

# Data model component reusability

T2TRG IETF 104 pre-meeting

22.3.2019

Ari Keränen

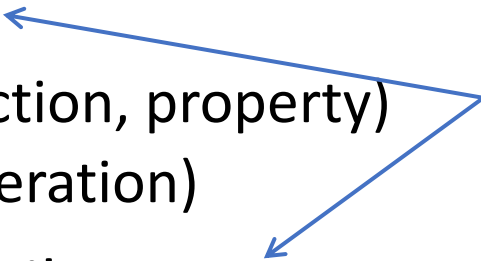
# Reusability for data models

- Re-usability: design once, use many times
  - Save time and money
  - Promote good design patterns
  - Re-use of tools, processes, etc.
  - Facilitate interoperability
- Levels of reusability
  - **Within a model / system**
  - Between models and systems
    - Enables (less lossy) translation

# Goal of this exercise

- Guidance for design of re-usable data model components
- What have we learned?
  - What works, what not?
  - What do you gain / lose with certain choices?

# Levels of modeling

- (System)
  - Device (thing)
  - Object (capability)
  - Resource
    - (event, action, property)
    - (path, operation)
  - Value & attributes
    - (data item, payload)
- Focus today
- 
- A blue arrow originates from the text 'Focus today' and splits into two branches. One branch points to the 'Resource' level, and the other points to the 'Value & attributes' level.

# ”Resource level” features

- The value
- Identifier (machines) & name (humans)
- Data type: number, string, boolean, time, etc.
- Operations: read/observe, write, execute
- Description: human readable information
- Range: min and max values
- Unit: celsius, meter, lux
- Mandatory / optional (within object)
- What else?
  - Further data shape constraints, e.g., step?
  - Feature of Interest (e.g., iotschema); could also be object level

# Example: IPSO "current temperature" re-usable resource

- Identifier (machines) & name (humans)
  - 5700, "Current temperature"
- Data type
  - Float
- Operations (read, write, execute)
  - Read
- Description
  - "Current temperature of the sensor"
- Range
  - Undefined
- Unit
  - Discoverable from Unit resource

# How much fixed features in design time?

- (Nothing fixed)
  - No benefits for re-usability?
- All features always fixed
  - Limits re-usability
  - Simple
- Some features fixed; where's the sweet spot?
  - Static (model) decision which are fixed
  - Features fixed if defined in registration
  - Discover which features fixed
  - Discover values for attributes

# To fix or not to fix?

- Identifier (machines) & name (humans)
  - "Same resource, different context": **fixed**
- Data type
  - "Same purpose": **fixed**
- Operations (read, write, execute)
  - Same purpose again? Some object may allow write while others not
  - Current **measured** value no point writing to?
  - **Sometimes** fixed?



# To fix or not to fix?

- Description
  - Two types of description: generic and context specific?
- Range
  - Not always applicable
  - Fixed when defined?
- Unit
  - Like range?

# Cost of not fixing

- Less fixed -> less information conveyed by IDs
- Need discovery or out-of-band information
  - communication overhead
  - implementation complexity