Bluetooth – SDF conversion experiences

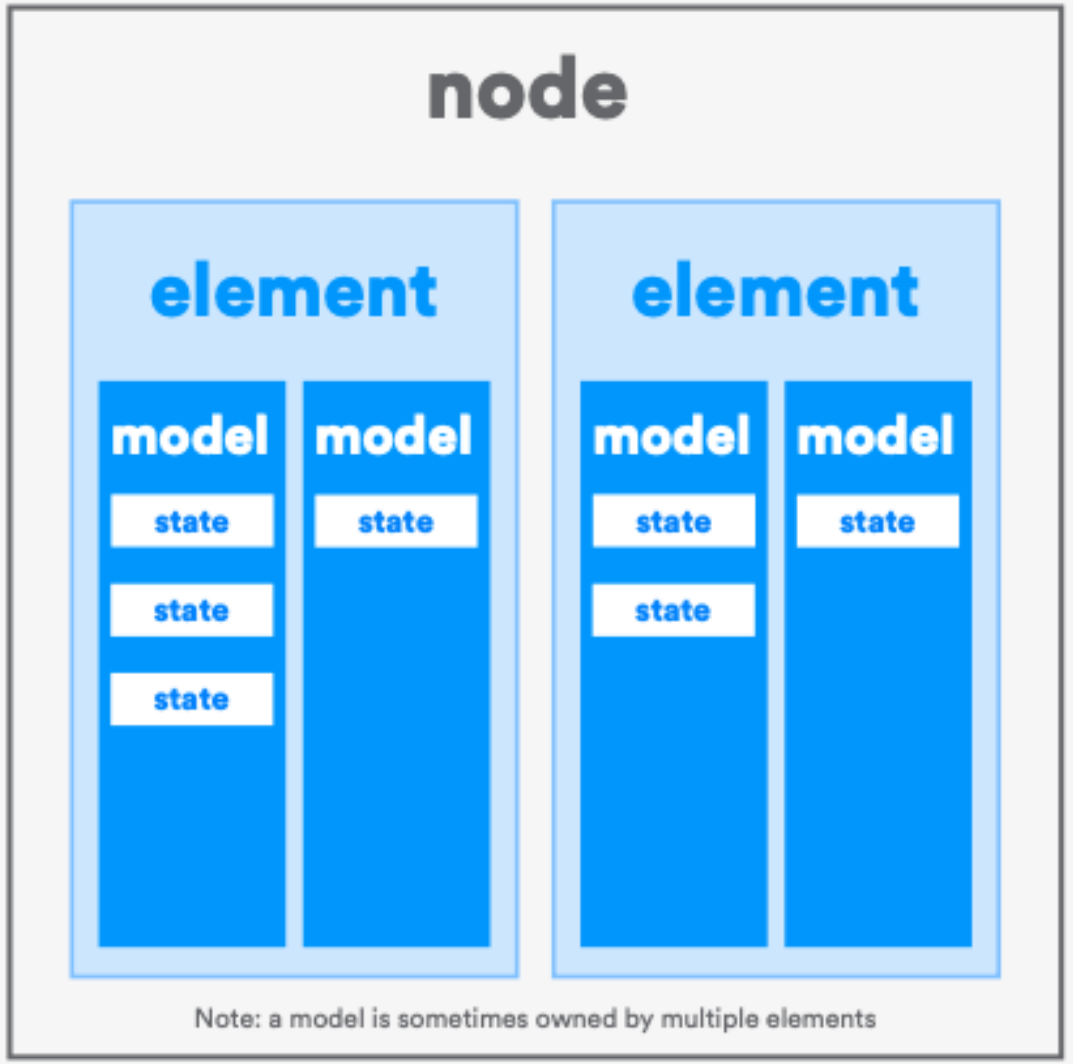
Petri Laari
21.8.2023 OneDM meeting

Target of the activity

- To test the capabilities of SDF as a meta model for different ecosystems
- Bluetooth is one of the biggest ecosystems of devices
  - Can SDF be used to describe Bluetooth nodes
  - In this work, we concentrated only on the Bluetooth Mesh models
  - Before Mesh models, all Bluetooth devices were modelled with “services” and “characteristics”
    - Some of the characteristics are used also by some Mesh models
  - Bluetooth radio is not considered

