

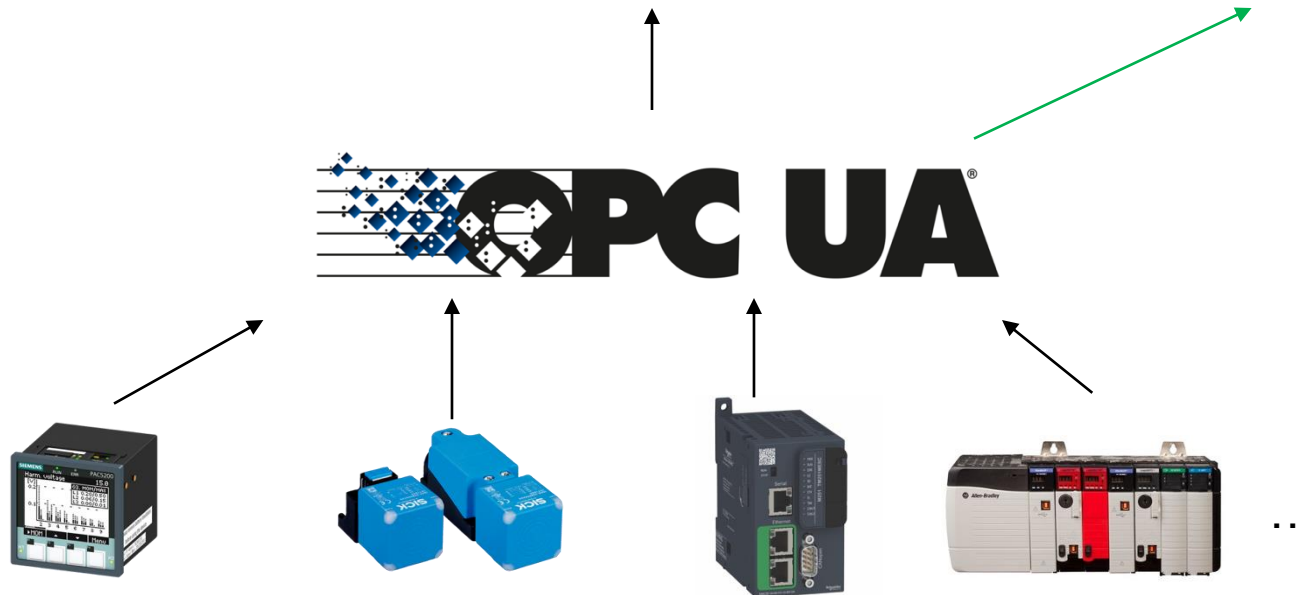
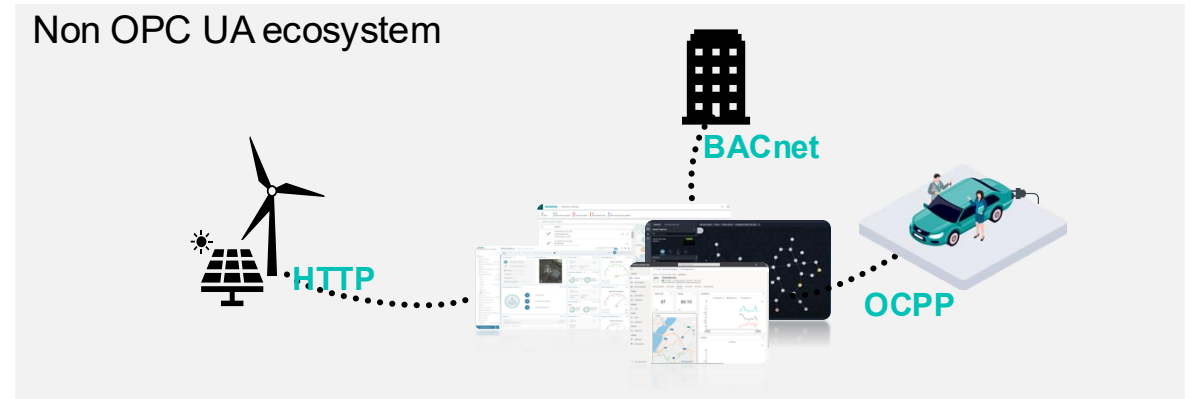
