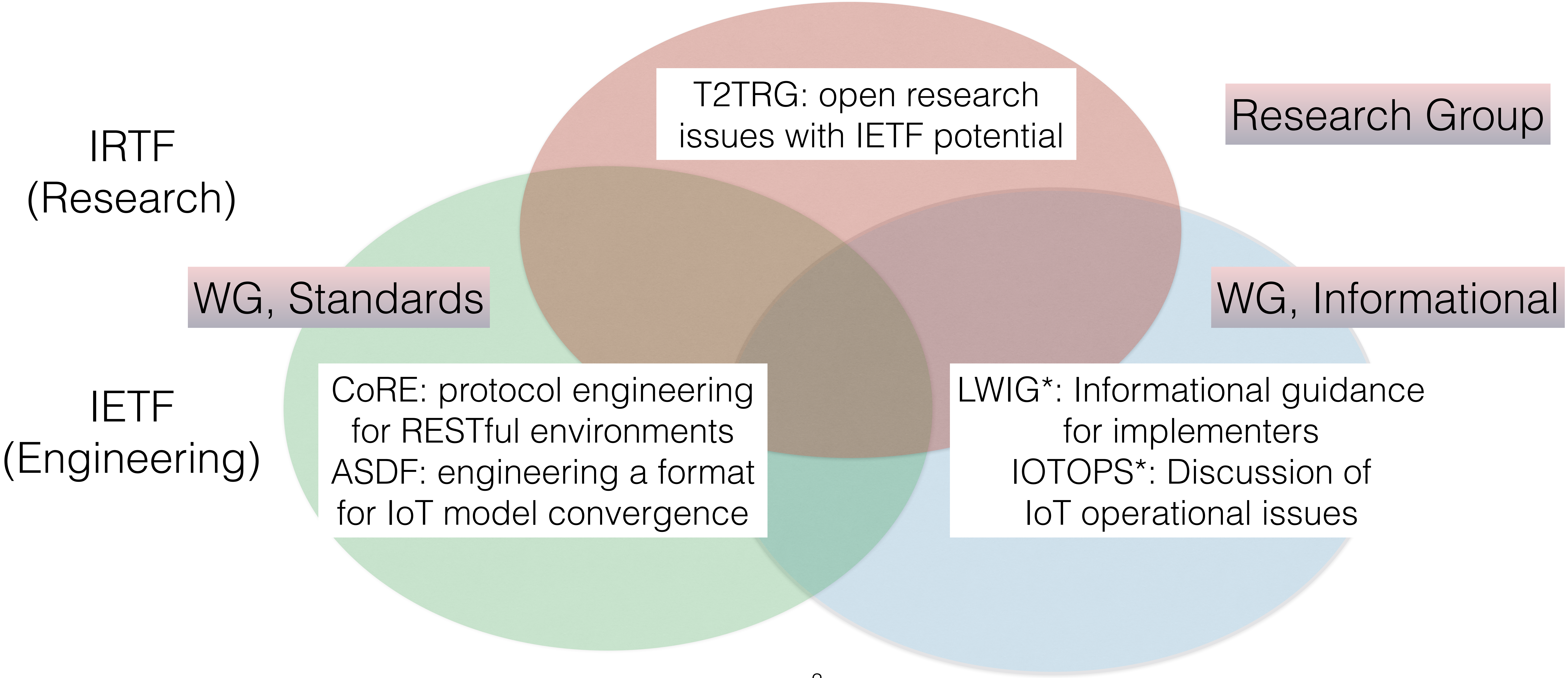
IETF/T2TRG update

W3C WoT Open Day 2021-06-28

Carsten Bormann

IRTF and IETF



IRTF
(Research)

IETF
(Engineering)

T2TRG: open research
issues with IETF potential

Research Group

WG, Standards

WG, Informational

CoRE: protocol engineering
for RESTful environments
ASDF: engineering a format
for IoT model convergence

LWIG*: Informational guidance
for implementers
IOTOPS*: Discussion of
IoT operational issues

IETF WGs and IRTF RGs

- IETF: ~14 IoT-related WGs (next slides), foundational WGs (e.g., TLS)
- IRTF: Research arm, work on more far-out issues, no standards
    - **T2TRG**: Thing-to-Thing Research Group (e.g., WISHI: wishi.space; Work on IoT Semantic/Hypermedia Interoperability)
    - DINRG: Decentralized Internet Infrastructure RG
    - COINRG: Computing in the Network RG
    - CFRG: Crypto Forum Research Group (e.g., HPKE, AEAD limits)

