

A Simple JSON Format for ODM Definitions

Michael Koster

April 4, 2019

Overview

- Simple definition format for the ODM ontology
 - Object
 - Event, Action, Property
 - Data types
- Defined namespaces using curie notation
- Flat definition space with cross-references
- Can be single file or multiple (e.g. separate data type definitions)
- Examples of SmartThings, ZCL, and OCF definitions

Simple example – Header Part

<https://github.com/mjkoster/ODM-Examples/SDF-example.json>

keywords

File Information

```
"info": {  
  "title": "Example file for ODM Simple JSON Definition Format",  
  "version": "20190404",  
  "copyright": "Copyright 2019 Xcorp, Inc. All rights reserved.",  
  "license": "http://example.com/license"  
},
```

curies resolved

```
"namespace": {  
  "odm": "http://onedm.example.org/vocab/core",  
  "js": "http://onedm.example.com/vocab/jschema",  
  "st": "http://smarththings.example.com/capability/odm"  
},  
"defaultnamespace": "odm",
```

Definition Part

Default curie
resolves to
"odm:type", etc.

References link to
definitions

```
"define": {  
  "st:Switch": {  
    "type": "Object",  
    "hasProperty": "st:Switch.value",  
    "hasAction": [  
      "st:Switch.on",  
      "st:Switch.off"  
    ],  
  },  
  "st:Switch.value": {  
    "type": "Property",  
    "hasData": "st:Switch.valueData"  
  },  
  "st:Switch.on": {  
    "type": "Action"  
  },  
  "st:Switch.off": {  
    "type": "Action"  
  },  
  "st:Switch.valueData": {  
    "type": "Data",  
    "js:type": "string",  
    "js:enum": ["on", "off"]  
  }  
}
```

ZCL Example

<https://github.com/mjkoster/ODM-Examples/SDF-ZCL.json>

- More complex definition, some Action parameters defined with Data definitions

ZCL Example

```
"define": {  
  "zcl:OnOff": {  
    "type": "Object",  
    "hasProperty": "zcl:onoff.onoff",  
    "hasAction": [  
      "zcl:OnOff.On",  
      "zcl:OnOff.Off",  
      "zcl:OnOff.Toggle"  
    ]  
  },  
  "zcl:OnOff.OnOff": {  
    "type": "Property",  
    "hasData": "zcl:onoff.OnOffData",  
    "readable": true,  
    "writeable": false,  
    "required": true  
  },  
  "zcl:OnOff.OnOffData": {  
    "type": "Data",  
    "js:type": "js:boolean",  
    "js:default": false  
  }  
}
```

