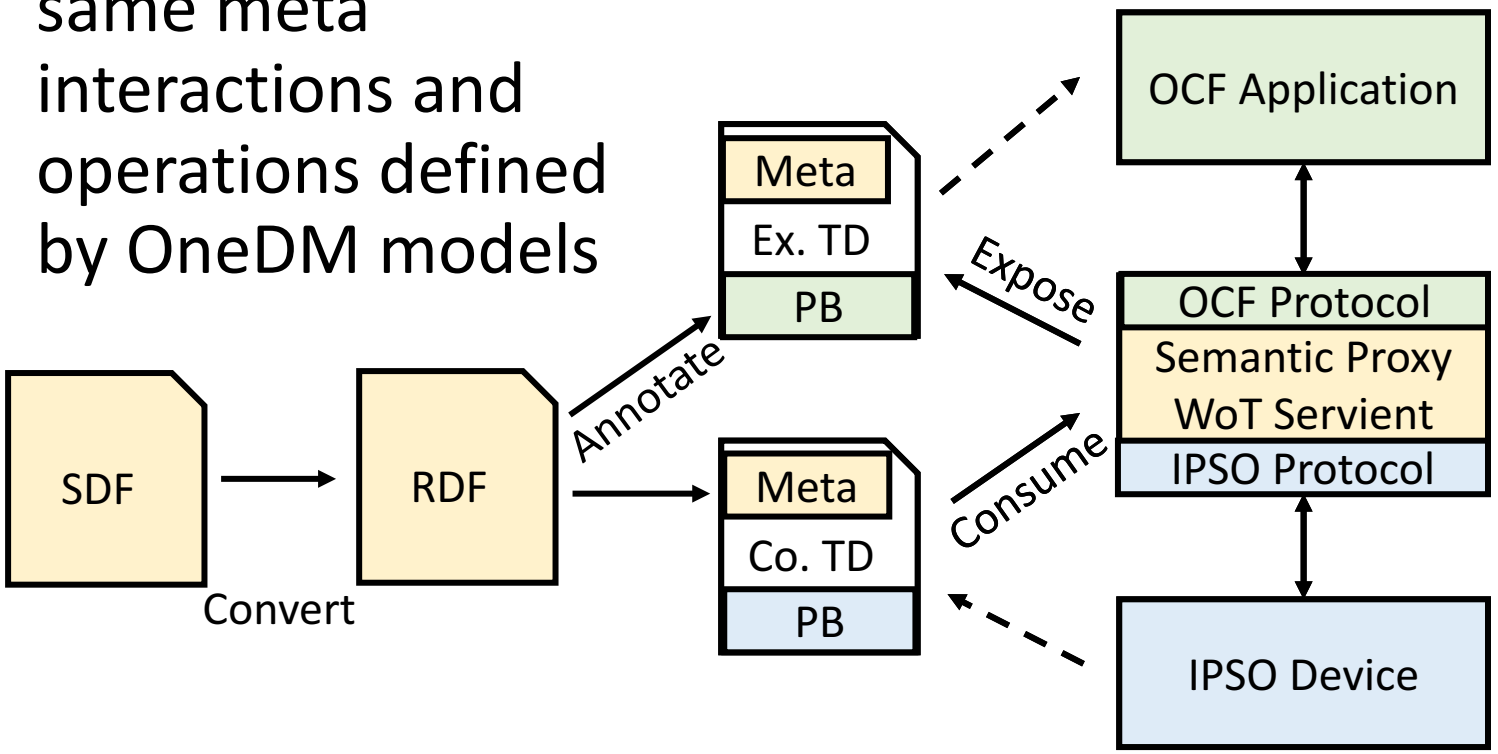
Semantic Proxy - Schematic

- Exposed Thing TD has OCF protocol binding
- Consumed Thing TD has IPSO protocol binding



