W3C WoT Face-to-Face Meeting
Bundang, Korea, July 2018
W3C Web of Things – Summary

• Counter fragmentation in the IoT
  – Web of Things (WoT) vs Internet of Things (IoT) is similar to World Wide Web vs Internet
  – Take patterns from the World Wide Web and adapt and apply them to the IoT
    ▪ JSON, Schema, and Linked Data
    ▪ URIs and Media Types
    ▪ JavaScript runtime

• By Describing and Complementing
  – Not competing with existing IoT standards, as not prescribing a full-stack solution
  – W3C WoT offers building blocks to pick that enable semantic interoperability
    ▪ WoT Thing Description (TD)
    ▪ WoT Binding Templates
    ▪ WoT Scripting API
W3C Web of Things – Activity

• WoT Workshop, Berlin, 2014
  – Identify stakeholders for W3C activity
  – Siemens took a leading role chairing the activity

• W3C WoT Interest Group (IG)
  – Started spring 2015
  – ~200 participants
  – Exploration and liaisons (e.g., OPC, oneM2M, OCF)
  – Continues to exists parallel to Working Group

• W3C WoT Working Group (WG)
  – Started end of 2016
  – ~100 participants
  – Normative work on deliverables
W3C WoT Approach – Building Blocks

**WoT Thing Description (TD)**

- **JSON-LD** representation format to describe Thing **instances** with **metadata**. Uses **formal interaction model** and **domain-specific vocabularies** to uniformly describe how to use Things, which enables semantic interoperability.

**The index.html for Things**

- **Properties**
- **Events**
- **Actions**

**WoT Scripting API**

- Standardized **JavaScript** object API for an IoT runtime system **similar to the Web browser**. Provides an interface between applications and Things to simplify IoT application development and enable **portable apps** across vendors, devices, edge, and cloud.

**Runtime System**

- **Application**
- **SDK**

**Information Model**

- **Protocol**
- **Information Model**

**WoT Binding Templates**

- Capture how the **formal Interaction Model** is mapped to concrete protocol operations (e.g., CoAP) and platform features (e.g., OCF). These templates are re-used by concrete TDs.

**WoT Thing Description (TD)**

- **Properties**
- **Events**
- **Actions**

**JavaScript**

**WoT Scripting API**

- **JSON-LD** representation format to describe Thing **instances** with **metadata**. Uses **formal interaction model** and **domain-specific vocabularies** to uniformly describe how to use Things, which enables semantic interoperability.
W3C WoT Approach – Batteries Included

**WoT Thing Description (TD)**
JSON-LD representation format to describe Thing instances with metadata. Uses formal interaction model and domain-specific vocabularies to uniformly describe how to use Things, which enables semantic interoperability.

**JavaScript**

**WoT Scripting API**
Standardized JavaScript object API for an IoT runtime system similar to the Web browser. Provides an interface between applications and Things to simplify IoT application development and enable portable apps across vendors, devices, edge, and cloud.

**WoT Binding Templates**
Capture how the formal Interaction Model is mapped to concrete protocol operations (e.g., CoAP) and platform features (e.g., OCF). These templates are re-used by concrete TDs.

**WoT Security and Privacy**
metadata and guidelines for existing security (e.g., IETF ACE) Bindings

The *index.html* for Things

Properties

Events

Actions

HTTP

CoAP

MQTT

UA Binary

Modbus

BACnet
Deployment Scenarios – Full-stack Devices

- **WoT Client**
  - **WoT Scripting API**
  - **WoT Runtime**
  - **Client Script**
  - **Protocol Bindings**:
    - HTTP(S)
    - CoAP(S)
    - MQTT
    - ...

- **Servient Hosting Scripted Things**
  - **Thing Script**
  - **WoT Scripting API**
  - **WoT Runtime**
  - **Exposed Thing**
  - **Security Metadata**
  - **Protocol Bindings**:
    - HTTP(S)
    - CoAP(S)
    - MQTT
    - ...

- **System API**
  - **Local Hardware**
Deployment Scenarios – Constrained Devices

WoT Client

- Client Script
- WoT Scripting API
- WoT Runtime
- Protocol Bindings
  - HTTP(S)
  - CoAP(S)
  - MQTT
  - ...

