ASDF
IETF 109
2020-11-17

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-109-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)
- SDF 1.1 features
- AOB

Administrative

- CodiMD for notes https://codimd.ietf.org/notes-ietf-109-asdf
- Meetecho for speaking. https://meetings.conf.meetecho.com/ietf109/?group=asdf&short=&item=1
- jabber: xmpp:asdf@jabber.ietf.org
- Roll call/Blue sheet
- Note takers
- AOB?

Status update

- **Status:** ASDF WG was chartered in October
- **Chairs:** Michael Richardson and Niklas Widell
- **Progress so far:**
  - Unofficial hallway meeting — SDF 1.0 tutorial: https://youtu.be/_8i6X4AxuOk
  - Virtual interim — **SDF 1.0 draft adopted** as draft-ietf-asdf-sdf
  - IETF 109 Hackathon — Preparations for SDF 1.1
- **Meeting plans:**
  - IETF 109 — today
  - Virtual interim in December

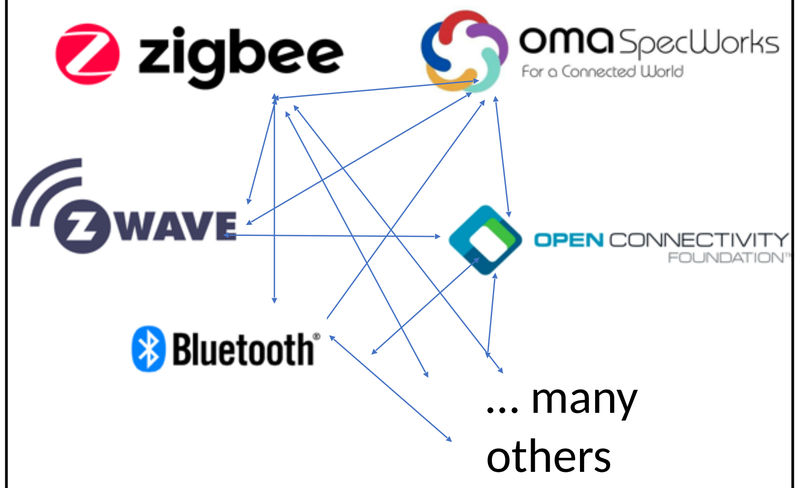
WG procedures

- Decisions on mailing list: https://www.ietf.org/mailman/listinfo/asdf
- Work on Github: https://github.com/ietf-wg-asdf
  - Use Issue tracker for issues (new features and fixes)
- Schedule (doodle) regular virtual interims between virtual physical meetings
- Plan to release intermediate “Implementation Drafts” of SDF
  - E.g., to be used by OneDM as update target for toolchains

One Data Model in a nutshell

The problem

Several standardised data models

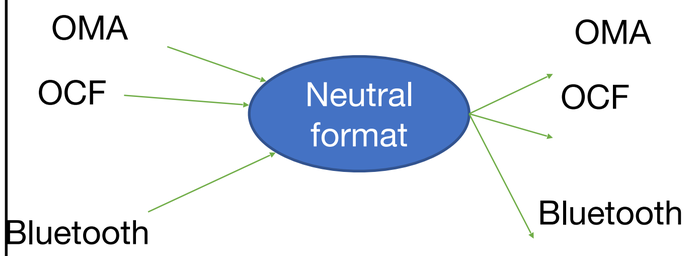


… many others

Lack of semantic interoperability across ecosystems ➔ high integration cost

Potential Solutions

- ❑ Everyone join my (or your) ecosystem ➔ *does not work*
- ✓ Facilitate translation between ecosystems via a neutral format



One Data Model

- ✓ Neutral format: Semantic Definition Format (SDF), now destined for IETF
- ✓ Create toolchains SDF ⇔ ecosystems
- ✓ Create set of adopted device models for wider re-use
- ✓ https://onedm.org/

ASDF Outreach

- Value of ASDF increases with increased adoption of SDF by organisations doing IoT data models

- Outreach so far (beyond OneDM)
  - ASDF presented to DMSE (LwM2M) group in OMA SpecWorks
  - SDF proposed to ISO/IEC JTC1 SC41 for IoT Thing modeling

