

# ECHONET Lite and/or Web API as Web of Things

Van Cu **PHAM**

Center for Digitalization Endeavors

Japan Advanced Institute of Science and Technology (JAIST)

*cupham@jaist.ac.jp*

June 16, 2021

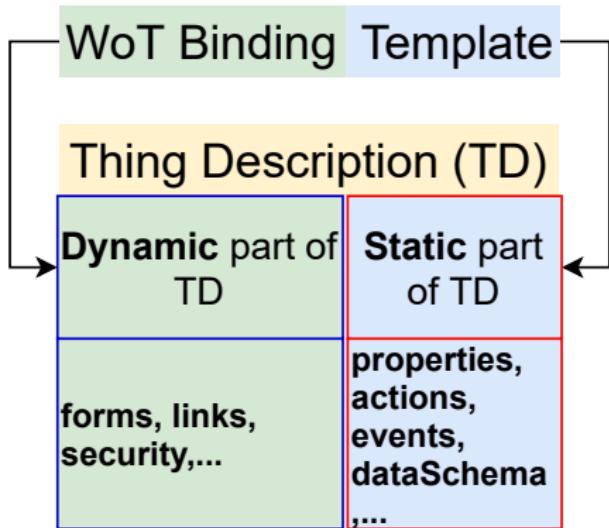
# Outline

- 1.1 WoT Binding Template Overview
- 1.2 ECHONET Lite WoT Cooperation Overview
- ① Thing Description Schemes
  - 2.1 WoT Thing Description (Overall)
  - 2.2 WoT Thing Description (Data Schema)
  - 2.3 Summary
- ② WoT Scripting API Usage
  - 3.1 Expose Thing Description API
- 3.2 Example: Expose a thing by API
- 3.3 Add corresponding handlers
- ③ Implementation
  - 4.1 EL-WoT Proxy Building Blocks
  - 4.2 Demonstration with 実験クラウド (Pattern 2)
  - 4.3 Demonstration with 実験クラウド (Pattern 3)
  - 4.4 Solutions for Pattern 3
- ④ Summary

# 1.1 WoT Binding Template Overview

## TODO:

- ① MRA (DD) → WoT **Thing Description** Mapping (**Static part**)
- ② Use WoT Scripting API to add **Dynamic** part at **run-time**
  - Library: **node-wot**
  - Self-Implementation
- ③ ECHONET Lite-WoT Gateway/Proxy Implementation



**Pattern 1:** ECHONET Lite <-> **EL-WoT Gateway** <-> WoT Client

**Pattern 2:** EL WebAPI <-> **EL-WoT Proxy** <-> WoT Client

**Pattern 3:** **EL-WoT Binding Proxy** + EL WebAPI <-> WoT Client

# 1.2 ECHONET Lite WoT Cooperation Overview

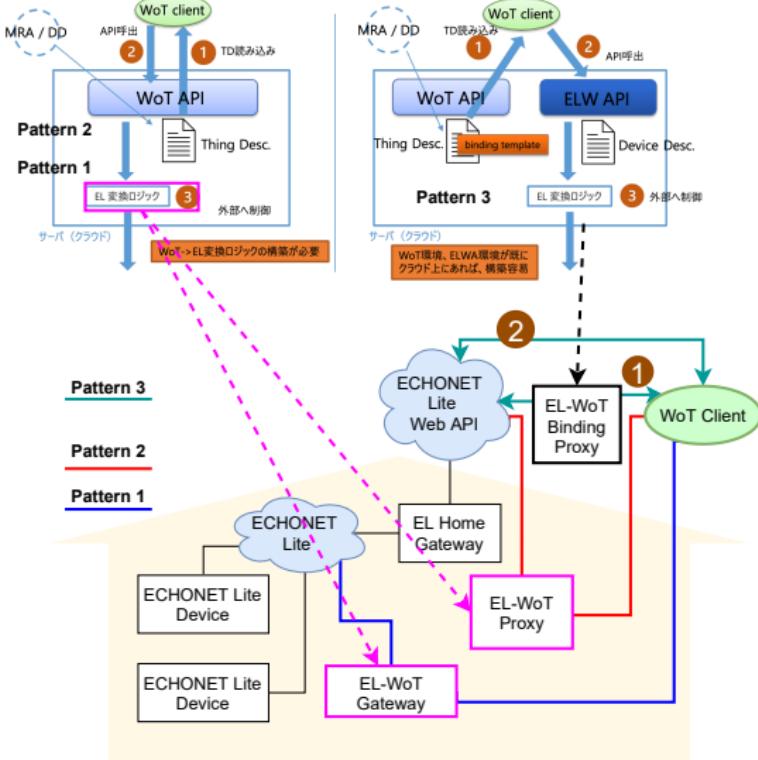
**Pattern 1** and **Pattern 2** are similar

