ECHONET Consortium Liaison Activity

ECHONET

October 14, 2021

Agenda

- Our liaison activities since the last WoT virtual meeting
- PlugFest result
- Mapping Device Description of ECHONET Lite Web API to Thing Description in the PlugFest
- HTTP message translation done by the intermediary in the PlugFest

Our liaison activities since the last WoT virtual meeting & PlugFest result

Use Case Proposal

Title: Leaving and Coming Home

Target Users:

- device user
- service provider
  (Home Management Service Provider)
- device manufacturer

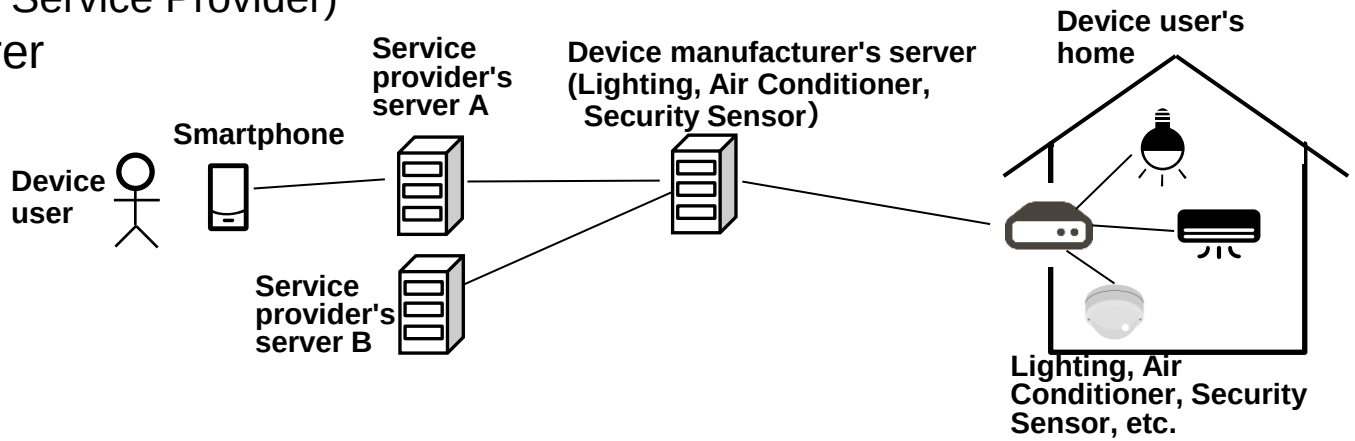


Figure: example system structure

Motivation:

