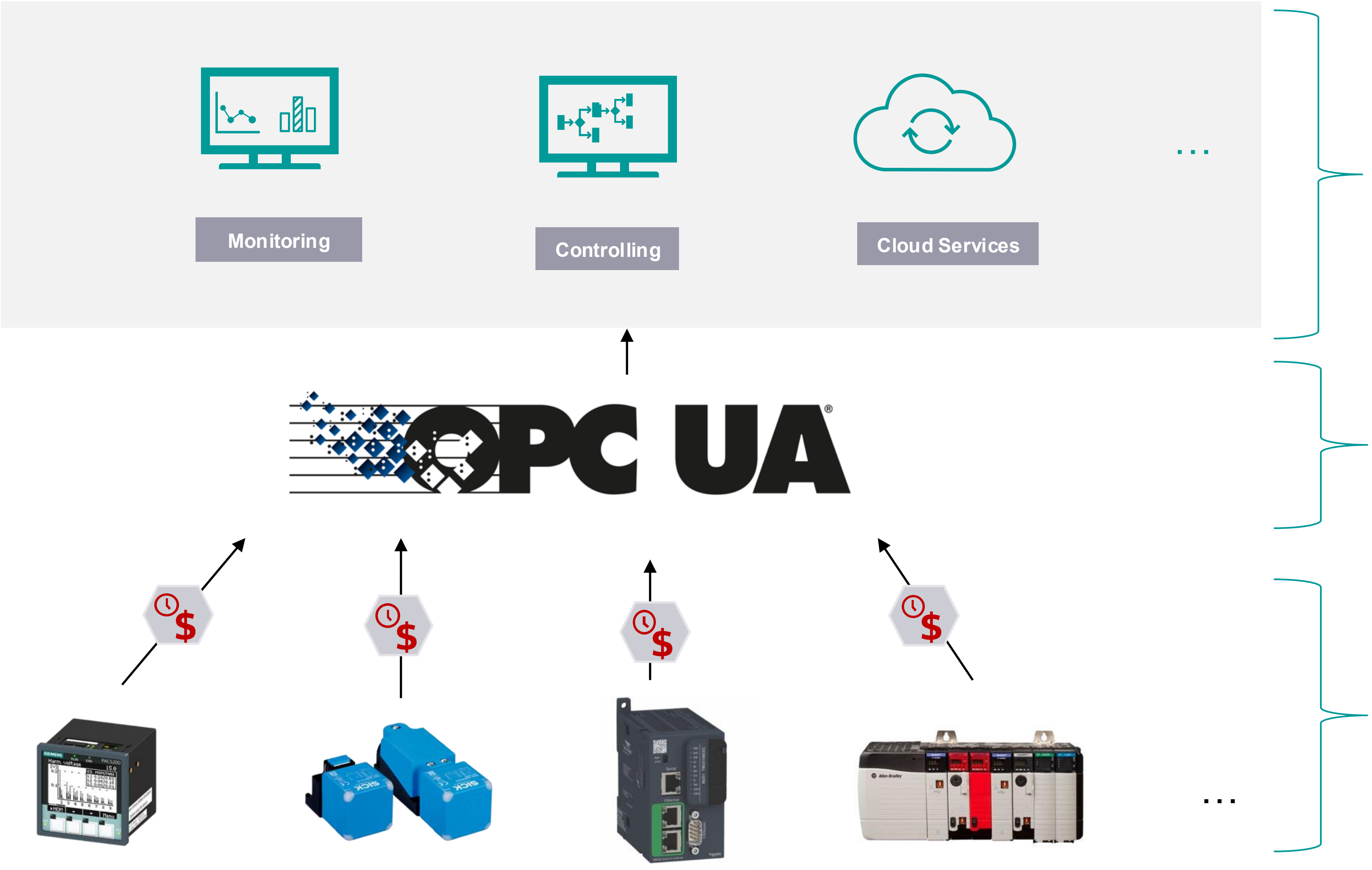
OPC UA Liaision

Sebastian **Käbisch** | Siemens AG

November 14, 2025

Example Use Case: Industrial Asset Onboarding to OPC UA



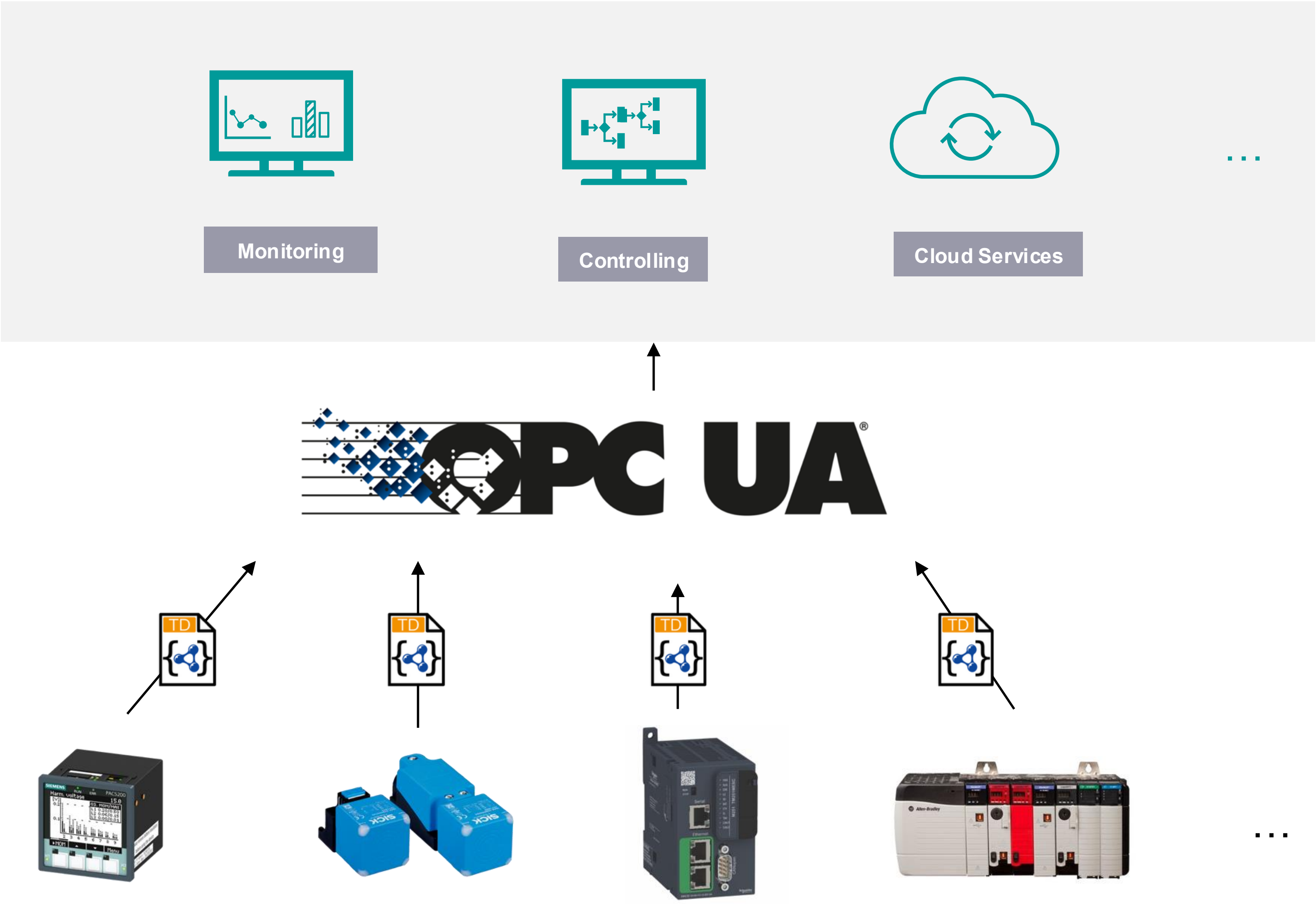
Target application based on OPC UA communication

Normalized, standardized, open data model & interface for all assets, including security!