Semantic Proxy - Schematic

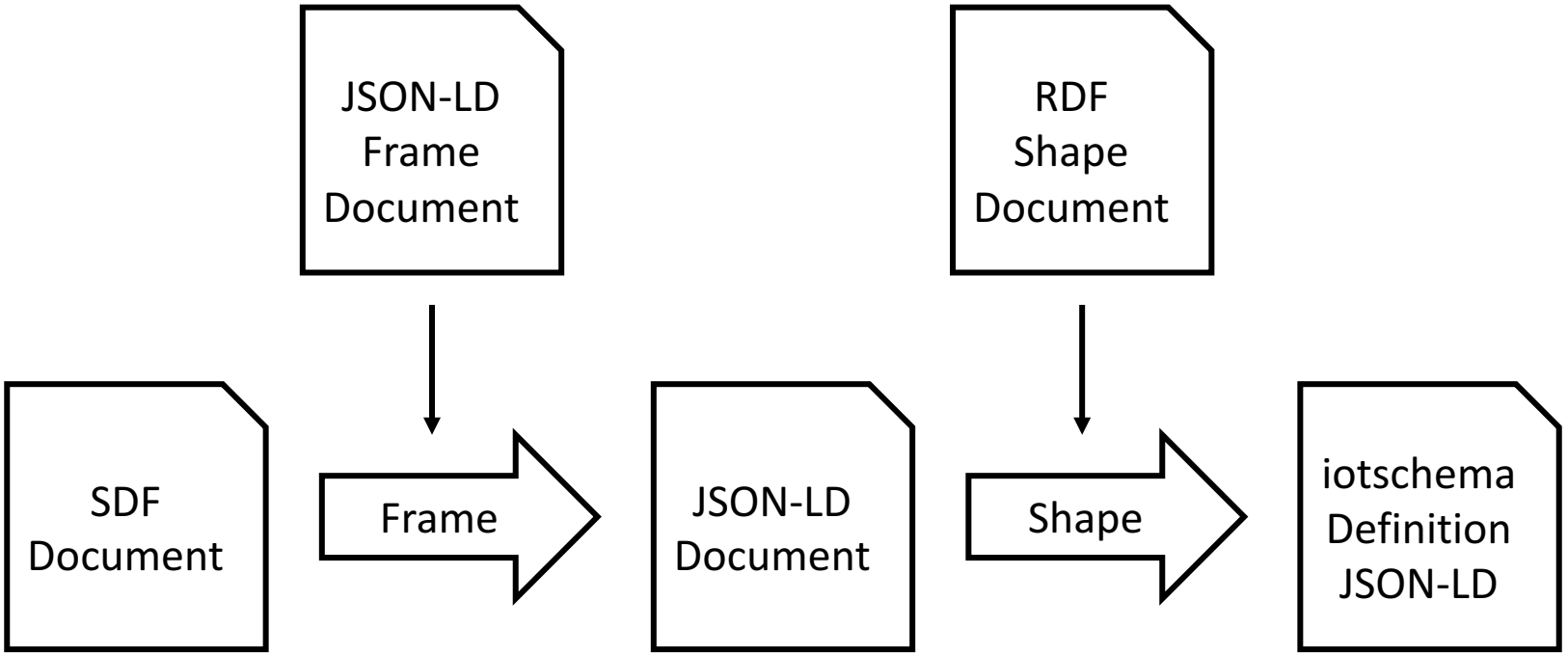
- Both TDs have the same meta interactions and operations defined by OneDM models