IoT-related IETF activities

- Adaptation layer: **6LO**, **6TISCH** (including join protocol), **LPWAN**,
  BOF: MADINAS (life goes on with random MAC addresses)
- Routing: **ROLL** (RPL), RAW* (Reliable and Available Wireless)
- Networks/operations:
  - ANIMA (e.g., BRSKI (RFC 8995) for automatable device identities)
  - **IOTOPS***, a general discussion group about ops aspects of IoT (including onboarding)
- Application layer: **CoRE** (CoAP, discovery), **CBOR** (representation),
  ASDF (next slides), JSONPath (“XPath for JSON”)
- Security: next slide
- Implementation advice: **LWIG***

IoT-related IETF activities: Security

- **COSE** (signing/encryption formats, ~ concise JOSE)
- **ACE** (~ OAuth, CWT)
- **SUIT**: Secure firmware updates
- **RATS**: Attestation
- **LAKE**: lightweight authenticated key establishment
- SACM: Enabling Security Automation, including SBOM-related formats
- DANISH*: using DANE (DNSSEC-based security) in IoT (BOF)
- DRIP: Drone identification
- TEEP: Protocols for speaking with trusted execution environments

SDF (ASDF WG) Update

SDF: One Data Model

- IoT standardization is dominated by **ecosystem**-specific SDOs
  - Ecosystem-specific data/interaction models,
    ecosystem-specific ways to document them
- IoT applications may need to work with *things* from multiple ecosystems:
  No single ecosystem can supply the whole variety needed
  - Can build protocol translators; harder to translate **hundreds** of data models
- One Data Model liaison group: People from different SDOs meet informally
  - Express hundreds of ecosystem-specific data models in **common format**
  - Work on merging and harmonizing data models
  - Make harmonized data models available for all SDOs (BSD license!)
  - Working in the open: https://github.com/one-data-model

SDF: The Simple Definition Format

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

- Defines **classes** of *things* (sdfObject, combine into sdfThing)

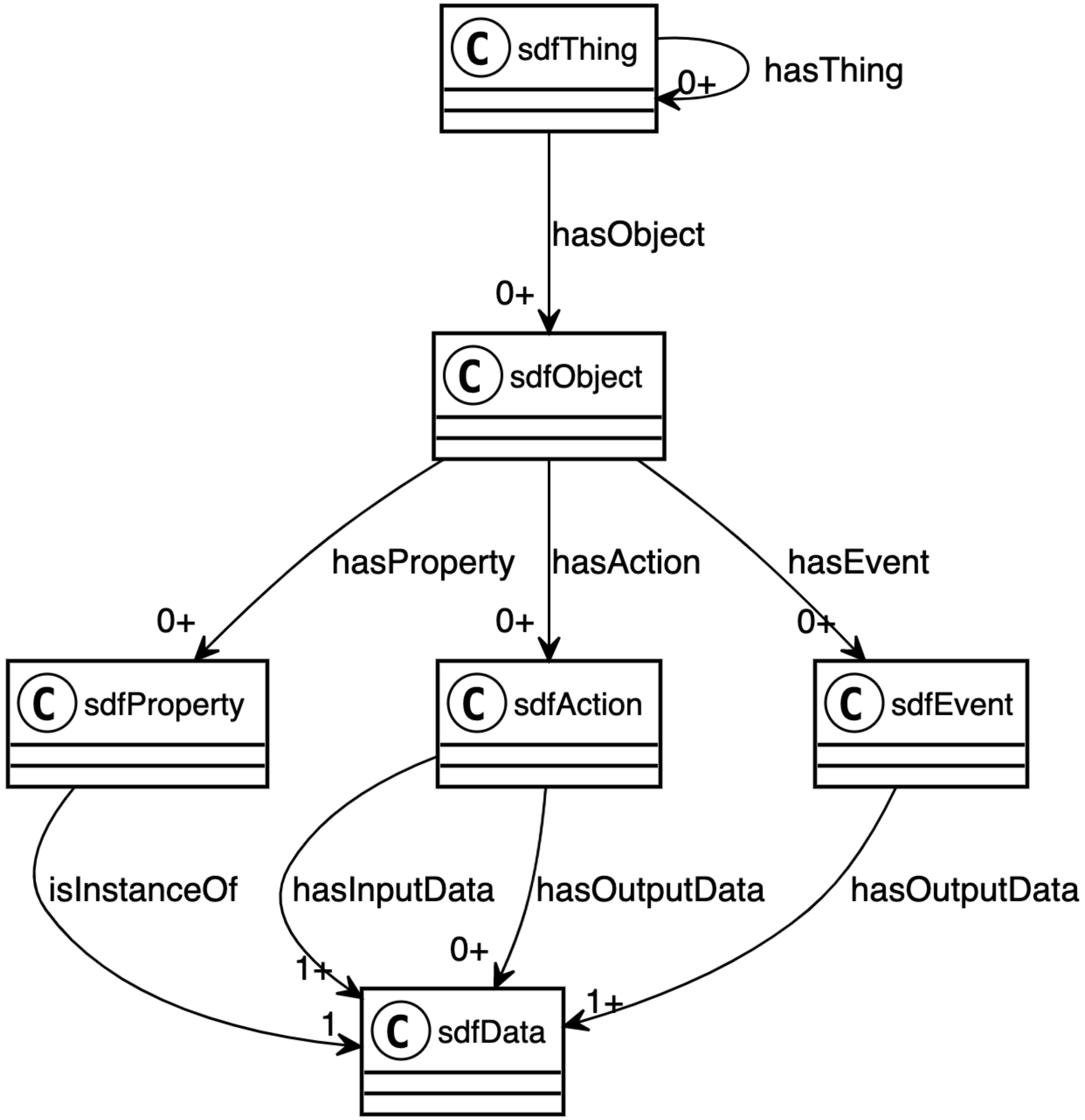
- Things don’t have data, they have **interactions** with their *clients*, provided by **affordances,** grouped into **interaction patterns**:
  For now, **Property, Action, Event**
  (Might add “data sheet” style affordances of electronic components.)