- WoT Clients talk to a **Proxy/Gateway** → No need to adjust ELW API and WoT API

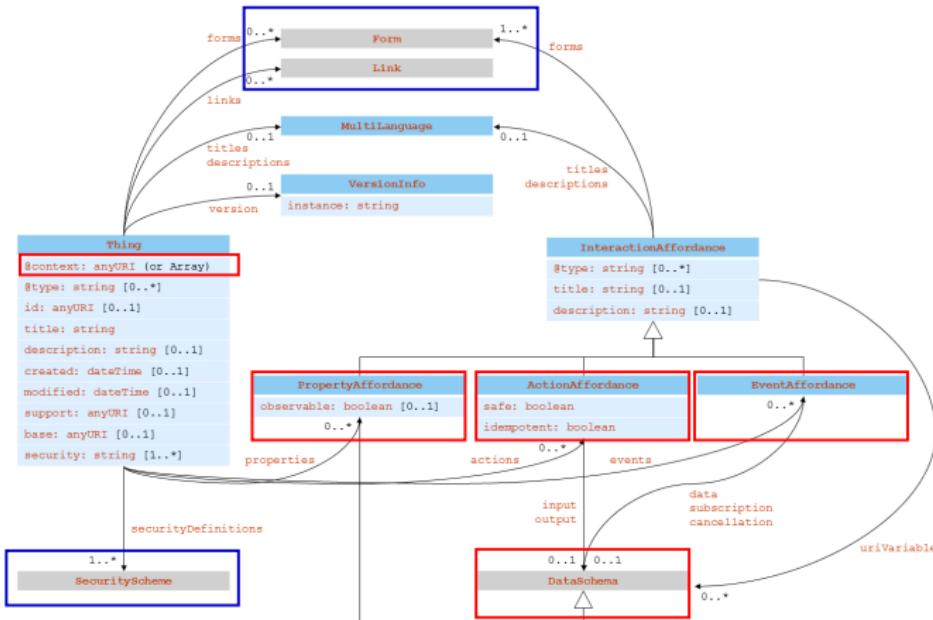
## Pattern 3

- WoT Clients talk directly with **ELW API** using **information from binding templates**

**Pattern 1, 2, 3:** Use the same **static parts of Thing Description**



## 2.1 WoT Thing Description (Overall)



**TD.Thing= DD.Device**, **TD.Properties= DD.Properties**, **TD.Actions= DD.Actions**, **TD.Events= DD.Events**, **TD.DataSchema= DD.Schema**

## 2.1.1 DD/MRA to TD Mapping (Overall)

The diagram illustrates the mapping between three JSON objects: MRA, DD, and TD.

**MRA:**

```
"0x0130": {  
    "validRelease": {  
        "from": "A",  
        "to": "latest"  
    },  
    "className": {  
        "ja": "家庭用エアコン",  
        "en": "Home air conditioner"  
    },  
    "shortName": "homeAirConditioner",  
    "elProperties": {...  
    }  
}
```

**DD:**

```
{  
    "deviceType" : "homeAirConditioner",  
    "eoj" : "0x0130",  
    "descriptions" : {  
        "ja" : "家庭用エアコン",  
        "en" : "Home air conditioner"  
    },  
    "properties" : {...  
    },  
    "actions" : {...  
    }  
}
```

