Thing-to-Thing Research Group K. Lynn
Internet-Draft L. Dornin
Intended status: Informational Verizon Labs
Expires: August 25, 2016 February 22, 2016

Modeling RESTful APIs with JSON Hyper-Schema
draft-lynn-t2trg-model-rest-apis-00

Abstract

This document explores JSON Hyper-Schema as a method of modeling Internet of Things (IoT) systems that follow the principles of the Representational State Transfer (REST) architectural style.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 25, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
2. Overview of JSON Schema and Hyper-Schema 3
3. Examples 3
4. IANA Considerations 8
5. Security Considerations 8
6. References 8
Appendix A. Future Work 9
Authors' Addresses 9

1. Introduction

The central problem in an IoT domain such as home control might be characterized as "translating intention into configuration". The challenge is to translate a high level goal such as "turn off all the lights on the first floor", expressed in a natural language, into action. An agent trying to accomplish this goal might look up "light" in a semantic registry, discover instances of "light", select those instances that fall within a given area, etc., and ultimately transmit control packets to devices that match the given criterea.

A user defining such a goal should not be concerned with the underlying technology, i.e., details of the protocols or data links over which the control packets are sent. Similarly, service providers or developers wishing to provide home control solutions would generally prefer them to be technology agnostic. As with other translation tasks, this can more easily be accomplished by using one or more intermediate abstraction layers.

Recent research in semantic interoperability in heterogeneous environments has underscored the need for semantic alignment at various levels [Semantic-Interop]. This document is concerned with identifying a fairly primitive abstraction that sufficiently models target state and behavior without overly specifying the details of the underlying technology. We proprose to explore REST APIs as a reasonable layer at which to abstract devices.

JSON Hyper-Schema [I-D.luff-json-hyper-schema] is a formalism for describing RESTful APIs. It supports a description of inbound and outbound data across the interface, together with link descriptions that identify the URIs, link-relations, and methods that apply to links. JSON Hyper-Schema therefore supports the principal of Hypertext as the Engine of Application State. It may also serve as input to documentation and code generation tools.

2. Overview of JSON Schema and Hyper-Schema

JSON Schema is a JSON based format for defining the structure of JSON data [RFC7159][I-D.zyp-json-schema]. JSON Hyper-Schema adds hyperlink- and hyper-media related keywords to JSON Schema. This section simply lists features of JSON Schema and Hyper-Schema used in the examples. For a detailed overview, see [Understanding-JSON-Schema]

2.1. JSON Schema

JSON Schema supports:

- JSON data types: object, array, number, string, boolean, null
- $schema keyword: value identifies meta-schema and version
- Definitions and JSON references promote fragment reuse
- Schema composition keywords: oneOf, anyOf, allOf
- Patterns, regular expressions, more...

2.2. JSON Hyper-Schema

JSON Hyper-Schema adds Link Description Objects which include:

- href: URI template
- rel: link relation
- title: a title for the link
- targetSchema: JSON Schema describing the link target
- mediaType: media type describing the link target
- method: REST method that applies to this link
- encType: media type of the request
- schema: Schema describing the data sent with the request

3. Examples

The following examples may be validated at http://json-schema-validator.herokuapp.com/. The first example may be converted to markdown using the prmd tool at https://github.com/interagent/prmd.

