One Data Model SDF: A brief tutorial and status

ASDF “Hallway meeting”, 2020-10-12

Carsten Bormann

The need for One Data Model

- IoT standardization is dominated by **ecosystem**-specific SDOs
- Each ecosystem has their own data models,
  and their own way 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

The One Data Model liaison group

- People from different SDOs meet in an informal liaison group
- Bring together hundreds of ecosystem-specific data models
  - Express 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
- Inevitably: standardize on a **common format**: SDF

SDF: The Simple Definition Format

- https://github.com/one-data-model/SDF
- Defines classes of *things* (sdfObject, combine into sdfThing)
- Things don't have data, they have **interactions** with their *clients(*)*, provided by **affordances**
- Interaction affordances grouped into **interaction patterns**:
  For now, **Property, Action, Event**
- Interactions input and output **data** (groupable into sdfData)

(*) Not a oneDM term

Overall Specification Structure

- One or more JSON documents; linked together with JSON pointers [RFC6901]
- An SDF 1.0 specification can **reuse** elements (such as sdfData definitions) of other SDF specifications
  - Goal: define a basic core set that every specification can reference (“common reusable definitions”)
  - Pretty much untested so far, develop further for **SDF 1.1**

SDF 1.1

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	Initiativ e	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

SDF 1.1

Data Model vs. Information Model (1)

- SDF 1.0 uses json-schema.org-style data modeling, enhanced by SDF qualities
- Really: This should be **information models** (RFC 3444):
  - Abstract from arbitrary representation decisions
  - Don’t commit to specific numbers, strings, etc. (**bindings** can do that)
  - Bind to **semantics** via RDF-style links

