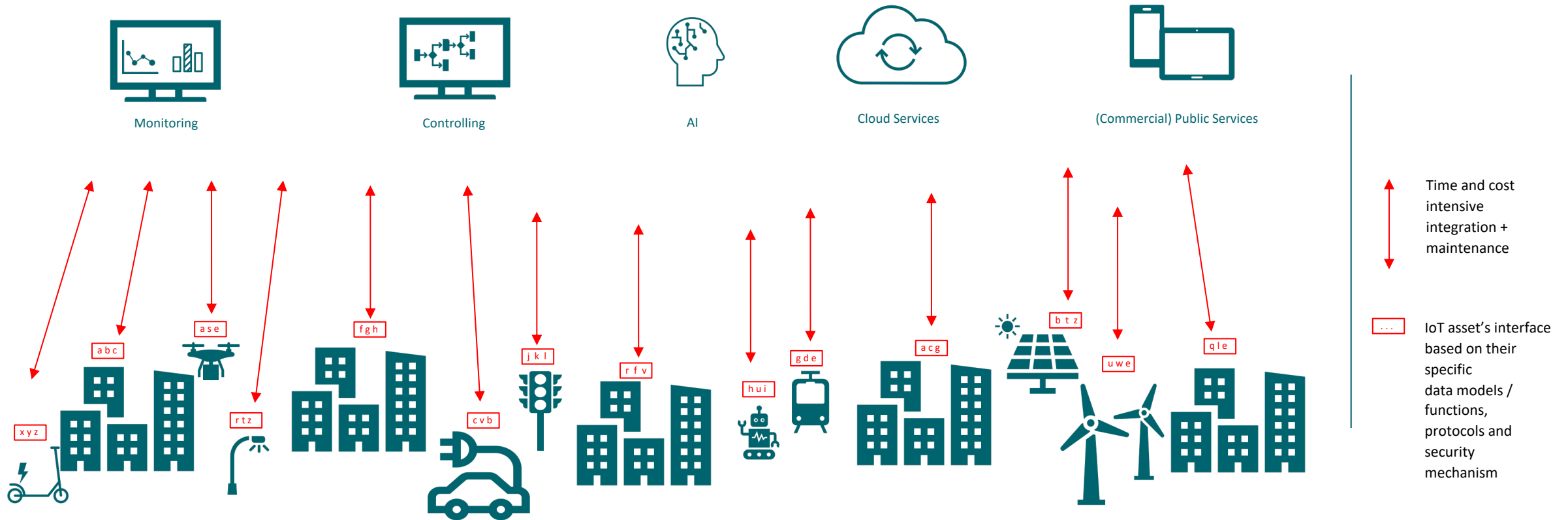


# Thing Description and its Applications

Sebastian Kaebisch, Siemens

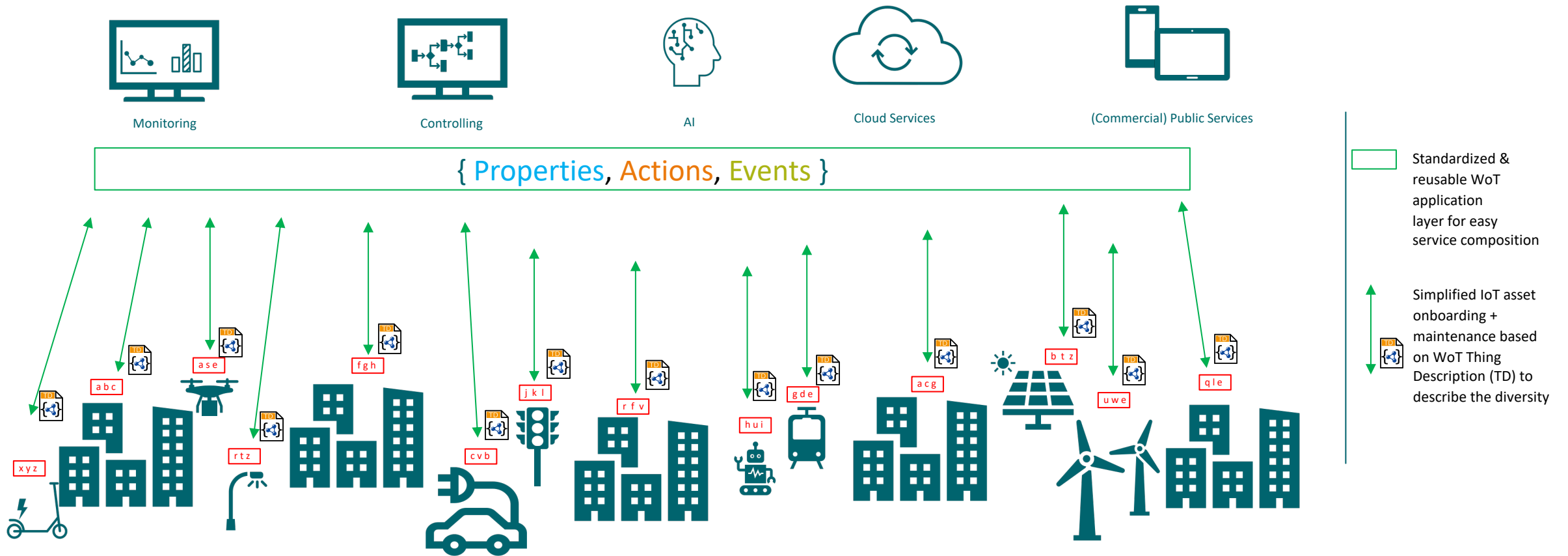
February 2022

# Challenges in Smart City Scenarios

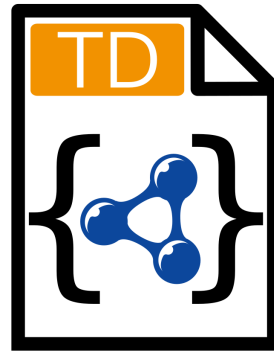


# Service Orchestration with Web of Things

## E.g., for increasing the usage of renewable energy



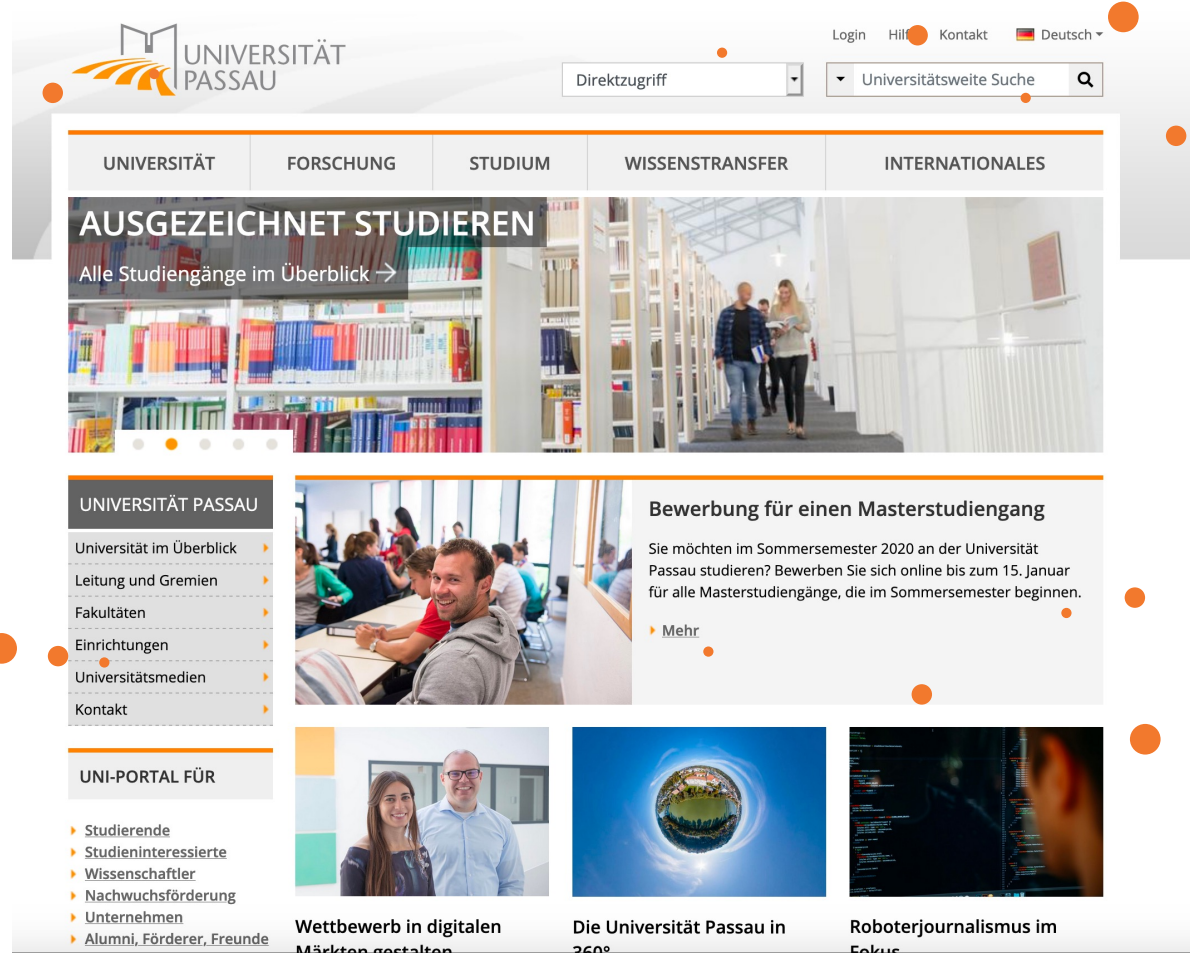
# Details about WOT Thing Description



# Websites are intended for *Humans*

## Typically, we know what we get and how!

Context of website