Device as Thing

- Exposed Thing
- Device Firmware
- Protocol
  - CoAP(S)

Driver API

- Local Hardware

Host

Diagram showing the interaction between WoT Client and Device as Thing, with Protocol Bindings and a Client Script.
Deployment Scenarios – Retrofitting Devices

WoT Client
- Client Script
- WoT Scripting API
- WoT Runtime

Protocol Bindings
- HTTP(S)
- CoAP(S)
- MQTT
- Modbus

Existing Device
- Device Firmware
- Driver API
- Local Hardware

Protocol
- Modbus

Host

Diagram showing the interaction between the WoT Client and the Existing Device, with various protocols and APIs involved.
Deployment Scenarios – Web Integration

Servient in Browser
- Application Script
- WoT Scripting API
- Protocol Bindings: HTTP(S), (S)RTP, FTP, ... 

Servient on Cloud
- Proxy Script(s)
- WoT Scripting API
- Protocol Bindings: HTTP(S), CoAP(S), MQTT, ...

Servient on Gateway
- Integration Scripts
- Security Metadata
- WoT Scripting API
- Protocol Bindings: HTTP(S), CoAP(S), MQTT, ...
- System API: Local Hardware, Proprietary Communication

Integration Scripts
- Exposed Thing
- Consumed Thing

Browser + Polyfill
- WoT Runtime
- Security Metadata

Thing
- Legacy Device
Deployment Scenarios – Web Integration

Servient in Browser
- Application Script
- WoT Scripting API
- Protocol Bindings: HTTP(S), (S)RTP, FTP

Servient on Cloud
- Cloud-native Implementation
- Protocol Bindings: HTTP(S), CoAP(S), MQTT

Servient on Gateway
- Integration Scripts
- Security Metadata
- WoT Scripting API
- Protocol Bindings: HTTP(S), CoAP(S), MQTT
- System API: Local Hardware, Proprietary Communication

Integration: Servient in Browser → Servient on Cloud → Servient on Gateway

Exposed Thing → Consumed Thing

Browser + Polyfill

WoT Runtime

Thing

Legacy Device
W3C WoT Integration Patterns – Things AND Cloud

- **Cloud**
  - Servient
    - Digital Twin
    - Orchestr. Client
    - Digital Twin
    - Scripting API
    - Interaction Model
    - Bindings
  - Remote Access and Synchronization

- **Edge**
  - Servient
    - Virtual Thing
    - Orchestr. Client
    - Proxy Thing
    - Scripting API
    - Interaction Model
    - Bindings
  - Integrate and Orchestrate Devices

- **Web Integration**
  - Web Browser
    - App Script
    - Scripting API
    - Interaction Model
    - Bindings
  - Direct Thing-to-Thing Interaction

- **Existing Device**
  - Complement Existing Devices

**Additional Notes:**
- **Web Browser**
  - Web Integration

- **Thing**
  - Application Logic
  - Scripting API
  - Interaction Model
  - Bindings

- **Constr. Thing**
  - Classic Firmware
  - Interaction Model
  - Protocol

**Diagram Descriptions:**

- **Remote Access and Synchronization:** Connections between Cloud and Edge services for seamless integration.
- **Direct Thing-to-Thing Interaction:** Connections between Things for direct communication.
- **Integrate and Orchestrate Devices:** Connections for integrating and orchestrating existing devices.
Editors’ Draft: https://w3c.github.io/wot-thing-description/

WOT THING DESCRIPTION
WoT WG Changed to “Simplified TD” in March 2018

• JSON-LD 1.1 processing
  – Objects instead of arrays
  – Default values (e.g., "writable": false)
  – Framing to serialize and preprocess

• Semantic annotations optional
  – TDs can be treated as simple JSON format
  – No JSON-LD keywords or processing required
  – properties, actions, events on top level
  – No LD convention of terms being singular
  – New Media Type application/td+json
  – Context and terms known implicitly

• JSON Schema compatible
  – Data schema syntax now also identical
  – Payloads can be validated directly with JSON Schema implementations