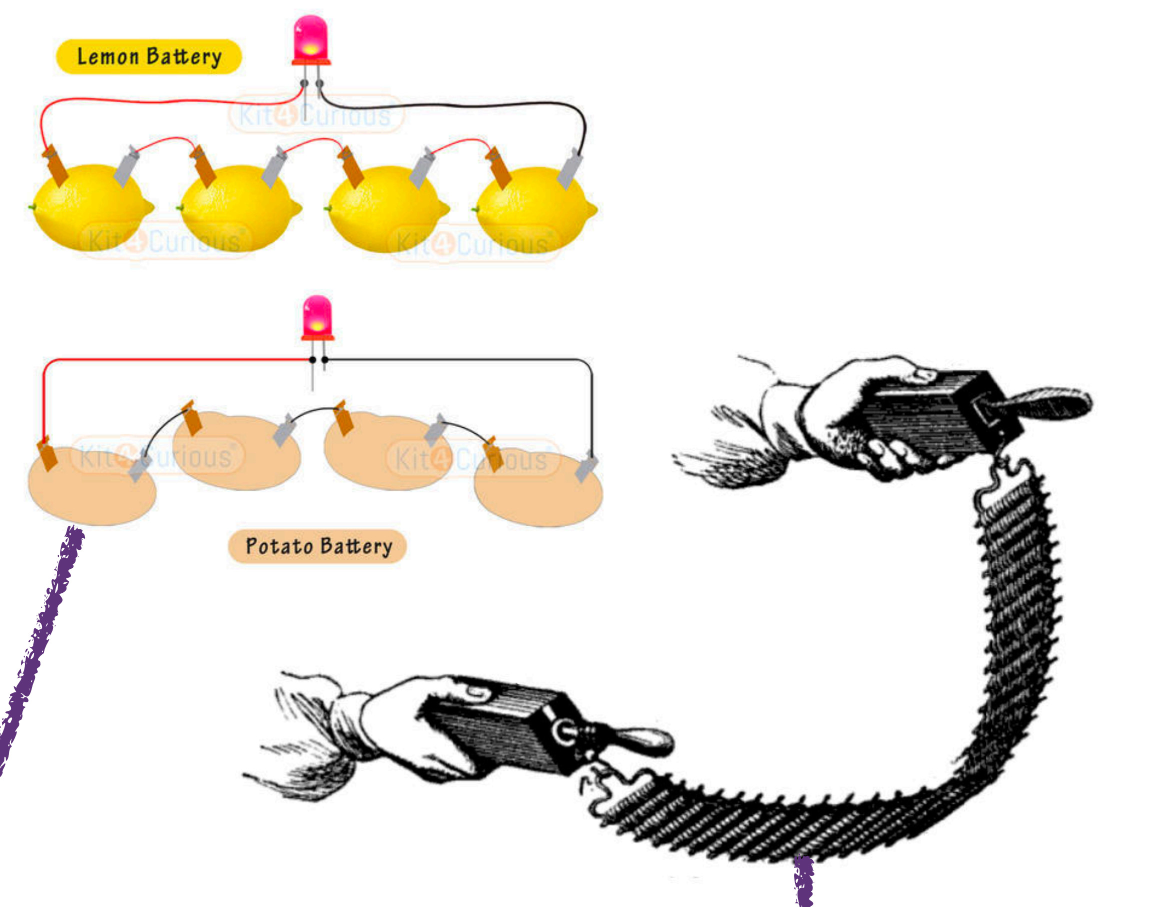
SDF 1.1

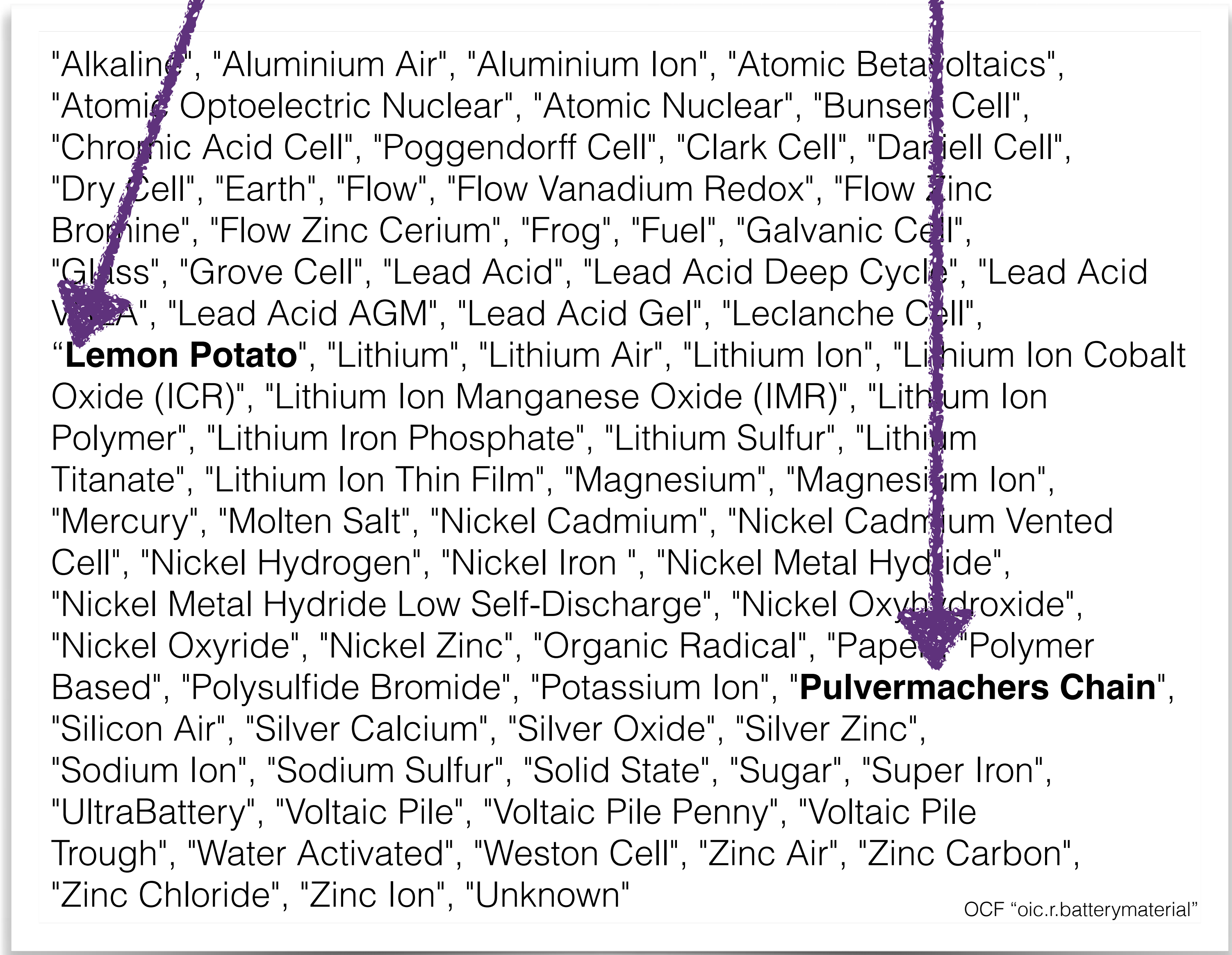
Data Model vs. Information Model (2)