Bluetooth Mesh Node

- Mesh Node
  - Consists of one or more elements
- Element
  - "model of models"
  - Supports multiple models
- Mesh model
  - 52 Standard mesh models (SIG models)
  - Clients, Servers, Setup Servers
- State
  - Indicates the condition of the device



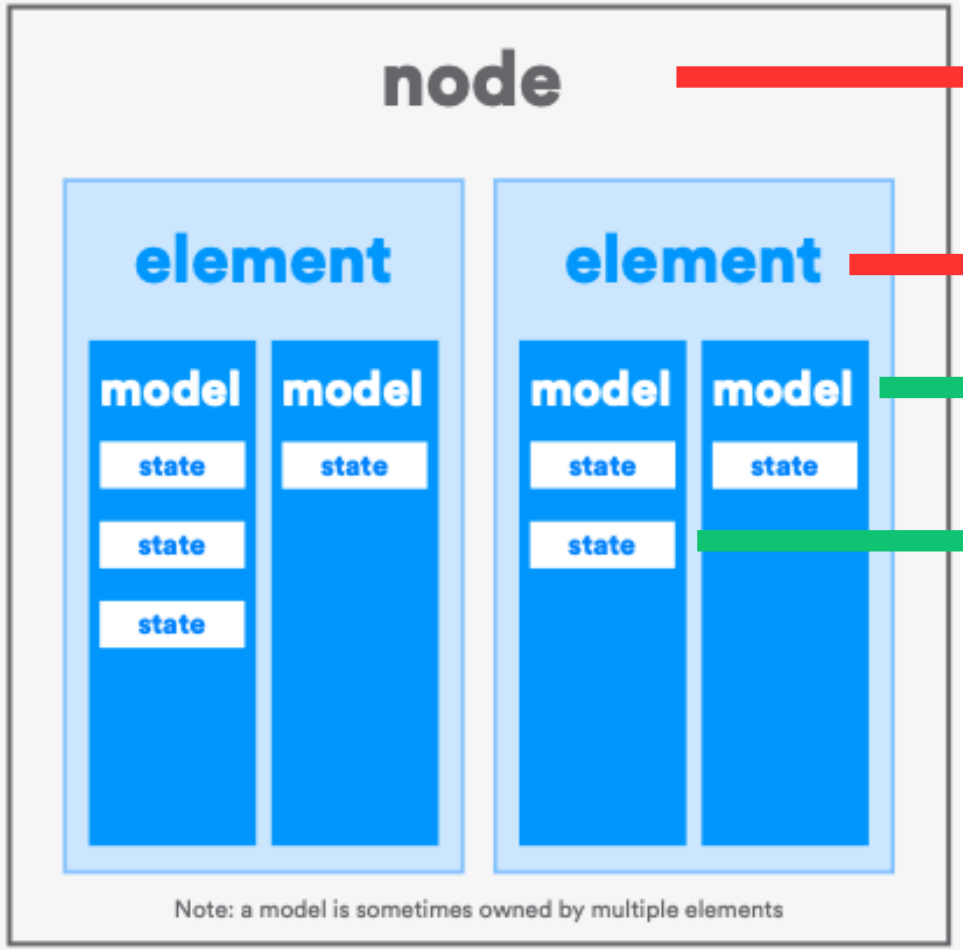
Mesh models

- Four categories
  - Generics
    - On/Off, levels, power on/off, battery, …
  - Sensors, Time and scenes
    - Setting up e.g. a room for particular purpose
  - Lighting
    - Controlling the lighting in various ways