• New terms
  – id
  – label
  – description
  – support
  – ... collecting more
WG changed from the “JSON-LD 1.0 based TD” to the “Simplified TD”

DIFF FOR SIMPLIFIED TD
WoT WG Changed to “Simplified TD”

```
{
  "@context": "https://./w3c-wot-td-context.jsonld",
  "name": "Lamp",
  "@id": "urn:dev:wot:com:example:servient:lamp",
  "base": "https://servient.example.com/things/lamp/",
  "interaction": [{
    "@type": "Property",
    "name": "on",
    "schema": { "type": "boolean" },
    "writeable": false,
    "form": [{
      "href": "properties/on",
      "mediaType": "application/json"
    }]
  }, {
    "@type": "Property",
    "name": "brightness",
    "writeable": true,
    "schema": {
      "type": "integer",
      "minimum": 0,
      "maximum": 100
    },
    "form": [{
      "href": "properties/status",
      "mediaType": "application/json"
    }]
  }, {
    "@type": "Action",
    "id": "urn:dev:wot:com:example:servient:lamp",
    "label": "Lamp",
    "description": "Corner torchiere",
    "base": "https://servient.example.com/things/lamp/",
    "properties": {
      "on": {
        "type": "boolean",
        "forms": [{
          "href": "properties/on",
          "mediaType": "application/json"
        }]
      },
      "brightness": {
        "type": "integer",
        "minimum": 0,
        "maximum": 100
      }
    },
    "actions": {
      "fade": {
        "type": "integer",
        "minimum": 0,
        "maximum": 100
      }
    }
  }
}
```
WoT WG Changed to “Simplified TD”

```
}, {
    "@type": "Action",
    "name": "fade",
    "inputSchema": {
        "type": "object",
        "fields": [
            {
                "name": "from",
                "schema": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                }
            },
            {
                "name": "to",
                "schema": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                }
            },
            {
                "name": "duration",
                "schema": { "type": "number" }
            }
        ]
    },
    "forms": [{
        "href": "actions/fade",
        "outputMediaType": "application/json",
        "inputMediaType": "application/json"
    }]
}],
"form": [{ "href": "actions/fade" }]
}, {
"actions": {
    "fade": {
        "input": {
            "type": "object",
            "properties": {
                "from": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                },
                "to": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                },
                "duration": { "type": "number" }
            }
        },
        "forms": [{
            "href": "actions/fade",
            "outputMediaType": "application/json",
            "inputMediaType": "application/json"
        }]
    }
},
"events": {
```

OLD

NEW
WoT WG Changed to “Simplified TD”

OLD

}, {
   "@type": "Event",
   "name": "overheated",
   "schema": {
      "type": "object",
      "fields": [{
         "name": "temperature",
         "schema": { "type": "number" }
      }]
   }
},
"form": [{
   "href": "events/overheated",
   "mediaType": "application/json"
}]
},
"link": [{
   "href": "https://servient.example.com/things/pir",
   "rel": "controlledBy",
   "mediaType": "application/td+json"
}]

NEW

"events": {
   "overheated": {
      "type": "object",
      "properties": {
         "temperature": { "type": "number" }
      },
      "forms": [{
         "href": "events/overheated",
         "subProtocol": "EventSource",
         "mediaType": "application/json"
      }]
   }
},
"links": [{
   "href": "https://servient.example.com/things/pir",
   "rel": "controlledBy",
   "mediaType": "application/td+json"
}]
}
WG changed from the “JSON-LD 1.0 based TD” to the “Simplified TD”

SIMPLIFIED TD IN DETAIL
WoT Thing Description (TD) – Minimal Form

```json
{
  "id": "urn:dev:org:32473:1234567890",
  "label": "MyLEDThing",
  "description": "RGB LED torchiere",
  "security": [{
    "scheme": "Bearer",
  }],
  "properties": {
    "brightness": {
      "type": "integer",
      "minimum": 0,
      "maximum": 100,
      "writable": true,
      "forms": [ ... ]
    }
  },
  "actions": {
    "fadeIn": {
      "input": {
        "type": "object",
        "properties": {
          "target": {
            "type": "integer",
            "minimum": 0,
            "maximum": 100,
          }
        }
      }
    }
  }
}
```