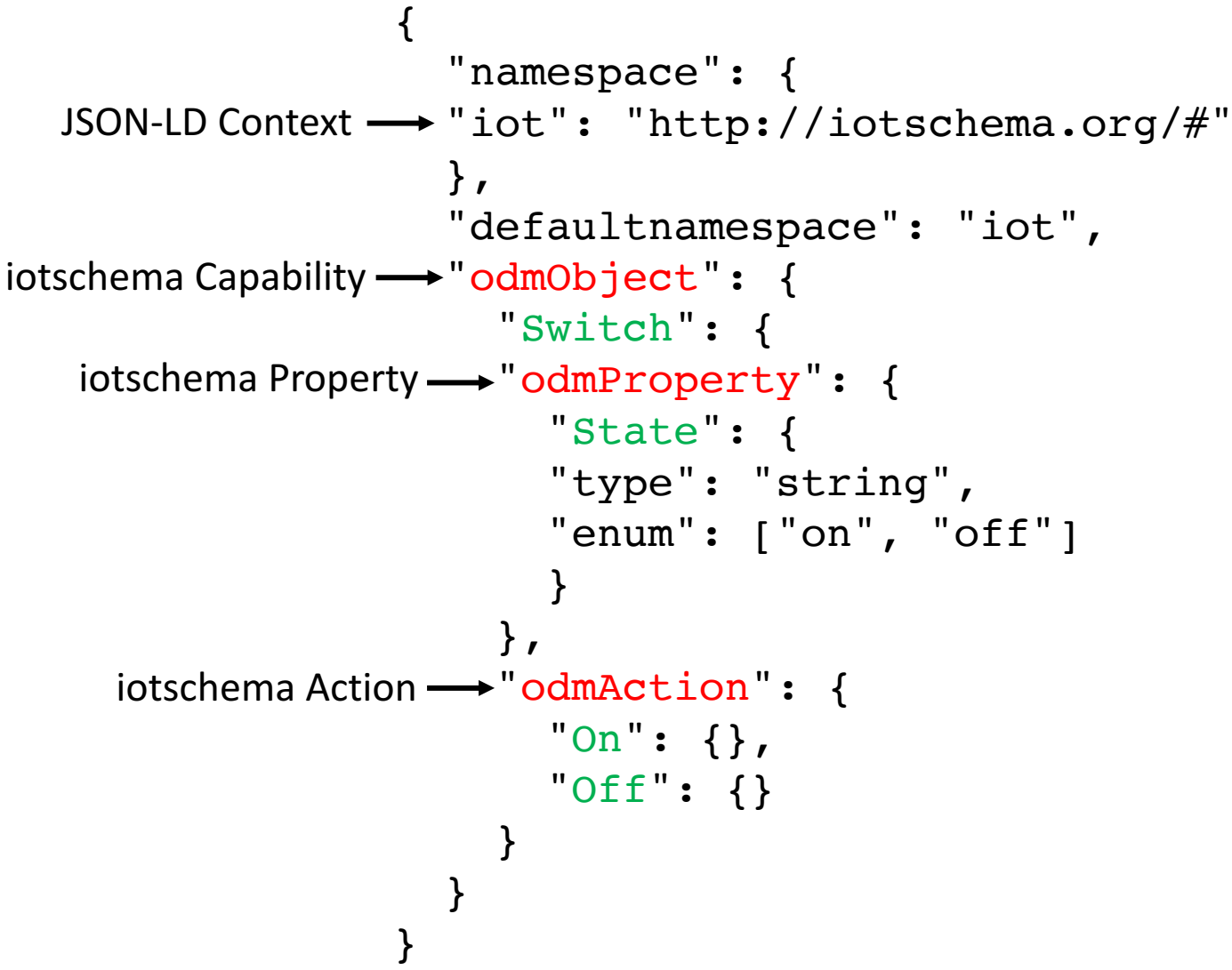
Semantic Proxy – RDF Converter

- WoT Thing Description can use iotschema definitions for annotation
  - WoT TD only has Thing and affordance (P/A/E) classes
- iotschema style RDF definitions are aligned with the OneDM SDF meta-model
- Create RDF statements from OneDM definitions for use in semantic tooling
- odmObject maps to iotschema Capability
- odmThing and odmView don't directly map but can extend iotschema Capability

Convert SDF Documents to an iotschema style Definitions



OneDM SDF Example Mapping




```
{
  "namespace": {
    "iot": "http://iotschema.org/#"
  },
  "defaultnamespace": "iot",
  "odmObject": {
    "Switch": {
      "odmProperty": {
        "State": {
          "type": "string",
          "enum": ["on", "off"]
        }
      },
      "odmAction": {
        "On": {},
        "Off": {}
      }
    }
  }
}
```

Expected Result – Object maps to Capability

```
{
  "@id": "iot:SwitchCapability",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchCapability",
  "rdfs:subClassOf": {
    "@id": "iot:Capability"
  },
  "iot:providesInteractionPattern": [
    {
    "@id": "iot:SwitchStateProperty",
    "@id": "iot:SwitchOnAction",
    "@id": "iot:SwitchOffAction"
    }
  ]
}
```

Expected Result - Actions

```
{
  "@id": "iot:SwitchOnAction",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchOnAction",
  "rdfs:subClassOf": {
    "@id": "iot:Action"
  }
},
{
  "@id": "iot:SwitchOffAction",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchOffAction",
  "rdfs:subClassOf": {
    "@id": "iot:Action"
  }
}
```