**TD:**

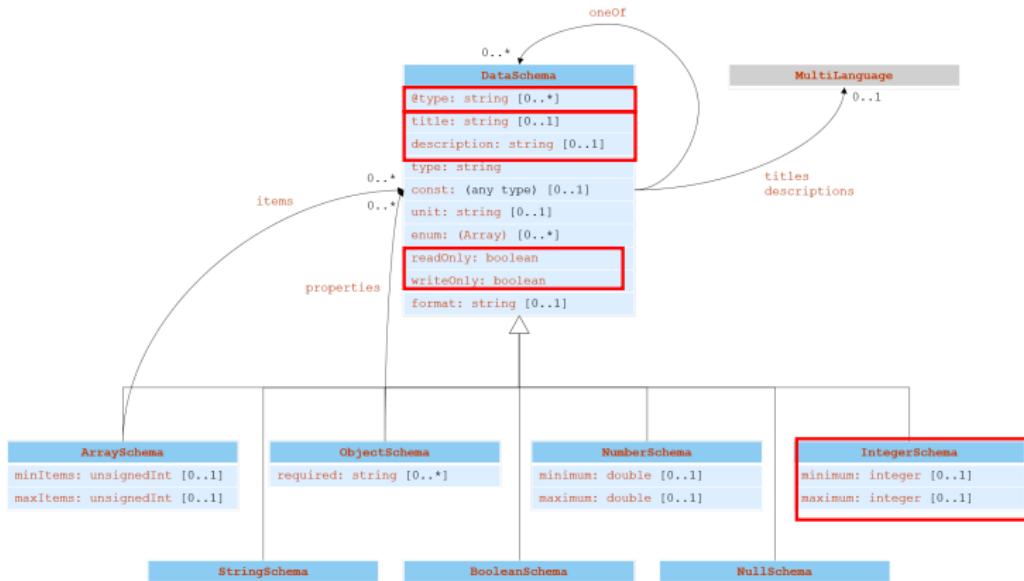
```
{  
    "@context" : [ "https://www.w3.org/2019/wot/td/v1",  
        "echonet" : "https://echonet.jp/"  
    ],  
    {  
        "@language" : "en"  
    },  
    "id" : "",  
    "echonet:ej" : "0x0130",  
    "title" : "homeAirConditioner",  
    "titles" : {  
        "en" : "homeAirConditioner",  
        "ja" : "家庭用エアコン"  
    },  
    "description" : "Home air conditioner",  
    "descriptions" : {  
        "en" : "Home air conditioner"  
        "ja" : "家庭用エアコン"  
    },  
    "properties" : {...  
    },  
    "actions" : {...  
    }  
}
```

Annotations highlight specific fields:

- Custom keyword:** Points to the prefix "echonet" in the TD "@context".
- Title = shortName**: Points to the "title" field in TD.
- Titles.en = shortName**: Points to the "titles.en" field in TD.
- Titles.ja = descriptions.ja (no SPACE)**: Points to the "titles.ja" field in TD.
- description = descriptions.en**: Points to the "description" field in TD.
- descriptions{} = descriptions {}**: Points to the "descriptions" field in TD.

**Customized keywords** = (1) define a prefix (e.g *echonet*) → (2)  
**prefix:propertyName** (e.g *echonet:ej*)

## 2.2 WoT Thing Description (Data Schema)



**TD.@Type**= Semantic label, **TD.title**, **TD.description**: in English

**TD.IntegerSchema** = **DD.Level** ?

→ Mapping from DD to TD is straightforward

## 2.2.1 DD/MRA to TD Mapping (Properties)

```
"humidity" : {  
    "epc" : "0xBA",  
    "descriptions" : {  
        "ja" : "室内相对湿度計測値",  
        "en" : "Measured value of room relative humidity"  
    },  
    "writable" : false,  
    "observable" : false,  
    "schema" : {  
        "oneOf" : [ {  
            "type" : "number",  
            "unit" : "%",  
            "minimum" : 0,  
            "maximum" : 100  
        }, {  
            "type" : "string",  
            "enum" : [ "unmeasurable" ],  
            "values" : [ ] { ... } ]  
        } ]  
    }  
},
```

