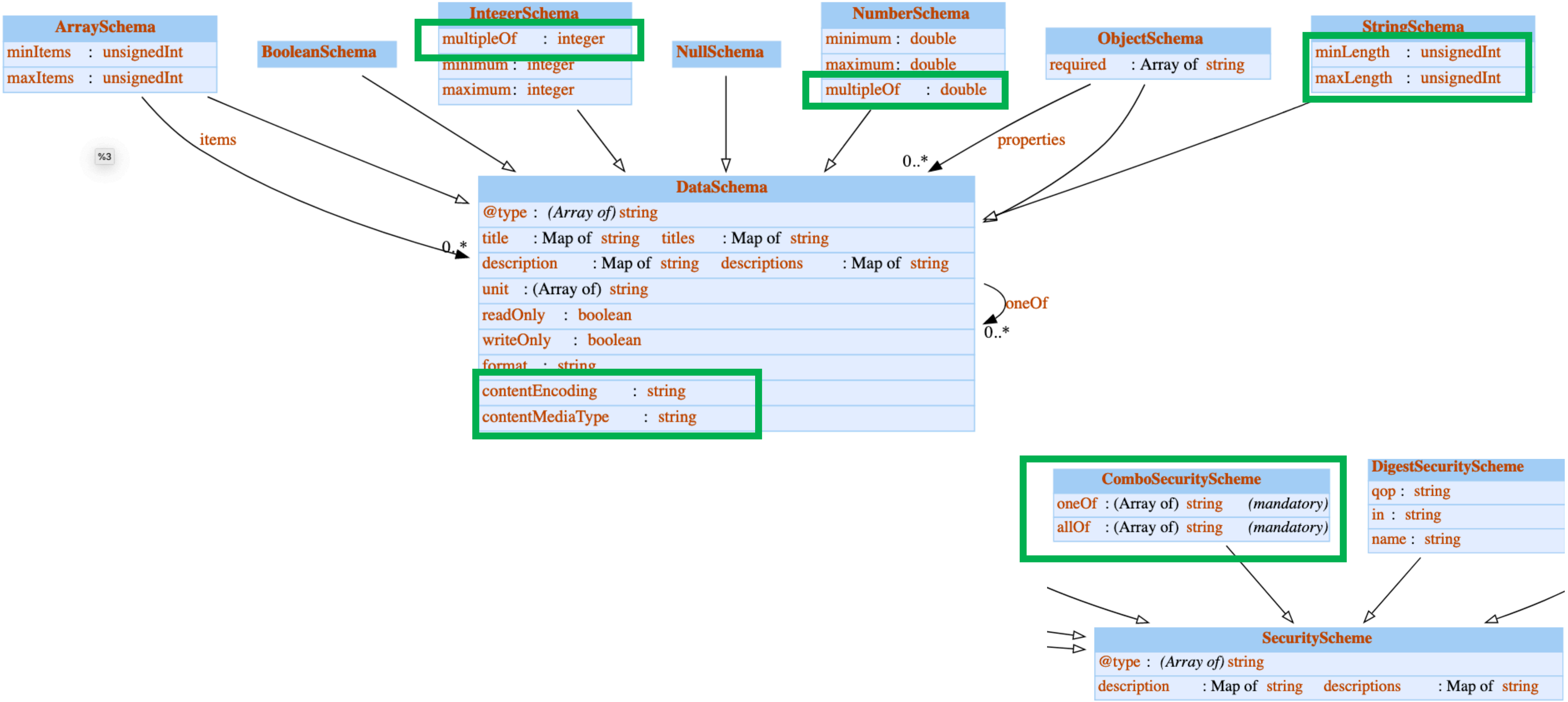
Thing Description Session

Sebastian Kaebisch
Oct 21, 2020

Agenda

- What’s new in TD 1.1 FPDW?
  Organizer: Sebastian, 10+5min
- Dynamic TD discussion (hyperlinks in actions)
  Organizer: Daniel, 10+5min
- Status Thing Model + next steps
  Organizer: Sebastian, 1h
- Component approach for TD
  Organizer: Taki, 1h
- System Description
  Organizer: Ege, 15min

What’s new in TD 1.1 FPDW?
New Terms



%3

ArraySchema
minItems : unsignedInt
maxItems : unsignedInt

BooleanSchema

IntegerSchema
multipleOf : integer
minimum : integer
maximum : integer

NullSchema

NumberSchema
minimum : double
maximum : double
multipleOf : double

ObjectSchema
required : Array of string

StringSchema
minLength : unsignedInt
maxLength : unsignedInt

items

0..*

properties

0..*

DataSchema
@type : *(Array of)* string
title : Map of string titles : Map of string
description : Map of string descriptions : Map of string
unit : (Array of) string
readOnly : boolean
writeOnly : boolean
format : string
contentEncoding : string
contentMediaType : string