Characteristics

- Services and Characteristics are/were used pre-mesh models to describe a device
- Some Mesh models utilize a set of characteristics
  - Device properties
  - Describe the server operations
- There are two types of characteristics (looking from the SDF point of view)
  - Simple ones; contain one descriptor (value)
  - Complex ones; contain multiple descriptors
- Characteristics are available as YAML files

Suggested high level mapping: BT Mesh <-> SDF



sdfThing

sdfThing (nested)

sdfObject

sdfProperty
(sdfAction, sdfEvent, sdfData)

Characteristics and SDF

- In Mesh models, all used characteristics are simple ones
  - Each characteristic can be represented using a single sdfProperty in the SDF
- Complex characteristics
  - *Out of the scope of this work!*
  - Could be either
    - Converted to sdfObject, with multiple sdfProperties
    - Opened in the related SDF directly as multiple sdfProperties

Example characteristic: Time msec 24

```
characteristic:
  identifier:
org.bluetooth.characteristic.time_millisecond_24
  name: Time Millisecond 24
  description: |-
    The Time Millisecond 24 characteristic is used to
represent a period of time with a resolution of 1
millisecond.
  structure:
    - field: Time Millisecond 24
      type: uint24
      size: "3"
      description: |-
        Unit is second with a resolution of 0.001.
        Minimum: 0
        Maximum: 16777.214
        Represented values: M = 1, d = -3, b = 0
        Unit: org.bluetooth.unit.time.second
        A value of 0xFFFFFF represents 'value is not known'.
```

```
{
  "sdfProperty": {
    "Time Millisecond 24": {
      "description": "Unit is second with a resolution of
          0.001.Minimum: 0 Maximum: 16777.214 Represented
          values: M = 1, d = -3, b = 0 Unit:
          org.bluetooth.unit.time.second A value of
          0xFFFFFF represents 'value is not known'.",
      "type": "integer",
      "sdfChoice": {
        "value": {
          "minimum": 0,
          "maximum": 16777215
        },
        "value not known": {"const": 16777216}
      }...
```

- Unit: as text in description; not parsed now
- Scaling information: to mapping file?

Proof of concept implementation

Proof of Concept (PoC) implementation

- There are no Mesh models in machine readable format
  - Definitions in specifications
  - Automatic parsing and conversion from PDF files not feasible
- Simple solution to start with: create manually SDF files from the Mesh model specifications
  - Test the conversion possibilities

Simple YAML design for the Mesh models

- Solution for the PoC implementation
  - Created our own simplified design of the Mesh models in YAML
- An example of a YAML model
  - May miss details or definitions
  - Only for the Proof of concept work
  - YAML selected because the characteristics files were in YAML => same conversion mechanism