DD

```
"humidity" : {  
    "title" : "humidity",  
    "titles" : {  
        "en" : "humidity",  
        "ja" : "室内相对湿度計測値"  
    },  
    "echonet:epc" : "0xBA",  
    "description" : "Measured value of room relative humidity"  
    "descriptions" : {  
        "en" : "Measured value of room relative humidity",  
        "ja" : "室内相对湿度計測値"  
    },  
    "readOnly" : true,  
    "writeOnly" : false,  
    "observable" : false,  
    "oneOf" : [ {  
        "type" : "number",  
        "unit" : "%",  
        "minimum" : 0,  
        "maximum" : 100  
    }, {  
        "type" : "string",  
        "enum" : [ "unmeasurable" ]  
    } ]  
},
```

TD

**setOnly** property → **DD.actions** → **TD.writeOnly** is **false** in all cases

**DD.values** → **TD.echonet:edt**, value pair ?

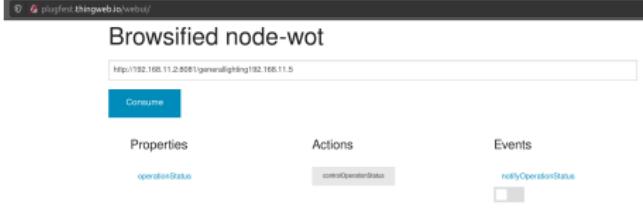
## 2.2.2 DD/MRA to TD Mapping (Actions/Events)



## 2.3 Summary

- **Device Description to Thing Description** is simply converted using the **rule-based** approach
- **MRA to DD and TD** conversion is added to the **eDataModelsGen** tool available at <https://github.com/Tan-Lab/eDataModelsGen>
- All **Thing Descriptions** in this slide are created from MRA automatically using the above tool
- Customized keywords such as **echonet:ej**, **echonet:epc**, **echonet:edt** are used in **Pattern 1** for **ECHONET Lite Frame** to **TD** conversions
- **TD.** forms, links, securitySchema are available at **run-time** → Depend on cooperation patterns

# 3.1 Expose Thing Description API

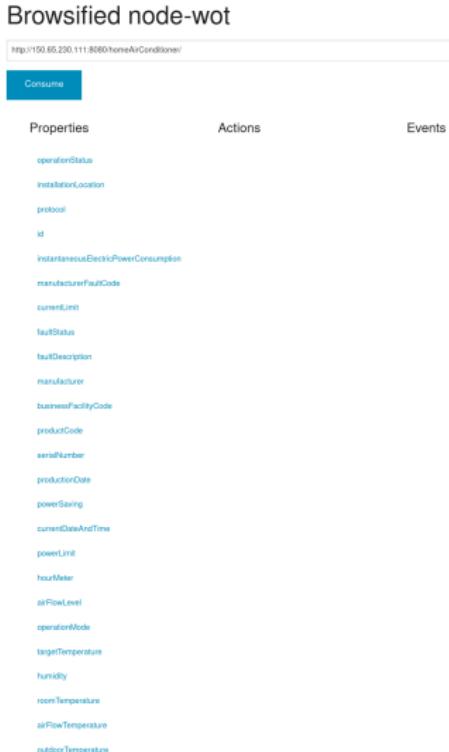


Browsified node-wot

http://192.168.11.2:8081/generalighting/192.168.11.5

Properties: operationStatus  
Actions: controlOperatorStatus  
Events: notifyOperationStatus

**Consume**



Browsified node-wot

http://192.168.11.2:8080/homeAirConditioner/

Properties: operationStatus, installationLocation, protocol, id, instantaneousElectricPowerConsumption, manufacturerFaultCode, currentUml, faultStatus, faultDescription, manufacturer, businessFacilityCode, productCode, serialNumber, productionDate, powerSaving, currentDateAndTime, powerLimit, hourMeter, airFlowLevel, operationMode, targetTemperature, humidity, roomTemperature, airFlowTemperature, outdoorTemperature

Actions: **Consume**

Events:

To allow things to be discoverable by WoT clients

- Display **Properties, Actions, Events**

## Library: node-wot

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

## Sample from

<http://www.thingweb.io/smart-coffee-machine.html>

### 3.2 Example: Expose a thing by API

```

1  const Servient = require('@node-wot/core').Servient;
2  const HttpServer = require('@node-wot/binding-http').HttpServer;
3  const CoapServer = require("@node-wot/binding-coap").CoapServer
4
5  const servient = new Servient();
6  servient.addServer(new HttpServer());
7  servient.addServer(new CoapServer());
8
9  servient.start().then((WoT) => {
10    WoT.produce({
11      title: "Light"+anode.addr,
12      properties: [
13        operationStatus: {
14          type: "boolean",
15          readOnly : false,
16          observable: true
17        },
18        actions:{...},
19        events: {...}
20      }
21    }).then((thing) => {
22      thing.writeProperty("operationStatus", getStatusFromRealDevice(device));
23      thing.setPropertyWriteHandler('operationStatus', setStatusOfDevice(device))
24    });
25  });
26
27  thing.expose().then(() => {
28    });
29  });
30  });
31  });
32  });
33  });
34  });
35  });
36  });
37  });
38  });
39  });
40  });
41  });
42  });
43  });
44  });
45  });
46  });
47  });
48  });
49  });
50  });
51  });
52  });
53  });
54  });
55  });
56  });
57  });
58  });
59  });
60  });
61  });

```

**Protocol Support**

- HTTP ✓
- HTTPS ✓
- CoAP ✓
- CoAPS ✓
- MQTT ✓
- Websocket ✎ (Server only)
- OPC-UA ✎ (Client only)
- NETCONF ✎ (Client only)
- Modbus ✎ (Client only)

**WoT ThingDescription Binding**

```

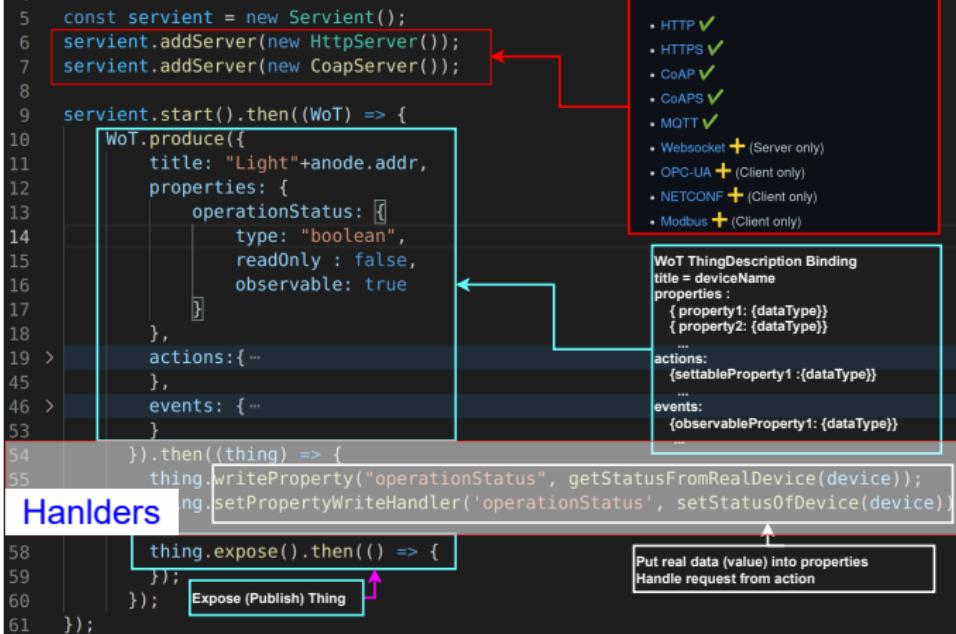
title: deviceName
properties:
  { property1: {dataType} }
  { property2: {dataType} }
...
actions:
  {settableProperty1 :(dataType)}
...
events:
  {observableProperty1: {dataType}}
...

```

**Hanlders**

Put real data (value) into properties  
Handle request from action

Expose (Publish) Thing



Exposing things of **Pattern 1**, **Pattern 2**, and **Pattern 3** are similar

### 3.3 Add corresponding handlers

To handle **read, write, invoke action, subscribe events** requests from WoT clients

- **not readOnly** → PropertyWriteHandler
- **not writeOnly** → PropertyReadHanlder
- observable or events → emitEvent
- actions → ActionHanlder

