One Data Model SDF: A brief tutorial and status

T2TRG summary meeting @ IETF 107+, April 14, 2020

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/language
- Defines classes of *things* (odmObject, combine into odmThing)
- 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 odmData)

(*) Not a oneDM term

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)?

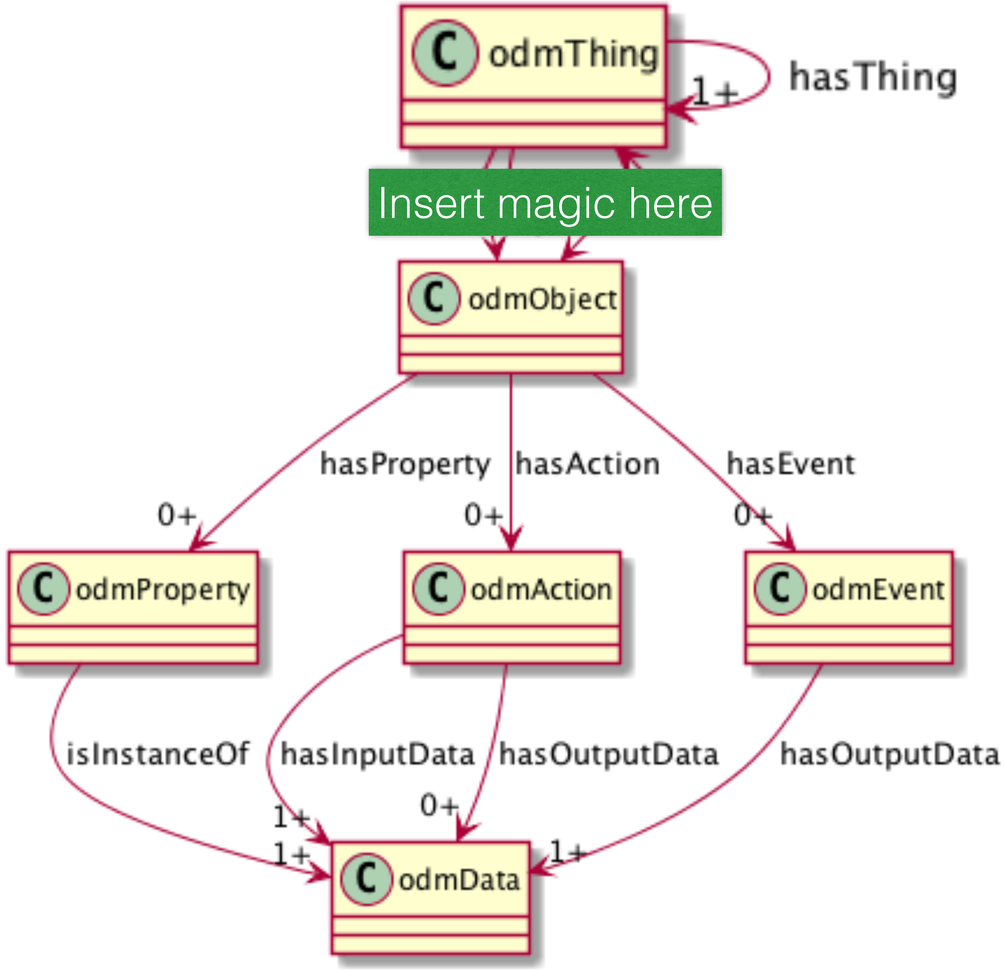
Name	cf. REST	Initiative	Input	Output
Property	GET	Client	—	Data
Property (writable)	PUT	Client	Data	Data
Action	POST	Client	Input	Output
Event	?	Thing	—	Output

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 subset of json-schema.org terms,
  and/or SDF-specific terms such as contentFormat, nullable, scale…

odmThing, odmProduct

- odmObject definitions can be combined into top-level structures
- odmThing can contain odmObject and odmThing
- odmProduct similar, as a (not to be harmonized) top-level product definition



[figure modified from Michael J Koster]

Overall Specification Structure

- One or more JSON documents; linked together with JSON pointers [RFC6901]
- SDF specification can reuse elements (such as odmData definitions) of other SDF specifications
  - Goal: define a basic core set that every specification can reference (“common reusable definitions”)

Specifying SDF

- SDF specs are JSON documents, can be specified in a data definition language
  - https://github.com/one-data-model/language/blob/master/sdf-schema.json using json-schema.org “JSON Schema” format
  - Do not confuse with selected json-schema.org terms used **in** SDF
- Of course, also needs semantics
  - Definition: https://github.com/one-data-model/language/blob/master/sdf.md
  - Best practices: https://github.com/one-data-model/language/blob/master/bestpractices.md
- De-facto specifics via tooling at https://github.com/one-data-model/playground

Status 2020-04-14

- SDF spec is stable enough to submit data models
  - Stabilized in Stockholm F2F meeting (2019-10-01..-04)
  - **Several hundred** data models now collected at playground
  - Ecosystem SDOs have developed tools to convert their corpus to SDF
- Specification itself needs more cleanup and an editorial round
  - 4-day online conference tentatively scheduled for weeks 19–21
  - Should be completed by **end of May**

What's next

- Continue implementation work on the model-*consuming* side (e.g., WISHI hackathon on 2020-04-24)
- Solve remaining issues for SDF 1.0 (to be done in liaison group)
  - Existing "playground" definitions serve as a corpus
  - Can fix all of these in place if needed for a non-backwards compatible change!
- Next: Find a venue for standardization of SDF?