# TD Payload Pattern

Sebastian Kaebisch

# Motivation

- Many existing IoT systems have defined an own message pattern that have to be followed and exchanged between Thing and client

- E.g., https://github.com/glenndehaan/ikea-tradfri-coap-docs#payload

```
{
  "3311": [
    {
      "5850": 1, // on / off
      "5851": 254, // dimmer (1 to 254)
      "5706": "f1e0b5", // color in HEX (Don't use in combination with: color X and/or color Y)
      "5709": 65535, // color X (Only use in combination with color Y)
      "5710": 65535, // color Y (Only use in combination with color X)
      "5712": 10 // transition time (fade time)
    }
  ]
}
```

- Thing Description can be defined to provide a status of the bulb and actions to change the state like on/off, dim value, etc.

# TD Snippet

```
"properties": {
   "state": {
    "description": "LED state",
    "type": "object",
    "properties": {
     "3311": {
       "items": [
        {
         "type": "object",
         "properties": {
          "5851": {
           "title": "on/off state",
           "type": "integer",
           "minimum": 0, "maximum": 1
          }
         }
        },
        {
         "type": "object",
         "properties": {
          "5851": {
           "title": "dim state",
           "type": "integer",
           "minimum": 1, "maximum": 254
          }
         }
        },
        {
         "type": "object",
         "properties": {
          "5706": {
           "title": "color temp as HEX RGB",
           "type": "string",
           "enum": ["f5faf6","f1e0b5","efd275"]
          }
         }
...
```

ID is / has to provided in each message

```
...
"actions": {
  "dim": {
    "input": {
      "description": "set the dim level",
      "type": "object",
      "properties": {
        "3311": {
          "items": [
            {
              "type": "object",
              "properties": {
                "5851": {
                  "title": "dim state",
                  "type": "integer",
                  "minimum": 1,
                  "maximum": 254
                },
                "5712": {
                  "title": "transition time",
                  "type": "integer"
                }
              }} ]
...
```

Actual command to change the state

Time parameter

# Application Development Today

```
WoTHelpers.fetch("coaps://localhost:5683/bulb").then( async (td) => {
// using await for serial execution (note 'async' in then() of fetch())
try {
let thing = await WoT.consume(td);
// read status
let status = await thing.readProperty("status");
console.info("Bulb status: ", status); // returns value in format {"3311": [{"5850": 1, "5851": 254, ...
…
// change dim value
await thing.invokeAction("dim", {"3311":[{"5851":254, "5712":3}]}); // set dim value to 254 in 3s

} catch(err) {
console.error("Script error:", err);
}

}).catch( (err) => { console.error("Fetch error:", err); });
```

Developer has to know that is the “on/off state” id

Useless data

Developer should not forget this

Developer has to know that is the “transition time” id

(Web) developer

# Dream of Application Development

```
WoTHelpers.fetch("coaps://localhost:5683/bulb").then( async (td) => {
// using await for serial execution (note 'async' in then() of fetch())
try {
let thing = await WoT.consume(td);
// read status
let status = await thing.readProperty("status");
console.info("Bulb status: ", status); // returns value in format {"OnOffState": 1, "DimValue": 254, ...
…
// change dim value
await thing.invokeAction("dim", {"DimValue":254, "TransitionTime":3}); // set dim value to 254 in 3s

} catch(err) {
console.error("Script error:", err);
}

}).catch( (err) => { console.error("Fetch error:", err); });
```

(Web) developer

# Idea

Define TD Payload Pattern & Mark which parameter member(s) of the pattern is passed to / changed from application level (e.g., via WoT API)

# Proposal – Use Semantic Annotations

Is not passed to application level

Signalize, that this member can be passed to application level

Signalize, that the following data schema definition is a Payload template

Title can be used to create parameter name (e.g., OnOffState).
Alternative, define new term like “parameterName”

```
"properties": {
  "state": {
    “@type“ : “PayloadTemplate“,
    "description": "LED state",
    "type": "object",
    "properties": {
      "3311": {
        "items": [
          {
            "type": "object",
            "properties": {
              "5851": {
                “@type“ : “ApplicationParameter",
                "title": "on/off state",
                "type": "integer",
                "minimum": 0, "maximum": 1
              }
            }
          },
          {
            "type": "object",
            "properties": {
              "5851": {
                “@type“ : “ApplicationParameter",
                "title": "dim state",
                "type": "integer",
                "minimum": 1, "maximum": 254
              }
            }
          },
          {
            "type": "object",
            "properties": {
              "5706": {
                "title": "color temp as HEX RGB",
                "type": "string",
                "enum": ["f5faf6","f1e0b5","efd275"]
              }
```

# Miscellaneous

- WoT Runtime: Semantic annotation is used to map application parameter to the message pattern and vice versa
- Application parameters are submitted in flat JSON object
- The developer must ensure that the parameter name is unique (→ take care on title values)
- Message pattern may also define on global level and may be valid for all interaction affordances (also see https://github.com/w3c/wot-thing-description/issues/307)