Select something

Enter data in a web form

Content / Information

Get more information

Get more information

# Thing's Interface - Situation Today

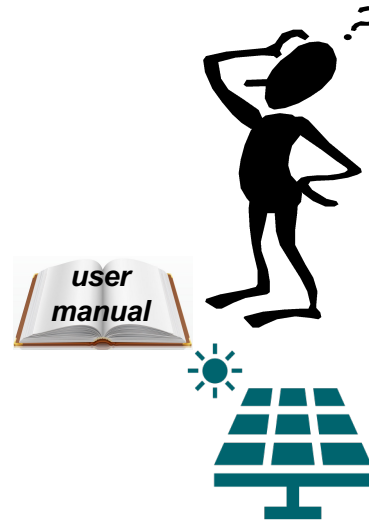
What kind of data do you serve?

Who are you?

How does the payload structure look like?

Are there some context information  
(e.g., kind of actuator/sensor, unit)?

How can I access the data/function?



What kind of functions do you have?

What kind of protocols & serializations do you  
support?

Are there some security constrains?

Do you have other relations to other Things?

# The WoT Thing Description

The “index.html” for Things – A common language based on JSON-LD / RDF

What kind of data do you serve?

Who are you?

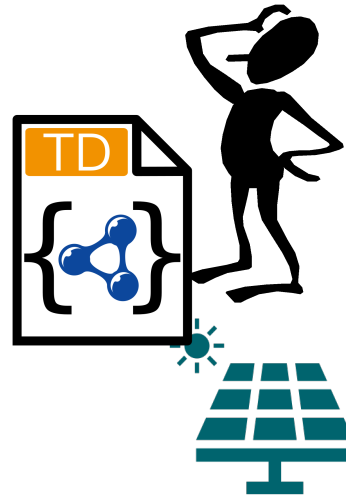
How does the payload structure look like?

How can I access the data/function?

What kind of protocols & serializations do you support?

Are there some security constrains?

Do you have other relations to other Things?



