TD Format definition in the spec: use of JSON Schema, ABNF

Michael.Lagally@oracle.com

Bundang F2F

4.7.2018

What is the problem?

The current TD format is primarily described by examples

These leave a lot of room for interpretations

Difficult to track different versions / No change marks, no diffs

Normative descriptive language still leaves room for interpretation

No formal TD validation possible

This hurts interoperability

A formal syntax description

Enforces to get conceptual clarity

Allows validation using public tools

Ensures interoperability

Clear definition of lexical tokens, e.g. valid characters for identifiers, number formats etc.

-> Spec gets more precise

How to describe a data format

There are multiple ways to define data formats, Wikipedia has 63 pages on data model modeling languages.

For the WoT TD spec there are two obvious candidates for a formal format description:

ABNF

JSON-Schema

ABNF

Augmented Backus Naur Form is a common way to precisely define a data format

Specified by https://tools.ietf.org/html/rfc5234

Used for describing the syntax of programming languages, data formats, JSON format, …

Very compact format, fit for multiple purposes

Not specifically designed for JSON

ABNF Example (Oracle Device Model)

```
device_model_description = '{'
        metadata_section
        [ attribute_section ]
        [ action_section ]
       <snip>
    '}' ;

metadata_section =
    standard_metadata
    [ vendor_extension_metadata ] ;

standard_metadata =
    '"urn"' ':' '"'urn_type'"' ','
    '"name"' ':' '"'js_identifier'"' ',’
    [ '"author"' ':' js_string ',' ]
    '"version"' ':' js_string ','
    '"description"' ':' "<human-readable description as js_string>" ','
    '"created"' ':' js_number ',' /* time in milliseconds since 01 January, 1970 UTC */
    '"createdAsString"' ':' js_date ','
    '"lastModified"' ':' js_number ',' /* time in milliseconds since 01 January, 1970 UTC */
        '"lastModifiedAsString"' ':' js_date ','
        '"userLastModified"' ':' js_string ;

attribute_section = '”attributes"' ':' '[' attribute_declarations ']' ',' ;

attribute_declaration = attribute |
    [ attribute ',' attribute_declaration ] ;

attribute = attribute_name ":" '{'
            [ '"name"' ':' attribute_name "," ]
            [ '"alias"' ':' js_identifier "," ]
            '"description"' ':' js_string ","
            '"type"' ':' attribute_type ",”
            [ '"unit"' ':' js_string "," ]
            [ '"range"' ':' '"' js_number ',' js_number '"' "," ]
            '"writable"' ':' 'true' | 'false'
        '}' ;
<snip>
```

JSON-Schema

JSON Schema is not an RFC yet, only an informational internet draft

https://tools.ietf.org/html/draft-handrews-json-schema-01

The current version expires: September 20, 2018

The JSON Schema Spec is rather stable − fit for purpose as demonstrated by the TD-playground

Several validation tools are available online, e.g.:

https://jsonschemalint.com/#/version/draft-06/markup/json

More verbose than ABNF.

JSON Schema example

```
{
  "title": "WoT TD Schema for Bundang Plug Fest",
  "description": "JSON Schema representation of the TD serialisation format.",
  "$schema ": "http://json-schema.org/draft-06/schema#",
  "type": "object",
  "properties": {
    "base": {"$ref": "#/definitions/url”},
    "@type": {"$ref": "#/definitions/type_declaration”},
    "@context": {"$ref": "#/definitions/context”},
    "name": {"type": "string” },
    "id": {"type": "string”},
    "description": {"type": "string”},
    "properties": {
      "type": "object",
      "items": {"$ref": "#/definitions/properties”}
    },
    "actions": {
      "type": "object",
      "items": {"$ref": "#/definitions/actions }
    },
    "events": {
      "type": "object",
      "items": {"$ref": "#/definitions/events”}
    },
    "links": {
      "type": "array",
      "items": {"$ref": "#/definitions/links”}
    }
< snip >
```

Proposal

Add JSON Schema definition for the JSON serialization to the TD spec.

Make it a normative part of the TD spec.

Add JSON Schema to TD spec

Pro:

- Clear specification of the data format
- No ambiguity
- Formal validation is ensured
- No divergence between JSON Schema and human readable spec text

Con:

- Additional (small) spec maintenance effort