**Howto:**

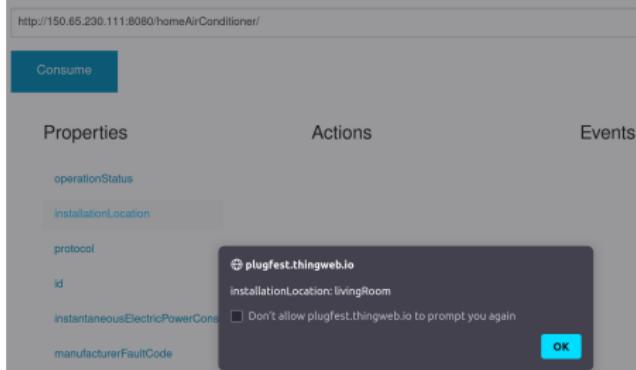
- **Pattern 1:** send EL frames
- **Pattern 2:** call web APIs
- **Pattern 3: No Need handler**

Browsified node-wot



The screenshot shows a browsified node-wot interface. At the top, there is a URL bar with the address `http://192.168.11.2:8081/generalLighting192.168.11.5`. Below the URL bar, there is a blue "Consume" button. To the right of the button, there are three sections: "Properties", "Actions", and "Events". The "Properties" section contains a single entry: "operationStatus". The "Actions" section contains a single button labeled "controlOperationStatus". A red box highlights this button. A red arrow points from the "controlOperationStatus" button to a red box containing the "controlOperationStatus" field. This field has the value "true" and a dropdown arrow. Below the "controlOperationStatus" field is a blue "Invoke" button.

Browsified node-wot



The screenshot shows a browsified node-wot interface. At the top, there is a URL bar with the address `http://150.65.230.111:8080/homeAirConditioner/`. Below the URL bar, there is a blue "Consume" button. To the right of the button, there are three sections: "Properties", "Actions", and "Events". The "Properties" section contains several entries: "operationStatus", "installationLocation", "protocol", "id", "instantaneousElectricPowerConsumption", and "manufacturerFaultCode". The "Actions" section is currently empty. A modal dialog box is open over the interface. The dialog box has a title "plugfest.thingweb.io" and a message "installationLocation: livingRoom". There is a checkbox below the message with the text "Don't allow plugfest.thingweb.io to prompt you again". At the bottom right of the dialog box is a blue "OK" button.

## 4.1 EL-WoT Proxy Building Blocks

### (1) Get ECHONET Lite devices

- **Pattern 1:** Broadcast *NodeFindingMessage* → `getInstance` list
  - RESULT: **EOJs**
- **Pattern 2, Pattern 3:** httpGET `/elapi/v1/devices`
  - RESULT: **deviceType**, **ID**, etc.

### (2) Generate Thing Description

- **Pattern 1:** Compare `this.EOJ` and `TD.echonet:ej` → **xxxTD.json**
- **Pattern 2, Pattern 3:** This.`deviceType` → **xxxTD.json**

→ **Not supported properties are included** → **Solution**

- **Pattern 1:** Get **GET**, **SET**, **INF** property maps → `Gettable`, `Settable`, `Observable` properties → Compare supported properties and defined property → **xxxTD.json**
- **Pattern 2, Pattern 3:** httpGET `/elapi/v1/devices/id/properties` → Compare supported properties and defined property → **xxxTD.json**

## 4.1.2 EL-WoT Proxy Building Blocks (cont.)

### (3) Expose TD and create **Handlers**

**handlers for Pattern 3 are supported by the ELW API**

**Manually** → Simple, less exception, time-consuming

**Automatically** → Complex, more exceptions, Fast, **Adaptable**

- **Pattern 1:** Complex dataTypes (objects, array) to Hex codes problem

### Pattern 2 automatic generated handlers

**FOR** *property* in *TD.properties*

**IF** *this.property.isNOTWriteOnly* → **PropertyReadHanlder(property)**

**httpGET** elapi/v1/devices/{id}/properties/{thisProperty} → res

**IF** *this.property.isNOTReadOnly* → **PPWriteHanlder(property, value)**

**httpPUT** elapi/v1/devices/{id}/properties/{thisProperty}, **body:**  
*thisProperty:value* → res

### Pattern 2: Thing Description

→ forms

- **href**: address of Gateway/Proxy to handle requests from WoT Client
- **op**: *readProperty* = **httpGET** (http binding)
- **op**: *writeProperty* = **httpPUT** (http binding)