# The WoT Thing Description

Reuse existing domain knowledge

What kind of data do you serve?

Who are you?

How does the payload structure look like?

How can I access the data/function?

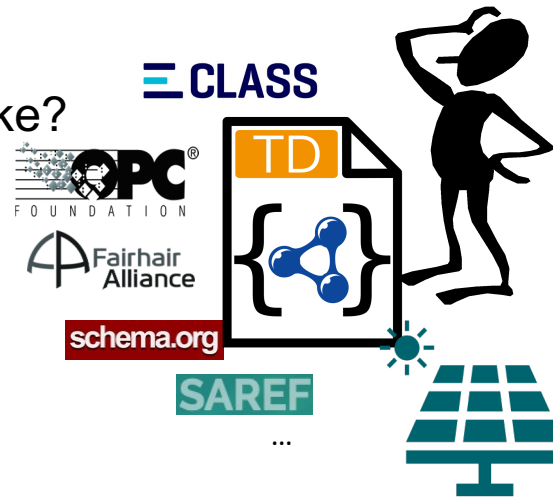
What kind of protocols & serializations do you support?

Are there some context information (e.g., kind of actuator/sensor, unit)?

What kind of functions do you have?

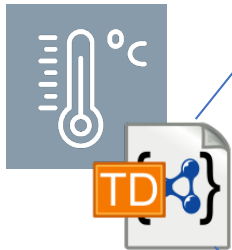
Are there some security constrains?

Do you have other relations to other Things?





# Describe any Thing's Interface with a TD



```
{
  "@context": [ "https://www.w3.org/2019/wot/td/v1",
                { "saref": "https://w3id.org/saref#" } ],
  "@type": "Thing",
  "id": "urn:dev:ops:13473-temp-12",
  "title": "Temperature",
  "security": { "scheme": "oauth2" },
  ....
  "properties": {
    "value": {
      "type": "number",
      "minimum": "-40.2",
      "maximum": "48.4",
      "unit": "Celsius",
      "@type": "saref:Measurement",
      "forms": [{
        "href": "http://192.168.0.1/temp",
        "contentType": "application/json",
        ...
      }
    ]
  }
}
```



```
{
  "@context": [ "https://www.w3.org/2019/wot/td/v1",
                { "eclass": "https://www.eclasscontent.com/owl/v11.1" } ],
  "@type": ["Thing", "eclass:0173-1#01-AKE162#016"],
  "id": "urn:dev:ops:42473-engine-12",
  "title": "Engine",
  "security": { "scheme": "basic" },
  ....
  "properties": {
    "status": {
      ...
      "forms": [{
        "href": "modbus+tcp://192.168.0.2:502 ..."}],
      "speed": {...}
    },
    "actions": {...}
  }
  "events": {...}
}
```

# WoT Binding Templates – Uniform Documentation of IoT Protocols

HTTP

```
"properties": {
  "forms": [
    {
      "op": "readproperty",
      "href": "https://myled.example.com:8080/livingroom/lamp/status",
      "contentType": "application/json",
      "htv:methodName": "GET"
    }
  ]
  ...
}
```

MQTT

```
"events": {
  "forms": [
    {
      "op": "subscribeevent",
      "href": "mqtt://mybroker.example.com:1883/livingroom/lamp/criticalCond",
      "contentType": "application/json",
      "mqv:controlPacketValue": "SUBSCRIBE"
    }
  ]
  ...
}
```

MQTT Broker address

MQTT Topic

CoAP

```
"actions": {
  "forms": [
    {
      "op": "invokeaction",
      "href": "coaps://myled.example.com:5684/lr/l/fi",
      "contentType": "application/ocf+cbor",
      "cov:methodName": "POST",
      "cov:options": [ {
        "cov:optionNumber": 2053,
        "cov:optionValue": "1.1.0"
      } ]
    }
  ]
}
```

