Noise in specifications hurts

Abstract
This short position paper analyses a collection of schemas in JSON schema format. These schemas appear to be hard to use as they contain significant syntactic noise. After translation to CDDL, the specifications appear to become more useful. The translation also has uncovered errors in the original specifications that probably were buried by their bulk.

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
Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

[I-D.zyp-json-schema] is a popular way to describe a set of JSON documents. JSON schema files are themselves JSON files.

Previously, in the definition of Relax-NG, a schema format for XML documents, it became clear that the XML-based format originally envisaged was hard to use. Relax-NG Compact was the alternative format developed to be useful for humans working with the specifications.

The present position paper repeats the exercise, using as a corpus a collection of JSON schema files provided by OCF [OCF]. Subjectively at least for this author, after translation into CDDL [I-D.greevenbosch-appsawg-cbor-cddl], the specifications became much more accessible.

2. Method

The OCF data models were automatically translated to CDDL using a rough tool developed for this purpose (at this time, the tool does not contain a prettyprinter function). Minimal hand editing was required to make the result useful within this document.

While a cursory comparison works out, it is likely that the tool still contains some errors; it was written mainly to support the present position paper. A number of guesses were required because of errors in the OCF data models. In 9 cases, a structure that was obviously intended as a JSON object was not defined as such. oic.r.media does not appear to conform to the structure of an object definition at all. oic.r.mediaSourceList.json gave an incorrect schema type. The type oic.web-link is not defined.

3. Result

The OCF data models [OCF] are described in 1725 lines of JSON. After translation to CDDL, with some comments and boilerplate added, 462 lines remain. 53150 characters turned into 15993. The more striking difference is the accessibility of the result, as demonstrated by this simple example:

```json
{
  "id": "http://openinterconnect.org/schemas/oic.r.door#",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "© 2016 Open Interconnect Consortium, Inc. All rights reserved.",
  "title": "Door",
  "definitions": {
    "oic.r.door": {
      "type": "object",
      "properties": {
        "openState": {"enum": ["Open","Closed"],
        "description": "ReadOnly, The state of the door (open or closed)"
      },
      "openDuration": {
        "type": "string",
        "description": "ReadOnly, The time duration the door has been open"
      },
      "openAlarm": {
        "type": "boolean",
        "description": "The state of the door open alarm"
      }
    }
  }
},
"type": "object",
"allOf": [
  {"$ref": "oic.core.json#/definitions/oic.core"},
  {"$ref": "oic.baseResource.json#/definitions/oic.r.baseResource"},
  {"$ref": "#/definitions/oic.r.door"}
],
"required": ["openState"]
}
```

turns into

```json
oic.r.door = { oic.r.baseResource,
  openState: "Open" / 'Closed' ; ReadOnly, The state of the door (open or closed)
  ? openDuration: text ; ReadOnly, The time duration the door has been open
  ? openAlarm: bool ; The state of the door open alarm
}
```

The reader is encouraged to have a look at the cited github repository and the appendix of the present document to judge for themselves.

4. Conclusion

Schema definitions are made to be used by humans. They probably should be designed primarily for their intended use. Excessive syntactic noise [Noise] detracts from the usability of the format.

5. Informative References

[I-D.greevenbosch-appsawg-cbor-cddl] Vigano, C. and H. Birkholz, "CBOR data definition language
Appendix A. CDDL format of OCF specifications

