Introduction to ECHONET Lite Web API

November 29, 2024

Agenda

- Overview of ECHONET Lite Web API
- Summary of ECHONET Lite Web API

Overview of ECHONET Lite Web API Guideline

Structure of ECHONET Lite Web API Guideline

- Guideline is comprised of API Specification part and Device Specifications part.
- API Specification part describes the guideline for Web API that enables a client to access ECHONET Lite devices via a server.

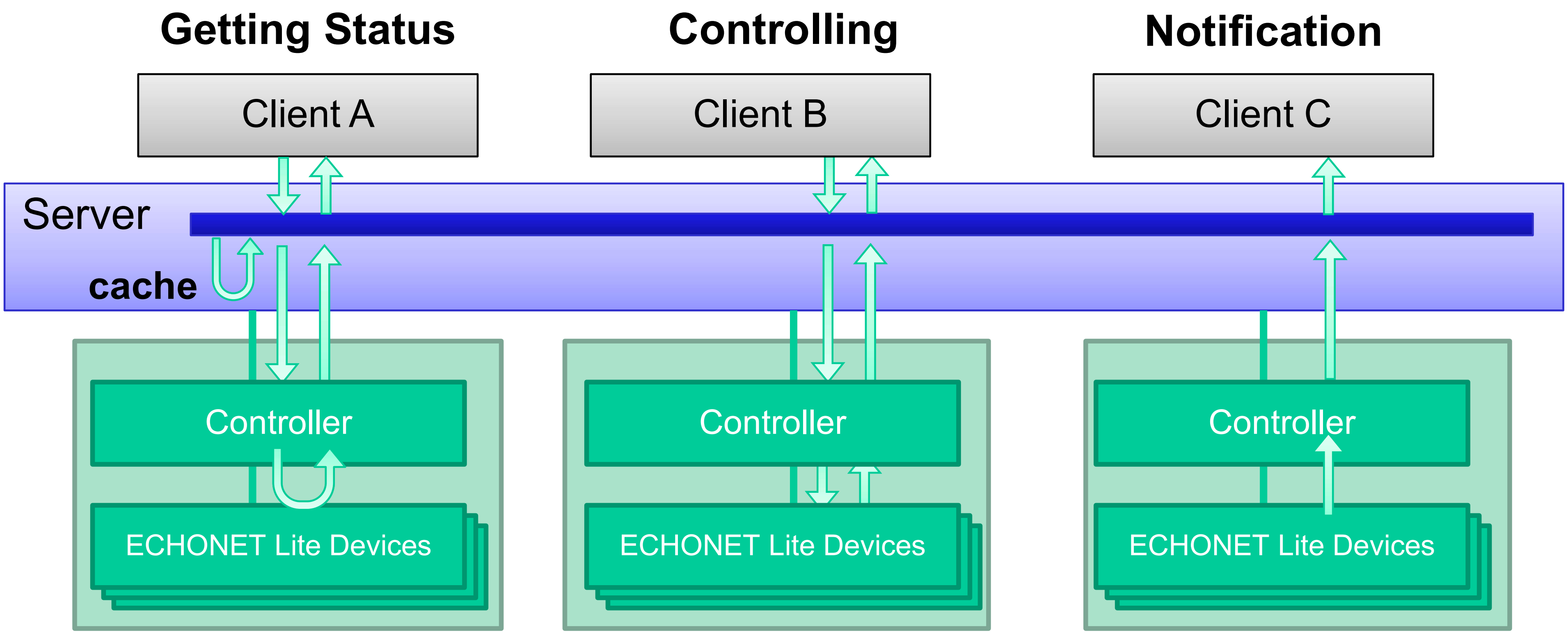
  EL Web API can be used to access "Things" other than ECHONET Lite devices.
- Device Specification part describes device models of some ECHONET Lite devices.

  Device model is serialized with JSON.

Examples of devices specified in Device Specification part	
Home Air conditioner	Low-Voltage Smart Electric Energy Meters
Heat Pump Water Heaters	High-Voltage Smart Electric Energy Meters
Instantaneous Water Heaters	Lighting / Extended Lighting System
Household Solar Power Generations	EV Chargers
Fuel Cell	EV Chargers/Dischargers
Storage Battery	Controller

Functions : Read Status / Control / Notification

- Device is modeled with property, action and event as in WoT.
- Read status of a target device (including cache data)
- Control a target device
  For ECHONET Lite devices, mainly by setting a property value.
  For logical objects, by invoking an action and by setting a property value.
- Notify an event from a target device to registered clients
  Currently, only notification of a change of a property value, INF, is supported.