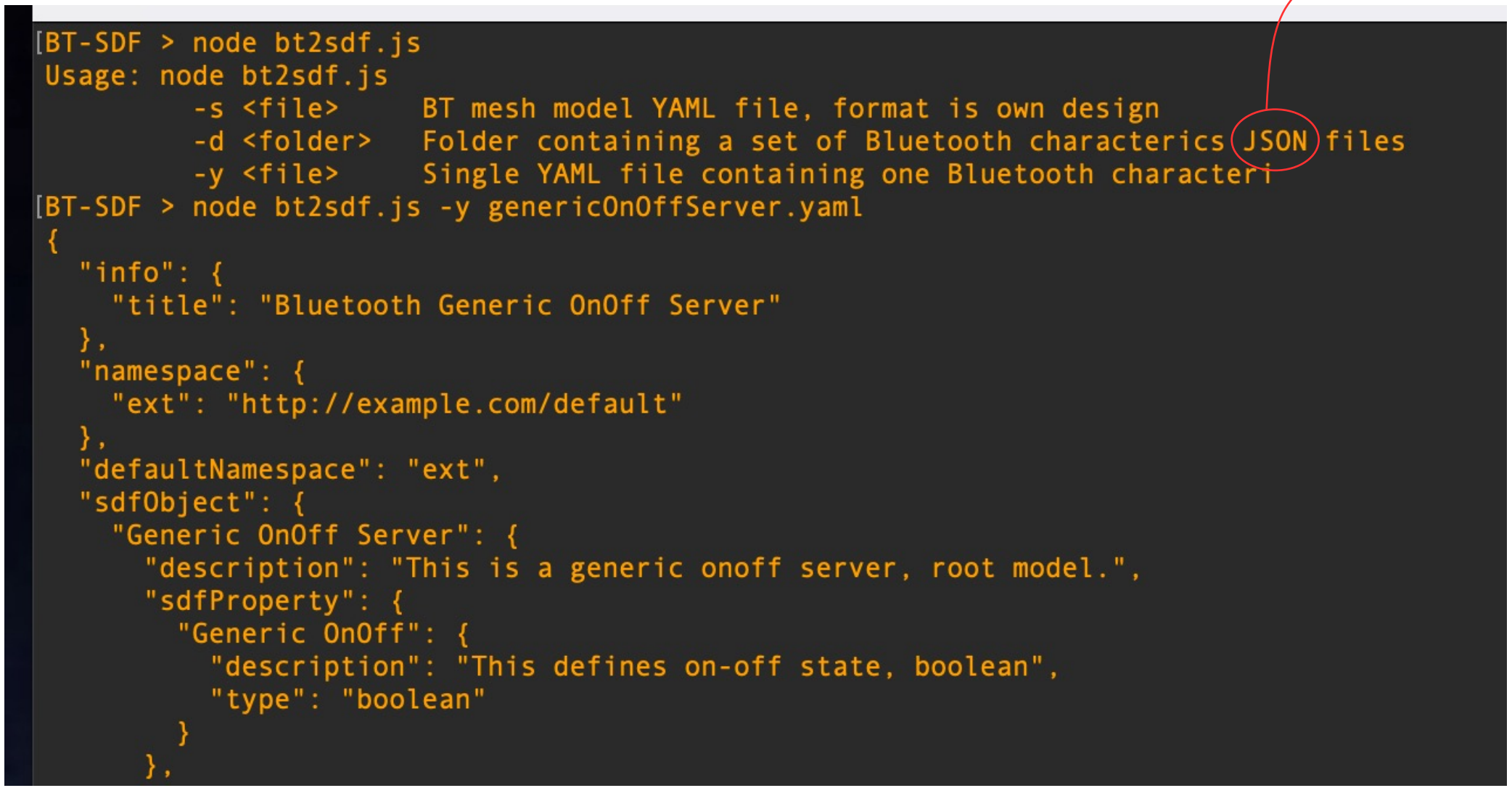
```
model:
  identifier:
org.bluetooth.meshmodel.generic_onoff_server
  name: Generic OnOff Server
  description: |-
    This is a generic onoff server, root model.
  states:
    - name: Generic OnOff
      description: |-
        This defines on-off state, boolean
      type: uint8
      values:
        - value: '0'
          description: Off
        - value: '1'
          description: On
  messages:
    - message: Generic OnOff Get
      description: |-
        Return the current GenericOnOff state value
      fields:
        - field: Response
          direction: out
          outdata: Generic OnOffStatus
...
```

Conversion from BT Mesh to SDF

- What has been done
  - Initially, implemented a conversion from the Characteristic to sdfProperties
    - Single files, or the whole directory into a single SDF file with multiple sdfProperties
    - This was a good rehearsal, but not exactly what we were after
  - Continued with making our own YAML specification of a Mesh model and implementing a conversion from that to sdfObject
- Potential next steps (no real plans at the moment)
  - Add more features from different Mesh models
  - Requires first defining them in YAML and then implementing the conversion
  - In addition to PoC, there is no real urge to implement more
    - Our simple YAML design may be wrong (in a way that it is not flexible for all the models)
    - Implementation would be broken in future if the design is something completely different