Properties and Data Types

```
{
  "@id": "iot:SwitchStateProperty",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateProperty",
  "rdfs:subClassOf": {
    "@id": "iot:Property"
  },
  "iot:providesOutputData": {
    "@id": "iot:SwitchStateData"
  }
},
{
  "@id": "iot:SwitchStateData",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateData",
  "rdfs:subClassOf": {
    "@id": "schema:PropertyValue"
  },
  "schema:propertyType": {
    "@id": "schema:String"
  }
}
```

Expected Result - Enums

- How do we describe enums in iotschema style RDF?
- How do we map enum values in RDF (which are URIs) to concrete representation values in schemas?
  - `[true, false], [1, 0], ["on", "off"]`
- Could use schema annotation, like @type but for enum values

```
"enum": [
  {"iot:Switch.State.On": "on"},
  {"iot:Switch.State.Off": "off"}
]
```

Enum Data Item and Type in RDF

```
{
  "@id": "iot:SwitchStateData",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateData",
  "rdfs:subClassOf": {
    "@id": "schema:PropertyValue"
  },
  "schema:propertyType": {
    "@id": "iot:SwitchStateValue"
  }
}
{
    "@id": "iot:SwitchStateValue",
    "@type": "rdfs:Class",
    "rdfs:comment": "Enumeration of Switch states",
    "rdfs:label": "SwitchStateType",
    "rdfs:subClassOf": {
      "@id": "schema:Enumeration"
    }
},
```

Enum Data Values in RDF

```
{
    "@id": "iot:SwitchOnState",
    "@type": "iot:SwitchStateValue",
    "rdfs:comment": "Switch On State",
    "rdfs:label": "SwitchOnState"
},
{
    "@id": "iot:SwitchOffState",
    "@type": "iot:SwitchStateValue",
    "rdfs:comment": "Switch Off State",
    "rdfs:label": "SwitchOffState"
}
```

Enum Data Values in RDF

```
{
    "@id": "iot:SwitchOnState",
    "@type": "iot:SwitchStateValue",
    "rdfs:comment": "Switch On State",
    "rdfs:label": "SwitchOnState",
    "schema:propertyType": {"@id": "schema:String"}
    "schema:defaultValue": {"rdfs:Literal": "on"}
},
{
    "@id": "iot:SwitchOffState",
    "@type": "iot:SwitchStateValue",
    "rdfs:comment": "Switch Off State",
    "rdfs:label": "SwitchOffState"",
    "schema:propertyType": {"@id": "schema:String"}
    "schema:defaultValue": {"rdfs:Literal": "off"}

}
```

TD Semantic Annotation

iotschema Definition	TD @type annotation
Capability	iot:SwitchCapability
Property	iot:SwitchStateProperty
Data Type	iot:SwitchStateData
Property Value	iot:SwitchOnState, iot:SwitchOffState
Action	iot:SwitchOnAction, iot:SwitchOffAction
OneDM Definition	**TD @type annotation**
Object	odm:odmObject/Switch
Property	odm:odmObject/Switch/odmProperty/State
Data Type	odm:odmData/SwitchData
Property Value	odm:odmData/SwitchData/enum/0, odm:odmData/SwitchData/enum/1
Action	odm:odmObject/Switch/odmAction/On, odm:odmObject/Switch/odmAction/Off

Semantic Proxy – WoT TD


```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    {"iot": "http://iotschema.org/"}
  ],
  "@type": [ "Thing", "iot:SwitchCapability" ],
  "properties": {
    "SwitchStateProperty": {
      "@type": ["iot:SwitchStateProperty", "iot:SwitchStateData"],
      "type": "string",
      "enum": ["on", "off"],
      "writeOnly": false,
      "readOnly": false,
      "observable": true,
      "forms": [
        {
         "href": "/SwitchCapability/properties/SwitchStateProperty",
         "op": ["readproperty", "writeproperty", "observeproperty"],
         "contentType": "application/json"
        }
      ]
    }
  },
```

Semantic Proxy – WoT TD

```
"actions": {
  "SwitchOnAction": {
    "@type": ["iot:SwitchOnAction"],
    "input": {},
    "forms": [
      {
        "href": "/SwitchCapability/actions/SwitchOnAction",
        "op": ["invokeaction"],
        "contentType": "application/json"
      }
    ]
  },
  "SwitchOffAction": {
    "@type": ["iot:SwitchOffAction"],
    "input": {},
    "forms": [
      {
        "href": "/SwitchCapability/actions/SwitchOffAction",
        "op": ["invokeaction"],
        "contentType": "application/json"
      }
    ]
```

