ASDF
Virtual interim
2021-06-02

Chairs: Michael Richardson, Niklas Widell

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-2021-asdf-02-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

1. Note Well. https://www.ietf.org/about/note-well/
2. Logistics for Meeting/admin
    1. CodiMD for notes https://codimd.ietf.org/notes-ietf-interim-2021-asdf-02-asdf
    2. Agenda bash
3. WG status update and future plans
4. SDF.next
5. Handling non-SDF info in mapping file (Carsten)
6. SDF: Relationships and Instances (Ari)
7. AOB?

Administrative

- CodiMD for notes https://codimd.ietf.org/notes-ietf-interim-2021-asdf-02-asdf
- Meetecho/webex: https://ietf.webex.com/ietf/j.php?MTID=m1c76ec067cec2124744ad29ccce6c616
- All ASDF notes at: https://github.com/ietf-wg-asdf/asdf-working-group-notes.git

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

Status update & future plans

- **Status:** ASDF WG was chartered in October 2021
- **Chairs:** Michael Richardson and Niklas Widell
- **Progress so far:**
  - Three virtual interims
  - IETF 109 & 110 with Hackathon
  - SDF 1.1 Implementation Draft published https://datatracker.ietf.org/doc/draft-ietf-asdf-sdf/
- **Meeting plans:**
  - Additional interims to be planned
  - No meeting at IETF 111

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
  - W3C Web of Things
  - Other use cases: Electronic Data Sheets, Textile manufacturing

- **If your organisation works with IoT data models, we want to hear from you!**

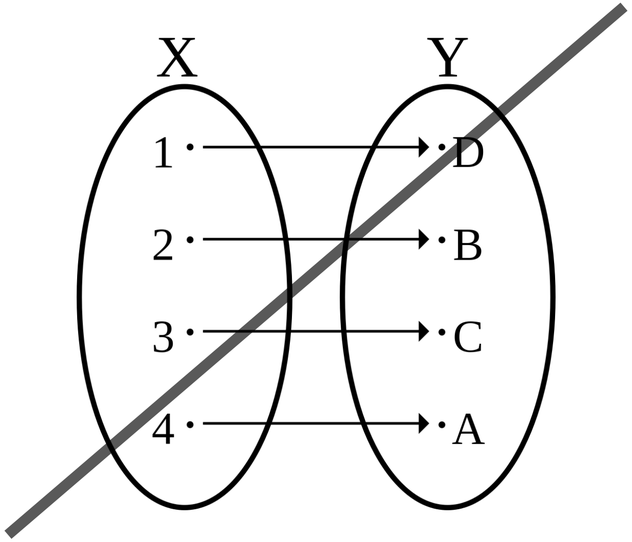
SDF.next

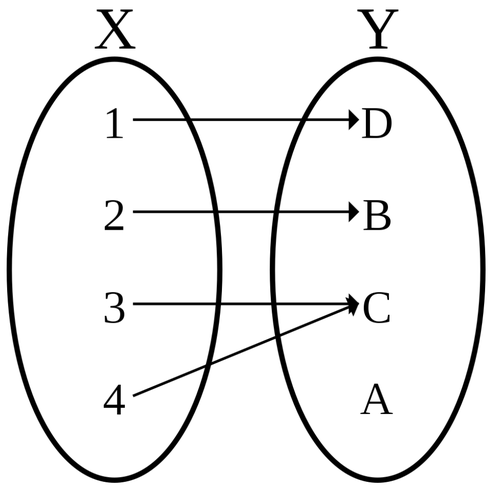
- Multi-instance sdfThings (discussion in OneDM, PR to be written)
- **#12: Allow URIs as unit values (beyond SenML registry)?**
- #26: Rules for combining infoblocks
- #27: Factoring sdfChoice
- **#28: Allow one model to contribute to more than one namespace?**
- **#29: Split version field into date, version, token-holder, feature-tags?**
- **PR #30: Fully define semantics of sdfRef + overrides (use RFC 7396)**
- (Editorial: PR #31)

SDF.next

SDF Quality for Round Trips

- Problem: A model converted to SDF and back (i.e. making a round trip) will differ substantially from the original if no additional measures are taken
- Why? Conversion between SDF and other formats does not always work in an injective, one-to-one manner
- Solution: *a new SDF quality*





Images: Wikipedia article on injective functions (https://en.wikipedia.org/wiki/Injective_function)