oneOf

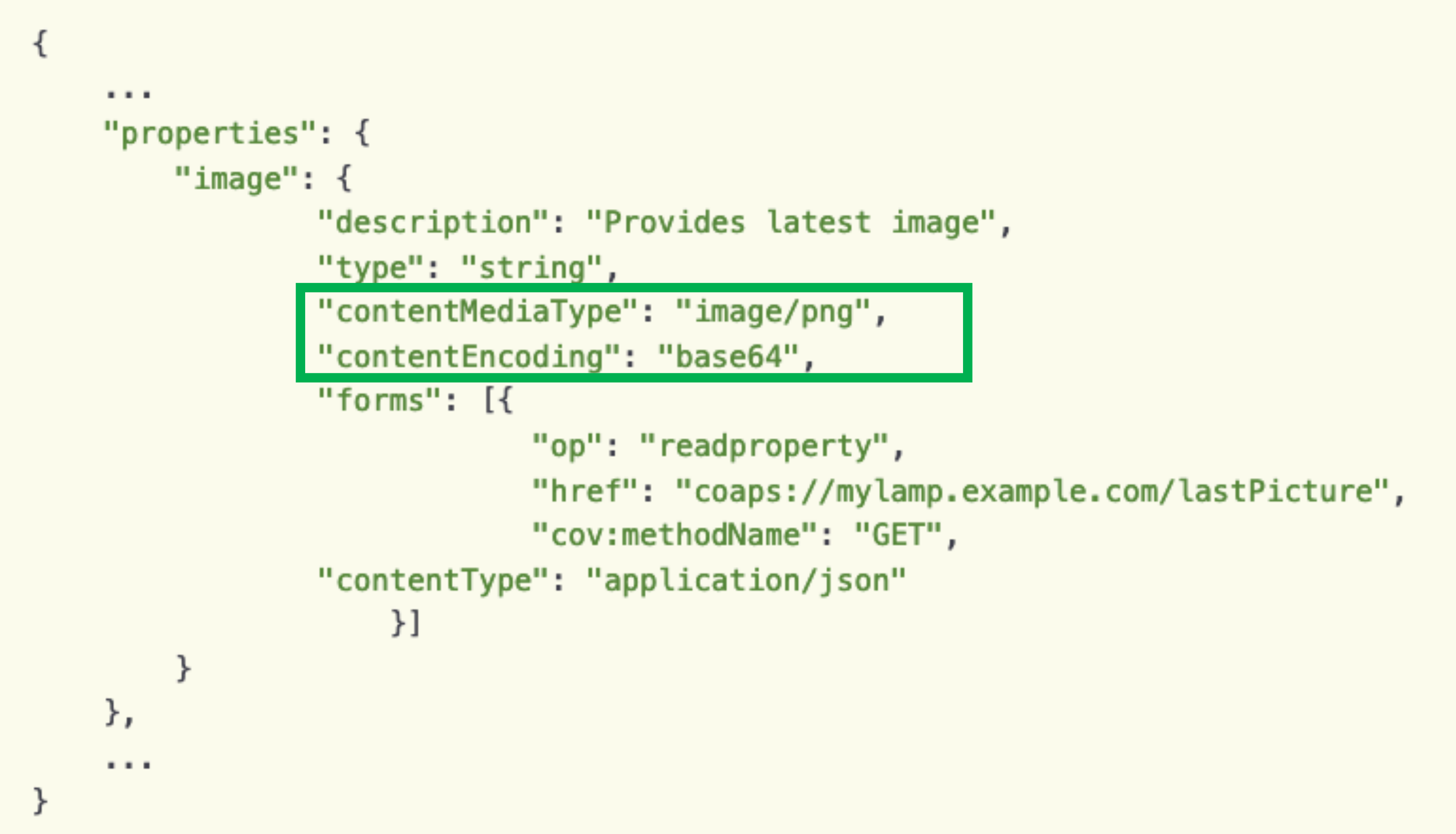
0..*

ComboSecurityScheme
oneOf : (Array of) string *(mandatory)*
allOf : (Array of) string *(mandatory)*

DigestSecurityScheme
qop : string
in : string
name : string

SecurityScheme
@type : *(Array of)* string
description : Map of string descriptions : Map of string

What’s new in TD 1.1 FPDW?
Example: contentMediaType and contentEncoding

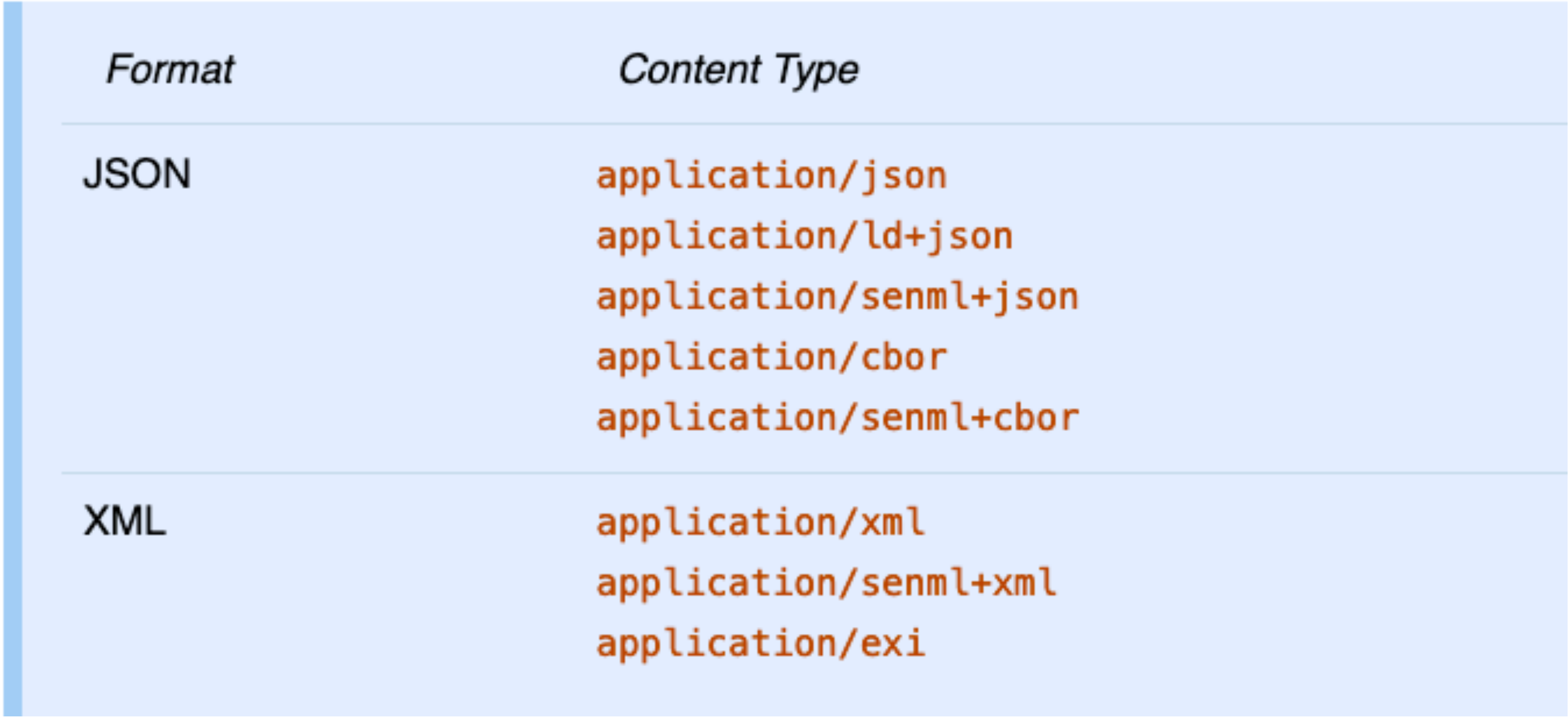


```
{
    ...
    "properties": {
        "image": {
                "description": "Provides latest image",
                "type": "string",
                "contentMediaType": "image/png",
                "contentEncoding": "base64",
                "forms": [{
                            "op": "readproperty",
                            "href": "coaps://mylamp.example.com/lastPicture",
                            "cov:methodName": "GET",
                "contentType": "application/json"
                    }]
        }
    },
    ...
}
```

https://w3c.github.io/wot-thing-description/#example-26

What’s new in TD 1.1 FPDW?
DataSchema mappings to content types

The following table is at risk but contains content types which *MAY* use data schema.



