One Data Model

Update for W3C WoT WG

August 21, 2019

One Data Model

- Liaison group consisting of SDOs and device vendors
- Zigbee, OCF, OMA, Google, Comcast, Amazon…
- Drive to a common set of data models for device definitions across SDOs and vendors
- Focus initially on smart home devices
- Emerged from the Hive meeting fall 2018
- Weekly meetings since February
- Three face to face meetings

One Data Model – high level process

- Initial focus is developing a common language
- Pressure test the language with definitions from the participating SDOs
- Improve the language as needed
- Create definitions for non-controversial device models taken from most likely SDOs
- Work out a process by which common models can be agreed upon where models overlap

One Data Model - Status

- A working definition of the language exists
- Simple Definition Format – SDF
- Markdown document and JSON Schema define the language
- Pressure testing in process
- 30+ github issues being addressed

SDF Language

- SDF is plain JSON with JSON Schema (v7) constructions
- For creating portable definitions of devices
- Property/Action/Event meta model
- Like iotschema but high detail features like bitfields
- Superset approach to accommodating SDO design patterns
- High level composition features
- Depends on separate protocol binding

SDF Language - Composition

- odmObject class corresponds to iotschema capability
- odmObject contains a set of Properties, Actions, and Events related to a specific function
- odmThing class to group a related set of Objects that work together
- odmProduct and odmView for specific compositions
- anyOf and allOf constructs at the odmThing level

SDF Language – Data Typing

- odmData class for data type definitions
- JSON Schema plus additional constraints
- subType for e.g. uint8, bytestring
- widthInBits to define constraints and bitfields
- JSON Schema predefined formats like datetime
- odmData has semantic typing e.g. temperatureData
- Only data type constraints are defined, not payload schemas; however, structured types are allowed