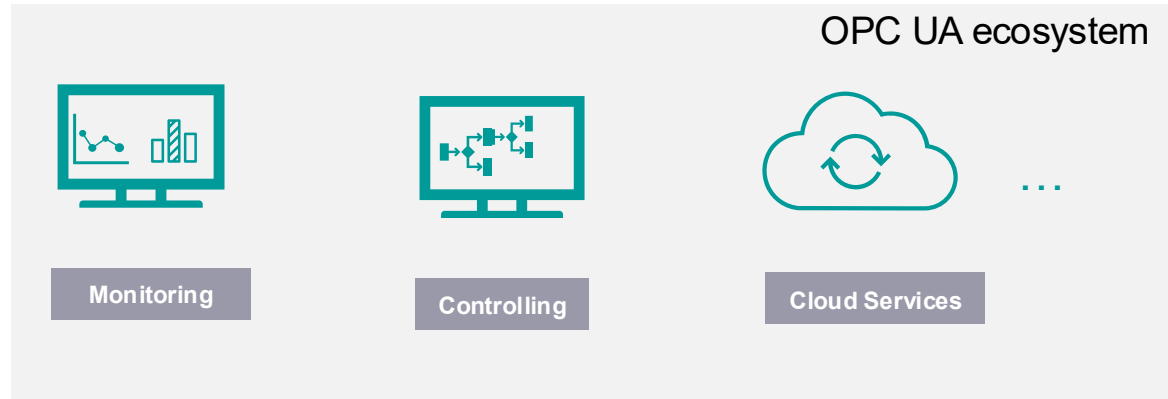
OPC UA WoT Binding