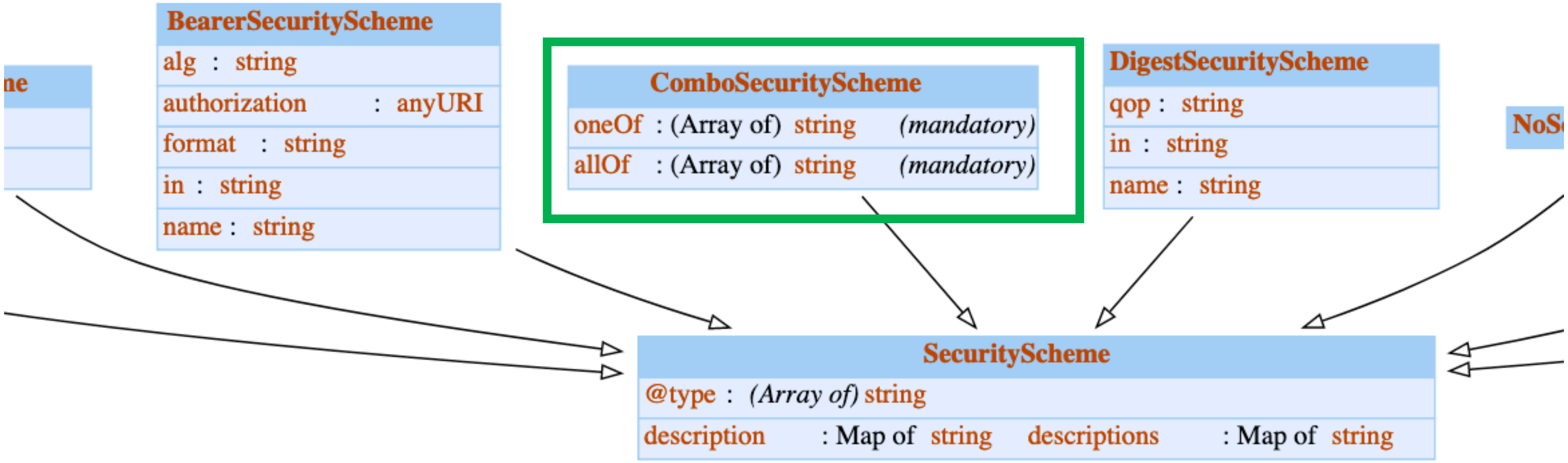
Format	*Content Type*

JSON	`application/json`
	`application/ld+json`
	`application/senml+json`
	`application/cbor`
	`application/senml+cbor`

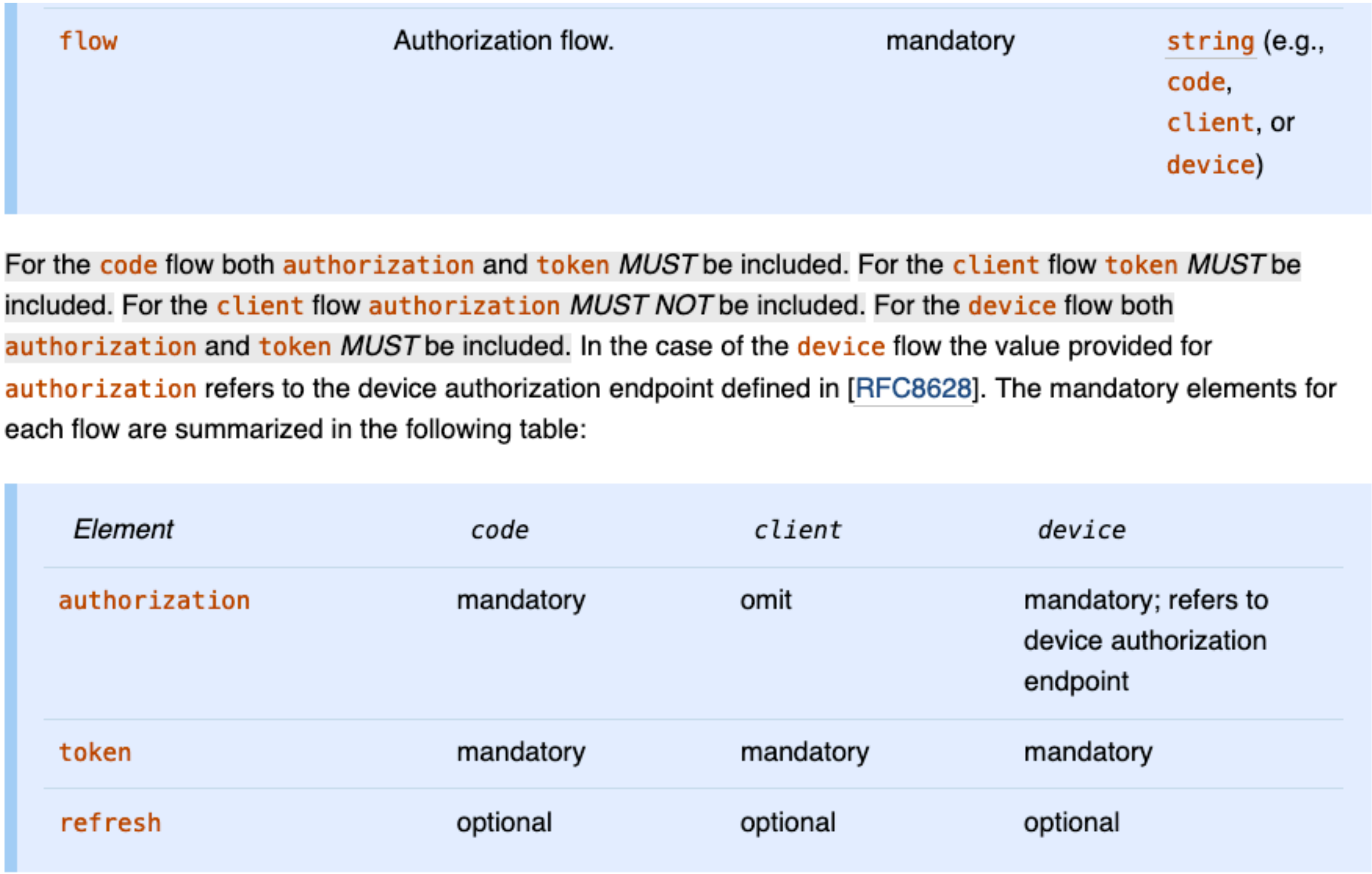
XML	`application/xml`
	`application/senml+xml`
	`application/exi`

Note: Other content types can indirectly use data schema definitions to sketch the data structure.

What’s new in TD 1.1 FPDW?
New security scheme: ComboSecurityScheme



What’s new in TD 1.1 FPDW?
OAuth2SecurityScheme with two additional flows: client and device

