Web Thing REST API

TDfying and Differences

Ege Korkan, Siemens AG
@egekorkan

Web Thing REST API

Documentation of how the REST API works with example request and responses

EXAMPLE 11: Request an action

```
POST https://mythingserver.com/things/lamp/actions/
Accept: application/json

{
  "fade": {
    "input": {
      "level": 50,
      "duration": 2000
    }
  }
}
```

EXAMPLE 12: Response to creating an action request

```
201 Created

{
  "fade": {
    "input": {
      "level": 50,
      "duration": 2000
    },
    "href": "/things/lamp/actions/fade/123e4567-e89b-12d3-a456-426655",
    "status": "pending"
  }
}
```

Web Thing Description


```
{
    "title": "Virtual Actions & Events Thing",
    "@context": "https://iot.mozilla.org/schemas",
    "href": "/things/virtual-things-10",
    "actions": {
        "basic": {
            "title": "No Input",
            "description": "An action with no inputs, fires an event",
            "links": [{
                    "rel": "action",
                    "href": "/things/virtual-things-10/actions/basic"
                }]
        },
        "single": {
            "title": "Single Input",
            "description": "An action with a single, non-object input",
            "input": {
                "type": "number"
            },
            "links": [{
                    "rel": "action",
                    "href": "/things/virtual-things-10/actions/single"
                }]
        },
```

Web Thing Description


```
"events": {
    "virtualEvent": {
        "description": "An event from a virtual thing",
        "type": "number",
        "links": [{
                "rel": "event",
                "href": "/things/virtual-things-10/events/virtualEvent"
            }]
    }
},
```

```
"links": [ {
        "rel": "properties", "href": "/things/virtual-things-10/properties"
    }, {
        "rel": "actions",     "href": "/things/virtual-things-10/actions"
    }, {
        "rel": "events",      "href": "/things/virtual-things-10/events"
    }, {
        "rel": "alternate", "href": "ws://localhost:8080/things/virtual-things-10"
    } ],
"id": "http://localhost:8080/things/virtual-things-10",
"base": "http://localhost:8080/",
"securityDefinitions": {...}
"security": "oauth2_sc"
}
```

Building the Requests

Web Thing Description is not enough on its own to build the requests.

It is required to read the Web Thing REST API Specification, similar to knowing the default values in the W3C TD.

Building the Requests

Example: Invoking (Requesting) an Action called *basic* with no Input

Request:

```
{
  "basic":{
            "input":null
           }
}
```

Response:

```
{
  “basic”:{
            “input”:null,
            “href”:”actions/fade/123”,
            “status”:”created”
           }
}
```

Example: Writing to a Property called *on* with bool input

Request:

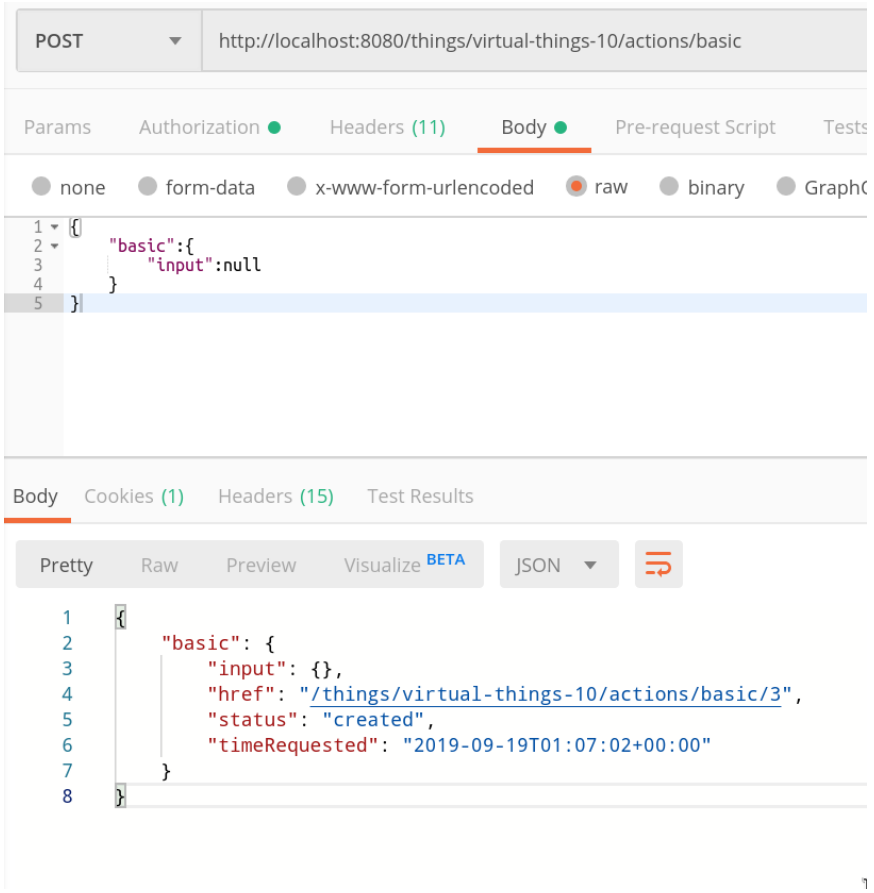
```
{
  “on”: true
}
```

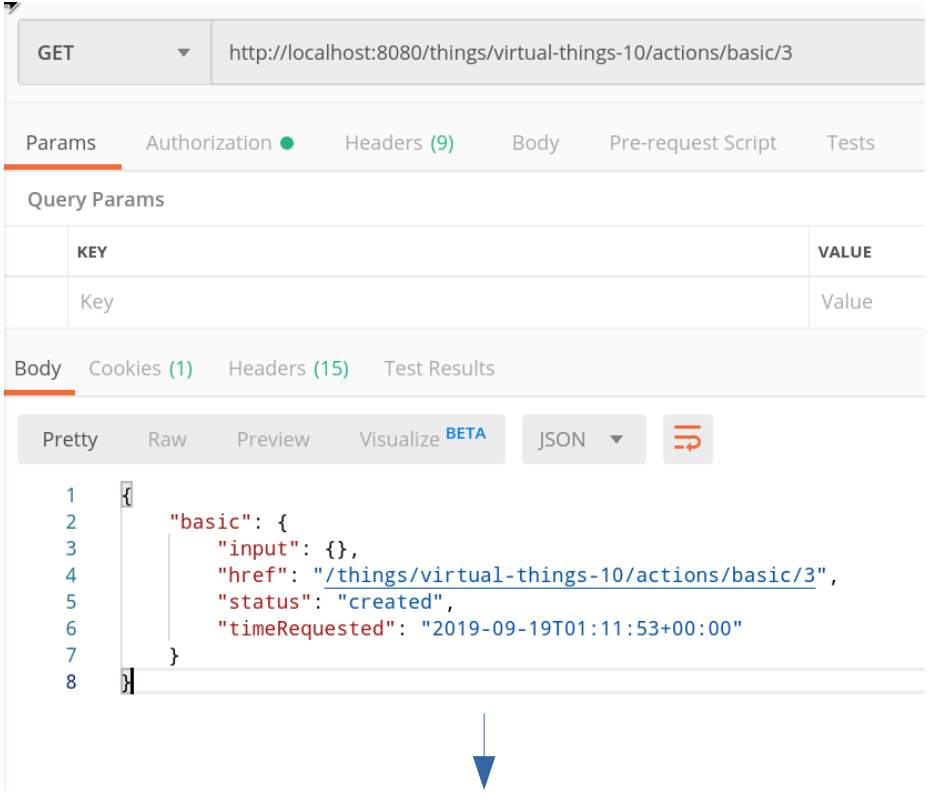
Response:

```
{
  “on”: true
}
```

Same for property read

Screenshots from Postman





This href also accepts DELETE and PUT

Building the Requests

Thus the sent and received data are always wrapped in a certain object structure. This results in the WebThing TD type being valid only inside this object.

This is easy to fix and we can describe WebThing payloads in a W3C TD Payload. See the attached TD.

Queues

Event queues for:

- Checking the past events
- Also Events are only a queue in HTTP, so not asynchronous

Can be described with a W3C TD but not a clear semantic definition

Queues

Action queues for:

- Monitoring actions' runtime => Can be described with a W3C TD
- Updating or canceling an action => **Cannot** be described with a W3C TD since the modifying an action requires the unique id that is assigned by the gateway when the action is invoked also we do not have an op value for these
  - Old issue by Kajimoto-san

W3C TD Transformation

- Most of the WebThing API can be described in a single TD which results in a very long TD.
- Steps:
  - Transform all the JSON Schemas into an object
  - Add 2 Property Affordances, one for event and one for action queues
  - Add a Property Affordance for each single event and action queue (not unique)
- Check the supplied TDs for all the information

What We Need

- We would need the following the be able to describe the Web Thing REST API in a W3C TD
  - op values for updating and deleting actions
  - A way to describe dynamic hrefs
    - RFC6570 → http://example.com/things/mything/actions/myaction/{actionID}
  - A way to pass “variables” between different positions of a TD
    - Using something like JSON Pointers → actionID is #/actions/*/output/href

Summary of Findings

- Describing simple capabilities of a Web Thing **device** is possible through W3C TD
- Describing the full capabilities of the Web Thing **Gateway** needs more features in the TD or another protocol
- WebThing API is purely about the gateway, client never accesses the end device