Sebastian **Käbis**ch | Siemens AG

October 16, 2025

WoT CG Meetup

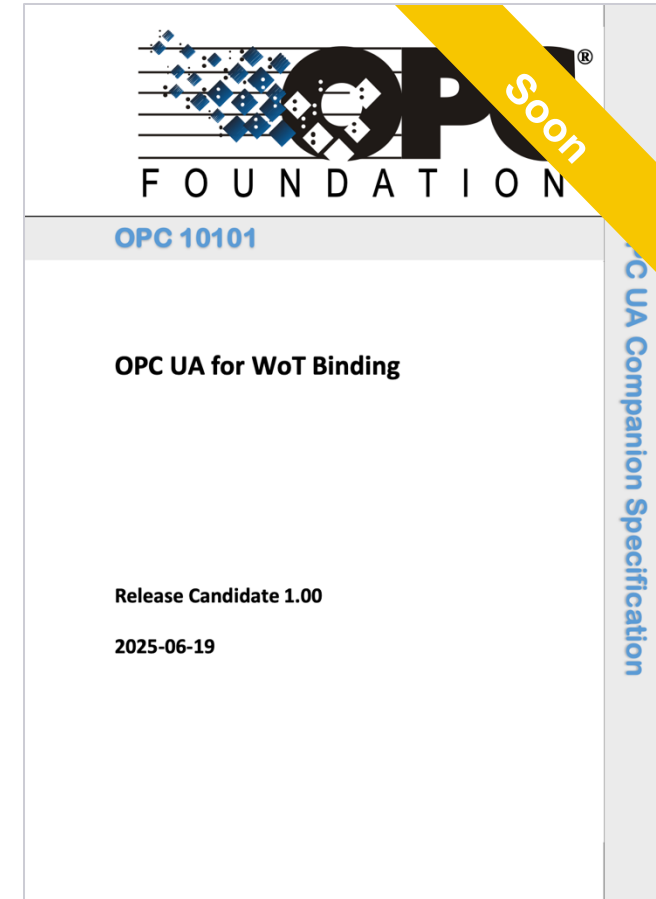
Motivations



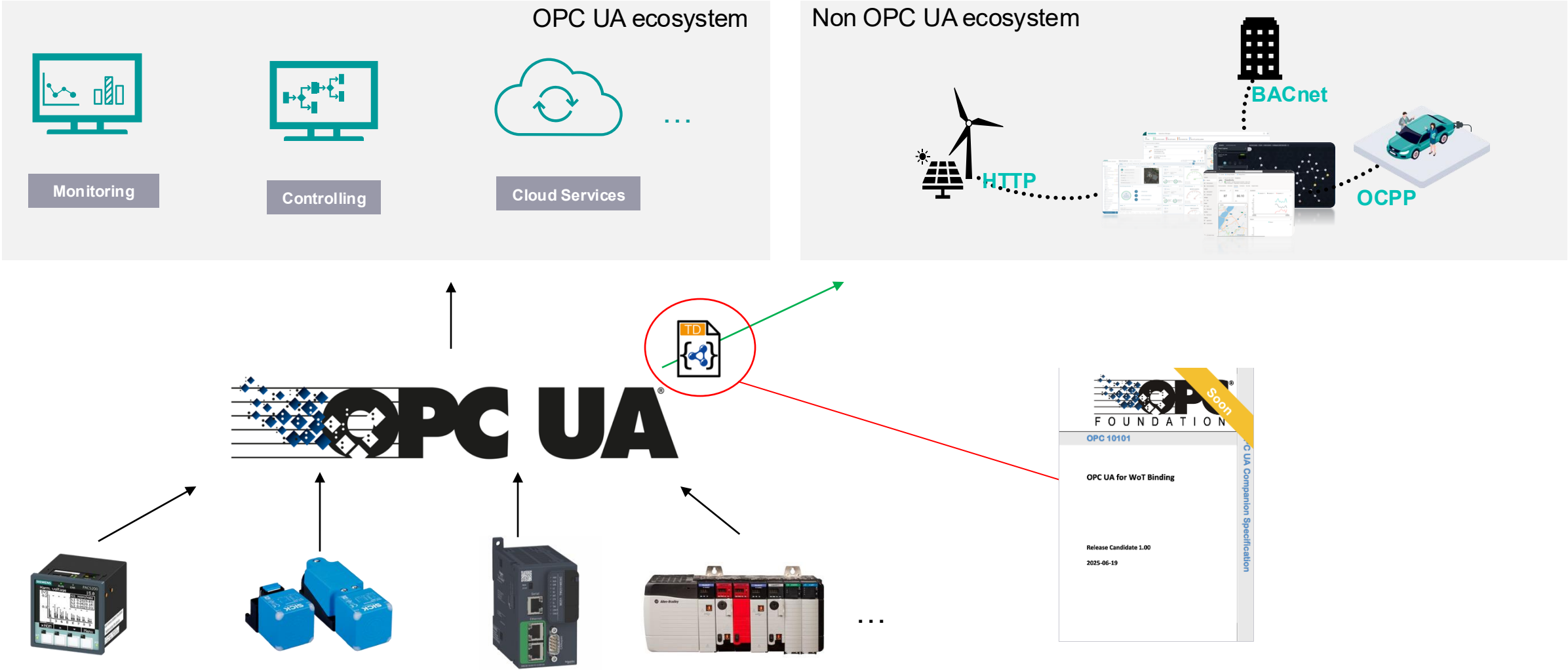
- Simple integration of OPC UA system into non-OPC UA ecosystem (e.g., for cross domain scenarios)
- Describe in a lightweight style an OPC UA interfaces in JSON based format
- Allow to define a specific subset of the UA data model that should be used for target application
- Allow semantic annotations based on RDF