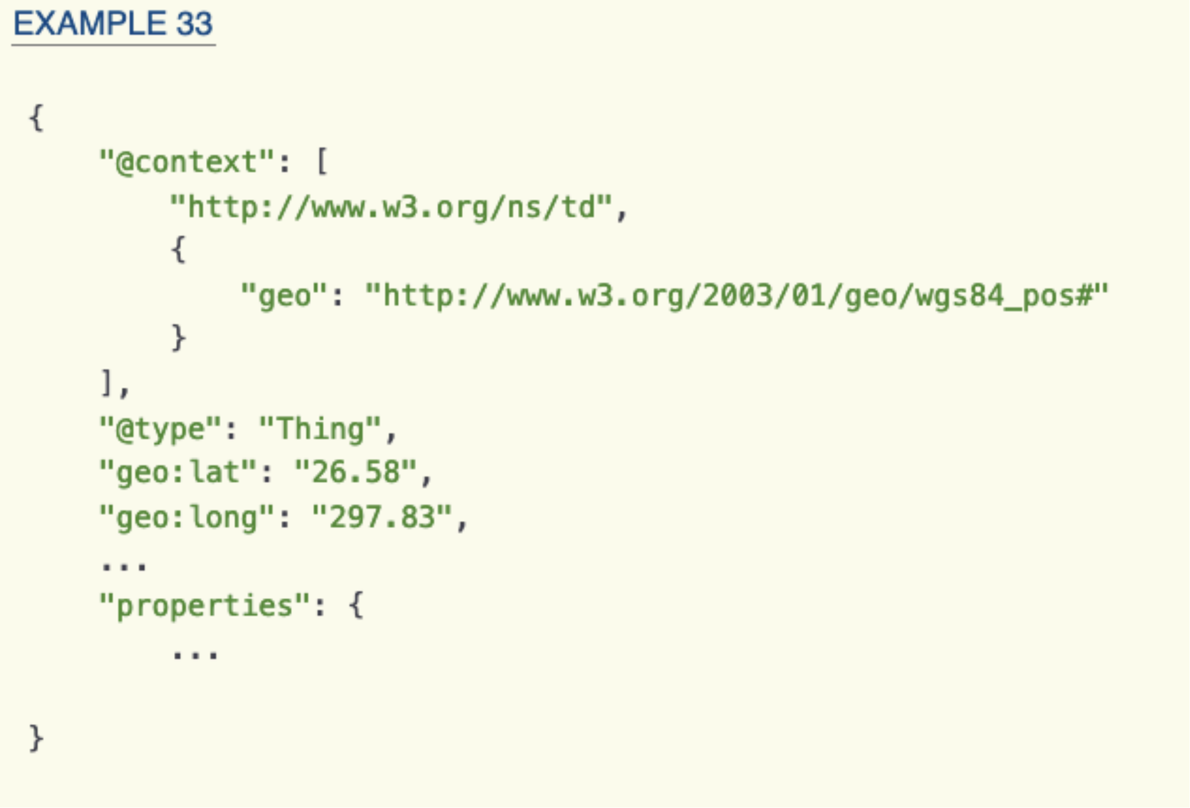


flow	Authorization flow.	mandatory	string (e.g., code, client, or device)

For the code flow both authorization and token *MUST* be included. For the client flow token *MUST* be included. For the client flow authorization *MUST NOT* be included. For the device flow both authorization and token *MUST* be included. In the case of the device flow the value provided for authorization refers to the device authorization endpoint defined in [RFC8628]. The mandatory elements for each flow are summarized in the following table:

Element	*code*	*client*	*device*
authorization	mandatory	omit	mandatory; refers to device authorization endpoint
token	mandatory	mandatory	mandatory
refresh	optional	optional	optional

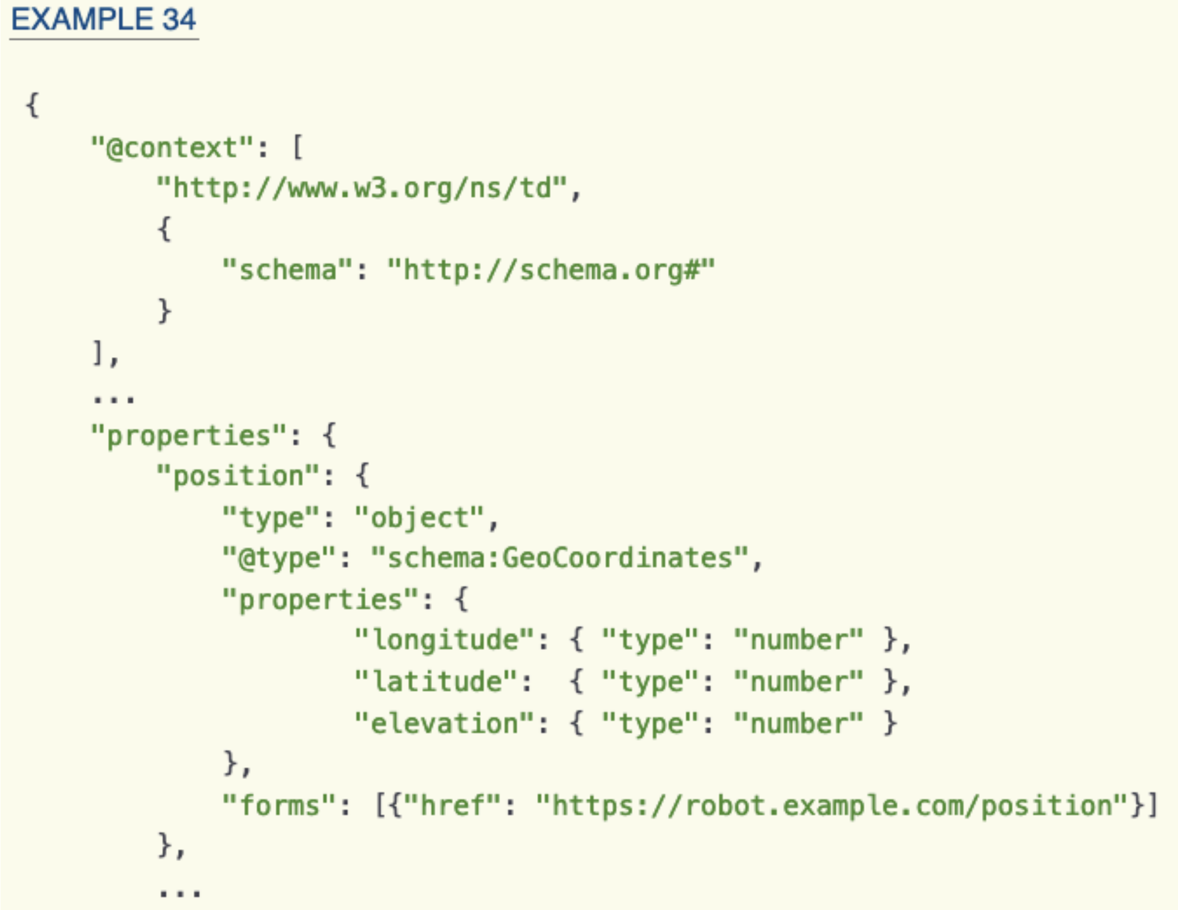
What’s new in TD 1.1 FPDW?
7.1.3 Example III: Geolocation Annotations



EXAMPLE 33

```
{
    "@context": [
        "http://www.w3.org/ns/td",
        {
            "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#"
        }
    ],
    "@type": "Thing",
    "geo:lat": "26.58",
    "geo:long": "297.83",
    ...
    "properties": {
        ...

}
```