- **If your organisation works with IoT data models — talk to us!**

SDF 1.1

SDF 1.1: Open Issues

- #3: Have we tested the cross-referencing/reuse well-enough?
  - #7: override/augment after reference
- #4: support for compound/composite data (can fix parameter-lists as well)
- #2: enums, #5: choices: handle more than optionality in composition
  - Related, but somewhat different
  - #2 Enum: uniformity of alternatives; need to add descriptive information
  - #5 “anyOf”: type union; misses out on description of alternatives

Hackathon: What got done

- Tested out resolutions to issue 3: cross-model referencing
  - Came up with convention on curie usage
  - Discussed transitions in namespaces during model progress and completion
- Made progress on issue 4: aggregation (compound types)
  - Fixing the way data are composed for actions and events

#3, #7

Reference:
share elements inside and outside spec

- https://github.com/ietf-wg-asdf/SDF/issues/3 (“#3”)
- Use JSON pointers right now; management of the space was 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? (#7)

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: old 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

Reference: updated Example

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

Export

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

Over in example.com namespace:

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

Import

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

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

- Keep the # with the curie; this allows uniform referents for local and defaultNamespace

Reference: updated Example

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

Export

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

Over in example.com namespace:

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

Import

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

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

- Keep the # with the curie; this allows uniform referents for local and defaultNamespace

#4

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": {
    "type": "object",
    "required": ["element1"],
    "properties": {
      "element1": { "type": "number" },
      "element2": { "type": "string" }
    }
  }
}
```

  - Mirror “type”: “`object`” like in json-schema.org

- express optionality like in json-schema.org

Aggregation: get rid of pointer-lists

- 1.0: types, but not names, for parameters

```
"sdfAction": {
  "hug": {
    "sdfInputData": ["#/foo/bar/squeeze", … ],
    "sdfOutputData": ["#/foo/bar/comfort", … ]
  }
}
```

- 1.1 (in –01): can have both, plus other descriptions and semantic tags

```
"sdfAction": {
  "hug": {
    "sdfInputData": {"type": "object", "properties": { "squeeze": … }},
    "sdfOutputData": {"type": "object", "properties": { "comfort": … }}
  }
}
```

#2, #5

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)

Choice: choices

- A choice can:
  - Be a type union
    - The branches reliably differ, or
    - It doesn’t matter which branch is used (i.e., the same value can occur in multiple branches but means the same)
  - Be a set of named alternatives
    - This meshes well with attaching semantic information
- **Proposal:**
  - **Always go for named alternatives**
  - **Unify syntax for “enum” and “anyOf” flavors**

Defining the choice syntax

- json-schema.org “Enum” is an array of values
  - Need not be of same “type” (whatever that is for you)
  - A convention is to use text strings only
  - (But these text strings don’t ***mean*** anything)
- json-schema.org “anyOf” is an array of schema expressions
  - No labels for the branches

Choice: Example

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

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

- Use the names (map keys — must be strings) as labels (could add “const” if needed)

- Can easily add semantic tags beyond human-only “description”

Choice: Example

- `StartUpCurrentLevel` — reformulated as a data type with three alternatives:

```
"sdfProperty": {
  "StartUpCurrentLevel": {
    "choice": {
      “ActualSetting”: {
        "type": "number",
        "minimum": 1,
        "maximum": 254
      },
      "MinimumDeviceValuePermitted": {
        "const": 0
      },
      "SetToPreviousValue": {
        "const": 255
      }
    }
  }
}
```

- Unnecessary nesting fixed; outer alternatives are named (and can have descriptions)
- Numeric identifiers for branches are not at an information model level (but could still be agreed)

1.1

Next steps to SDF 1.1

- https://github.com/ietf-wg-asdf/SDF/issues
- Discuss here and on the mailing list
- Complete the choice discussion and see if we really can close #2 and #5
- Get some work done on #7, look for other small details
- Timebox this; target date?
  - Proposal: 2020-12-08 final draft in repo, **WG has a look**,
  - submit SDF 1.1 as an implementation draft 2020-12-22

Beyond SDF 1.1

- SDF success factors:
  - Enable conversion of existing ecosystem models to SDF
  - SDF attractive as development language for new ecosystem models
- How can we increase these success factors
  - Test with more ecosystems
  - Strengthen the feedback links from ecosystems to SDF development
- Proposal: Aim for SDF 1.2 in February/March (before IETF 110)

AOB?