- Interactions input and output **data** (groupable into sdfData)

- Model = JSON text, can reference other models (JSON Pointer)

Composition: sdfThing, sdfProduct

- sdfObject definitions can be combined into top-level structures
- sdfThing can contain sdfObject and sdfThing
- sdfProduct similar, as a (not to be harmonized) top-level product definition



[draft-ietf-asdf-sdf-06]

Interaction Patterns

- SDF is about modeling data
- Interaction Patterns mostly defined along input and output data

Name	cf. REST	Initiative	Input	Output
Property	GET	Client	—	Data
Property (writable)	PUT	Client	Data	(Data)
Action	POST	Client	Input	Output
Event	?	Thing	—	Output

Action

- Actions can have different input and output data
- Some actions take time (not modeled): Initiative to return output moved to Thing (~ Event)

Name	cf. REST	Initiative	Input	Output
Property	GET	Client	—	Data
Property (writable)	PUT	Client	Data	Data
Action	POST	Client	Input	Output
Event	?	Thing	—	Output

Property

- Property is used for data items that can be read by the client
- Writable properties can also be “set” (no special output)
- Observable properties look like an Event

Name	cf. REST	Initiative	Input	Output
Property	GET	Client	—	Data
Property (writable)	PUT	Client	Data	(Data)
Property (observable)	GET (observe)	Client, Thing	—	Data
Event	?	Thing	—	Output

Event

- Least well-defined interaction pattern
- Is an Event just a notification (similar to observable property)?
- Are Events just status updates (temperature) or is any single one of them precious (coin insertion)?

Name	cf. REST	Initiative	Input	Output
Property	GET	Client	—	Data
Property (writable)	PUT	Client	Data	Data
Action	POST	Client	Input	Output
Event	?	Thing	—	Output