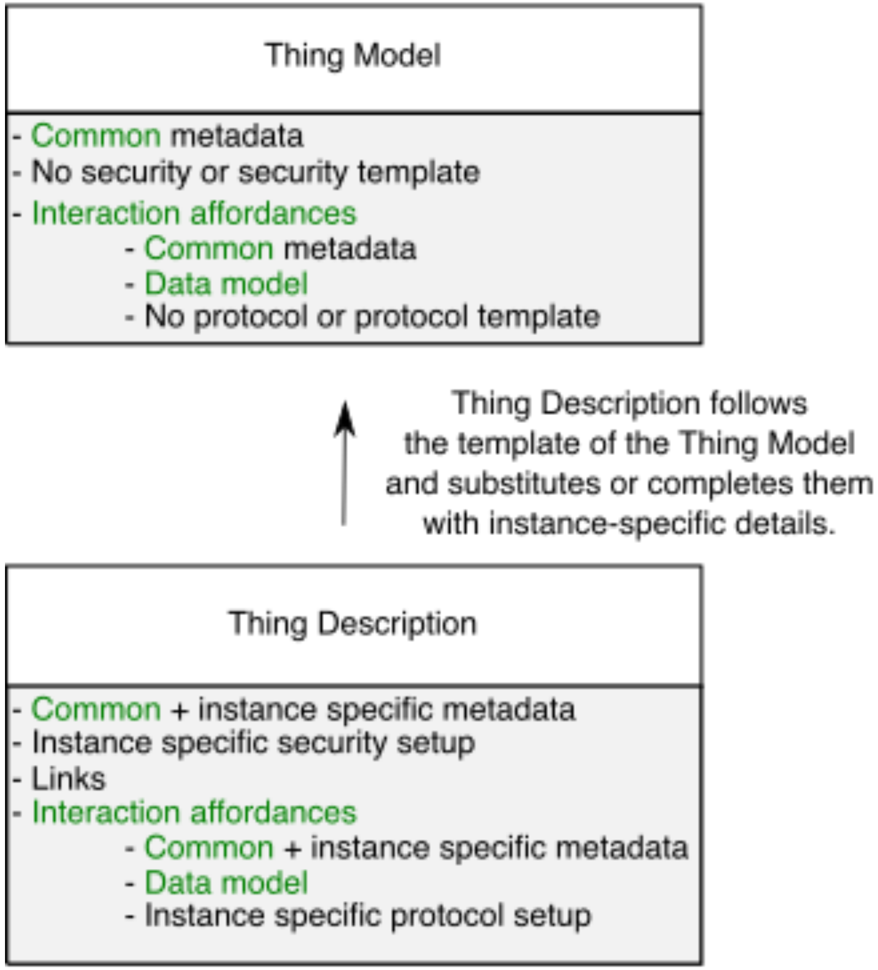
EXAMPLE 34

```
{
    "@context": [
        "http://www.w3.org/ns/td",
        {
            "schema": "http://schema.org#"
        }
    ],
    ...
    "properties": {
        "position": {
            "type": "object",
            "@type": "schema:GeoCoordinates",
            "properties": {
                    "longitude": { "type": "number" },
                    "latitude":  { "type": "number" },
                    "elevation": { "type": "number" }
            },
            "forms": [{"href": "https://robot.example.com/position"}]
        },
        ...
```

What’s new in TD 1.1 FPDW?
10. Thing Model I / II

Definition Thing Model

A Thing Model is a description for a class of Things that have the same capabilities. It describes the Properties, Actions, and Events and common metadata that are shared for an entire group of Things. Compared to a Thing Description, a Thing Model does not contain enough information to identify or interact with a Thing instance.



Thing Model

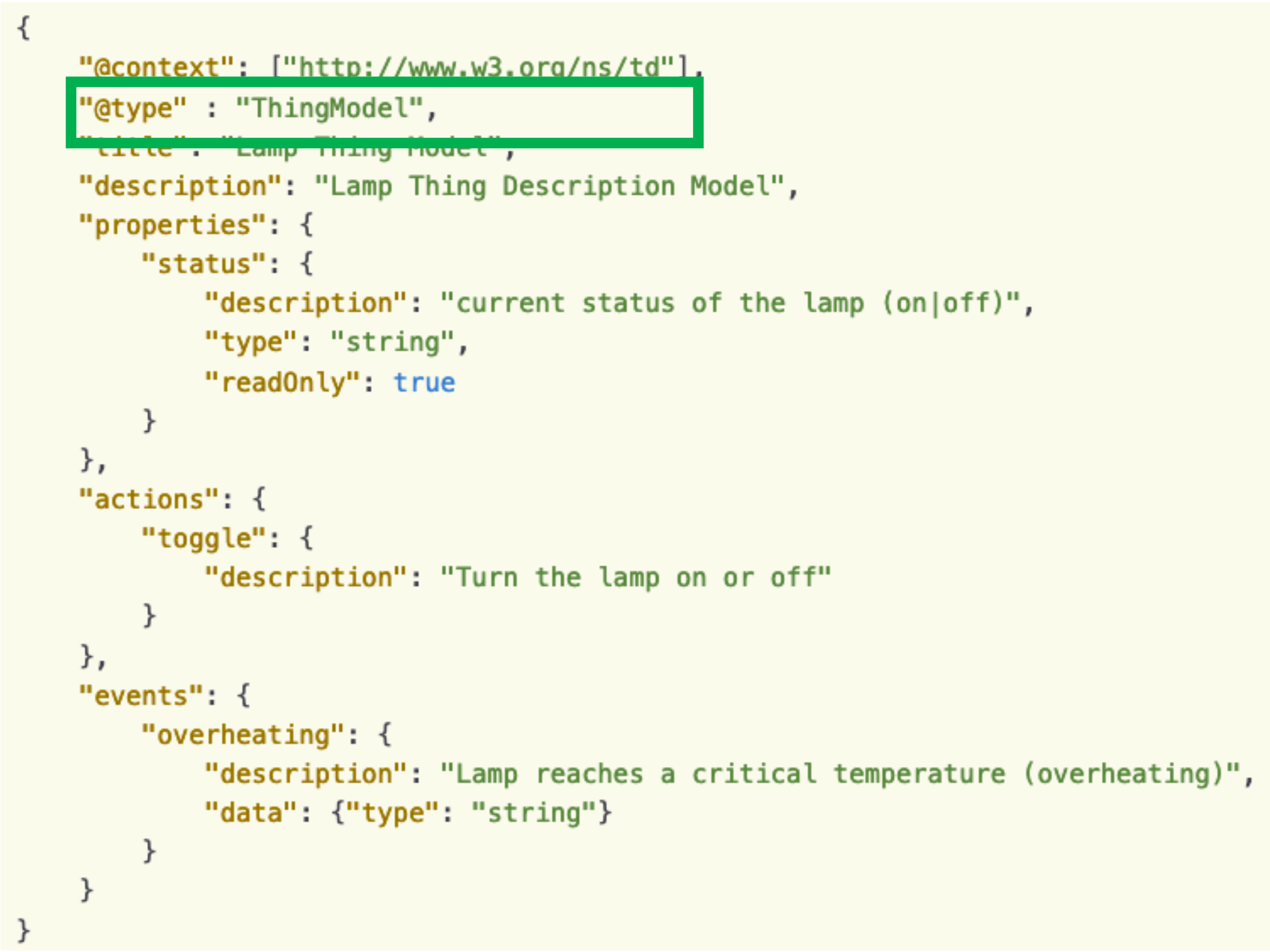
- Common metadata
- No security or security template
- Interaction affordances
  - Common metadata
  - Data model
  - No protocol or protocol template

Thing Description follows the template of the Thing Model and substitutes or completes them with instance-specific details.