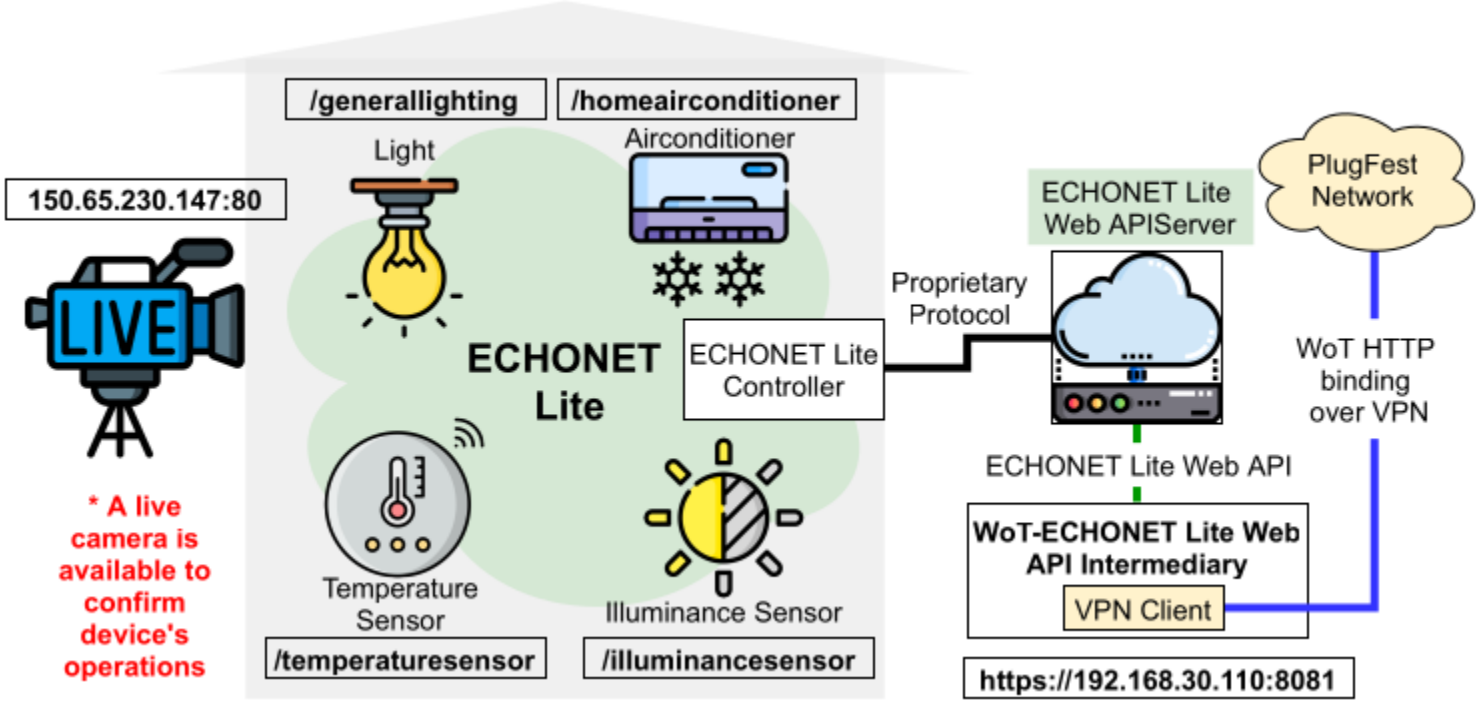
The purpose of this use case is to improve the usability of home appliances for device users by allowing device users to configure the operation modes of all devices at home without configuring those devices one by one when they leave and come home.

Status: Merged into the editor's draft of Use Case document.

PlugFest

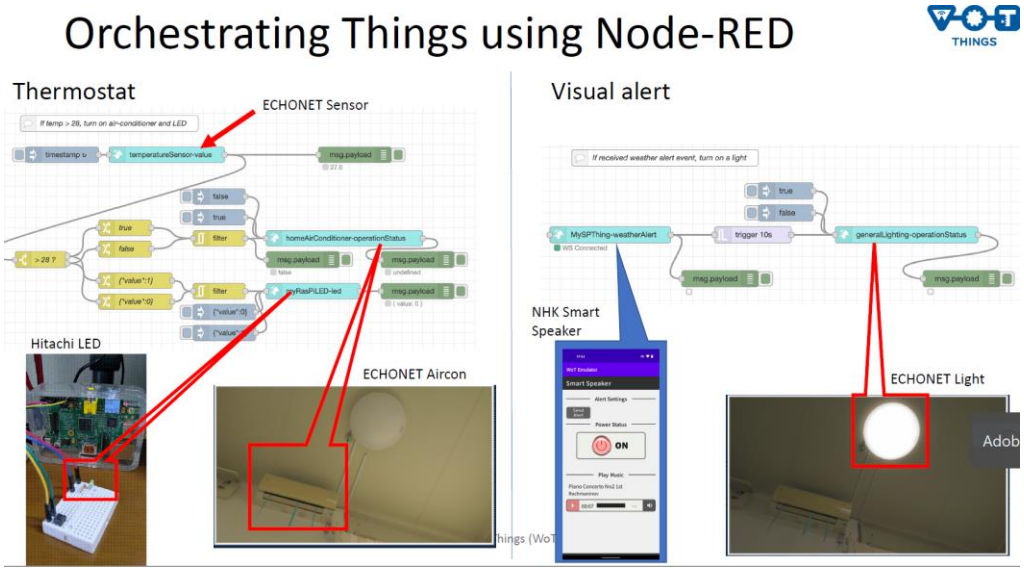
- We provided four ECHONET Lite devices via Intermediary (translation proxy).
  - LED light, Air conditioner, Illuminance sensor, Temperature sensor
  - Read and write on one property and readallproperties were implemented.
- We also provided a live camera with which the operation of the light and the air conditioner could be monitored.