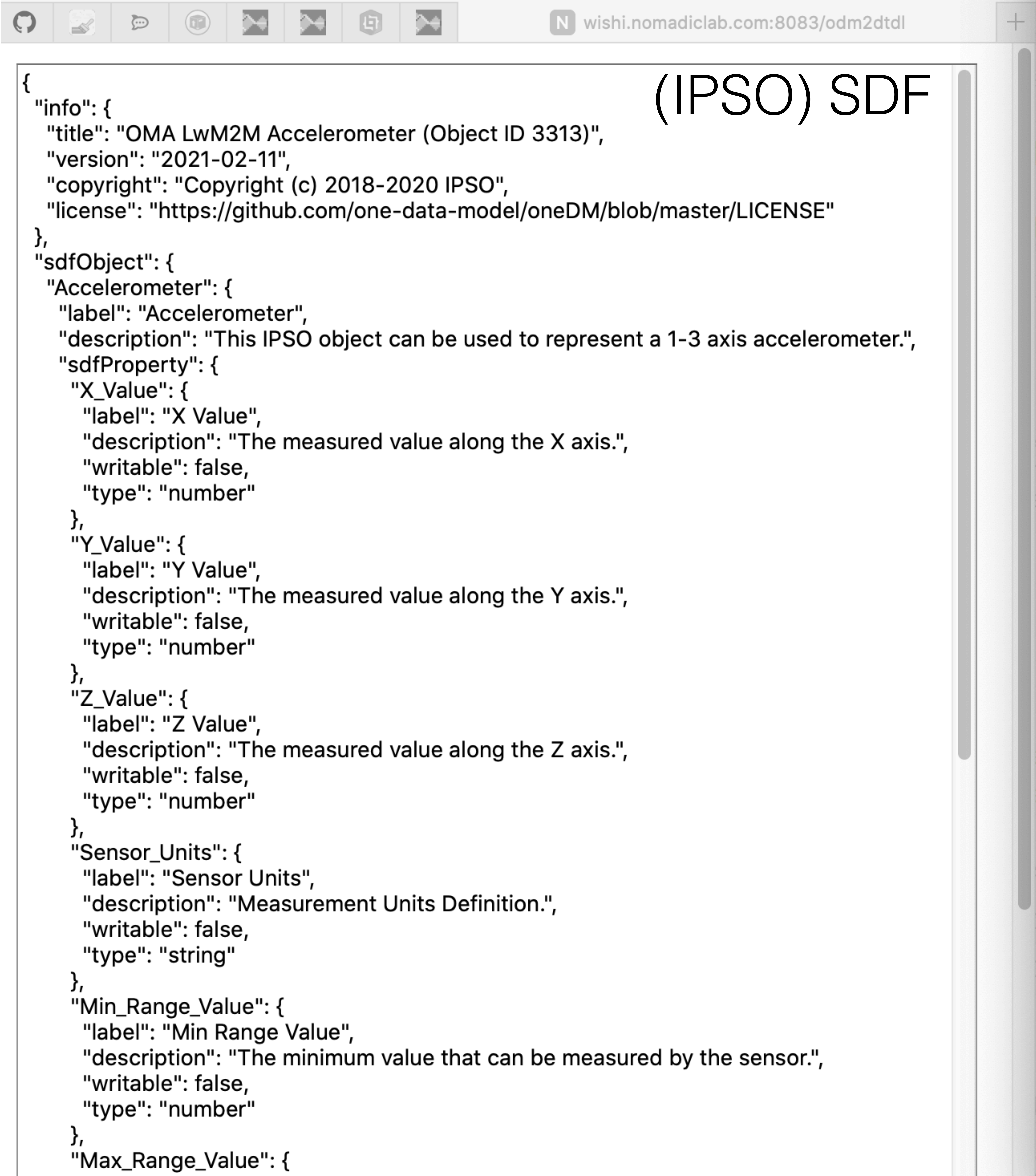
Data

- Data is defined by their *shape* (as in data definition/“schema” languages)
- Data definitions can be made inline in an affordance definition or separately, for later reference
- Definitions can use curated **subset** of json-schema.org terms, and/or SDF-specific terms such as contentFormat, nullable, scale…
- Mapping information (**protocol bindings**) helps bind these data to ecosystem specific formats and encodings