Semantic Proxy – Semantic API

- Names of affordances are resolved through Semantic Discovery
  - PropertyName=discover(FilterParameters)
- Applications use meta-model affordances and operations
  - data=readProperty(PropertyName)
  - writeProperty(PropertyName, data)
  - result=invokeAction(ActionName, parameters)
  - data=subscribeEvent(EventName)
- Supports modular, declarative programming models – Node-RED

References

One Data Model SDF and Model work in progress

- https://github.com/one-data-model/language
- https://github.com/one-data-model/playground

Semantic Proxy and W3C WoT

- https://github.com/tum-ei-esi/virtual-thing
- https://www.w3.org/TR/2019/CR-wot-thing-description-20191106/
- https://www.w3.org/TR/2019/CR-wot-architecture-20191106/

Expected Result – Object maps to Capability (.)

```
{
  "@id": "iot:Capability.Switch",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchCapability",
  "rdfs:subClassOf": {
    "@id": "iot:Capability"
  },
  "iot:providesInteractionPattern": [
    {
    "@id": "iot:Capability.Switch.Property.State",
    "@id": "iot:Capability.Switch.Action.On",
    "@id": "iot:Capability.Switch.Action.Off"
    }
  ]
}
```

Expected Result – Actions (.)

```
{
  "@id": "iot:Capability.Switch.Action.On",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchOnAction",
  "rdfs:subClassOf": {
    "@id": "iot:Action"
  }
},
{
  "@id": "iot:Capability.Switch.Action.Off",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchOffAction",
  "rdfs:subClassOf": {
    "@id": "iot:Action"
  }
}
```

Properties and Data Types (.)

```
{
  "@id": "iot:Capability.Switch.Property.State",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateProperty",
  "rdfs:subClassOf": {
    "@id": "iot:Property"
  },
  "iot:providesOutputData": {
    "@id": "iot:Capability.Switch.Property.State.Data"
  }
},
{
  "@id": "iot:Capability.Switch.Property.State.Data",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateData",
  "rdfs:subClassOf": {
    "@id": "schema:PropertyValue"
  },
  "schema:propertyType": {
    "@id": "schema:String"
  }
}
```

Enum Data Item and Type in RDF (.)

```
{
  "@id": "iot:Capability.Switch.Property.State.Data",
  "@type": "rdfs:Class",
  "rdfs:label": "SwitchStateData",
  "rdfs:subClassOf": {
    "@id": "schema:PropertyValue"
  },
  "schema:propertyType": {
    "@id": "iot:Capability.Switch.Property.State.Data.Value"
  }
}
{
    "@id": "iot:Capability.Switch.Property.State.Data.Value",
    "@type": "rdfs:Class",
    "rdfs:comment": "Enumeration of Switch states",
    "rdfs:label": "SwitchStateValue",
    "rdfs:subClassOf": {
      "@id": "schema:Enumeration"
    }
},
```

Enum Data Values in RDF (.)

```
{
    "@id": "iot:Capability.Switch.Property.State.Data.Value.On",
    "@type": "iot:Capability.Switch.Property.State.Data.Value",
    "rdfs:comment": "Switch On State",
    "rdfs:label": "SwitchOnState"
},
{
    "@id": "iot:Capability.Switch.Property.State.Data.Value.Off",
    "@type": "iot:Capability.Switch.Property.State.Data.Value",
    "rdfs:comment": "Switch Off State",
    "rdfs:label": "SwitchOffState"
}
```

iotschema Review: Status and Planning

September 2019

Contents

- Charter and Objectives – What we started out to do
- Status – What we have accomplished
- W3C WoT integration – Test case and results
- Schema.org integration – Proposal and issues
- W3C Community Group – IPR policy and continuity
- OneDM Liaison Group – Report out
- Going forward – a proposal