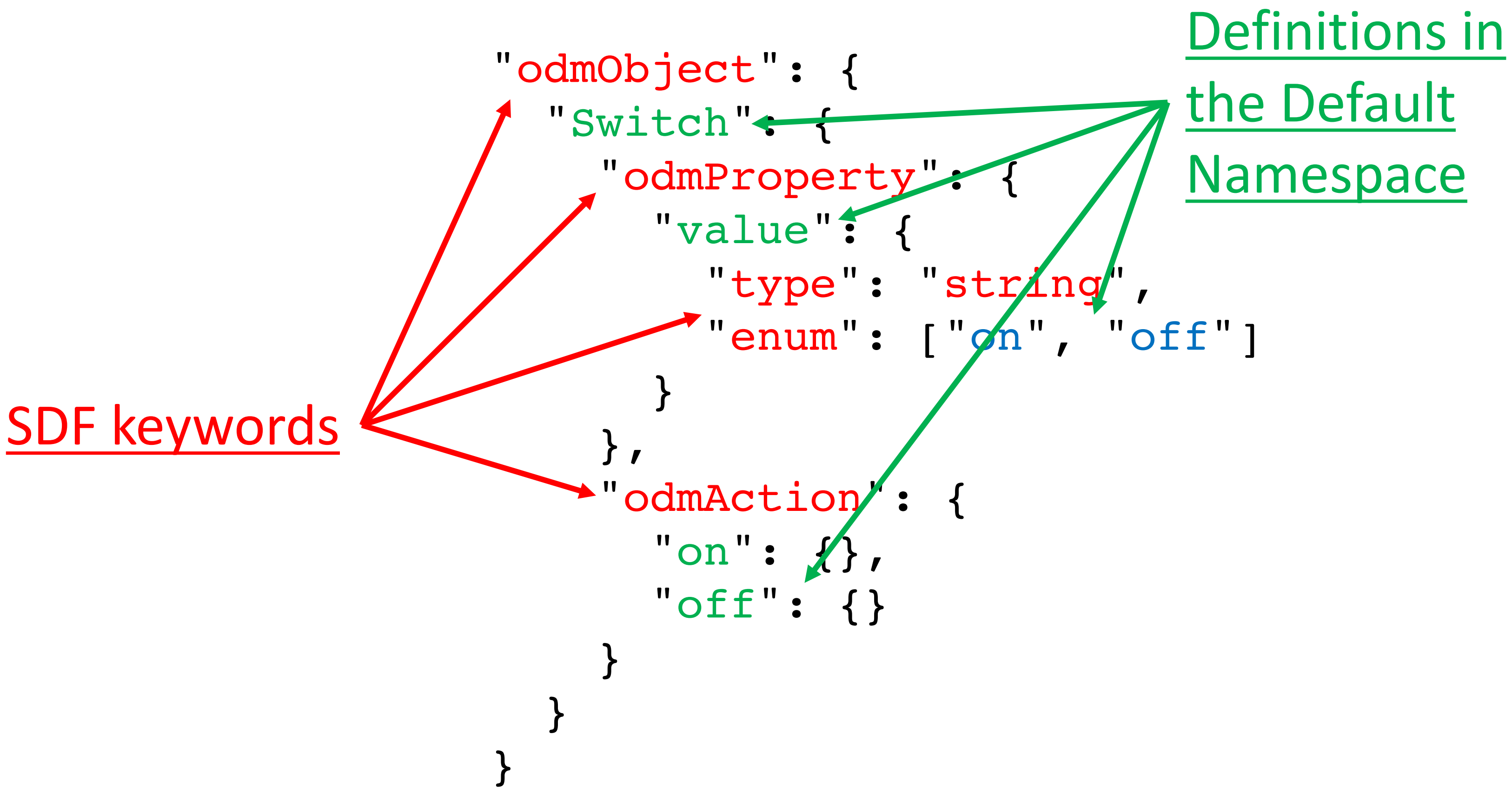
SDF Language - References

- SDF has a simple namespace model that works like JSON-LD
- Prefixes are defined for URIs and used in names with a colon ':' separator
- A default namespace allows names without prefixes
- JSON Pointer is used to refer to elements in a document
- definition => ocf : http://example.com/ocf#
- use => $ref: ocf:/odmData/temperatureData

SDF - Example

```
{
  "info": {
    "title": "Example file for ODM Simple JSON Definition Format",
    "version": "20190404",
    "copyright": "Copyright 2019 Example Corp. All rights reserved.",
    "license": "http://example.com/license"
  },
  "namespace": {
    "st": "http://example.com/capability/odm"
  },
  "defaultNamespace": "st",
  "odmObject": {
    "Switch": {
      "odmProperty": {
        "value": {
          "type": "string",
          "enum": ["on", "off"]
        }
      },
      "odmAction": {
        "on": {},
        "off": {}
      }
    }
  }
}
```

Definitions



```
"odmObject": {
  "Switch": {
    "odmProperty": {
      "value": {
        "type": "string",
        "enum": ["on", "off"]
      }
    },
    "odmAction": {
      "on": {},
      "off": {}
    }
  }
}
```

Definitions

- A definition consists of a defined term and a map of it's defined qualities

```
"value": {
  "type": "string",
  "enum": ["on", "off"]
}
```

SDF Documents

- SDF format description document

https://github.com/one-data-model/language/blob/master/sdf.md

- JSON Schema for validation and conversion

https://github.com/one-data-model/language/blob/master/sdf-schema.json

- SDF Examples

https://github.com/mjkoster/ODM-Examples/tree/master/examples

OneDM SDF FAQ

OneDM is for creating standard and reusable type definitions
- targeted at device vendors and existing devices
- for domain experts to maintain, reuse, and extend

OneDM definitions are similar in scope to iotschema definitions
- address some additional requirements like numeric identifiers
- adds extensions for composition into product types

OneDM definitions are semantically aligned with WoT TD and iotschema
- Property, Action, Event model
- Additional class for Objects, similar to the iotschema Capability class
- Thing class, View Class, and Product class for composing Objects together
- use of JSON schema fragments for data constraint definitions

SDF is plain JSON
- tools for mapping the SDF model into other models, including RDF
- will be used as a developer front-end for iotschema

OneDM SDF FAQ

SDF does not define payloads (by practice)
- use of JSON schema to define data element constraints and data types
- WoT Thing Description or protocol binding defines payload schemas
- SDF defined terms are used as application semantics (@type statements) for TD

SDF uses external protocol bindings
- WoT Thing Description provides bindings for common protocols and formats
- also OCF, LWM2M, Zigbee, Bluetooth for specialized ecosystems
- Some interest in a standard compact binding for OneDM network interoperability -> single constrained device ecosystem

Vendors and SDOs in OneDM will create definitions that can be used for standard application semantics
- TD annotation using OneDM defined Properties, Actions, and Events
- iotschema format will be available for the same definitions