httpGET, httpPUT requests are handled by **EL-WoT Gateway/Proxy**

→ Pattern 3: href → ELW API server

```

▼ object {11}
  ► @context [3]
    id : FE00007702905D99427FDB571E1EBDA7D1
    title : FE00007702905D99427FDB571E1EBDA7D1
  ► titles {2}
    description : General lighting
  ► descriptions {2}
  ▼ properties {21}
    ▼ operationStatus {10}
      title : operationStatus
    ► titles {2}
      echonet:epc : 0x80
      description : Operation status
    ► descriptions {2}
      readOnly : false
      writeOnly : false
      observable : true
      type : boolean
    ▼ forms [2]
      ▼ 0 {3}
        href : http://192.168.11.2:8080
          /FE00007702905D99427FDB571E1EBDA7D1/properties
          /operationStatus
        contentType : application/json
      ▼ op [2]
        0 : readproperty
        1 : writeproperty
  
```

## 4.3 Demonstration with 実験クラウド (Pattern 3)

forms = ELW API + pppath

- propertyPaths are compatible
- op read/write are compatible
- actions are compatibile
- ✗ writeProperty **body**

**WoT:** true/false

**ELW API:** operationStatus:  
true/false

- ✗ getAllProperty **path**

**WoT:** id/all/properties

**ELW API:** id/properties

```
▼ object {11}
  ▶ @context [3]
    id : FE0000770288B93A16E1A5C0C59A16AA32
    title : FE0000770288B93A16E1A5C0C59A16AA32
  ▶ titles {2}
    description : Low-voltage smart electric energy meter
  ▶ descriptions {2}
  ▼ properties {27}
    ▼ operationStatus {10}
      title : operationStatus
    ▶ titles {2}
      echonet:epc : 0x80
      description : Operation status
    ▶ descriptions {2}
      readOnly : false
      writeOnly : false
      observable : true
      type : boolean
    ▼ forms [2]
      ▼ 0 {3}
        href : https://webapiechonet.com/elapi/v1/devices
               /fe0000770288b93a16e1a5c0c59a16aa32/properties
               /operationStatus
        contentType : application/json
      ▶ op [2]
        0 : readproperty
        1 : writeproperty
```

## 4.4 Solutions for Pattern 3

- WoT Generated **href** = my **href** → not yet supported by current **node-wot** library → Pull Request?

```
// set contentType (extend with more?)  
public static updatePropertyFormWithTemplate(form: TD.Form, tdTemplate: WoT.ThingDescription, propertyName: string) {  
    if (form && tdTemplate && tdTemplate.properties && tdTemplate.properties[propertyName] && tdTemplate.properties[propertyName].forms) {  
        for (let formTemplate of tdTemplate.properties[propertyName].forms) {  
            // 1. Try to find match with correct href scheme  
            if (formTemplate.href) {  
                // TODO match for example http only? form.href = formTemplate.href  
            }  
  
            // 2. Use any form  
            if (formTemplate.contentType) {  
                form.contentType = formTemplate.contentType;  
                return; // abort loop  
            }  
        }  
    }  
}
```

- ELW API **httpPUT body** → **only data** → remove **propertyName** from **body**

## 5 Summary

- WoT Binding Templates for ECHONET Lite = EL-WoT **Thing Description + Protocol Bindings**
- Thing Description Schemes are generated from Device Descriptions
  - Auto conversion tool is available (**eDataModelsGen**)
- Three Patterns for EL-WoT Cooperation are implemented
- Mechanism to create WoT-ECHONET Lite Gateway
  - Init WoT Servient
  - Init ECHONET Lite interface → When detect a device → create Thing Description for the device + getValue from device → expose device as WoT consumable resources
  - **Pattern 1 and 2** → Create operation handlers for operations
  - **Pattern 3** → Redirect to corresponding ELW API
- Need some adjustments of ELW API to implement **Pattern 3**

### **EL-WoT Proxy, EL-WoT Binding Proxy:**

<https://github.com/Cupham/eWoTPProxy>

### **MRA-DD-TD Conversion**

**tool:** <https://github.com/Tan-Lab/eDataModelsGen>