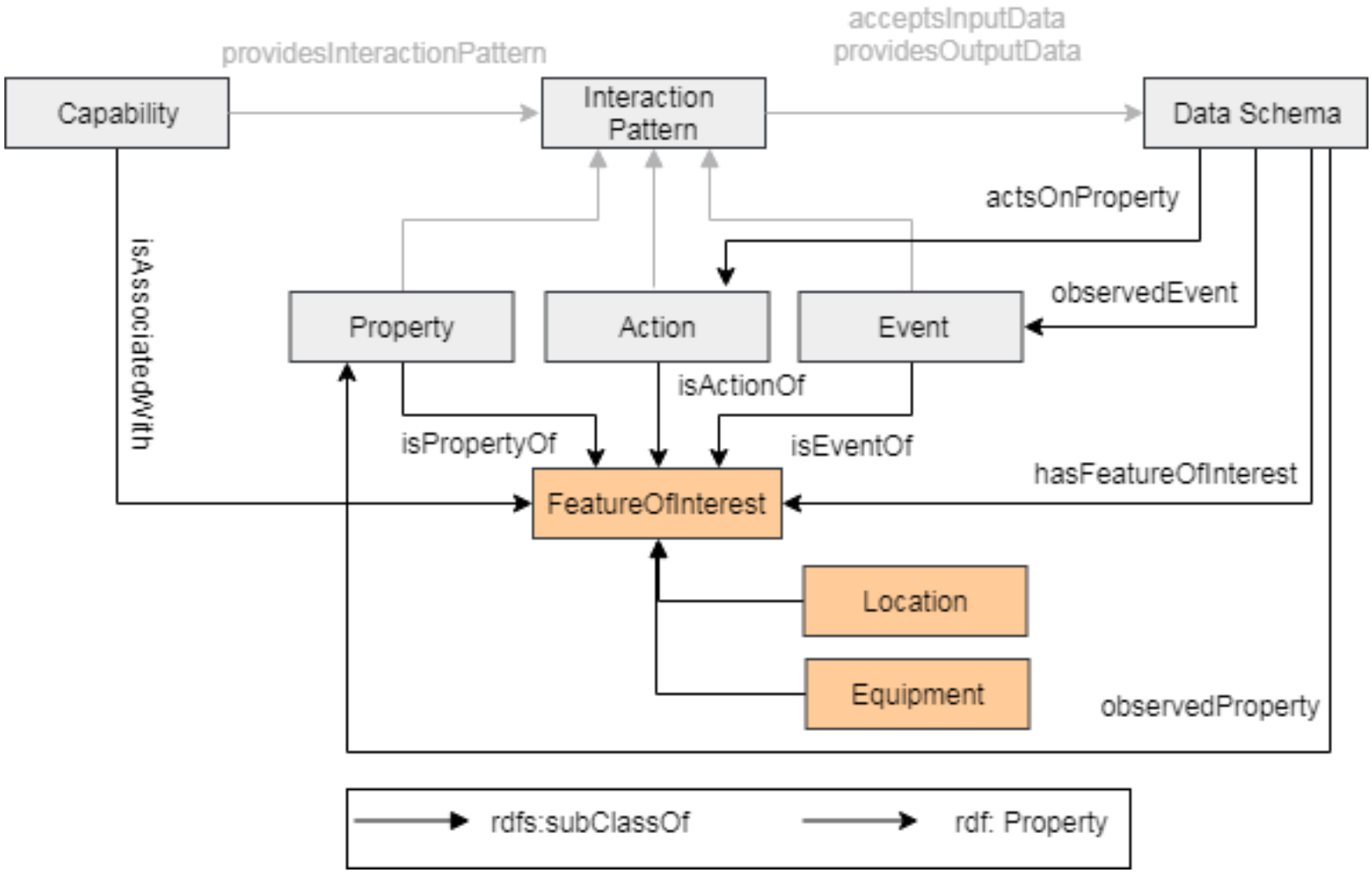
Charter and Objectives

- Create an extension to Schema.org that will enable IoT semantics
- Work with relevant vendors and SDOs to architect a common information model for IoT
- Develop a process to enable free contribution from diverse organizations, driving toward normalization to a common model
- Enable domain and subject experts to create and maintain definitions and models
- Work on practical integration with Schema.org

Status

- Developed a common meta-model which is well adopted in industry (graphic)
  - Functional/semantic capabilities with common Property, Action, and Event classes of affordances
  - Feature of Interest integration from other namespaces – GENIVI/VSS, BrickSchema, Project Haystack
  - Integration with W3C SOSA/SSN
- Published strawman definitions for some common IoT capabilities
- Conducted test case evaluation with W3C Web of Things and IRTF T2TRG WISHI including hands-on