Unfortunately, RFC format does not allow for syntax coloring. The following CDDL specification should still be useful to get an overview over the OCF JSON schemas.

```cddl
start = oic.create / oic.r.airFlow / oic.r.airflowControl-Batch / oic.r.airflowControl / oic.r.audio / oic.r.autofocus / oic.r.automaticDocumentFeeder / oic.r.button / oic.r.colour.autowhitebalance / oic.r.colour.chroma / oic.r.colour.rgb / oic.r.colour.brightness / oic.r.colorbattery / oic.r.energy.consumption / oic.r.energy.drlc / oic.r.energy.load / oic.r.energy.usage / oic.r.humidity-Update / oic.r.humidity / oic.r.icemaker-Update / oic.r.icemaker / oic.r.light.brighness / oic.r.light.dimming / oic.r.light.rampTime / oic.r.lock.code / oic.r.lock.status / oic.r.media / oic.r.mediaSource / oic.r.mediaSourceList / oic.r.mode-Update / oic.r.mode / oic.r.movement.linear / oic.r.nightMode / oic.r.openLevel / oic.r.operational.state-Update / oic.r.operational.state / oic.r.ptz / oic.r.refrigeration-Update / oic.r.refrigeration / oic.r.sensor.activity.count / oic.r.sensor.atmosphericPressure / oic.r.sensor.carbonDioxide / oic.r.sensor.carbonMonoxide / oic.r.sensor.contact / oic.r.sensor.illuminance / oic.r.sensor / oic.r.sensor.magneticFieldDirection / oic.r.sensor.motion / oic.r.sensor.presence / oic.r.sensor.radiation.uv / oic.r.sensor.touch / oic.r.sensor.water / oic.r.signalStrength / oic.r.speech.tts / oic.r.switch.binary / oic.r.temperature-error / oic.r.temperature / oic.r.time.period

;;; oic.core.json: Core

oic.core = {
    ? lt: text ; ReadOnly, Resource Type
    ? fo: [+ "oic.if.def" / "oic.if.ll" / "oic.if.b" / "oic.if.rp" / "oic.if.p" / "oic.if.a" / "oic.if.s"]; ReadOnly, The interface
    set supported by this resource
    ? p: text ; ReadOnly, bitmap indicating observable and discoverable
    ? n: text ; Friendly name of the resource
}

;;; oic.baseResource.json: Base Resource

oic.r.baseResource = ( oic.core,
    id: text ; ReadOnly, Instance ID of this specific resource
    ? value: text / bool
    ? range: text
)

;;; oic.create.json

oic.create = {
    ? ResURI: text
}

;;; oic.r.airFlow.json: Air Flow

oic.r.airFlow = { oic.r.baseResource,
    ? direction: text ; Directionality of the air flow
    speed: int ; Current speed level
    ? range: text ; ReadOnly, Min,max values for the speed level
}

;;; oic.r.airflowControl-Batch.json: Air Flow Control

oic.r.airflowControl-Batch = { oic.r.baseResource,
    airflowControl: [* oic.r.switch.binary / oic.r.airFlow]
}

;;; oic.r.airflowControl.json: Air Flow Control

oic.r.airflowControl = { oic.r.baseResource,
    airflowControl: [2*2 oic.r.web-link]
}

;;; oic.r.audio.json

oic.r.audio = { oic.r.baseResource,
    volume: int ; Volume setting of an audio rendering device.
    mute: bool ; Mute setting of an audio rendering device
}

;;; oic.r.autofocus.json: Auto Focus

oic.r.autofocus = { oic.r.baseResource,
    autofocus: bool ; Status of the Auto Focus
}
/// oic.r.automaticDocumentFeeder.json: Automatic Document Feeder
oic.r.automaticDocumentFeeder = { oic.r.baseResource,
  adfStates: text                      ; ReadOnly, Comma separated list of the possible adf states.
currentAdfState: text                 ; ReadOnly, Current adf state.
}

/// oic.r.button.json: Button Switch
oic.r.button = { oic.r.baseResource,
  value: bool                         ; ReadOnly, Status of the button
}

/// oic.r.colour.autowhitebalance.json: Auto White Balance
oic.r.colour.autowhitebalance = { oic.r.baseResource,
  autoWhiteBalance: bool              ; Status of the Auto White balance
}

/// oic.r.colour.chroma.json: Colour Chroma
oic.r.colour.chroma = { oic.r.baseResource,
  hue: int                            ; Hue as defined by the CIECAM02 model definition
  saturation: int                     ; Saturation as defined by the CIECAM02 model definition
  colourspacevalue: text              ; CSV of chromaX, chromaY, colourTemperature (X,Y,T).
}

/// oic.r.colour.rgb.json: Colour RGB
oic.r.colour.rgb = { oic.r.baseResource,
  rgbValue: text                      ; RGB value
  ? range: text                       ; min max value of RGB
}

/// oic.r.colour.saturation.json: Colour Saturation
oic.r.colour.saturation = { oic.r.baseResource,
  colourSaturation: int               ; The colour saturation value
}

/// oic.r.door-update.json: Door
oic.r.door-update = { oic.r.baseResource,
  ? openAlarm: bool                   ; The state of the door open alarm
}

/// oic.r.door.json: Door
oic.r.door = { oic.r.baseResource,
  openState: "Open" / "Closed"        ; ReadOnly, The state of the door (open or closed)
  ? openDuration: text                ; ReadOnly, The time duration the door has been open
  ? openAlarm: bool                    ; The state of the door open alarm
}

/// oic.r.energy.battery.json: Battery
oic.r.energy.battery = { oic.r.baseResource,
  charge: int                         ; ReadOnly, The current charge percentage.
}

/// oic.r.energy.consumption.json: Energy Consumption
oic.r.energy.consumption = { oic.r.baseResource,
  power: number                       ; ReadOnly, Instantaneous Power
  energy: number                      ; ReadOnly, Energy consumed
}

/// oic.r.energy.drlc.json
oic.r.energy.drlc = { oic.r.baseResource,
  DRType: int                         ; The to be applied demand-response type
  ? start: text                       ; The start time for the application of DR