Time & cost-intensive* integration due to the heterogeneous data models, protocols and security requirements of the shop floor

*Softing AG at “Beyond IoT 2025”: Up to 4000,- EUR per asset

Solution: WoT TD can be used for Asset Onboarding



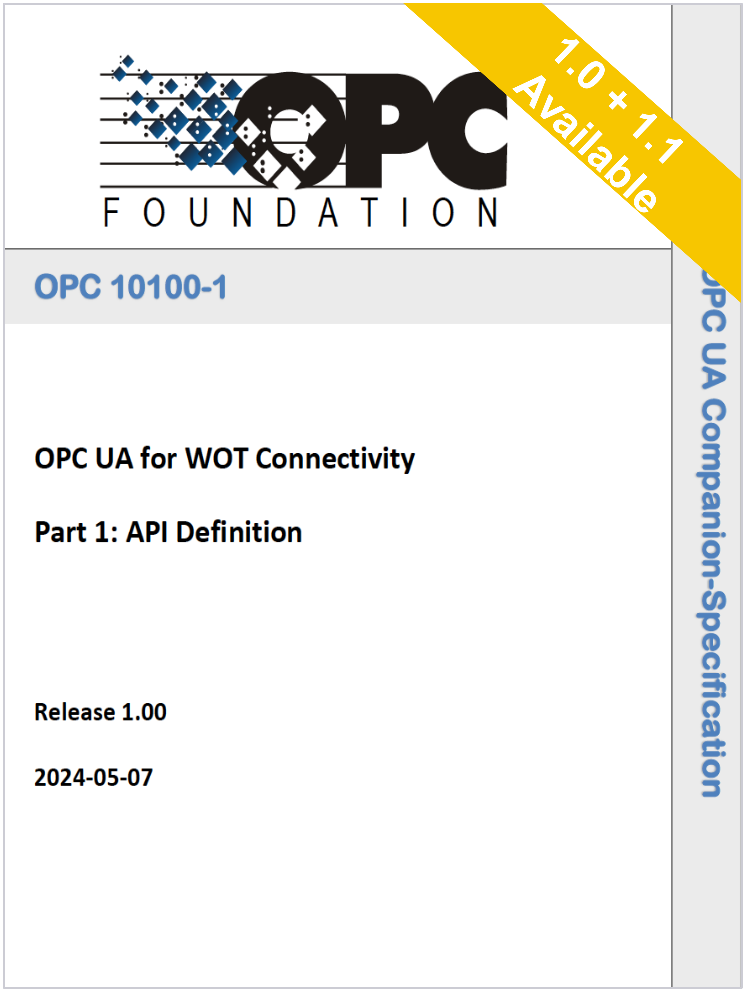
Target application based on OPC UA communication

Normalized, standardized, open data model & interface for all assets, including security!