Work in Progress

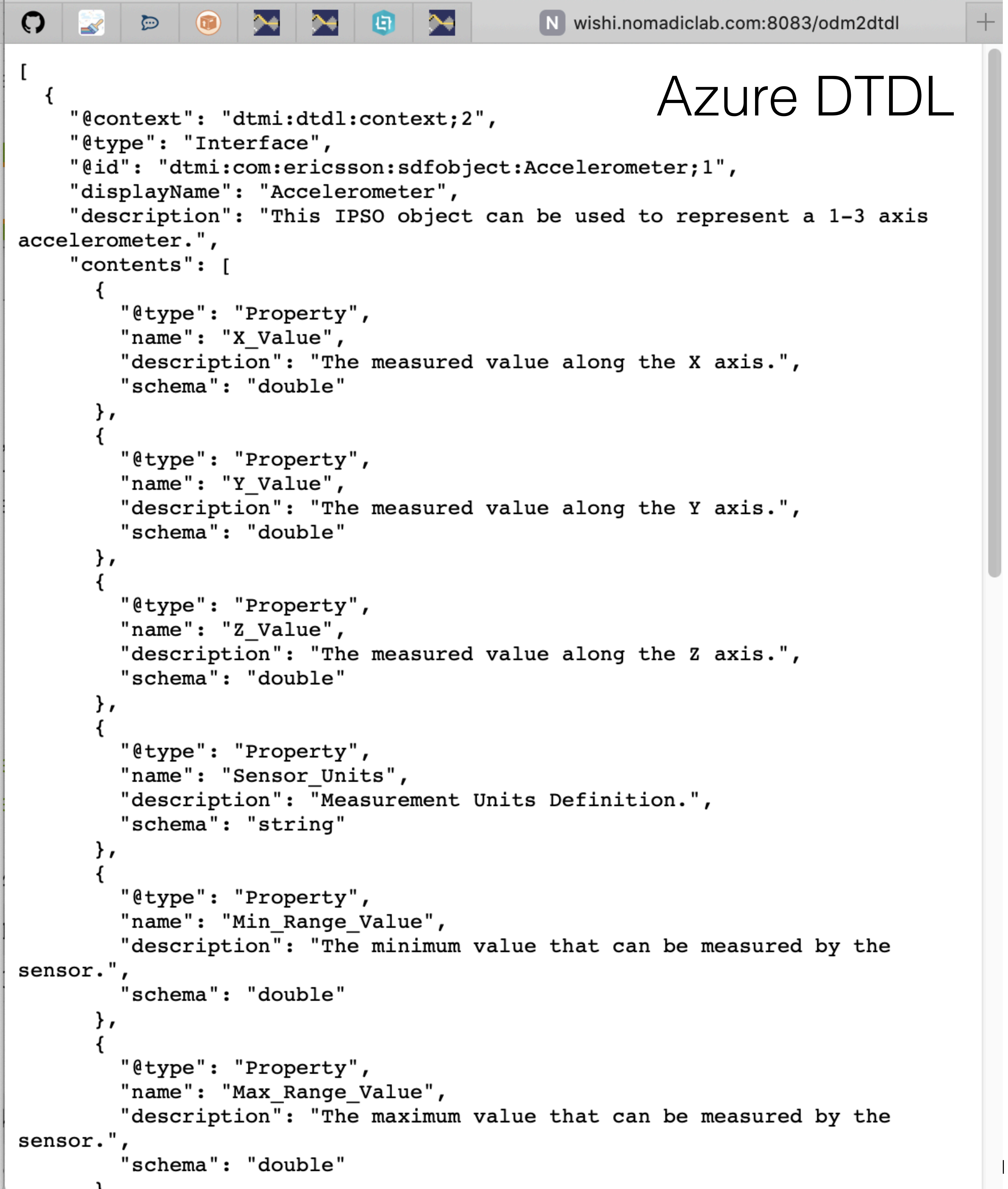
SDF: Status 2021-06-28

- SDF 1.1 has been stable since March (draft-ietf-asdf-sdf-05)
- Since, draft-ietf-asdf-sdf-06 (June) defines:
  (1) derived items (referencing and overriding qualities),
  (2) array-like sdfObjects (“outlet strip”)
- ~ 200 data models in playground, exploratory, unit_test repos
  - Ecosystem SDOs have developed tools to convert their corpus to SDF
  - Recent: Tools for converting to and from:
    Azure **DTDL** (Digital Twin Definition Language),
    IETF **YANG**