No semantic annotations

Media Type application/td+json identifies document as TD and TD context and vocabulary apply as default
WoT Thing Description (TD) – Annotated Form

```json
{
    "@context": [
        "https://w3c.github.io/wot/w3c-wot-td-context.jsonld",
        { "iot": "http://iotschema.org/" }
    ],
    "id": "urn:dev:org:32473:1234567890",
    "label": "MyLEDThing",
    "description": "RGB LED torchere",
    "@type": ["Thing", "iot:LightControl"],
    "security": [{
        "scheme": "OAuth2",
        "token": "PoP",
        "as": "https://authority-issuing.example.org"
    }],
    "properties": {
        "brightness": {
            "type": "integer",
            "minimum": 0,
            "maximum": 100,
            "@type": ["iot:CurrentDimmer"],
            "iot:Unit": "iot:Percent",
            "writable": true,
            "forms": [ ... ]
        }
    },
    "actions": {
        "fadeIn": {
            // Explicit context for additional vocabularies
            // W3C WoT TD vocabulary
            // Domain-specific vocabulary
            // Security metadata
            // Protocol Bindings
            // Semantic annotations
            // Domain-specific vocabulary
        }
    }
}
```
WoT Binding Templates – Instantiated in TDs

... "properties": {
  "brightness": {
  ...
  "forms": [ 
    { /* Default: GET to read, PUT to write */
      "href": "https://myled.example.com:8080/pwr",
      "mediaType": "application/json"
    } 
  ]
},
"actions": {
  "fadeIn": {
  ...
  "forms": [ 
    { /* PUT instead of POST to invoke */
      "href": "coaps://myled.example.com:5684/pwr",
      "mediaType": "application/ocf+cbor",
      "coap:methodCode": 3,
      "coap:options": [ { 
        "coap:optionNumber": 2053,
        "coap:optionValue": "1.1.0"
      } ]
    }
  ]
}
WoT Thing Description (TD) – Model

[Diagram of WoT Thing Description (TD) – Model]
WoT Thing Description – Terms

- **@context**
- **@type**
- **id**
- **name**
- **description**
- **support**
- **base**
- **security**
- **properties**
- **actions**
- **events**
- **links**

- **JSON-LD Context**
  - Gives semantic meaning to terms used in JSON document
  - Configures semantic processing (e.g., how to handle object keys)
  - See JSON-LD spec: [https://json-ld.org/spec/latest/json-ld/#the-context](https://json-ld.org/spec/latest/json-ld/#the-context)

- **Optional**
  - Default points to TD context file
    - [https://w3c.github.io/wot/w3c-wot-td-context.jsonld](https://w3c.github.io/wot/w3c-wot-td-context.jsonld)
    - Will be changed to w3.org namespace once finished

- **Used to include domain-specific vocabularies**
  - Example: `{"iot": "https://iot.schema.org/"}`
    - Note: [iotschema.org](http://iotschema.org) is the temporary work-in-progress domain

- **Default TD context file always applied last**
  - Protects against overrise of terms
WoT Thing Description – Terms

- `@context`
- `@type`
- `id`
- `name`
- `description`
- `support`
- `base`
- `security`
- `properties`
- `actions`
- `events`
- `links`

- RDF Type of the Thing
  - Also see JSON-LD specification
  - [https://json-ld.org/spec/latest/json-ld/#typed-values](https://json-ld.org/spec/latest/json-ld/#typed-values)

- Optional
  - Default is “Thing” from TD core vocabulary

- Used for Capabilities
  - Things can offer multiple Capabilities
  - Come from domain-specific vocabularies
  - See “iot.schema.org” activity for examples
    - Note: [iotschema.org](https://iotschema.org) is the temporary working domain

- Strict Thing types are discouraged
  - Forcing Things into strict types does not scale
  - Might be required in well-defined, closed systems
  - Strict type can be seen as a very specific “capability”
WoT Thing Description – Terms

- **@context**
- **@type**
- **id**
- **name**
- **description**
- **support**
- **base**
- **security**
- **properties**
- **actions**
- **events**
- **links**