Information Meta-Model



W3C WoT Test Case

- Integrated iotschema definitions with WoT Thing Description as semantic annotation (example)
- Created strawman definitions for common use
- Investigated use in discovery and configuration
- Several organizations used semantic annotation
- Node-RED application for semantic interoperability
  - Uses WoT Thing Description with iotschema annotation
- WoT discovery will be further developed

iotschema for Node-RED
Recipe-based applications

- iotschema embedded in Node-RED tool
  - Enables an easy configuration of things using iotschema definitions
- Easies the use of semantics for IoT developers
  - No need for a developer to know RDF(S), JSON-LD, RDF Shapes ...
- Simplify creation of applications with W3C WoT
  - Avoids translations of serializations formats, data types, units ...
- Demonstrates semantic discovery and processing
  - Integrates WoT Thing Directory
- GitHub project location:
  - https://github.com/iot-schema-collab/iotschema-node-red

Schema.org Integration

- Class names Event, Action, Property conflict
- iotschema has diverse semantic types for objects, schema.org has diverse property types
- Property types could be synthesized from objects but…
- iotschema will potentially define hundreds of types for physical quantities (temperature, humidity, voltage, acceleration…), control affordances (open/close, brightness, color control, camera controls, operating modes…), and features of interest (rooms, machines…)

Schema.org Integration

- The WoT use case is based on annotation consisting of RDF @type statements that point to URIs of defined terms for specialized types that conform to the classes in the meta-model
- These meta-model classes would only add about 6 new property types to schema.org
  - iotCapability, iotEvent, iotAction, iotProperty, iotData, iotFeatureofInterest
  - new types like iotInterface, iotThing, etc. as needed

Schema.org Integration

- There is a potential example pattern in schema.org
  - MedicalEntity, with about 7 property types
- Likewise, an instance of IoT Schema would contain some set of iotCapability, iotAction, iotProperty, iotEvent, iotFeatureOfInterest properties
- Specialization of iot types would happen at the next level in the graph
  - URIs that point to accepted specialized definitions in a different namespace (?)
  - lighting controls, thermostats, etc. that conform to the base types but have their own properties

Schema.org Integration

- The base properties line up with semantic classes that have software handlers,e.g. to discover, display, control
- The specialization information enables selection of application elements that can understand and interoperate with the specialized affordances, for example lighting controls
- More general applications might reason about semantic constructions based on their ability to parse the semantic graph
- Adaptation software may need special properties to describe data types, units, and ranges

W3C Community Group

- Started early 2019
- A few members have joined but not active yet
- IPR policy for contributions based on CG membership
- CG membership will become part of the community and be required for contributors and participants
- Can we adopt the BSD 3-Clause license for our contributed and published content?

OneDM Liaison Group

- Semi-private group of SDOs and associated vendors
- Goal is to create a common IoT data model and normalized device definitions
- Creating open source JSON language and developer tools for definitions, using BSD 3-Clause license
- High level alignment with the Property-Action-Event meta-model
- OneDM definitions can feed iotschema
- We can share OneDM language and tools

Going Forward

- Create an experimental area on schema.org for normalized, accepted iotschema content
- Create a namespaced area per contributing org in the public github, allow open contribution of raw content
  - CI tools validate the contributed definitions
- The definition can immediately be dereferenced in the contributor's namespace (on schema.org?)
- Move definitions into the official github repository and schema.org experimental area when they are accepted

To Do

- Integrate the iotschema W3C Community Group into the process, define licensing and publication
- Design a namespace scheme for contributing orgs and for linked ontologies
- Refactor the vocabulary as necessary to remediate the name conflicts
- Build and deploy CI tools to check contributions

iotschema: Resources

- W3C Community Group:

  The Schema Extensions For IoT

- https://www.w3.org/community/iotschema/

- Web site:

  Current location

- http://iotschema.org/docs/full.html

- GitHub repository:

- https://github.com/iot-schema-collab/iotschema

  Teleconferences:

- https://github.com/iot-schema-collab/teleconferences

  Contributions:

- https://github.com/iot-schema-collab/iotschema

  Charter:

- https://github.com/iot-schema-collab/ws-charter

- Tools:

  iotschema for Node-RED

- https://github.com/iot-schema-collab/iotschema-node-red