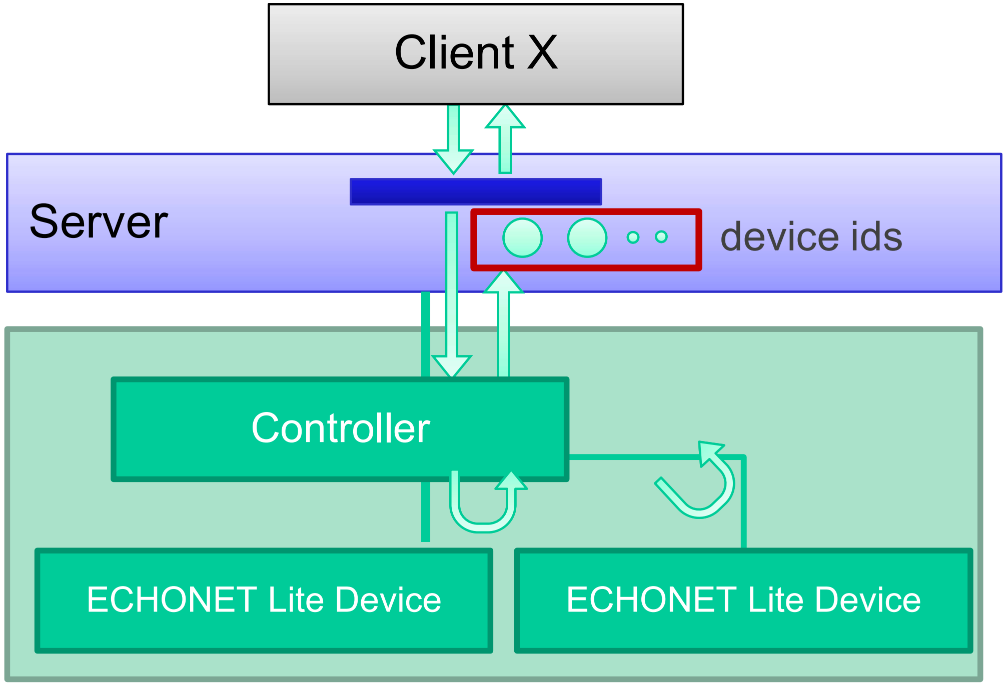


Functions : Get a List of Devices / Get Device Description

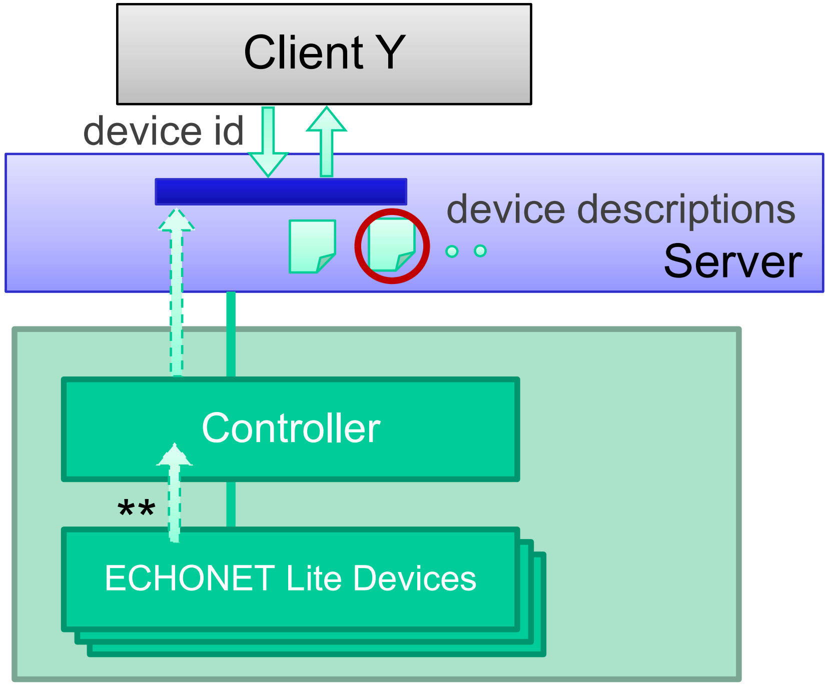
Discovery function is partially supported.