Enter OPC UA for WoT Binding

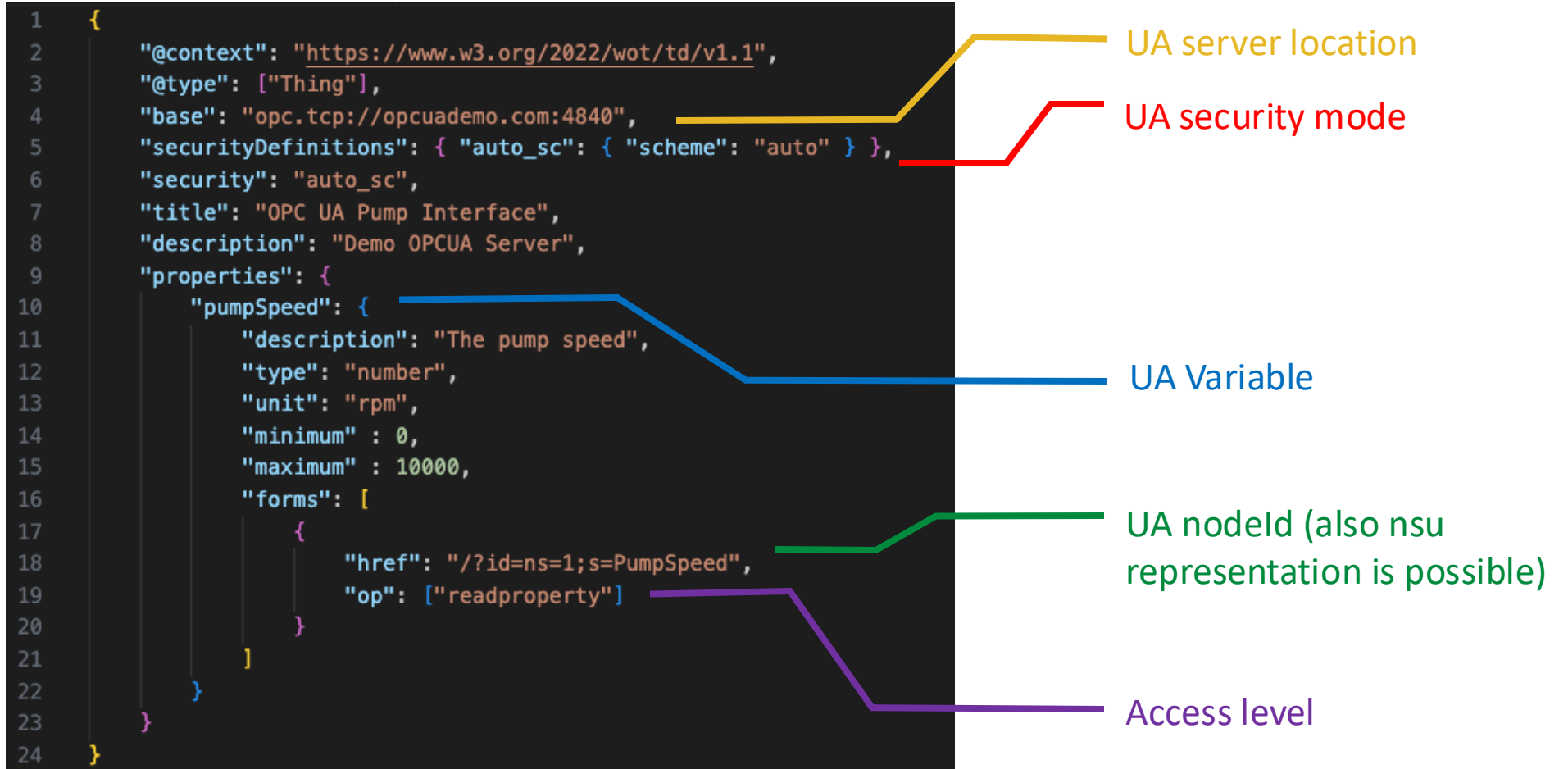
- WG lead by Sebastian Käbisch (Siemens);
Co-lead by Erich Barnstedt (Microsoft)
- Main goal: describe how the WoT Thing Description should be specified to describe an OPC UA interface
- TAG review completed
- Bi-weekly on Tuesday, 4-5CET (next meeting October 21)



Use WoT Thing Description to describe OPC UA interface



Example (Minimal)



URL Scheme

6.2 URL Format for base and href

The following BNF format shows the structure of an URI of the OPC UA Client / Server protocol that have to be followed in a WoT Thing Description:

```
opc.tcp://<address>:<port>[/<resourcePath>]/?id=<nodeId>
```

Where is:

- {address} OPC UA server endpoint (IP) address
- {port} OPC UA server port number
- {resourcePath} If used by the OPC UA server, an added resource path at the endpoint address.
- {nodeId} OPC UA NodeId with the following expectations:
 - 1) any hash character (#) shall be URL encoded (%23)
 - 2) any ampersand character (&) shall be URL encoded (%26)

Note that the URL can be used as a whole in *href* within a forms definition or divided into the global *base* term (for sever location only) and *href* term (relative to the *base* that provides only the nodeId information).