Thing Description

- Common + instance specific metadata
- Instance specific security setup
- Links
- Interaction affordances
  - Common + instance specific metadata
  - Data model
  - Instance specific protocol setup

What’s new in TD 1.1 FPDW?
10. Thing Model II / II



```
{
    "@context": ["http://www.w3.org/ns/td"],
    "@type" : "ThingModel",
    "title": "Lamp Thing Model",
    "description": "Lamp Thing Description Model",
    "properties": {
        "status": {
            "description": "current status of the lamp (on|off)",
            "type": "string",
            "readOnly": true
        }
    },
    "actions": {
        "toggle": {
            "description": "Turn the lamp on or off"
        }
    },
    "events": {
        "overheating": {
            "description": "Lamp reaches a critical temperature (overheating)",
            "data": {"type": "string"}
        }
    }
}
```

https://w3c.github.io/wot-thing-description/#example-39-mylampthingmodel-serialized-in-json

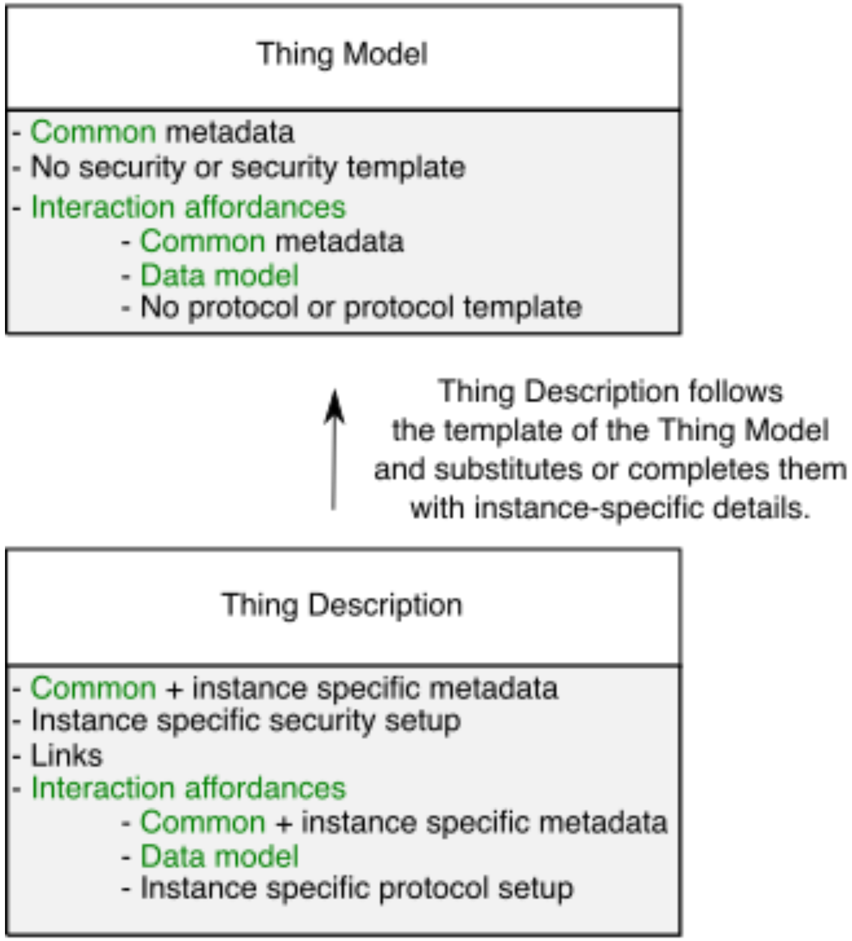
Dynamic TD Discussion (Daniel)

Thing Model
(import / extend feature)

Conceptional Idea

- Thing Model as stand-alone concept such as for
  - onboarding the model into a system (e.g., cloud services) where no details about communication or security metadata is required
  - Things simulation, when (real) Things not yet been developed or deployed
  - **combining multiple or reuse models to avoid redundant Thing Model definitions**

- Thing Model as template for creating Thing Description instances, e.g., for
  - Mass production of real Things. Thing Model can be used to generate individual Thing Description, e.g., in the deployment phase
  - multiple variations of protocol / security support depending of target system. The core model and metadata, however, will be the same.



Thing Model

- Common metadata
- No security or security template
- Interaction affordances
  - Common metadata
  - Data model
  - No protocol or protocol template

Thing Description follows the template of the Thing Model and substitutes or completes them with instance-specific details.

Thing Description

- Common + instance specific metadata
- Instance specific security setup
- Links
- Interaction affordances
  - Common + instance specific metadata
  - Data model
  - Instance specific protocol setup

Combining Multiple Models
Mechanism to import / extend TM Models

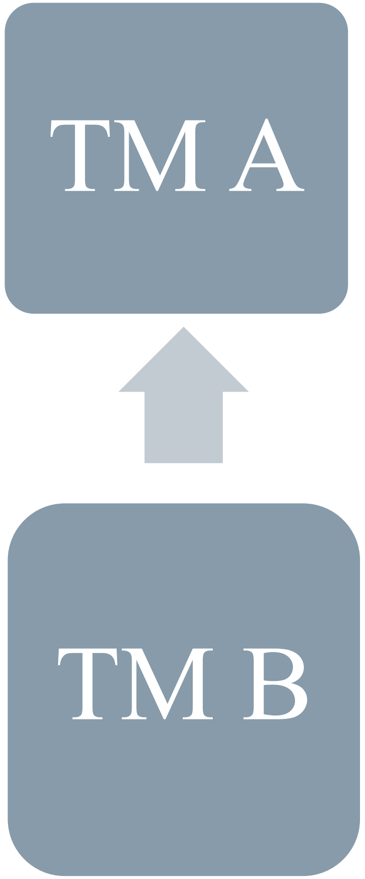
- How we import / extend Thing Models?

- JSON Schema
  - Import schema definitions by **$ref** and **definitions**
  - **$ref** can point to internal and external (e.g., different file) definitions
  - potential, we can use the **$id** (identify schema by URI) similar as we do with **id** in the Thing Description
  - Open question: Name clashes?

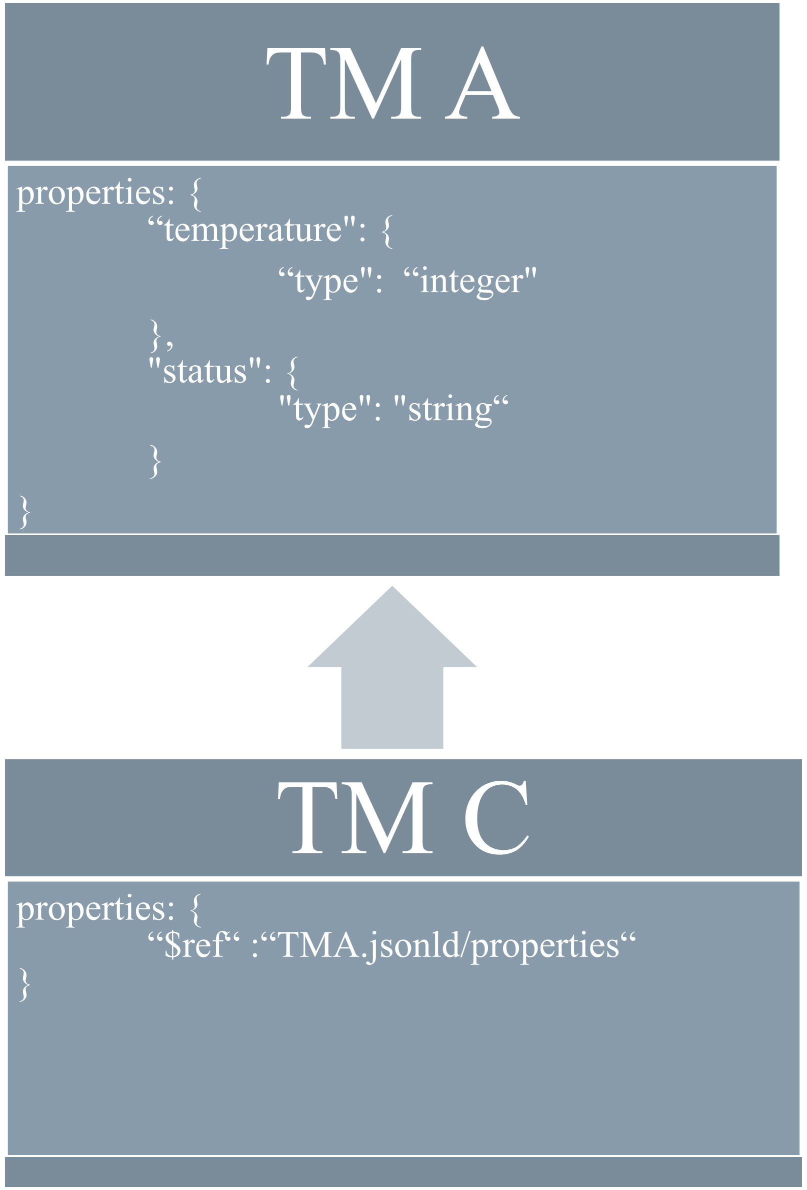
- JSON-LD
  - @context allows to integrate other context definitions
  - Name clashes are solved by the usage of prefixes
  - However, ‘imported’ models by @context are not automatically adopted