- “Enums”: choices of values (strings, integers), each usually denoting some specific concept
- Information model: Don’t commit to specific representation (**bindings** can do that)
- Do bind to **semantics** via RDF-style links

SDF 1.1

- Enums in SDF 1.0 patterned after json-schema.org: but should it really be "General", "Fire", "Flood", "Weather", “Security"? How to provide extensibility for enums (IANA? URIs?) ➔ **SDF 1.1**

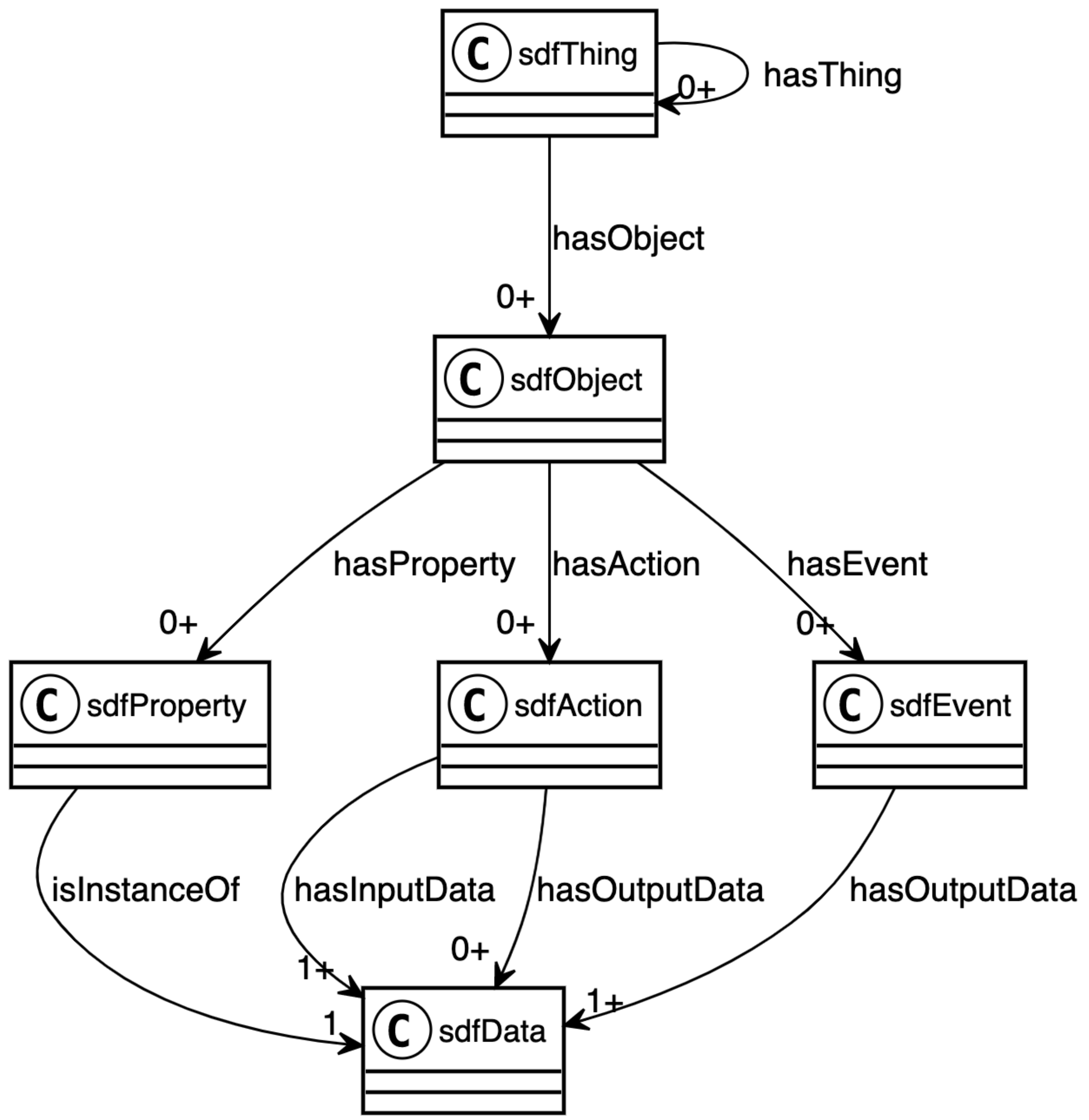



"Alkaline", "Aluminium Air", "Aluminium Ion", "Atomic Betavoltaics", "Atomic Optoelectric Nuclear", "Atomic Nuclear", "Bunsen Cell", "Chromic Acid Cell", "Poggendorff Cell", "Clark Cell", "Daniell Cell", "Dry Cell", "Earth", "Flow", "Flow Vanadium Redox", "Flow Zinc Bromine", "Flow Zinc Cerium", "Frog", "Fuel", "Galvanic Cell", "Glass", "Grove Cell", "Lead Acid", "Lead Acid Deep Cycle", "Lead Acid VRLA", "Lead Acid AGM", "Lead Acid Gel", "Leclanche Cell", "**Lemon Potato**", "Lithium", "Lithium Air", "Lithium Ion", "Lithium Ion Cobalt Oxide (ICR)", "Lithium Ion Manganese Oxide (IMR)", "Lithium Ion Polymer", "Lithium Iron Phosphate", "Lithium Sulfur", "Lithium Titanate", "Lithium Ion Thin Film", "Magnesium", "Magnesium Ion", "Mercury", "Molten Salt", "Nickel Cadmium", "Nickel Cadmium Vented Cell", "Nickel Hydrogen", "Nickel Iron ", "Nickel Metal Hydride", "Nickel Metal Hydride Low Self-Discharge", "Nickel Oxyhydroxide", "Nickel Oxyride", "Nickel Zinc", "Organic Radical", "Paper", "Polymer Based", "Polysulfide Bromide", "Potassium Ion", "**Pulvermachers Chain**", "Silicon Air", "Silver Calcium", "Silver Oxide", "Silver Zinc", "Sodium Ion", "Sodium Sulfur", "Solid State", "Sugar", "Super Iron", "UltraBattery", "Voltaic Pile", "Voltaic Pile Penny", "Voltaic Pile Trough", "Water Activated", "Weston Cell", "Zinc Air", "Zinc Carbon", "Zinc Chloride", "Zinc Ion", "Unknown"