Examples:

- "href": "opc.tcp://192.168.120.237:4840/?id=ns=10;i=12345"
- "href": "opc.tcp://192.168.120.237:4840/?id=nsu=http://widgets.com/schemas/hello;s=水 World"
- "href": "opc.tcp://192.168.120.237:4840/UA/Factory/?id=ns=10;i=12345"
- "href": "/?id=nsu=http://example.com/hello%23;s=temperature"
- "href": "/?id=ns=10;i=12345"

Namespaces (see Section 6.5.5)

Example:

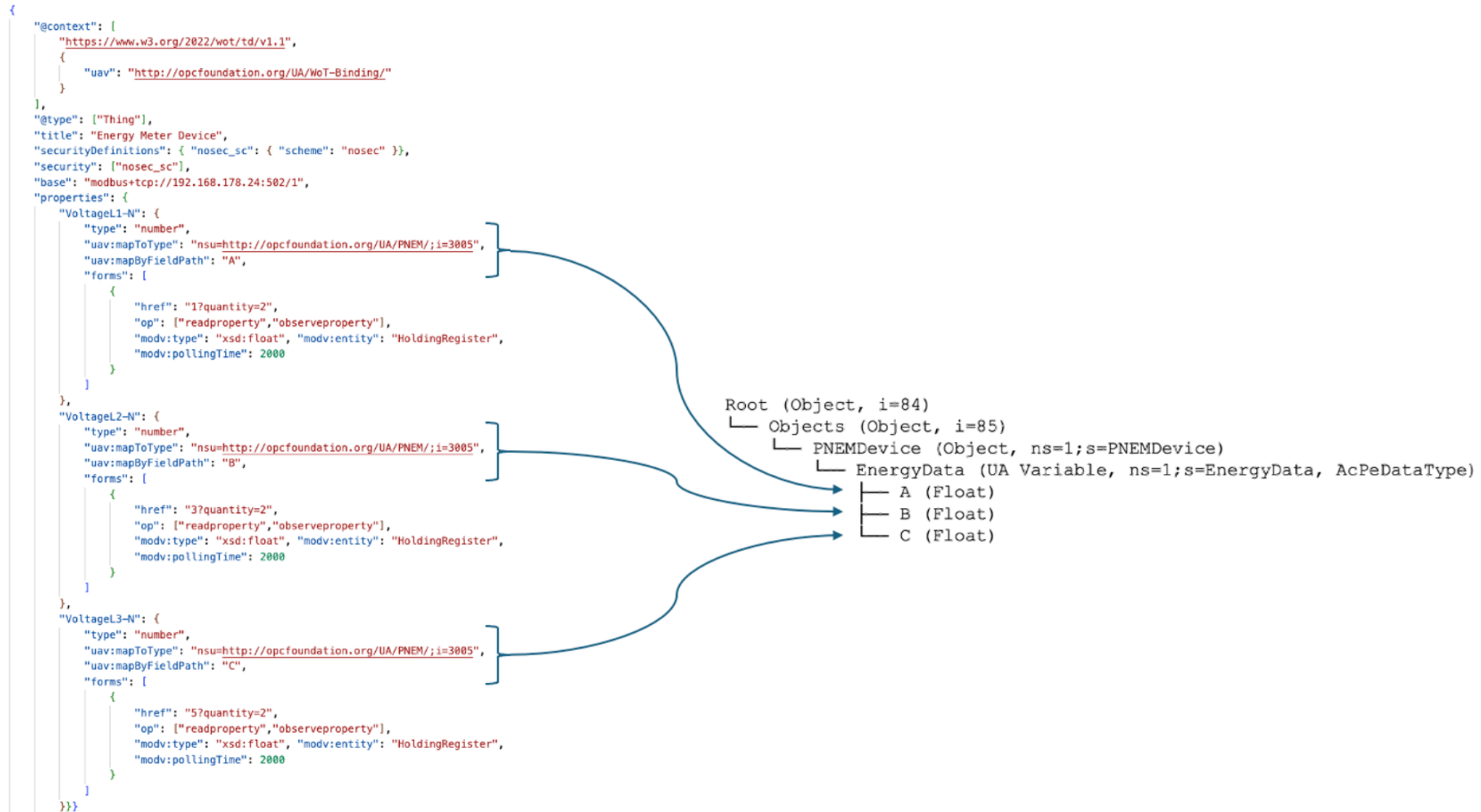
Lets assume there are namespaces used in a OPC UA server as described in the following nodeset:

```
<UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <NamespaceUris>
    <Uri>http://opcfoundation.org/UA/</Uri> <!-- Namespace Index 0 -->
    <Uri>http://opcfoundation.org/UA/DI/</Uri> <!-- Namespace Index 1 -->
    <Uri>http://yourcompany.com/UA/Namespaces/<!-- Namespace Index 2 -->
  </Uri>
</NamespaceUris>
...
```