Proof of Concept (PoC) implementation

Source format to be changed to YAML



```
[BT-SDF > node bt2sdf.js
 Usage: node bt2sdf.js
          -s <file>     BT mesh model YAML file, format is own design
          -d <folder>   Folder containing a set of Bluetooth characterics JSON files
          -y <file>     Single YAML file containing one Bluetooth characteri
[BT-SDF > node bt2sdf.js -y genericOnOffServer.yaml
 {
   "info": {
     "title": "Bluetooth Generic OnOff Server"
   },
   "namespace": {
     "ext": "http://example.com/default"
   },
   "defaultNamespace": "ext",
   "sdfObject": {
     "Generic OnOff Server": {
       "description": "This is a generic onoff server, root model.",
       "sdfProperty": {
         "Generic OnOff": {
           "description": "This defines on-off state, boolean",
           "type": "boolean"
         }
       },
```

Open questions and discussion topics

Discussion topics / open issues
Bluetooth related issues

- Mesh model subscribe / publish defined as requirement for some models
  - How does subscription and publication work related this concept in real life?
  - No messages defined for subscription, status publication with sdfEvent when status changes?
- Bound states; how these can be handled?
  - Are these left to the implementor to decide how to do it? BT Specification defines how one state affects another one, but there is no defined mechanism how these can be implemented.
  - Formulas and calcuations in SDF *mapping file?*
  - sdfRelation between states?

Discussion topics / open issues
SDF related issues

- sdfRef usage when an object extends another object
  - "sdfRef": "ns://example.com/bluetooth/meshmodels/basemode" ?
  - The extended model can be copied directly into this model (as done in the Bluetooth Mesh also)
- Data type conversion
  - Currently: Numeric values, e.g. uint8 => type integer, min = 0, max = 255
    - Allows also reverse translation
  - Some uint8 fields define boolean (0=off, 1=on, 2-255 prohibited)
    - How should this be presented (in reversable way) in SDF so that it can be reversed?
    - Are booleans always uint8 in Bluetooth?
    - sdfChoice? Off = 0; On = 1; prohibited; 2-255

Discussion topics / open issues
SDF related issues

- How to express the requirement to implement another model on the same node as “this” model?
  - sdfRelation?
- Characteristics:
  - The ones used with Mesh models are trivial: each is one sdfProperty
  - More complex characteristics probably to be converted to sdfObjects, *but they are **out of the scope of this work, they are not used in Mesh models***
- TID: Transaction identifier
  - In message data fields in Mesh, but should these be in sdfProperties or in mapping files?
- Mesh models: *Set*—messages are defined with or without acknowledgement (two separate messages)
  - How these should be handled in SDF?
  - Two separate sdfActions? Allows reverse conversion SDF -> BT

Discussion topics / open issues
SDF related issues

- Scaling of values (for some states)
  - State values are the base values, and the scaling is defined in the Bluetooth documentation
  - How the scaling should be defined in SDF? Mapping file to contain the scaling formula?
    - Or should the values be the real ones when describing with SDF?
      - 0x0000 – 0xFFFE with scale factor 0.01 => 0 – 655.34
      - 0xFFFF "value not known" => 655.35

Discussion topics / open issues
SDF related issues

- Device Properties
  - Define how the device acts when messages arrive (internal configuration)
  - Defined by characteristics, so they can be converted to SDF (not in the examples at the moment)
  - Example:
    - Light LC Server:
      - Device properties can be queried from the Server
    - Light LC Setup Server
      - Modifying the device properties
  - *TBD: if needed, I can extend our YAML and implement also needed parts in the PoC implementation*

Conclusions (so far)

- At the moment, it looks like SDF can cope with all the Bluetooth Mesh model details
- While the BT Mesh specification does not necessarily give strict design to models, there may be things that affect the SDF part also
- Note: we have to go through the open issues still