OCF “oic.r.batterymaterial”

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-onedm-t2trg-sdf-00]

Composition (SDF 1.1)

SDF 1.1

- Property data and Action/Event input/output often have multiple fields/members
- SDF 1.0 deliberately does **not** include json-schema.org “object” type
- Question for SDF 1.1: What is the **right** way to handle data composition? (i.e., within a property — or are there composite properties?)

Specifying SDF

- SDF specs are JSON documents, can be specified in a data definition language
  - Using CDDL:
    https://github.com/one-data-model/SDF/blob/master/sdf-feature.cddl
  - Translated into json-schema.org "JSON Schema" format:
    https://github.com/one-data-model/SDF/blob/master/sdf-framework.jso.json and
    https://github.com/one-data-model/SDF/blob/master/sdf-validation.jso.json
    (do not confuse with the selected json-schema.org terms used **inside** SDF)
- Of course, also needs semantics (in English text);
  - Definition: https://github.com/one-data-model/SDF/blob/master/sdf.md
- De-facto specifics via tooling (e.g., at https://github.com/one-data-model/tools)

Status 2020-10-12

- SDF 1.0 has been stable since June (draft-onedm-t2trg-sdf-00)
  - ~ 200 data models in playground, exploratory, unit_test repos
  - Ecosystem SDOs have developed tools to convert their corpus to SDF
- Work on establishing ASDF WG, defining OneDM processes since
- Next: Define objectives and solutions for SDF 1.1; then decide!
- Test new ideas at Hackathon week before IETF109, 2020-11-03..-09
  https://trac.ietf.org/trac/ietf/meeting/wiki/109hackathon:
  ASDF: Getting ready for SDF 1.1