CoAP header settings

# WoT Binding Templates – Uniform Documentation of IoT Protocols (cont.)

**Modbus**

```
"properties": {
  "forms": [
    {
      "op": "readproperty",
      "href": "modbus+tcp://127.0.0.1:60000/1/",
      "contentType": "application/octet-stream;byteSeq=BIG_ENDIAN;length=4",
      "modbus:function": "readHoldingRegisters",
      "modbus:address": 40001,
      "modbus:quantity": 2,
      "modbus:pollingTime": 500
    }
  ]
}
```

**OPC UA**

```
"properties": {
  "forms": [
    {
      "op": "readproperty",
      "href": "opc.tcp://localhost:26543/ns=3;s=\"Case_Lamp_Variable\"",
      "contentType": "application/x.opcua-binary",
      "opc:method": "READ"
    }
  ]
}
```

Alternative addressing possible (e.g via browse path)

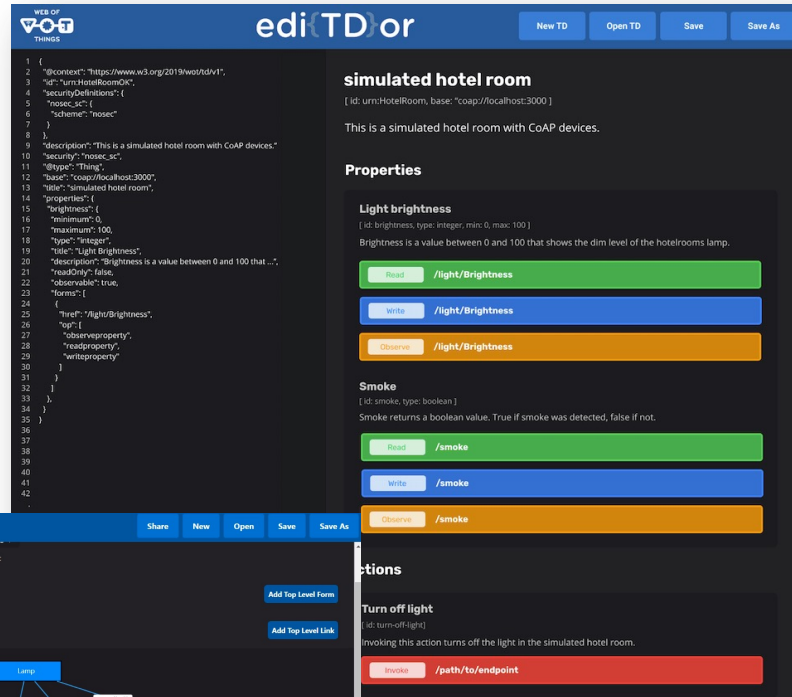
**M-Bus**

```
"properties": {
  "forms": [
    {
      "op": "readproperty",
      "href": "mbus+tcp://127.0.0.1:8182",
      "contentType": "application/octet-stream",
      "mbus:unitID": 3,
      "mbus:offset": 1,
      "mbus:timeout": 2000
    }
  ]
}
```

...

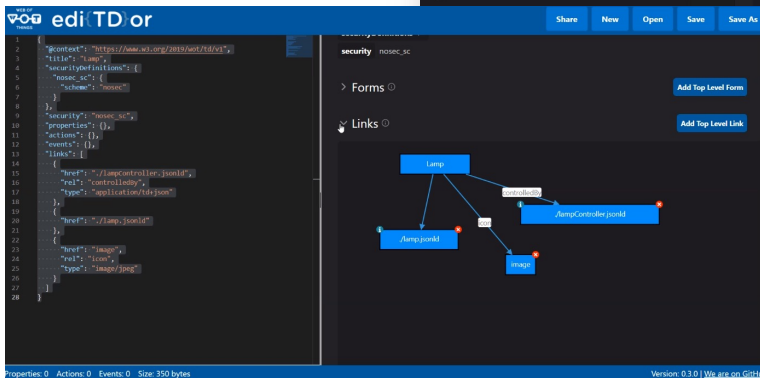
# Eclipse edi{TD}or

web based IDE



The screenshot shows the Eclipse edi{TD}or web-based IDE interface. The top navigation bar includes buttons for 'New TD', 'Open TD', 'Save', and 'Save As'. The main content area is divided into several sections:

- Code Editor:** Displays a JSON schema for a simulated hotel room, including fields like 'id', 'title', 'securityDefinitions', 'properties', and 'actions'.
- simulated hotel room:** Provides a description of the simulated room and its CoAP devices.
- Properties:** Lists properties such as 'Light brightness' and 'Smoke', each with a description and associated REST endpoints (e.g., '/light/Brightness', '/smoke').
- Actions:** Lists actions like 'Turn off light' with their descriptions and endpoints.
- Forms and Links:** Offers options to 'Add Top Level Form' and 'Add Top Level Link'.
- Diagram:** A visual representation of the schema elements, showing relationships between 'lamp', 'lampcontrol', and 'smoke'.



This screenshot shows a different view of the Eclipse edi{TD}or interface, focusing on a diagram. The diagram illustrates the relationships between different components of the simulated hotel room, such as 'lamp', 'lampcontrol', and 'smoke'. The interface includes a navigation menu with 'Forms' and 'Links' sections, and buttons for 'Add Top Level Form' and 'Add Top Level Link'. The bottom status bar shows 'Properties: 0', 'Actions: 0', 'Events: 0', and 'Size: 350 bytes'.

- Open Source: [eclipse.github.io/editdor/](https://eclipse.github.io/editdor/)
- Supports you to create your first Thing Description
- Validates TDs/TMs
- Renders TDs/TMs like Swagger
- Can be extended with vendor context
- Supports the new Thing Model feature

# Checkout for more Information + Tools

<https://www.w3.org/WoT/developers/>



## Developer Tools

### Thing Description (TD) Tooling

- [Thing Description Playground](#) (TD validation)
- [Eclipse Ed\(TD\)](#) or (Editor for easy creation of Thing Description instances and Thing Models)
  - [Try it live here](#)
- [WoTify](#) (a collection of devices that have been WoT-enabled)
- [Shadow Thing](#) (creates and deploys a thing based on its TD)
- [Web of Things Test Bench](#) (tests a WoT Thing by executing interactions automatically, based on its TD)
- [TD code](#) (TD validation and code snippets for Visual Studio Code)
  - See a short presentation about TD Code used together with the [WoT Application Manager \(WAM\)](#): [slides](#) or [video](#)
- [Java API for Thing Descriptions of WoT \(JDTs\)](#) (creates Java Thing Description ORM from a TD in JSON-LD or RDF triples)

### WoT Implementations

- [Eclipse Thingweb node-wot](#) (W3C Web of Things implementation in Node.js with support for multiple bindings.)
  - [Browsified node-wot](#) (Web UI)
  - See [hands-on tutorials](#) and [videos](#) for node-wot
- [WoT FXUI](#) (UI for desktop, mobile, browser)
  - See [running Web-UI instance](#)
- [Node generator](#) (Generate a WoT Consumer Node for [Node-RED](#) from TD)
  - See a short introduction [slides](#) or [video](#) for Node Generator
- [WoT API Development Environment \(WADE\)](#) (Desktop application based on node-wot, Vue.js and Electron)
- [SANE WoT Servient](#) (Java)
- [WoTPy](#) (Experimental implementation in Python)
- [sayWoT!](#) (for web and cloud developers)

### Thing Description Directory Implementations

- [LinkSmart Thing Directory](#)
- [WoTHive Thing Directory](#)

### WoT application development tools

- [WoT Application Manager \(WAM\)](#) CLI tool to set up node-wot application projects. See the presentation and video for further information: [slides](#) or [video](#)

### Others

- [WoT Plugin for AASX Package Explorer](#) Plugin to import/export WoT Thing Description into Asset Administration Shell definitions.

# Contact

**Dr. Sebastian Kaebisch**

W3C Web of Things Co-Chair,  
TF Lead Thing Description

[sebastian.kaebisch@siemens.com](mailto:sebastian.kaebisch@siemens.com)