  ? duration: int                     ; The duration of the to be applied DR type
  ? override: bool                    ; Whether the consumer has overriden the application of DR
}

/// oic.r.energy.overload.json: Energy Overload Sensor
oic.r.energy.overload = oic.r.sensor

/// oic.r.energy.usage.json: Energy Usage
oic.r.energy.usage = { oic.r.baseResource,
  resources: [2*2 oic.web-link]
}

/// oic.r.humidity-update.json: Humidity
oic.r.humidity-update = { oic.r.baseResource,
  ? desiredHumidity: int              ; Desired value for Humidity
}

/// oic.r.humidity.json: Humidity
oic.r.humidity = { oic.r.baseResource,
  humidity: int                      ; ReadOnly, Current sensed value for Humidity
  ? desiredHumidity: int             ; Desired value for Humidity
}
;; oic.r.iceMaker-Update.json: Ice Maker

oic.r.iceMaker-Update = { oic.r.baseResource,
  status: "on" / "off" ; Set the status of the Ice Maker
}

;; oic.r.iceMaker.json: Ice Maker

oic.r.iceMaker = { oic.r.baseResource,
  status: "on" / "off" / "full" ; Status of the Ice Maker
}

;; oic.r.light.brightness.json: Brightness

oic.r.light.brightness = { oic.r.baseResource,
  brightness: int ; Current sensed or set value for Brightness
}

;; oic.r.light.dimming.json: Dimming

oic.r.light.dimming = { oic.r.baseResource,
  dimmingSetting: int ; Current dimming value
  ? step: int ; ReadOnly, step increment for dimming values
  ? range: text ; ReadOnly, Min and Max values for the dimming setting
}

;; oic.r.light.rampTime.json: Ramp Time

oic.r.light.rampTime = { oic.r.baseResource,
  rampTime: int ; Actual speed of changing between 2 dimming values
  ? range: text ; ReadOnly, Min and Max of possible values
}

;; oic.r.lock.code.json: Lock Code

oic.r.lock.code = { oic.r.baseResource,
  ; each value is a Value for the lock code:
  lockCodeList: [* text]
}

;; oic.r.lock.status.json: Lock

oic.r.lock.status = { oic.r.baseResource,
  lockState: "Locked" / "Unlocked" ; State of the lock.
}

;; oic.r.media.json: Media

object1 = {
  ? url: text ; url for the media instance
  ; each value is a SDP media or attribute line:
  ? sdp: [* text] ; Array of strings, one per SDP line
}

oic.r.media = { oic.r.baseResource,
  media: [* object1]
}

;; oic.r.mediaSource.json: Media Source

oic.r.mediaSource = { oic.r.baseResource,
  sourceName: text ; Specifies a pre-defined media input or output
  ? sourceNumber: int / text ; ReadOnly, Numeric identifier to specify the instance
  ? sourceType: "audioOnly" / "videoOnly" / "audioPlusVideo" ; ReadOnly, Specifies the type of the source
  status: bool ; Specifies if the specific source instance is selected or not
}

;; oic.r.mediaSourceList.json: Media Source List

oic.r.mediaSourceList = { oic.r.baseResource,
  sources: [* oic.r.mediaSource]
}

;; oic.r.mode-Update.json: Mode

oic.r.mode-Update = { oic.r.baseResource,
  modes: text ; Desired mode
}

;; oic.r.mode.json: Mode

oic.r.mode = { oic.r.baseResource,
  supportedModes: text ; ReadOnly, Comma separated list of possible modes the device supports.
  modes: text ; Comma separated list of the currently active mode(s)
}

;; oic.r.movement.linear.json: Linear Movement

oic.r.movement.linear = { oic.r.baseResource,
  movementSettings: text .regexp csv-regexp ; ReadOnly, comma separated list of possible movement values
  movement: text ; Current movement value
  ? movementModifier: text ; Modified to the movement value (e.g. spin-90, left-20),
  ? range: text ; units are device dependent
}

;; oic.r.nightMode.json: Night Mode
oic.r.nightMode = { oic.r.baseResource,
    nightMode: bool ; Status of the Night Mode
}

oic.r.openLevel = { oic.r.baseResource,
    openLevel: int ; How open or ajar the entity is
    ? increment: int ; ReadOnly, The step between possible values
    ? range: text ; ReadOnly, Lower bound=closed, Upper bound=open
}

oic.r.operational.state = { oic.r.baseResource,
    ? currentMachineState: text ; Current state of operation of the device.
    ? currentJobState: text ; Currently active jobState
}

oic.r.ptz = { oic.r.baseResource,
    pan: number ; horizontal pan in degrees
    tilt: number ; vertical tilt in degrees
    ? pan_range: text .regexp csv-regexp; ReadOnly, Min and Max values for the pan setting
    ? tilt_range: text .regexp csv-regexp; ReadOnly, Min and Max values for the tilt setting
    ? zoomFactor: text ; The zoom factor value
    ? zoomFactorRange: "linear, 1x, 2x, 4x, 8x, 16x, 32x"; ReadOnly, allowed Zoom Factor values. Linear equates to a 1-100 min/max.
}

oic.r.refrigeration = { oic.r.baseResource,
    ? rapidFreeze: bool ; Indicates whether the unit has a rapid freeze capability active.
    ? rapidCool: bool ; Indicates whether the unit has a rapid cool capability active
    defrost: bool ; Indicates whether a defrost cycle is currently active
}

oic.r.sensor = { oic.r.baseResource,
    ? filter: int ; ReadOnly, Percentage life time remaining for the water filter
    ? rapidFreeze: bool ; Indicates whether the unit has a rapid freeze capability active.
    ? rapidCool: bool ; Indicates whether the unit has a rapid cool capability active
    defrost: bool ; Indicates whether a defrost cycle is currently active
}

oic.r.illuminance = { oic.r.baseResource,
    illuminance: number ; ReadOnly, sensed luminous flux per unit area in lux.
}
oic.r.sensor.json: Generic Sensor