(IPSO) SDF

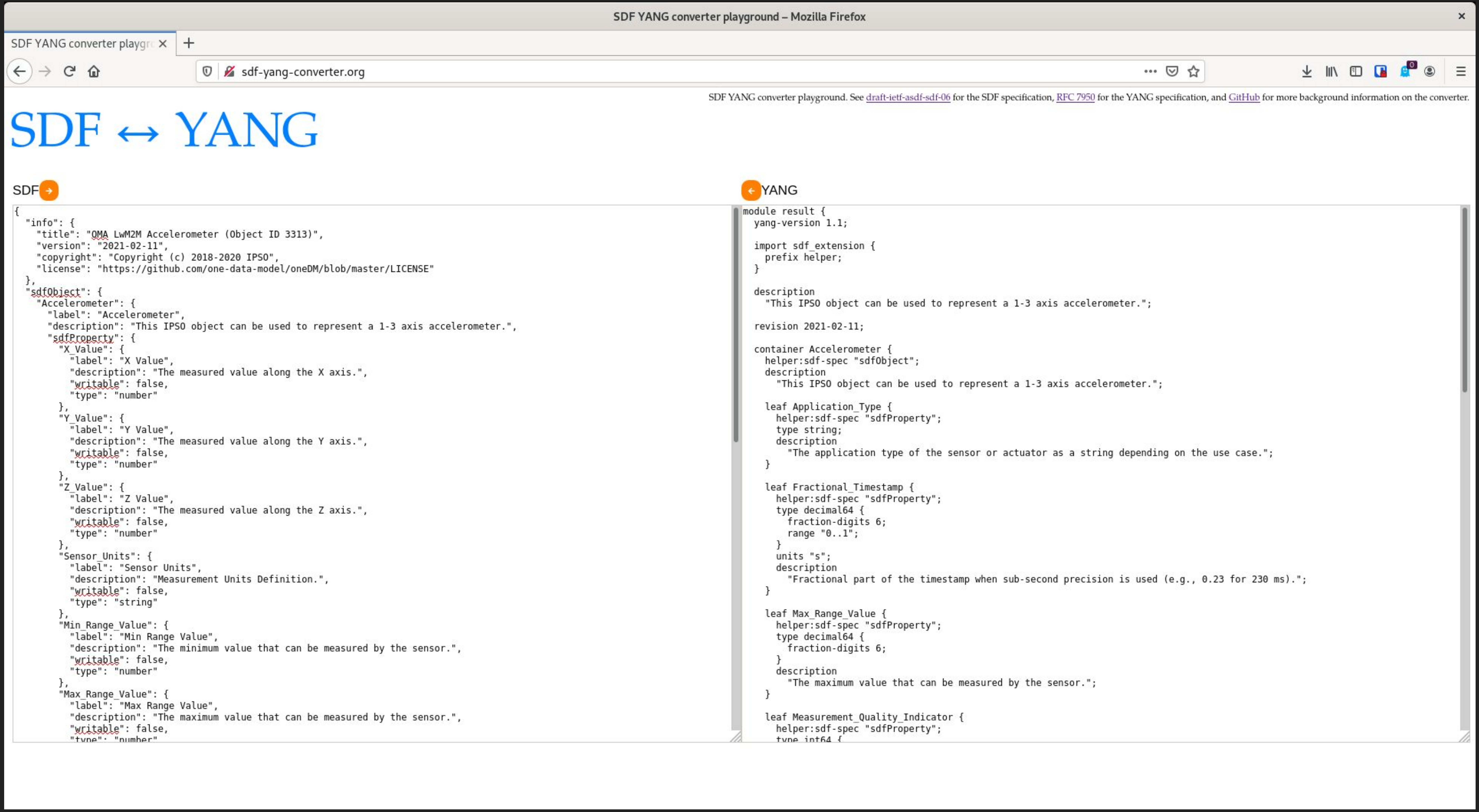
```
{
 "info": {
  "title": "OMA LwM2M Accelerometer (Object ID 3313)",
  "version": "2021-02-11",
  "copyright": "Copyright (c) 2018-2020 IPSO",
  "license": "https://github.com/one-data-model/oneDM/blob/master/LICENSE"
 },
 "sdfObject": {
  "Accelerometer": {
   "label": "Accelerometer",
   "description": "This IPSO object can be used to represent a 1-3 axis accelerometer.",
   "sdfProperty": {
    "X_Value": {
     "label": "X Value",
     "description": "The measured value along the X axis.",
     "writable": false,
     "type": "number"
    },
    "Y_Value": {
     "label": "Y Value",
     "description": "The measured value along the Y axis.",
     "writable": false,
     "type": "number"
    },
    "Z_Value": {
     "label": "Z Value",
     "description": "The measured value along the Z axis.",
     "writable": false,
     "type": "number"
    },
    "Sensor_Units": {
     "label": "Sensor Units",
     "description": "Measurement Units Definition.",
     "writable": false,
     "type": "string"
    },
    "Min_Range_Value": {
     "label": "Min Range Value",
     "description": "The minimum value that can be measured by the sensor.",
     "writable": false,
     "type": "number"
    },
    "Max_Range_Value": {
```



