# SDF 1.1 Gaps and Proposed Features

October 12, 2020

# Gaps in modeling of data types

- Enumerations, quantized categories
  - Semantic (URI) anchor for each selection
  - sdfEnum
- Reusable sdfProperty definitions
  - Quantities, units, ranges, application semantics
  - As used in OMA LWM2M
- Composite sdfData and sdfProperty
  - Semantic and named compositetypes for sdfProperty, equivalent to multiple data items for sdfAction, sdfEvent
- Overloaded data item: range + enum codes
  - Zigbee ZCL example
- https://github.com/one-data-model/SDF/issues

# Enumerations and quantized categories

- Semantic (URI) anchor for each selection
- Description field in the schema for each selection
- Flexible, binding-time mapping to representation type
- Extensible through loose coupling – mapping files
- JSON pointer to each selection for mapping files
- Like to preserve the JSON-ness of the current system e.g. mapping table

# Feature: sdfEnum

- New Class for sdfData and sdfproperty

```
"#/sdfData/OnOffState/sdfEnum/off"
"sdfData": {
  "OnOffState": {
    "sdfEnum": {
      "on" : {
        "description": "power on state"
      },
      "off": {
        "description": "power off state"
      }
    }
  }
}
```

# Reusable sdfProperty

- Define once and use in many objects
- Semantic reusability
- Can define sdfProperty that uses sdfRef to point to a sdfProperty definition
- The pattern is used in IPSO Smart Objects
- Needed for conversion to equate sdfProperty class
- Schema change

# Composite sdfProperty

- Semantic and named composite types for sdfProperty and sdfData
- Equivalent to multiple data items for sdfAction, sdfEvent
- Semantic tagging and description of elements
- JSON pointer addressable elements for mapping
- Optionality constructs e.g. anyOf, sdfRequired
- Data schema could be part of the protocol binding

# Feature: sdfData

- Could use the inline "sdfData" mechanism we already provide for sdfAction and sdfEvent

```
"sdfProperty": {
  "simple1" : {
    "type": "string"
  },
  "compound1": {
    "sdfData": {
      "element1": { "type": "number" },
      "element2": { "type": "string" }
    }
  }
}
```

# Overloaded data element

- Some values of an analog setting reserved for enum
- Zigbee example of startup parameter for light brightness reserves a few values for settings to "restore last brightness", "set to minimum brightness", etc.
- Preferred pattern for mapping this design?
- Example has both range and enum with exclusive values, should include a "anyOf" construct to select range vs. enum


```
"sdfProperty": {
  "StartUpCurrentLevel": {
    "label": "StartUpCurrentLevel",
    "anyOf": [
      {
        "type": "number",
        "minimum": 1,
        "maximum": 254
      },
      {
        "sdfEnum": {
          "MinimumDeviceValuePermitted": {
            "const": 0
          },
          "SetToPreviousValue": {
            "const": 255
          }
        }
      }
    ]
  }
}
```