```json
oic.r.sensor = { oic.r.baseResource,
    ? value: bool                   ; ReadOnly, true = sensed, false = not sensed. }
```

oic.r.sensor.magneticFieldDirection.json: Magnetic Field Direction Sensor

```json
oic.r.sensor.magneticFieldDirection = { oic.r.baseResource,
    value: text                     ; ReadOnly, CSV containing Hx, Hy, Hz. }
```

oic.r.sensor.motion.json: Motion Sensor

```json
oic.r.sensor.motion = oic.r.sensor
```

oic.r.sensor.presence.json: Presence Sensor

```json
oic.r.sensor.presence = oic.r.sensor
```

oic.r.sensor.radiation.uv.json: UV Radiation

```json
oic.r.sensor.radiation.uv = { oic.r.baseResource,
    measurement: number             ; ReadOnly, the measured UV Index }
```

oic.r.sensor.touch.json: Touch Sensor

```json
oic.r.sensor.touch = oic.r.sensor
```

oic.r.sensor.water.json: Water Sensor

```json
oic.r.sensor.water = oic.r.sensor
```

oic.r.signalStrength.json: Signal Strength

```json
oic.r.signalStrength = { oic.r.baseResource,
    lqi: number                     ; ReadOnly, current value of Link Quality Indicator
    rssi: number                    ; ReadOnly, current value of Received Signal Strength Indicator }
```

oic.r.speech.tts.json: Speech Synthesis-TTS

```json
oic.r.speech.tts = { oic.r.baseResource,
    utterance: text                 ; SSML document including the speak body
    ? supportedLanguages: text       ; ReadOnly, comma separated list of supported language tags
    ? supportedVoices: text          ; ReadOnly, SSML document fragment indicating supported voices }
```

oic.r.switch.binary.json: Binary Switch

```json
oic.r.switch.binary = { oic.r.baseResource,
    value: bool                     ; Status of the switch }
```

oic.r.temperature-Error.json: Temperature

```json
oic.r.temperature-Error = { oic.r.baseResource,
    ? units: "C" / "F" / "K"        ; ReadOnly, Units for the temperature value
    ? range: text                   ; ReadOnly, Comma separated min,max values for this temperature on this device }
```

oic.r.temperature.json: Temperature

```json
oic.r.temperature = { oic.r.baseResource,
    temperature: number             ; Current temperature setting or measurement
    ? units: "C" / "F" / "K"         ; ReadOnly, Units for the temperature value
    ? range: text                    ; ReadOnly, Comma separated min,max values for this temperature on this device }
```

oic.r.time.period.json: Time Period

```json
oic.r.time.period = { oic.r.baseResource,
    startTime: text                 ; Start time for the time period
    ? stopTime: text                ; Stop time for the time period }
```

```text
;;; types not defined in the JSON files (guessed)

oic.web-link = nil               ; certainly guessed wrong
csv-regexp = "([^,]*,)*[^,]*"
```

Author's Address

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen, D-28359
Germany
Phone: +49-421-218-63921
EMail: cabo@tzi.org