Acknowledgement: Our system was developed and operated by Van Cu PHAM-san, Japan Advanced Institute of Science and Technology and ECHONET Consortium.

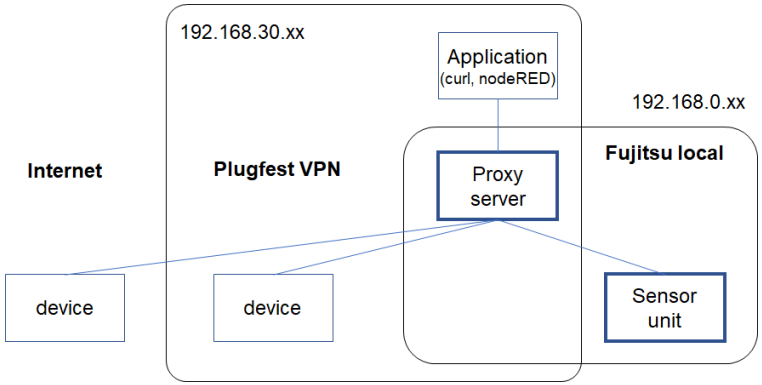


PlugFest Result (1)

- Hitachi connected to ECHONET Lite devices successfully with Node-RED. Thank you, Toumura-san !
- Fujitsu connected to ECHONET Lite devices successfully with Fujitsu local Proxy. Thank you, Matsukura-san !
- TUM connected to ECHONET Lite devices successfully with TUM's tool. Thank you, Marcus Schmidt-san !



Hitachi's Node-RED
(source Hitachi's material)



Fujitsu's local Proxy
(source Fujitsu's material)

PlugFest Result (2)

- Toumura-san raised the following point.
  - Direct access to ECHONET Lite web API
    - In this plugfest, we can access ECHONET device via gateway (WoT-ECHONET Lite Web API Intermediary).
      - Gateway removes an encapsulating object from a response payload.
    - By adding the encapsulating object schema when converting Device Description of ECHONET Lite Web API to TD, it is possible to access ECHONET Lite Web API directly … ?
- We think this proposal can be an option.
  However, we are not sure if it is acceptable to request an WoT client to interpret a value { "rgb": { "r" : 20, "g" : 255, "b" : 0 } } as a property value of "rgb" property for example, because JSON key "rgb" in the value is actually the property name and the real property value to be read is { "r" : 20, "g" : 255, "b" : 0 }.
  An WoT client application has to implement a special processing for ECHONET Web API.

Mapping Device Description of ECHONET Lite Web API to Thing Description in the PlugFest

Example: General Lighting (1)

ECHONET Lite Device Description (DD)

```
{
  "deviceType": "generalLighting",
  "eoj": "0x0290",
  "descriptions": {
  "ja": "一般照明",
  "en": "General lighting"
  },
  "properties": {
    "brightness": {
    "epc": "0xB0",
    "descriptions": {
      "ja": "照度レベル設定",
      "en": "Illuminance level"
    },
    "writable": true,
    "observable": false,
    "schema": {    <--removed
      "type": "number",
      "unit": "%",
      "minimum": 0,
      "maximum": 100
    }
  }
}
```