- Identifier of the physical/unique Thing
  - Mandatory

- **Must be a URI**
  - Maps to the “@id” of the Linked Data node (Thing)
  - Usually a URN is used, e.g., UUID or IMEI URN

- Same ID can occur in multiple TDs
  - Describes same physical/unique Thing (or “asset”)
  - TD directly from the device
  - TD registered in a Thing Directory
  - Alternative TD serialization, e.g., different language
WoT Thing Description – Terms

- `@context`
- `@type`
- `id`
- `name`
- `description`
- `support`
- `base`
- `security`
- `properties`
- `actions`
- `events`
- `links`

- Human-readable name for the Thing
  - Mandatory

- Used for displaying text in a UI or documentation
  - Can be language-dependent
  - One value, changed by serializing alternative TD
WoT Thing Description – Terms

- @context
- @type
- id
- name
- **description**
- support
- base
- security
- properties
- actions
- events
- links

- Human-readable description for the Thing
- Optional

- Used for displaying text in a UI or documentation
  - Can be language-dependent
  - One value, changed by serializing alternative TD

- **Discussion**: how to deal with language?
  - Serialize language-specific TDs (cf. Accept-Lang)?
  - Have a map with language identifiers as key?
  - Model must be able to hold multiple languages to serialize language-specific TDs!
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- **support**
- base
- security
- properties
- actions
- events
- links

- Web Link (URI) to TD author or Thing vendor information
  - Optional

- Used for contact/responsibility information
  - URI for Web page
  - mailto URI
  - Linked Data resource

- **Discussion**: make mandatory?
  - Force authors to fill out for better data quality of TDs
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Base URI for optimization to enable short relative URIs in forms
  - Optional

- Used for optimization
  - Usually, all Interactions have the same “href” prefix
  - Can use relative URIs there, which must be resolved to absolute URIs

- Only one base
  - Must use absolute URIs for alternate endpoints
  - Must use absolute URIs for alternate URI schemes

- Origin URI of the TD usually does not work
  - TDs often come from a Thing Directory, which is on different endpoint
  - TDs can be hosted externally
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Security metadata to interact with Thing
- Optional

- Takes TD security vocabulary
  - scheme, e.g., “basic”, “apikey”, “oauth2”
  - Scheme-specific fields, e.g., “in”, “pname”, “flow”

- Discussion: make mandatory?
  - No-security must be an explicit choice
  - Security should be used by default

- Discussion: vocabulary
  - Currently looked at OpenAPI and a few specific platforms (e.g., OCF)
  - Need more feedback
WoT Thing Description – Terms

- `@context`
- `@type`
- `id`
- `name`
- `description`
- `support`
- `base`
- `security`
- `properties`
- `actions`
- `events`
- `links`

- Entry point for Property Interactions
- Optional
  - Thing might not have Properties
- Expose internal state of Things for direct access
  - Always readable (don’t use Property, but Action if not)
  - Optionally writable
  - Optionally observable
    (different from Events as state transfer / eventual consistency)
- Object structure with handle as key and Property as value
  - With JSON-LD 1.1 processing, keys map to `@id` of value node
- Property nodes have JSON Schema terms and syntax
  - Property description can be fed to JSON Schema validator
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Property vocabulary
  - label (optional)
  - description (optional)
  - @type (optional)
  - type (mandatory)
    - any → const, enum
    - object → properties, required, dependencies, additionalProperties, ...
    - array → items, additionalItems, maxItems, minItems, uniqueItems, contains
    - string → maxLength, minLength, pattern
    - number → minimum, maximum, multipleOf, exclusiveMinimum/-Maximum
    - integer → minimum, maximum, multipleOf, exclusiveMinimum/-Maximum
    - boolean
      - writable (optional – default false)
      - observable (optional – default false)
      - forms (mandatory for Interaction)

- Any domain-specific vocabulary can be added
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Property vocabulary
  - label (optional)

- **Discussion**: use term “name”?
  - Maps to “rdfs:label”
  - Web Linking has term “title” (was for documents)
  - Has no identification purpose, so no name, but...
  - Corresponds to HTML’s “name” (DOM’s “getElementsByTagName()”)

- Maps to ”rdfs:label”

- Web Linking has term “title” (was for documents)

- Has no identification purpose, so no name, but...