Azure DTDL

```
[
  {
    "@context": "dtmi:dtdl:context;2",
    "@type": "Interface",
    "@id": "dtmi:com:ericsson:sdfobject:Accelerometer;1",
    "displayName": "Accelerometer",
    "description": "This IPSO object can be used to represent a 1-3 axis
accelerometer.",
    "contents": [
      {
        "@type": "Property",
        "name": "X_Value",
        "description": "The measured value along the X axis.",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "Y_Value",
        "description": "The measured value along the Y axis.",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "Z_Value",
        "description": "The measured value along the Z axis.",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "Sensor_Units",
        "description": "Measurement Units Definition.",
        "schema": "string"
      },
      {
        "@type": "Property",
        "name": "Min_Range_Value",
        "description": "The minimum value that can be measured by the
sensor.",
        "schema": "double"
      },
      {
        "@type": "Property",
        "name": "Max_Range_Value",
        "description": "The maximum value that can be measured by the
sensor.",
        "schema": "double"
```

Converter Demo at sdf-yang-converter.org


SDF YANG converter playground – Mozilla Firefox
SDF YANG converter playgr
sdf-yang-converter.org

SDF YANG converter playground. See draft-ietf-asdf-sdf-06 for the SDF specification, RFC 7950 for the YANG specification, and GitHub for more background information on the converter.

SDF ↔ YANG

SDF →

```
{
  "info": {
    "title": "OMA LwM2M Accelerometer (Object ID 3313)",
    "version": "2021-02-11",
    "copyright": "Copyright (c) 2018-2020 IPSO",
    "license": "https://github.com/one-data-model/oneDM/blob/master/LICENSE"
  },
  "sdfObject": {
    "Accelerometer": {
      "label": "Accelerometer",
      "description": "This IPSO object can be used to represent a 1-3 axis accelerometer.",
      "sdfProperty": {
        "X_Value": {
          "label": "X Value",
          "description": "The measured value along the X axis.",
          "writable": false,
          "type": "number"
        },
        "Y_Value": {
          "label": "Y Value",
          "description": "The measured value along the Y axis.",
          "writable": false,
          "type": "number"
        },
        "Z_Value": {
          "label": "Z Value",
          "description": "The measured value along the Z axis.",
          "writable": false,
          "type": "number"
        },
        "Sensor_Units": {
          "label": "Sensor Units",
          "description": "Measurement Units Definition.",
          "writable": false,
          "type": "string"
        },
        "Min_Range_Value": {
          "label": "Min Range Value",
          "description": "The minimum value that can be measured by the sensor.",
          "writable": false,
          "type": "number"
        },
        "Max_Range_Value": {
          "label": "Max Range Value",
          "description": "The maximum value that can be measured by the sensor.",
          "writable": false,
          "type": "number"
```

← YANG

```
module result {
  yang-version 1.1;

  import sdf_extension {
    prefix helper;
  }

  description
    "This IPSO object can be used to represent a 1-3 axis accelerometer.";

  revision 2021-02-11;

  container Accelerometer {
    helper:sdf-spec "sdfObject";
    description
      "This IPSO object can be used to represent a 1-3 axis accelerometer.";

    leaf Application_Type {
      helper:sdf-spec "sdfProperty";
      type string;
      description
        "The application type of the sensor or actuator as a string depending on the use case.";
    }

    leaf Fractional_Timestamp {
      helper:sdf-spec "sdfProperty";
      type decimal64 {
        fraction-digits 6;
        range "0..1";
      }
      units "s";
      description
        "Fractional part of the timestamp when sub-second precision is used (e.g., 0.23 for 230 ms).";
    }

    leaf Max_Range_Value {
      helper:sdf-spec "sdfProperty";
      type decimal64 {
        fraction-digits 6;
      }
      description
        "The maximum value that can be measured by the sensor.";
    }

    leaf Measurement_Quality_Indicator {
      helper:sdf-spec "sdfProperty";
      type int64 {
```

ASDF/WISHI Hackathon Week

- **2021-07-19..-23**, starting with WISHI call on 2021-07-19 (1400Z?) (Week before IETF111 ➔ register for hackathon: $0 and get a T-Shirt :-)
- Continue work on converters for **SDF ⬌ other**, such as:
  - DTDL converter (http://wishi.nomadiclab.com:8083/odm2dtdl)
  - sdf-yang-converter.org
  - WoT TD
- continue development of semantic additions (the "mapping files")