To next page

Thing Description

```
{
 "@context": [
  "https://www.w3.org/2019/wot/td/v1",
  {    Added for ECHONET Lite vocabulary
    "echonet": "https://echonet.jp/"
  },
  {
    "@language": "en"
  }
 ],
 "id": "echonet:generalLighting:192168117029001",
 "title": "generalLighting",
 "description": "General lighting",
 "properties": {
  "brightness": {
   "echonet:epc": "0xB0",
   "readOnly": false,
   "writeOnly": false,
   "observable": false,
   "type": "number",
   "unit": "%",
   "minimum": 0,
   "maximum": 100,
```

Generated for TD

Example: General Lighting (2)

ECHONET Lite Device Description (DD)

Thing Description

```
      "forms": [
       {            Generated for TD (Binding Template)
         "href":
"http://192.168.30.110:8081/generallighting/prop
erties/brightness",
         "contentType": "application/json",
         "op": [
           "readproperty",
           "writeproperty"
         ]
       }
      ],
      "title": "brightness",
      "description": "Illuminance level"
    },
```

From previous page

Example: General Lighting (3)

ECHONET Lite Device Description (DD)

Thing Description

```
"forms": [
   {                    Generated for TD (Binding Template)
     "href":
"http://192.168.30.110:8081/generallighting/all/properties",
     "contentType": "application/json",
     "op": [
       "readallproperties"
     ]
   }
 ],
 "securityDefinitions": {
   "nosec_sc": {
     "scheme": "nosec"
   }      Generated for TD (Security scheme)
 }
}
```

Summary of mapping from DD to TD

- A definition in @context for ECHONET Lite Web API vocabulary is added.
- "id" in TD is generated automatically from "deviceType" and "deviceId" in ECHONET Lite Web API.
- "title" in TD is taken from "deviceType" or property name in DD.
- "description" in TD is taken from "descriptions.en" in DD.
- "readable" and "writable" in DD are converted to "readOnly" and "writeOnly" in TD.
- "schema" in DD is removed.
- "forms" for read, write and readallproperty operations are added based on the URI supported by the intermediary.
- Security scheme is generated according to the authentication scheme implemented by the intermediary (in the PlugeFest, no security scheme).

HTTP message translation done by the intermediary in the PlugFest

Note: Translation of HTTP Header is mostly omitted in this presentation.

Read One Property

W3C WoT HTTP | **ECHONET Lite Web API HTTP**

- Read request

GET /light/properties/brightness HTTP/1.1

- Translate target path (if necessary)

GET
/elapi/v1/devices/012345/properties/brightness
HTTP/1.1

- Read response

HTTP/1.1 200 OK
{ "brightness" : 100 }

- Translate HTTP body to the value of a property

HTTP/1.1 200 OK
100

* The source code of the intermediary that implements the translation in slide 14 & 15 is available at https://github.com/Tan-Lab/WoTPlugfest/blob/main/src/baseProxy.ts .
If you want to use the source code, please contact Van Cu PHAM-san.

Write One Property

W3C WoT HTTP

ECHONET Lite Web API HTTP

- Write request

PUT /light/properties/brightness HTTP/1.1
80

- Translate target path (if necessary)
- Translate HTTP body to property : value pair

PUT
/elapi/v1/devices/012345/properties/brightness
HTTP/1.1
{ "brightness" : 80 }

- Write response

HTTP/1.1 200 OK
{ "brightness" : 80 }

- Empty HTTT body

HTTP/1.1 200 OK

Note: In ECHONET Lite Web API, the property value returned by a write response is the updated value of the property, which may be different from the written value due to some constraint of the target device.

Summary of HTTP Message Translation

- A Target path of Read and Write request is translated if necessary.
- ECHONET Lite Web API read response HTTP message body ({ property : value } pair) is changed to value in WoT HTTP read response message.
- WoT write request HTTP message body (value) is changed to { property : value } pair in ECHONET Lite Web API write request HTTP message.
- ECHONET Lite Web API write response HTTP message body ({ property : value } pair) is changed to empty string in WoT HTTP write response message.

Thank you for your attention.

Any questions?