WoT Thing Description enables auto IoT asset onboarding + enables rich semantic annotation

Enter OPC UA for WoT Connectivity

- WG lead by Erich Barnstedt (Microsoft)
- Main goal: simplify the asset onboarding by the usage of the WoT Thing Description Standard
- Version 1.1 available
- Currently working on Version 1.2
- Welcome to join: Bi-weekly on Tuesday, 3-4CET (next meeting October 14, 2025)



1.0 + 1.1 Available

OPC 10100-1

OPC UA for WOT Connectivity

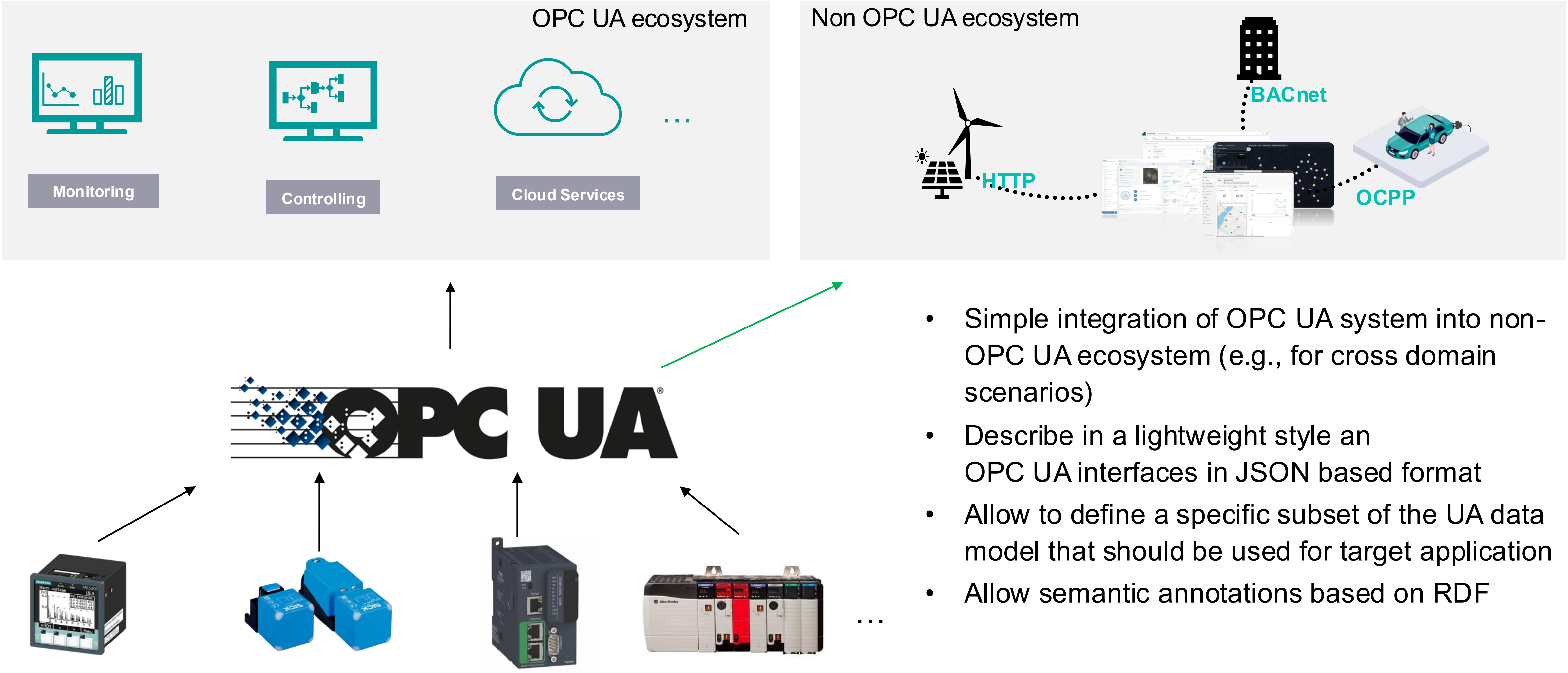
Part 1: API Definition

Release 1.00

2024-05-07

OPC UA Companion-Specification

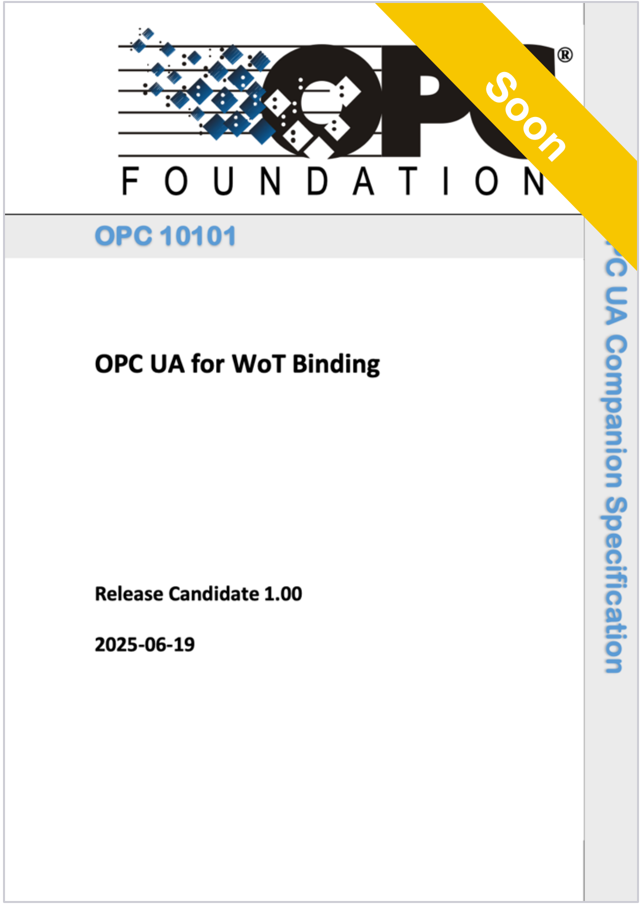
Motivations



- 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)