OCF Example

<https://github.com/mjkoster/ODM-Examples/SDF-OCF.json>

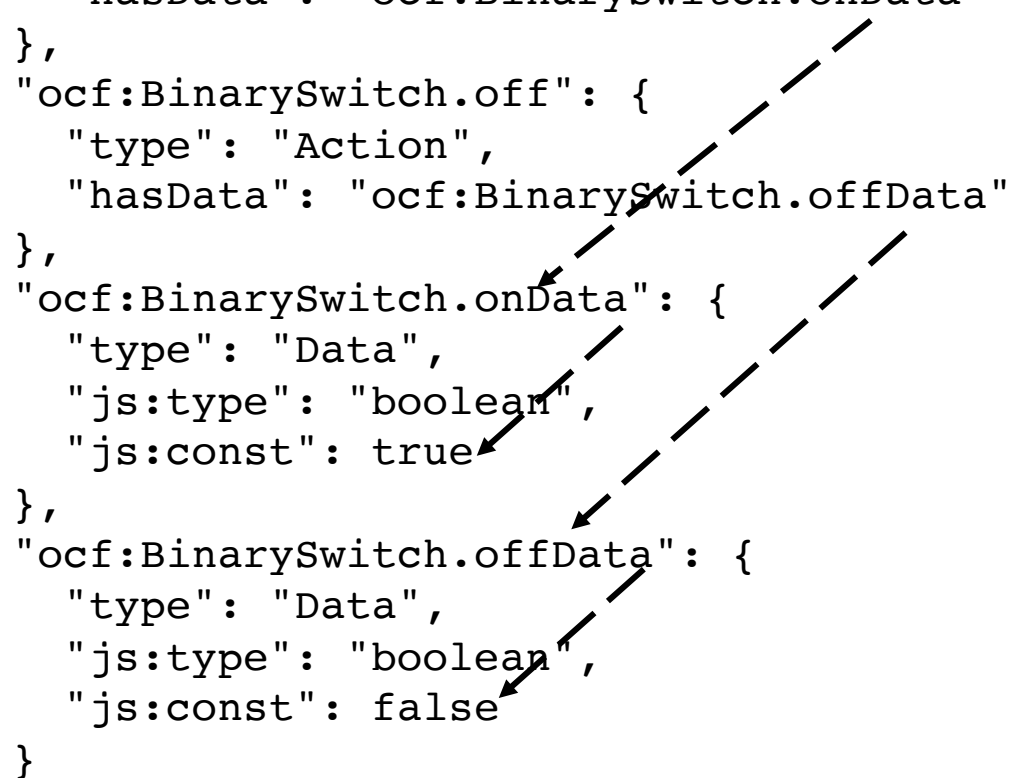
- ODM Object mapped to OCF Resource Type
- ODM Property mapped to OCF Property
- Action definitions added
- Actions supply values for Properties e.g. on=true, off=false

OCF Example

```
"define": {  
  "ocf:BinarySwitch": {  
    "type": "Object",  
    "hasProperty": "ocf:BinarySwitch.value",  
    "hasAction": [  
      "ocf:BinarySwitch.on",  
      "ocf:BinarySwitch.off"  
    ]  
  },  
  "ocf:BinarySwitch.value": {  
    "type": "Property",  
    "hasData": "ocf:BinarySwitch.valueData"  
  },  
  "ocf:BinarySwitch.valueData": {  
    "type": "Data",  
    "js:type": "boolean"  
  },  
}
```


OCF Example – Define Actions

```
"ocf:BinarySwitch.on": {  
  "type": "Action",  
  "hasData": "ocf:BinarySwitch.onData"  
},  
"ocf:BinarySwitch.off": {  
  "type": "Action",  
  "hasData": "ocf:BinarySwitch.offData"  
},  
"ocf:BinarySwitch.onData": {  
  "type": "Data",  
  "js:type": "boolean",  
  "js:const": true  
},  
"ocf:BinarySwitch.offData": {  
  "type": "Data",  
  "js:type": "boolean",  
  "js:const": false  
}  
}
```



Alternate Syntax

- Simpler for developers?
- Easier to process?
- Process using JSON Schema?

Alternate Syntax #2

```
{
  "object": {
    "st:Switch": {}
  },
  "property": {
    "st:Switch.value": {
      "js:type": "string",
      "js:enum": ["on", "off"]
    }
  },
  "action": {
    "st:Switch.on": {},
    "st:Switch.off": {}
  },
  "event": {},
  "data": {}
}
```

With One Target Namespace

```
{
  "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://smarththings.example.com/capability/odm"
  },
  "object": {
    "Switch": {}
  },
  "property": {
    "Switch.value": {
      "type": "string",
      "enum": ["on", "off"]
    }
  },
  "action": {
    "Switch.on": {},
    "Switch.off": {}
  }
}
```

Alternate Syntax #3

```
{
  "object": {
    "name": "st:Switch"
  },
  "property": [
    {
      "name": "st:Switch.value",
      "js:type": "string",
      "js:enum": ["on", "off"]
    }
  ],
  "action": [
    { "name": "st:Switch.on" },
    { "name": "st:Switch.off" }
  ],
  "event": [],
  "data": []
}
```