SDF Quality for Round Trips: Suggestion

- Statement *origin* in *sdfinfo* specifies orignal format
- *commonqualities* are extended by *origin* statement containing a named set of *originqualities*

```
17 sdfinfo = {
18  title: text
19  version: text
20  copyright: text
21  license: text
22  ? origin : {
23     formatname : text : e.g. "YANG"
24     ? formatref : text ; e.g. "RFC 7950"
25  }
26  EXTENSION-POINT<"info-ext">
27 }
...
37 originqualities = (
38  ? statement : text ; e.g. "type"
39  ? argument : text ; e.g. "union"
40 )
...
42 commonqualities = (
43  ? description: text
44  ? label: text
45  ? $comment: text
46  ? sdfRef: sdf-pointer
47  ? sdfRequired: pointer-list
48  ? origin : named<originqualities>
49 )
...
```

Extension of the formal SDF syntax
(Appendix A of draft-ietf-asdf-sdf-05)

SDF Quality for Round Trips: Example

```
module example {
  namespace "...";
  prefix "...";

  typedef exampleTpdf {
    type union {
      type string;
      type dec64;
    }
  }
}
```

YANG definition of a derived type based on the *union* built-in type

```
{
  "info": {
    "copyright": "...",
    "license": "...",
    "title": "...",
    "version": "...",
    "origin" : {
      "formatname" : "YANG",
      "formatref" : "RFC 7950"
    }
  },
  "sdfData": {
    "exampleTpdf" : {
      "sdfChoice" : {
        "string": { "type" : "string" }
        "dec64": { "type" : "number" }
      }
      "origin" : {
        "typedef" : {
          "statement" : "typedef"
        },
        "type" : {
          "statement" : "type",
          "argument" : "union"
        }
      }
    }
  }
}
```

Conversion to SDF

Attaching Semantics

Partial replay from T2TRG 2018-10-11

Attaching Semantics

"Semantic Style" (looking for a better word)

Attach information via **selectors** into the instance

— Similar to adding style semantics to HTML via CSS

One "style" can apply to:

— single instance
— "class" of instances (making it "metadata")

Models are instances

Can use JSON pointer as a selector syntax for a single item in a
single spec

More complex selectors may come in handy later ("all IPSO models"...)

Approaches

Transformation language (DSSSL/XSLT)

— What is the target language (generic data model)?

Augmentation language (CSS)

— In-model augmentation:
Stay in generic data model of input

— Extra-model augmentation:
Generate into expanded generic data model

SDF

Both transformation and augmentation might make sense

In-model augmentation might trigger existing extension points
Extra-model augmentation creates new extension points (i.e., they don't really differ that much)

Rules

selector → effect

```
.warning {color: red}
```

Selector selects zero or more structural elements into a "node set"

Effect is then applied to each selection

Start with JSON pointer

Use JSON syntax, as usual

```
{ ...
  "pg:#/sdfType/foo": ...
}
```

SDF relationships and instances

Ari Keränen

SDF relations

- SDF models today can present child-parent relationships / composition with sdfObjects in sdfThings (in sdfThings)
- Many ecosystems (e.g., OPC UA and DTDL) have support for more complex (arbitrary) relationships/references
- Common with more complex systems than just a single device

DTDL example:

```
{
   "@type": "Relationship",
   "name": "floor",
   "target": "dtmi:com:example:Floor;1"
}
```

SDF relations draft proposal

- Proposal: new SDF quality "sdfRelations" that contains map of relationships from object/thing to other definitions
- Target needed, can have type from ontology and additional properties

```
      "sdfRelations": {
        "containedIn" : {
          "type" : "saref:iscontainedIn",
          "target" : "ex:#/sdfObject/Room",
          "properties" : {
            "foo" : 42
          }
        }
      }
```

```
{
  "namespace": {
    "ex" : "https://example.com/room-stuff",
    "saref" : "https://saref.etsi.org/saref4bldg/"
  },
  "defaultNamespace": "ex",

  "sdfObject" : {
    "RoomSensor" : {
      "sdfProperty": {
        "Temperature" : {}
      },
      "sdfRelations": {
        "containedIn" : {
          "type" : "saref:iscontainedIn",
          "target" : "ex:#/sdfObject/Room",
          "properties" : {
            "foo" : 42
          }
        }
      }
    }
  }
}
```

Possible shorthand

- If only type and target needed, may be possible to combine them

```
      "sdfRelations": {
        "saref:iscontainedIn": "#/sdfInstances/room2"
      }
```

SDF instances

- When using SDF models in live system, often need to define also instance-specific information
  - Generating W3C WoT TD or CoRAL documents from SDF models
  - Provisioning devices from different ecosystems
- See discussion at previous CoRE interim for more examples

```
"sdfInstances": {
  "room1" : {
    "sdfInstanceOf": "api:#/sdfObject/Room/",
    "sdfProperty":{
      "Room-name": {
        "const": "foo" ...
```

End