ASDF
Virtual interim #1
2020-11-02

Note Well

- You will be recorded
- Be nice, and be professional
- The IPR guidelines of the IETF apply:
  see **http://ietf.org/ipr** for details.

Repo: https://github.com/ietf-wg-asdf/asdf-working-group-notes
Notes: https://codimd.ietf.org/notes-ietf-interim-2020-asdf-01-asdf

Note Well

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (https://www.ietf.org/contact/ombudsteam/) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- BCP 9 (Internet Standards Process)
- BCP 25 (Working Group processes)
- BCP 25 (Anti-Harassment Procedures)
- BCP 54 (Code of Conduct)
- BCP 78 (Copyright)
- BCP 79 (Patents, Participation)
- https://www.ietf.org/privacy-policy/(Privacy Policy)

IETF®

Agenda

- Introduction, admin, agenda bashing (10 min)
- ASDF status update (chairs) (10min)
- Result of adoption call (chairs) (2 min)
- SDF 1.1 features

Status update

- **Status:** We have been chartered!
- **Chairs:** Michael Richardson and Niklas Widell
- **Plans:**
  - Virtual interim - today
  - IETF 109 Hackathon - Preparations for SDF 1.1
  - IETF 109 - 2hr meeting slot requested
- **Activities since last meeting:**
  - Niklas presented ASDF to OMA SpecWorks DMSE group (LwM2M)

WG procedures

- Decisions on mailing list: https://www.ietf.org/mailman/listinfo/asdf
- Work on Github: https://github.com/ietf-wg-asdf
  - SDF spec will be migrated on adoption
  - Use Issue tracker for issues (new features and fixes)
- Schedule regular virtual interims between virtual physical meetings
  - Will send out doodle for finding meeting time
- Plan to release intermediate implementation drafts of SDF
  - E.g., to be used by OneDM as update target for toolchains

Call for adoption of SDF 1.0 draft

- Call for adoption of draft-onedm-t2trg-sdf sent on October 20th
- No objections have been raised

Next step SDF 1.1

- https://github.com/ietf-wg-asdf/SDF/issues

- Four features proposed

Choice:
Composing types out of alternatives

- https://github.com/ietf-wg-asdf/SDF/issues/2:
  named alternatives (`enum`s), with numbers (ranges) thrown in or not
  (limited to text/number/bool/*text/*number/*bool in 1.0)

- https://github.com/ietf-wg-asdf/SDF/issues/5:
  build choices out of general alternative types [`anyOf`]
  (not in 1.0; adding a variant of json-schema.org's `anyOf` allows types to be merged without naming the alternatives)

- There really isn’t much of a difference
  (except that json-schema.org has different names for these cases)

- Convenience value in:
  just giving text string names to a newly set up choice (`enum` of strings),
  vs. naming semantic tags locally (map a string to each semantic tag)

Choice: Example

- `state` — has a data type with two alternatives:

```
"sdfProperty": {
  "state": {
    "sdfEnum": {
      "on": {
        "description": "activated state, or powered state",
        "label": "On"
      },
      "off": {
        "description": "deactivated state, or non-powered state",
        "label": "Off"
      }
  }
```

- Redundant labels, no semantic foundation, but comment (`description`)

Choice: Example

- `StartUpCurrentLevel` — has a data type with two alternatives, one of which has two alternatives inside:

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

- Unnecessary nested choice (one as `anyOf`, one as `enum`); no name for outer alternatives
- Numeric identifiers are not at an information model level (but could still be agreed)

Choice: Example

- `brightnessWithColorTone` — has a data type with three labeled alternatives (more readable, less precise syntax):

```
sdfProperty {
  brightnessWithColorTone {
    anyOf [
      {
        type number
        label "warm"
        minimum 0
        maximum 86
      }
      {
         type number
         label "neutral"
         minimum 87
         maximum 166
      }
      {
         type number
         label "cool"
         minimum 167
         maximum 255
      }
    ]
  }
}
```

- Labels are part of the constituent types; can’t just reference a type
- Numeric identifiers are not at an information model level (but could still be agreed)

Choice: Example

- `brightnessWithColorTone` — has a data type with three labeled alternatives (more readable, less precise syntax):

```
“sdfProperty”: {
  “brightnessWithColorTone”: {
    “warm”: {
        type number
        minimum 0
        maximum 86
      }
    “neutral”: {
         type number
         minimum 87
         maximum 166
      }
    “cool”: {
         type number
         minimum 167
         maximum 255
      }
    ]
  }
}
```

- Labels are now part of the choice construct
- Numeric identifiers are not at an information model level (but could still be agreed)

Aggregation:
Composing types out of components

- https://github.com/ietf-wg-asdf/SDF/issues/4

- 1.0: Awkward composition in parameter lists (`sdf*Data` in Action/Event); unnecessarily different from Property, where Data are defined in place; `sdfRequired` is awkward way to describe optionality

- 1.1: go for (1) direct reference (only one, no “multiple inheritance”?) as well as (2) named fields (choice of specifier)

- Also: do we compose only types or entire affordances?

Aggregation: Example

- `compound1`: a property built out of two data items

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

- `sdfData` becomes a data schema construct; a set of named fields (with data schema definitions inside each)
  - Could call that "`object`" like in json-schema.org
- No obvious place to put in optionality (bikeshed issue)

Reference:
share elements inside and outside spec

- https://github.com/ietf-wg-asdf/SDF/issues/3
- Use JSON pointers right now; management of the space untested
- What can we reference?
  - 1.0: Data type (via `sdfData`)
  - 1.1:
    - Do we need to reference (and combine?) entire affordances?
    - How much can we alter a referenced type/affordance?

Reference: Example

- `sdfData` referencing another `sdfData`:

```
“sdfData": {
  "length" : {
    "type": "number",
    "minimum": 0,
    "units": "m"
    "description": "There can be no negative lengths."
  }
...
  "cable-length" : {
    "sdfRef": "#/sdfData/length"
    "minimum": 0.05,
    "description": "Cables must be at least 5 cm."
  }
```

- Does the second `description` override or add to the first `description`?

Reference: Example

- `sdfProperty` referencing another `sdfData`:

```
"sdfData": {
  "length" : {
    "type": "number",
    "minimum": 0,
    "units": "m"
    "description": "There can be no negative lengths."
  }
},
"sdfProperty: {
  "cable-length" : {
    "sdfRef": "#/sdfData/length"
    "minimum": 0.05,
    "description": "Cables must be at least 5 cm."
  } …
```

- `sdfData` just *defines* the type, does not *declare* an affordance; `sdfProperty` does

Reference: Example

- `sdfProperty` referencing another `sdfData` in a different specification:

```
"sdfData": {
  "length": {
    "type": "number",
    "minimum": 0,
    "units": "m"
    "description": "There can be no negative lengths."
  }
}
```

Export

Over in example.com namespace:

```
"namespace": {
  "moo": "https://example.com/#"
},
"defaultnamespace": "moo"
```

```
"sdfProperty: {
  "cable-length" : {
    "sdfRef": "foo:/sdfData/length"
    "minimum": 0.05,
    "description": "Cables must be at least 5 cm."
  } …
```

Import

```
"namespace": {
  "foo": "https://example.com/#"
}
```

- This is part of SDF 1.0, but we haven't exercised this

Next step SDF 1.1

- https://github.com/ietf-wg-asdf/SDF/issues

- Discuss here and on the mailing list

- Target date?

AOB?