If the Thing Description (TD) utilizes definitions from these namespaces, they should be specified within the TD @context with the corresponding namespace index as a prefix, as illustrated here:

```
{
  "@context": [
    "https://www.w3.org/2022/wot/td/v1.1",
    {
      "0": "http://opcfoundation.org/UA",
      "1": "http://opcfoundation.org/UA/DI/",
      "2": "http://yourcompany.com/UA/Namespaces/",
      "uav": "http://opcfoundation.org/UA/WoT-Binding/"
    }
  ],
  ...
}
```

Auto Data Model Mapping with UA Binding Terms



OPC UA Terms for annotating Thing Descriptions (from 6.5.4)

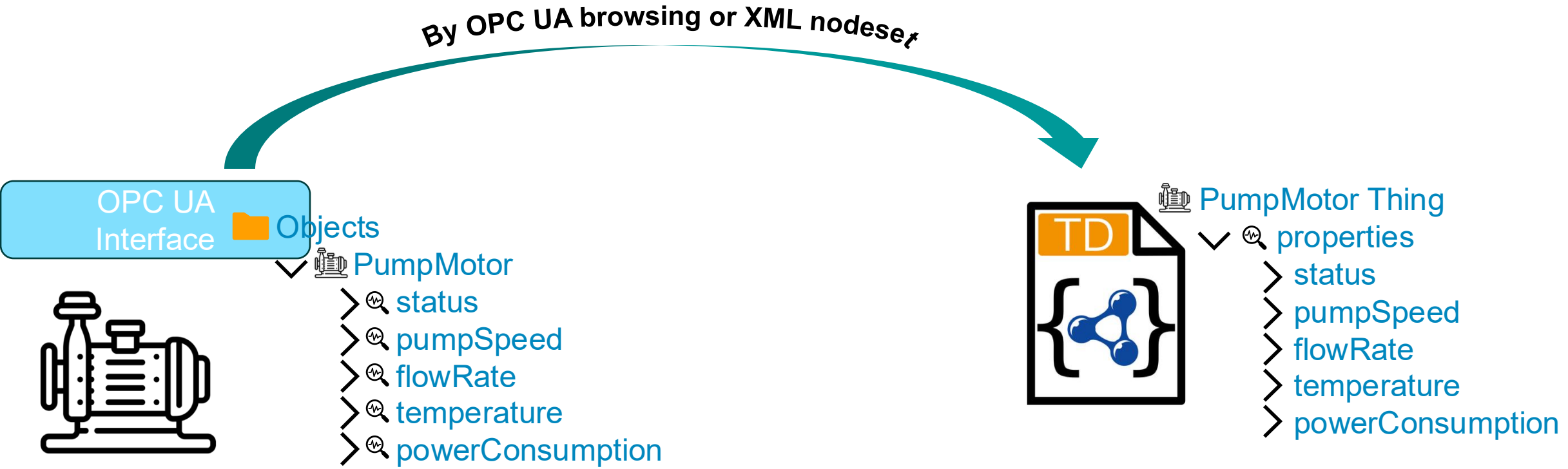
OPC 10101: WoT Binding

15

RC 1.00

Vocabulary term	Description	Assignment	Type
uav:object	Semantic annotation for @type of the TD root level to express that this TD instance represent a UA Object from the top level.	optional	-
uav:objectType	Semantic annotation for @type at root level to express that this instance represent a UA ObjectType (in TD context, this is called Thing Model (TM)) Note: This annotation is only applicable in a Thing Model definition.	optional	-
uav:variable	Semantic annotation for @type of a TD property to associate a UA Variable.	optional	-
uav:variableType	Semantic annotation for @type of a Thing Model property to associate a UA Variable Type. Note: This annotation is only applicable in a Thing Model definition.	optional	-
uav:method	Semantic annotation for @type of a TD action to associate a UA Method.	optional	-
uav:hasComponent	Term to express there is one or more has-child relationship. The value entry is a nodeid of the child node. Note: uav:hasComponent is equivalent to <Reference ReferenceType="HasComponent" IsForward="true">	optional	Array of String
uav:componentOf	Term to express there is one or more is-child-of relationship. The value entry is a nodeid of the parent node. Note: uav:componentOf is equivalent to <Reference ReferenceType="HasComponent" IsForward="false">	optional	Array of String
uav:browseName	Provides the origin browse name of the UA node. The uav:browseName can be used at the top level to provide the browse name of the UA Object or UA ObjectType, and within the property or action level to give the browse name of the UA variable or UA method respectively.	optional	String
uav:mapToNodeid	This term can be used within a property definition to express that the associated runtime datapoint can be mapped to a (external) target OPC UA node (e.g. UA variable) with the corresponding Nodeid. This term can be not used at the forms level. See Section 8.2 for example usage.	optional	string
uav:mapToType	This term can be used within a property definition to express that the associated runtime datapoint can be mapped to a (external) target OPC UA type with the corresponding Nodeid. This term can be not used at the forms level. See Section 8.2 for example usage.	optional	string
uav:mapByFieldPath	This term can be only used with the uav:mapToType term. In the case that the target type of uav:mapToType is complex (e.g. based on BaseType 0:Structure), a specific field name can be addressed within this complex type definition to which the runtime data can be mapped to. See Section 8.2 for example usage.	optional	string

Thing Description Mapping from OPC UA Core Definition



Thing Description Mapping from OPC UA Core Definition (Example UA Variables and UA Methods mapping)