3.1. Binary Switch Hyper-Schema

This JSON Hyper-Schema models a generic device on/off capability.

```
{
  "$schema": "http://json-schema.org/draft-04/hyper-schema#",
  "id": "http://example.com/schemata/switch-binary#",
  "description": "A simple API for a device that supports on/off.",
  "type": [
    "object"
  ],
  "definitions": {
    "uuid": {
      "description": "Unique identifier of the device",
      "example": "01234567-89ab-cdef-0123-456789abcdef",
      "format": "uuid",
      "readOnly": true,
      "type": [
        "string"
      ]
    },
    "identity": {
      "anyOf": [
        {
          "$ref": "#/definitions/uuid"
        }
      ]
    },
    "invalidated": {
      "description": "Time resource was invalidated",
      "example": "2015-01-01T12:00:00Z",
      "format": "date-time",
      "readOnly": true,
      "type": [
        "string"
      ]
    },
    "updated": {
      "description": "Time resource was last updated",
      "example": "2015-01-01T12:00:01Z",
      "format": "date-time",
      "readOnly": true,
      "type": [
        "string"
      ]
    },
    "SwBinary": {
      "title": "Binary Switch",
```

```
      "description": "Used to control devices with On/Off capability.",
      "stability": "prototype",
      "type": [
        "object"
      ],
      "definitions": {
        "SetValue": {
          "description": "0..99 (level) or 255 (on)",
          "example": 50,
          "type": "number",
          "multipleOf": 1,
          "oneOf": [
            {
              "minimum": 0,
              "maximum": 99
            },
            {
              "enum": [
                255
              ]
            }
          ]
        },
        "GetValue": {
          "description": "0 (off) or 255 (on)",
          "example": 255,
          "type": "number",
          "multipleOf": 1,
          "oneOf": [
            {
              "enum": [
                0,
                255
              ]
            }
          ]
        }
      },
      "properties": {
        "invalidated": {
          "$ref": "#/definitions/invalidated"
        },
        "updated": {
          "$ref": "#/definitions/updated"
        },
        "Value": {
          "$ref": "#/definitions/SwBinary/definitions/GetValue"
        }
```

```
      },
      "links": [
        {
          "title": "Set",
          "description": "Update a specific Binary Switch instance.",
          "href": "/id/{(%23%2Fdefinitions%2Fidentity)}/SwBinary/Set",
          "method": "POST",
          "rel": "update",
          "schema": {
            "type": [
              "object"
            ],
            "properties": {
              "Value": {
                "$ref": "#/definitions/SwBinary/definitions/SetValue"
              }
            },
            "required": [
              "Value"
            ],
            "strictProperties": true
          }
        },
        {
          "title": "Get",
          "description": "Read a specific Binary Switch instance.",
          "href": "/id/{(%23%2Fdefinitions%2Fidentity)}/SwBinary/Get",
          "method": "GET",
          "rel": "self",
          "targetSchema": {
            "$ref": "#/definitions/SwBinary"
          }
        }
      ]
    }
  },
  "properties": {
    "SwBinary": {
      "$ref": "#/definitions/SwBinary"
    }
  },
  "additionalProperties": false,
  "links": [
    {
      "href": "https:/",
      "rel": "self"
    },
    {
```

```
      "href": "/dev-schema",
      "method": "GET",
      "rel": "self",
      "targetSchema": {
        "additionalProperties": true
      }
    }
  ]
}
```

3.2. JSON Link-Format Document

With the addition of the required "rel" property to each Link Description Object, the link-format example from section 2.4 of [I-D.ietf-core-links-json] becomes a valid JSON Hyper-Schema document.

```
[
  {
    "href": "/sensors",
    "ct": "40",
    "title": "SensorIndex",
    "rel": "self"
  },
  {
    "href": "/sensors/temp",
    "rt": "temperature-c",
    "if": "sensor",
    "rel": "self"
  },
  {
    "href": "/sensors/light",
    "rt": "light-lux",
    "if": "sensor",
    "rel": "self"
  },
  {
    "href": "http://www.example.com/sensors/t123",
    "anchor": "/sensors/temp",
    "rel": "describedby"
  },
  {
    "href": "/t",
    "anchor": "/sensors/temp",
    "rel": "alternate"
  }
]
```

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed upon publication as an RFC.

5. Security Considerations

This document doesn't define new functionality and therefore doesn't introduce new security concerns. However, security considerations from related specifications apply:

- JSON security: section 12 of [RFC7159]

6. References

6.1. Normative References

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

6.2. Informative References

[I-D.luff-json-hyper-schema] Luff, G., Zyp, K., and G. Court, "JSON Hyper-Schema: Hypertext definitions for JSON Schema", draft-luff-json-hyper-schema-00 (work in progress), January 2013.

[I-D.zyp-json-schema] Galiegue, F., Zyp, K., and G. Court, "JSON Schema: core definitions and terminology", draft-zyp-json-schema-04 (work in progress), January 2013.

[I-D.ietf-core-links-json]
Li, K., Rahman, A., and C. Bormann, "Representing CoRE Formats in JSON and CBOR", draft-ietf-core-links-json-04 (work in progress), November 2015.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation , University of California, Irvine , 2000, <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[Understanding-JSON-Schema]
Droettboom, M., "Understanding JSON Schema, Release 1.0", Space Telescope Science Institute , February 2015, <http://spacetelescope.github.io/understanding-json-schema/>.

[Semantic-Interop]
Konstantinos, K. and A. Katasonov, "Semantic Interoperability on the Web of Things: The Smart Gateway Framework", VTT Technical Research Center Tampere, Finland, DOI 10.1109/CISIS.2012.200, 2012, <https://www.researchgate.net/publication/231203029_Semantic_Interoperability_on_the_Web_of_Things_The_Smart_Gateway_Framework>.

Appendix A. Future Work

- Provide more examples.
- Discuss relationship to higher semantic layer(s).

Authors' Addresses

Kerry Lynn
Verizon Labs
50-60 Sylvan Rd
Waltham , MA 02451
USA

Phone: +1 781 296 9722
Email: kerlyn@ieee.org

Laird Dornin
Verizon Labs
50-60 Sylvan Rd
Waltham , MA 02451
USA

Phone: +1 781 466 2062
Email: laird.dornin@verizon.com