Soon

FOUNDATION

OPC 10101

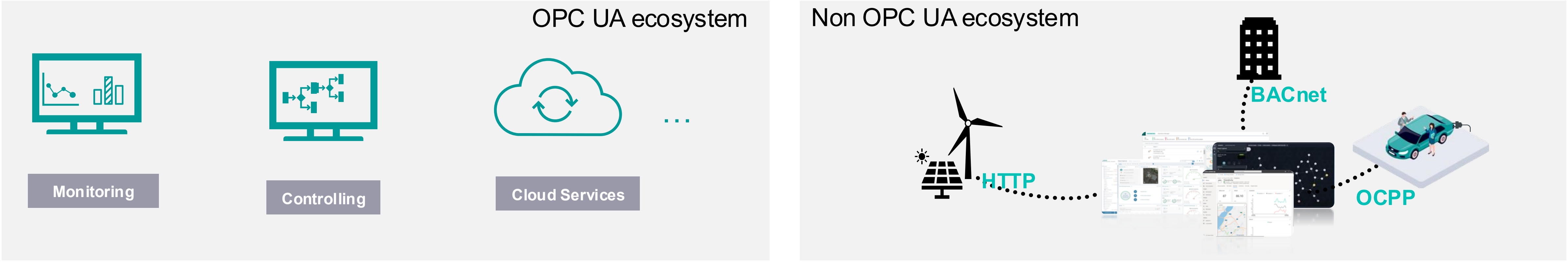
OPC UA for WoT Binding

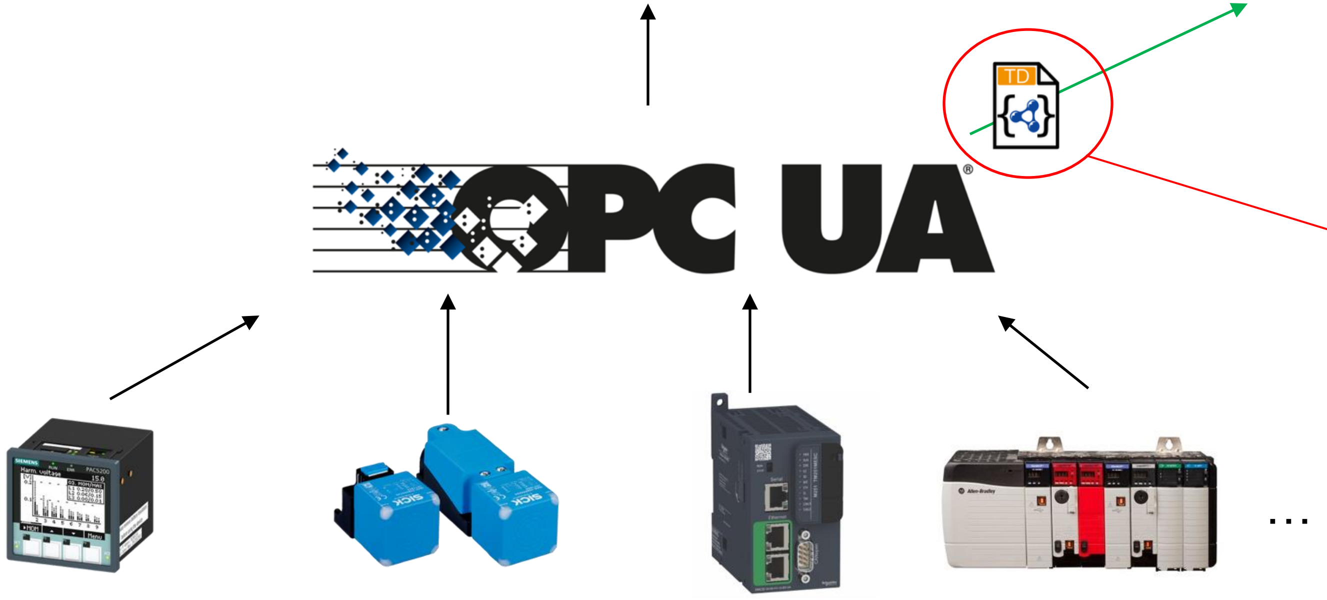
Release Candidate 1.00

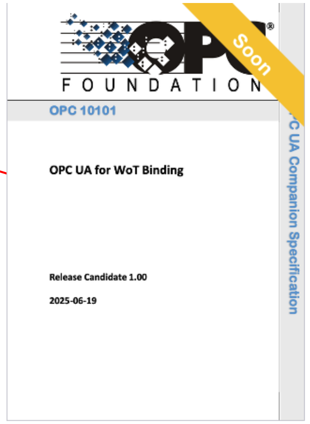
2025-06-19

C UA Companion Specification

Use WoT Thing Description to describe OPC UA interface






Soon

OPC FOUNDATION®

OPC 10101

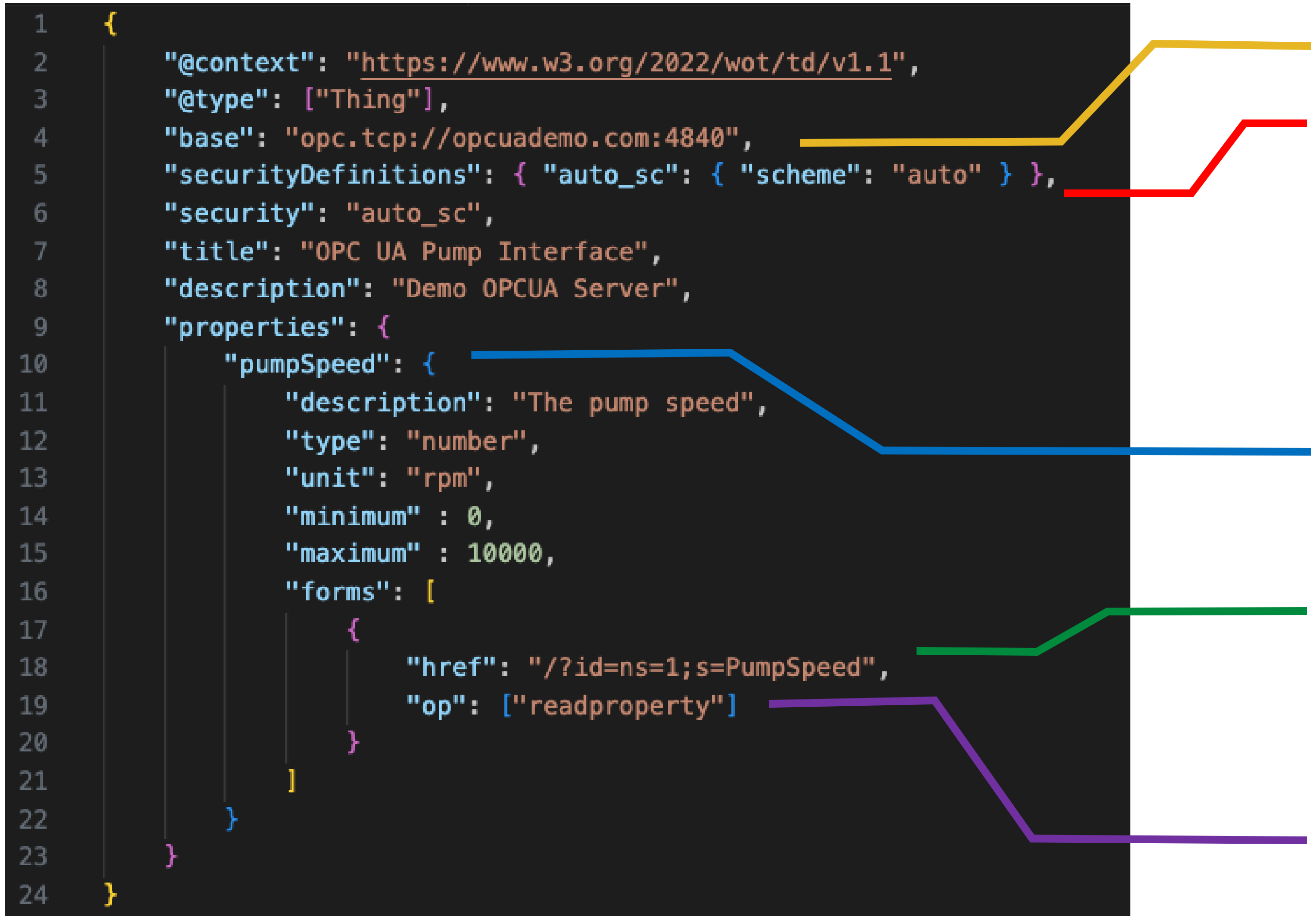
C UA Companion Specification

OPC UA for WoT Binding

Release Candidate 1.00

2025-06-19

Example (Minimal)



```
{
    "@context": "https://www.w3.org/2022/wot/td/v1.1",
    "@type": ["Thing"],
    "base": "opc.tcp://opcuademo.com:4840",
    "securityDefinitions": { "auto_sc": { "scheme": "auto" } },
    "security": "auto_sc",
    "title": "OPC UA Pump Interface",
    "description": "Demo OPCUA Server",
    "properties": {
        "pumpSpeed": {
            "description": "The pump speed",
            "type": "number",
            "unit": "rpm",
            "minimum" : 0,
            "maximum" : 10000,
            "forms": [
                {
                    "href": "/?id=ns=1;s=PumpSpeed",
                    "op": ["readproperty"]
                }
            ]
        }
    }
}
```