OPC UA Interface

```
Root (Object, i=84)
├── Objects (Object, i=85)
│   ├── IndustrialMotor (Object, ns=1;s=IndustrialMotor)
│   │   ├── Speed (UA Variable, ns=1;s=MotorSpeed, Double, AccessLevel: 3)
│   │   │   └── Unit (PropertyType, UA Variable, ns=1;s=SpeedUnit, String) [rpm]
│   │   ├── StartMotor (Method, ns=1;s=StartMotor)
│   │   │   ├── InputArguments
│   │   │   │   ├── StartDelay (Double) [seconds]
│   │   │   │   └── TargetSpeed (Double) [rpm]
│   │   │   └── OutputArguments
│   │   │       └── Status (String)
│   │   └── SensorData (Object, ns=1;s=SensorData)
│   │       ├── Temperature (UA Variable, ns=1;s=SensorTemperature, Double, AccessLevel:1)
│   │       │   └── Unit (PropertyType, UA Variable, ns=1;s=TemperatureUnit, String) [°C]
│   │       ├── Vibration (UA Variable, ns=1;s=SensorVibration, Double, AccessLevel:1)
│   │       │   └── Unit (PropertyType, UA Variable, ns=1;s=VibrationUnit, String) [g]
│   │       └── ... (other objects)
│   └── Types (Object, i=86)
└── Views (Object, i=87)
```



```
{
  "@context": [
    "https://www.w3.org/2022/wot/td/v1.1",
    {
      "I": "http://example.industrial.motor.com/ua",
      "uav": "http://opcfoundation.org/UA/WoT-Binding/"
    }
  ],
  "title": "IndustrialMotor",
  "uav:browseName": "1:IndustrialMotor",
  "base": "opc.tcp://192.168.1.10:4840/UA/IndustrialMotorServer",
  "securityDefinitions": {
    "auto_sc": {
      "scheme": "auto"
    }
  },
  "security": "auto_sc",
  "properties": {
    "1:Speed": {
      "title": "Speed",
      "type": "number",
      "unit": "rpm",
      "observable": true,
      "forms": [
        {
          "href": "/?id=ns=1;s=MotorSpeed",
          "contentType": "application/octet-stream",
          "op": [
            "readproperty",
            "writeproperty",
            "observeproperty"
          ],
          "uav:browsePath": "/Root/Objects/1:IndustrialMotor/1:Speed"
        }
      ]
    },
    "1:Temperature": {
      "title": "Temperature",
      "type": "number",
      "unit": "°C",
      "readOnly": true,
      "observable": true,
      "uav:componentOf": [
        "ns=1;s=SensorData"
      ],
      "forms": [
        {
          "href": "/?id=ns=1;s=SensorTemperature",
          "contentType": "application/octet-stream",
          "op": [
            "readproperty",
            "observeproperty"
          ],
          "uav:browsePath": "/Root/Objects/1:IndustrialMotor/1:SensorData/1:Temperature"
        }
      ]
    },
    "1:Vibration": {
      "title": "Vibration",
      "type": "number",
      "unit": "g",
      "readOnly": true,
      "observable": true,
      "uav:componentOf": [
        "ns=1;s=SensorData"
      ],
      "forms": [
        {
          "href": "/?id=ns=1;s=SensorVibration",
          "contentType": "application/octet-stream",
          "op": [
            "readproperty",
            "observeproperty"
          ],
          "uav:browsePath": "/Root/Objects/1:IndustrialMotor/1:SensorData/1:Vibration"
        }
      ]
    }
  },
  "actions": {
    "1:StartMotor": {
      "title": "StartMotor",
      "input": {
        "type": "object",
        "properties": {
          "StartDelay": {
            "type": "number"
          },
          "TargetSpeed": {
            "type": "number"
          }
        }
      },
      "output": {
        "type": "object",
        "properties": {
          "Status": {
            "type": "string",
            "description": "The status after starting the motor."
          }
        }
      },
      "uav:componentOf": [
        "ns=1;s=IndustrialMotor"
      ],
      "forms": [
        {
          "href": "/?id=ns=1;s=StartMotor",
          "contentType": "application/octet-stream"
        }
      ]
    }
  }
}
```



Implementations

SIEMENS



Simple *orchestrate* your application

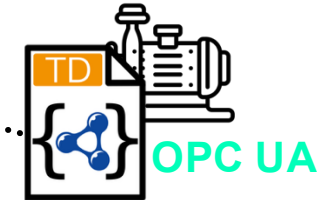
Example: Read pump speed (each 5s); increment counter service when speed >200; otherwise decrement



```
// OPCUA
const thingOPCUApump = await wot.consume(thingDescriptionOPCUA);

// HTTP counter
const thingHttpCounter = await wot.consume(thingDescriptionHTTPCounter);

// read pumpSpeed every 5 secs and update counter accordingly
// > 200 increment; <= 200 decrement
setInterval(async () => {
  const content = await thingOPCUA.readProperty("pumpSpeed");
  const value = await content.value();
  if (value !== null) {
    const speed = (value).valueOf();
    console.log("OPCUA Pump Speed is", speed);
    if (speed > 200) {
      console.log("\t increment HTTP counter");
      await thingHttpCounter.invokeAction("increment");
    } else {
      console.log("\t decrement HTTP counter");
      await thingHttpCounter.invokeAction("decrement");
    }
  }
}, 5000);
```



Concentrate on the actual business logic (e.g., blend out protocol -specific characteristics such as nodeId, http methods, contentType, ...)

<https://github.com/eclipse-thingweb/node-wot>

Contact



Dr. Sebastian Käbisch
Siemens AG

Telefon: +49 (172) 4087515

E-Mail: sebastian.kaebisch@siemens.com