- SDF / OneDM
  - uses **namespace,** same concept as @context
  - uses **sdfRef**, same concept as $ref
  - Open question: Name clashes?

- Introduce own import / extend feature for TM?



Import TM
Import models based on JSON Schema



TM A

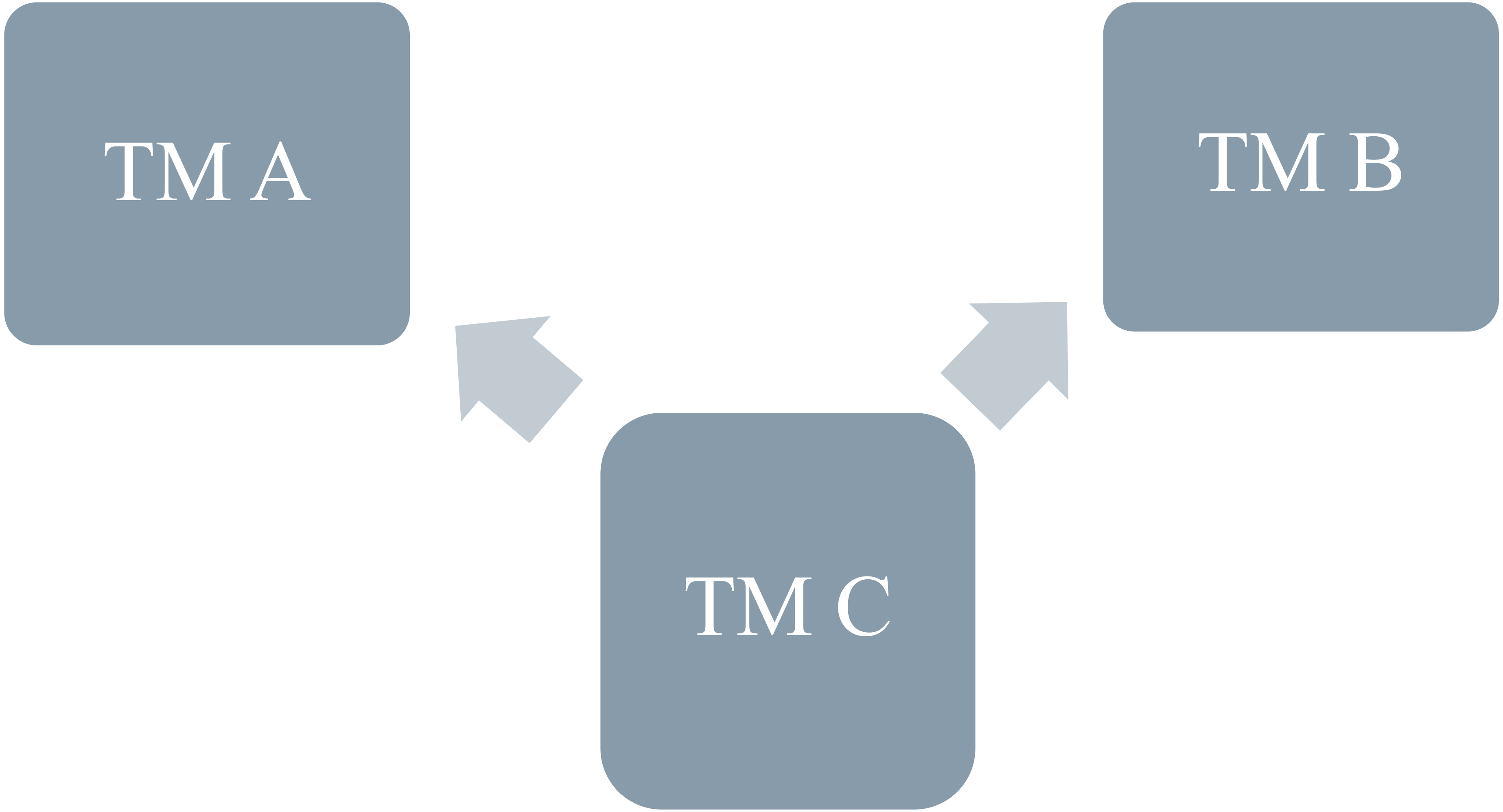
```
properties: {
          “temperature": {
                    “type":  “integer"
          },
          "status": {
                    "type": "string“
          }
}
```

TM C

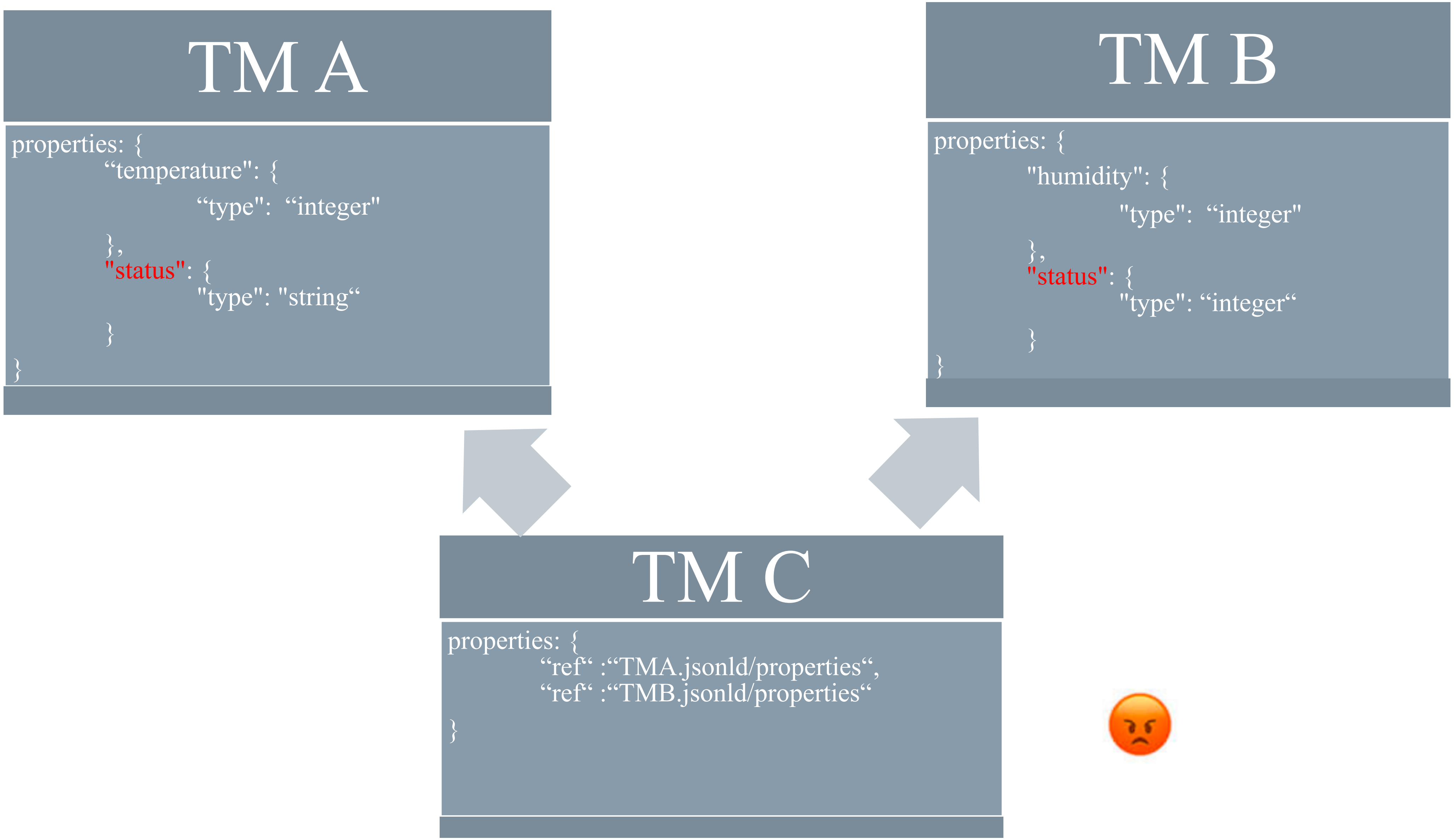
```
properties: {
          “$ref“ :“TMA.jsonld/properties“
}
```