UA server location

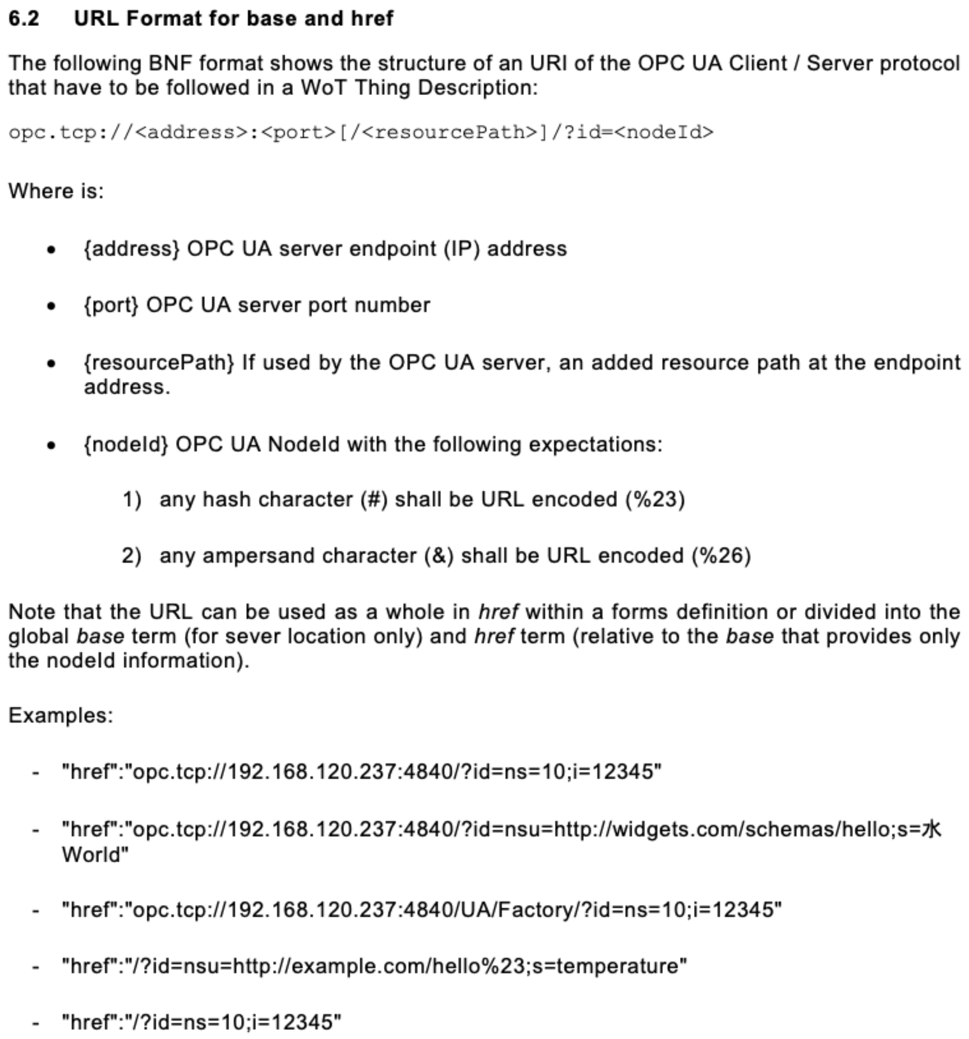
UA security mode

UA Variable

UA nodeId (also nsu representation is possible)

Access level

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:

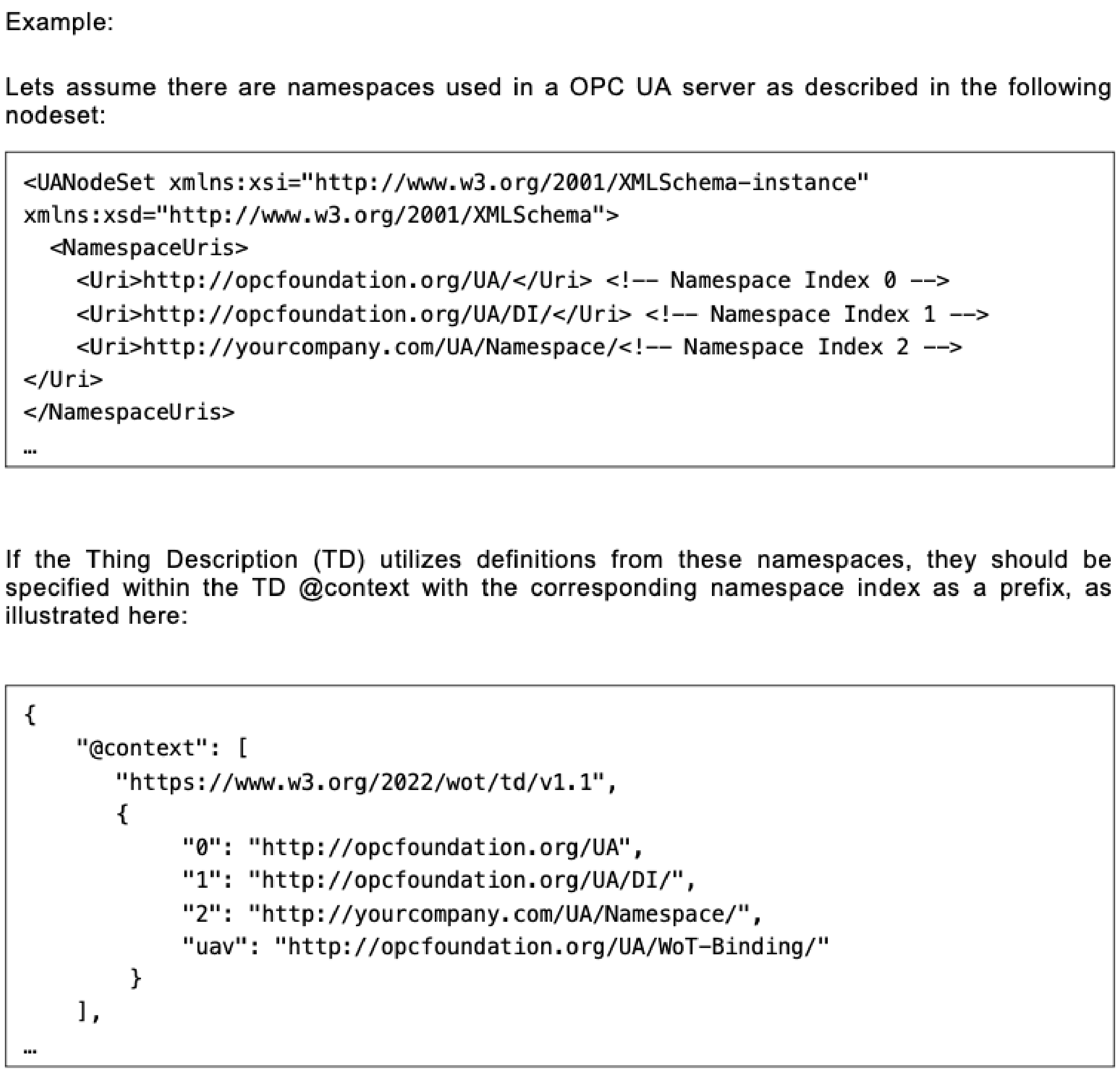
- {address} OPC UA server endpoint (IP) address
- {port} OPC UA server port number
- {resourcePath} If used by the OPC UA server, an added resource path at the endpoint address.
- {nodeId} OPC UA NodeId with the following expectations:
  1) any hash character (#) shall be URL encoded (%23)
  2) any ampersand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