- Corresponds to HTML’s “name” (DOM’s “getElementsByTagName()”)

- Discussion: use term “name”?
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Entry point for Action Interactions
- Optional
  - Thing might not provide Actions

- Modify internal state or actuate or process data
  - State that is not exposed directly (no Property)
  - Write only, read multiple, etc.

- Deal with intended vs actual state
  - Physical processes that take time
  - State that changes over time

- Data schema for input and output of the Action
  - input/output node can be fed to JSON Schema validator
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Action vocabulary
  - label (optional)
  - description (optional)
  - @type (optional)
  - input (optional)
    - data schema in JSON Schema syntax, see Property
  - output (optional)
    - data schema in JSON Schema syntax, see Property
  - forms (mandatory)

- Any domain-specific vocabulary can be added
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Entry point for Event Source Interactions
- Optional
  - Thing might not emit Events

- Push messages from Things
  - May be triggered through conditions that are not exposed as state
  - Not state, but state transitions (events) are communicated
  - In practice best effort consistency applies

- Events directly have data schema in JSON Schema syntax
  - Event source description can be fed to JSON Schema validator
WoT Thing Description – Terms

- @context
- @type
- id
- name
- description
- support
- base
- security
- properties
- actions
- events
- links

- Event vocabulary
  - label (optional)
  - description (optional)
  - @type (optional)
  - type (mandatory, data schema in JSON Schema syntax)
  - forms (mandatory)

- Discussion: have input and output?
  - Some subscription mechanisms need parameters such as filters
  - “events” describe event sources, not data instances, hence they are similar to Actions with their data specifications!
WoT Thing Description – Terms

- `@context`
- `@type`
- `id`
- `name`
- `description`
- `support`
- `base`
- `security`
- `properties`
- `actions`
- `events`
- `links`

Web Links to other Things or additional metadata
  - Optional

Resources in “links” are fetchable
  - Opposed to Linked Data predicates (e.g., domain-specific vocabulary)
  - `href` field for URI

Provide relations to other Things
  - `rel` gives relation type, e.g., contains, partOf, controls
  - `mediaType` is `application/td+json`

Provide additional metadata
  - Serves as extension point (cf. CSS for HTML)
WG changed from the “JSON-LD 1.0 based TD” to the “Simplified TD”

NEW SCRIPTING API
let thing = WoT.produce({
   name: "counter",
   // no support for
   // more metadata
});

console.log("Created thing " + thing.name);

thing.addProperty({
   name: "count",
   schema: '{ "type": "number"}',
   // no support for
   // custom metadata
   observable: true,
   writeable: true,
   value: 0
});

thing.addAction({ name: "increment" });
thing.setActionHandler("increment", () => {
   return thing.readProperty("count").then(res => {
      thing.writeProperty("count", ++res);
   });
});

let thing = WoT.produce({
   name: "counter",
   description: "counter example Thing",
   "@context": { "iot": "http://iotschema.org/" }
});

console.log("Created thing " + thing.name);

thing.addProperty("count",
   {
      type: "integer",
      description: "current counter value",
      "iot:Custom": "example annotation",
      observable: true,
      writeable: true
   },
   0);

thing.addAction("increment");
thing.setActionHandler("increment", () => {
   return thing.properties["count"].get().then(res => {
      thing.properties["count"].set(++res);
   });
});
WoT WG Changed to “Simplified TD”

### OLD

```javascript
WoT.fetch("http://localhost:8080/counter").then(td => {
  let thing = WoT.consume(td);

  thing.readProperty("count")
    .then(res => {
      console.info("count value is", res);
    })
    .catch(err => { console.error(err); });

  thing.invokeAction("increment");
});
```

### NEW

```javascript
let thing = WoT.consume(td);

thing.properties.count.get()
  .then(res => {
    console.info("count value is", res);
  })
  .catch(err => { console.error(err); });

thing.actions.increment.run();
```
Contact

Dr. Matthias Kovatsch
Senior Research Scientist

Siemens AG
CT RDA IOT EWT-DE

matthias.kovatsch@siemens.com