😃

Combining Multiple Models



Combining Multiple Models
Import models based on JSON Schema



TM A

```
properties: {
        “temperature": {
                “type":  “integer"
        },
        "status": {
                "type": "string“
        }
}
```

TM B

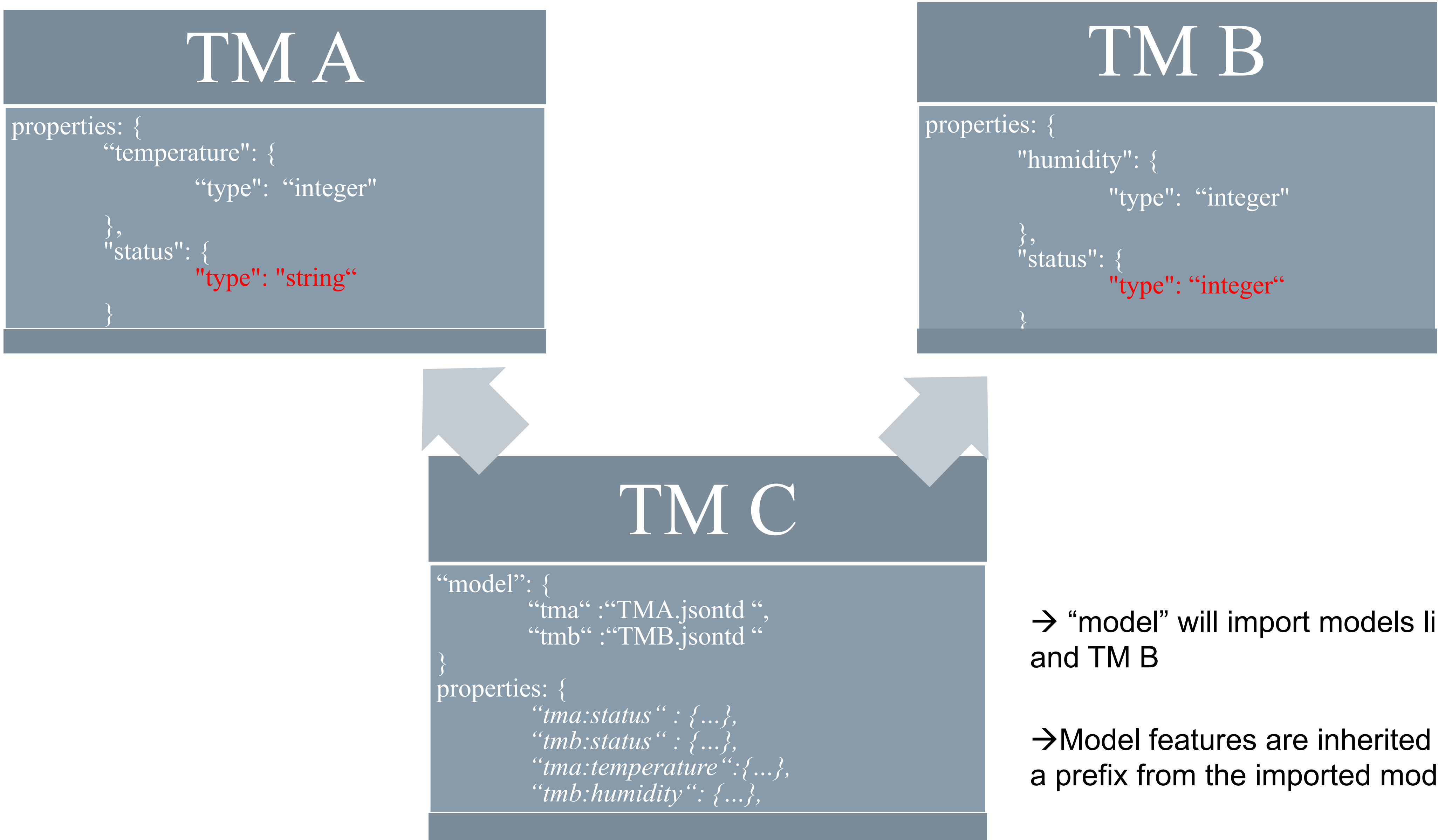
```
properties: {
        "humidity": {
                "type":  “integer"
        },
        "status": {
                "type": “integer“
        }
}
```

TM C

```
properties: {
        “ref“ :“TMA.jsonld/properties“,
        “ref“ :“TMB.jsonld/properties“
}
```

😡

Combining Multiple Models
Introduce new import feature for Thing Model?



TM A

```
properties: {
          “temperature": {
                    “type":  “integer"
          },
          "status": {
                    "type": "string“
          }
```

TM B

```
properties: {
          "humidity": {
                    "type":  “integer"
          },
          "status": {
                    "type": “integer“
          }
```

TM C

```
“model”: {
          “tma“ :“TMA.jsontd “,
          “tmb“ :“TMB.jsontd “
}
properties: {
          “tma:status“ : {…},
          “tmb:status“ : {…},
          “tma:temperature“:{…},
          “tmb:humidity“: {…},
```

→ “model” will import models like TM A and TM B

→Model features are inherited with a prefix from the imported models.

Next Steps

- Evaluate which kind of import / extend mechanism for Thing Model
  - JSON Schema vs. new import /extend feature

- We need to start to setup a pool of sample Thing Models + Use Cases
  - Oracle Cloud?
  - OneDM?
  - Vorto?