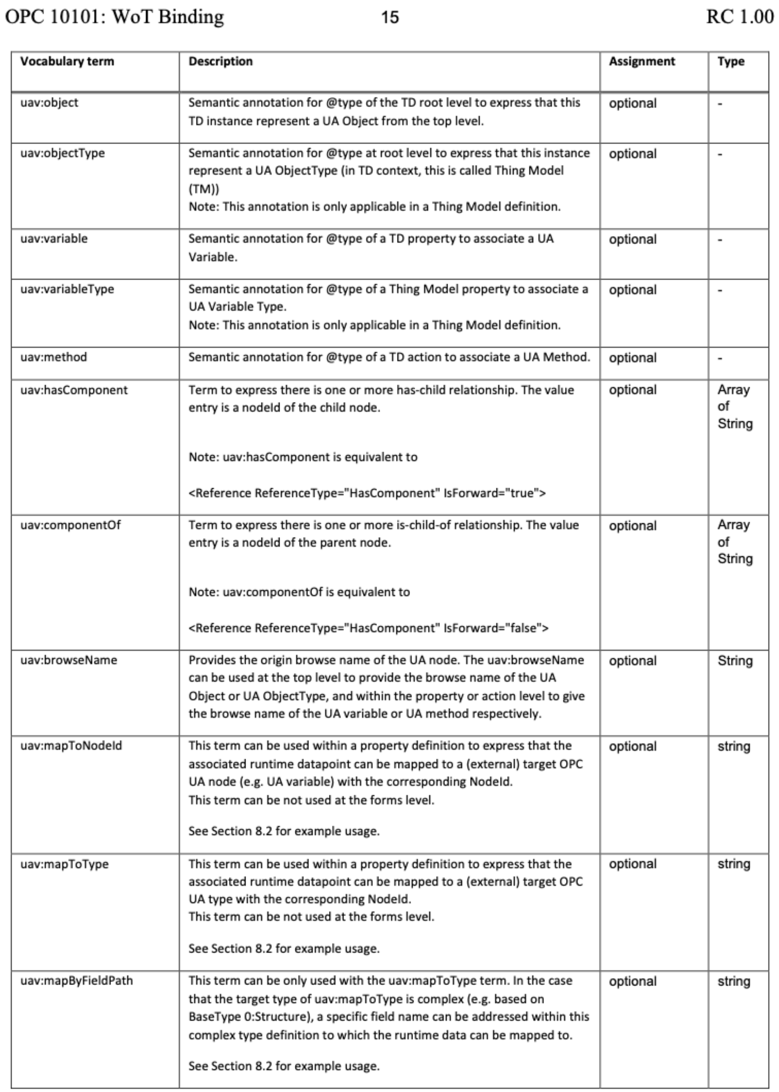
Lets assume there are namespaces used in a OPC UA server as described in the following nodeset:

```
<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/Namespace/<!-- 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:

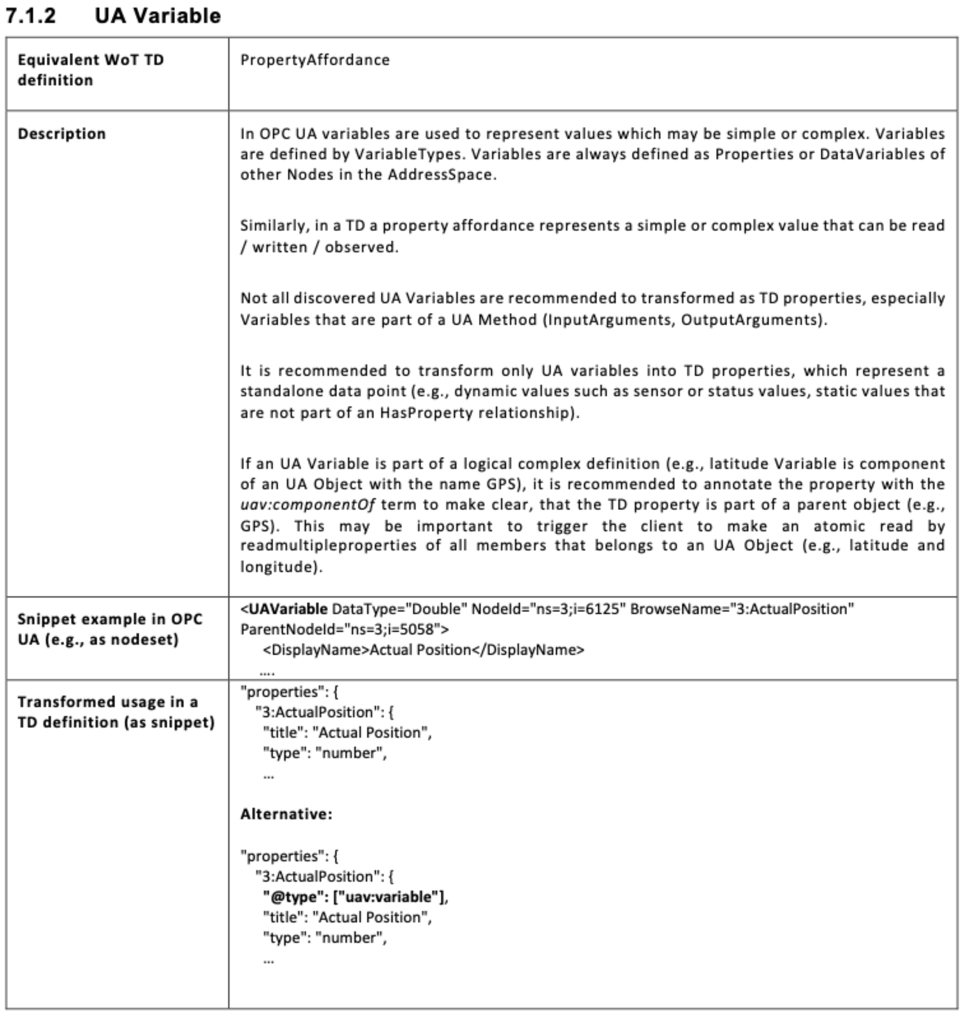
```
{
    "@context": [
       "https://www.w3.org/2022/wot/td/v1.1",
       {
            "0": "http://opcfoundation.org/UA",
            "1": "http://opcfoundation.org/UA/DI/",
            "2": "http://yourcompany.com/UA/Namespace/",
            "uav": "http://opcfoundation.org/UA/WoT-Binding/"
  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

OPC UA Terms for annotating Thing Descriptions (from 6.5.4)



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 (Example UA Variable; Chapter 7)



7.1.2 UA Variable

Equivalent WoT TD definition	PropertyAffordance
Description	In OPC UA variables are used to represent values which may be simple or complex. Variables are defined by VariableTypes. Variables are always defined as Properties or DataVariables of other Nodes in the AddressSpace. Similarly, in a TD a property affordance represents a simple or complex value that can be read / written / observed. Not all discovered UA Variables are recommended to transformed as TD properties, especially Variables that are part of a UA Method (InputArguments, OutputArguments). It is recommended to transform only UA variables into TD properties, which represent a standalone data point (e.g., dynamic values such as sensor or status values, static values that are not part of an HasProperty relationship). If an UA Variable is part of a logical complex definition (e.g., latitude Variable is component of an UA Object with the name GPS), it is recommended to annotate the property with the *uav:componentOf* term to make clear, that the TD property is part of a parent object (e.g., GPS). This may be important to trigger the client to make an atomic read by readmultipleproperties of all members that belongs to an UA Object (e.g., latitude and longitude).

Snippet example in OPC UA (e.g., as nodeset)	<**UAVariable** DataType="Double" NodeId="ns=3;i=6125" BrowseName="3:ActualPosition" ParentNodeId="ns=3;i=5058">
	    <DisplayName>Actual Position</DisplayName>
	    ....

Transformed usage in a TD definition (as snippet)	"properties": {
	  "3:ActualPosition": {
	    "title": "Actual Position",
	    "type": "number",
	    ...
	Alternative:
	"properties": {
	  "3:ActualPosition": {
	    "@type": ["uav:variable"],
	    "title": "Actual Position",
	    "type": "number",
	    ...