- Get a list of devices (all devices or some specific devices) on server
  - Can query the specified type devices: GET /XX/devices?type=homeAirConditioner
- Get the Device Description* of a device

Getting List of Devices



Getting a Device Description



** Controller can identify what properties are implemented by a device with ECHONET Lite

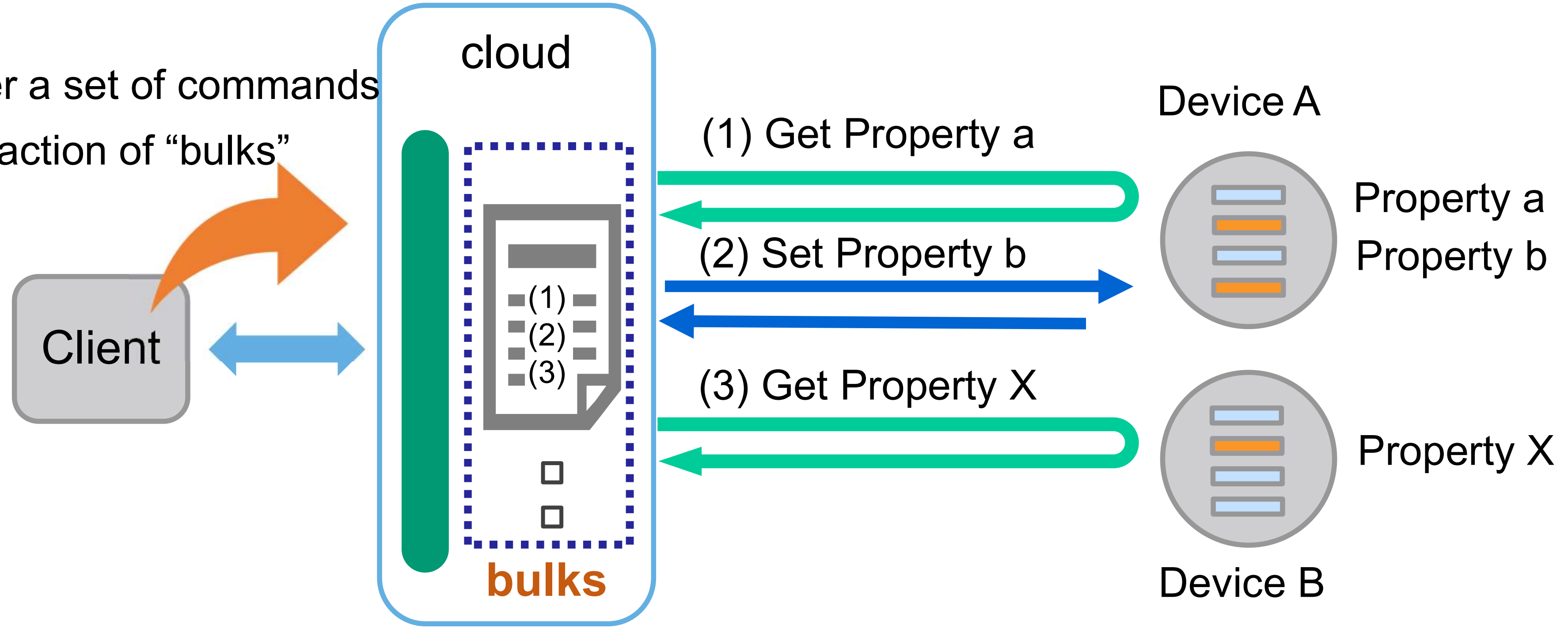
* Device Description is a meta data of a device (similar to TD in WoT)

Functions : Execute multiple commands with one API call

Create a "bulks" object that represents a set of commands and invoke action of "bulks" object.

- Client can issue multiple commands to multiple devices with one API call
- Client can specify whether commands are executed in parallel or sequentially.
- Client can check processing status of commands (useful when it takes time to execute).

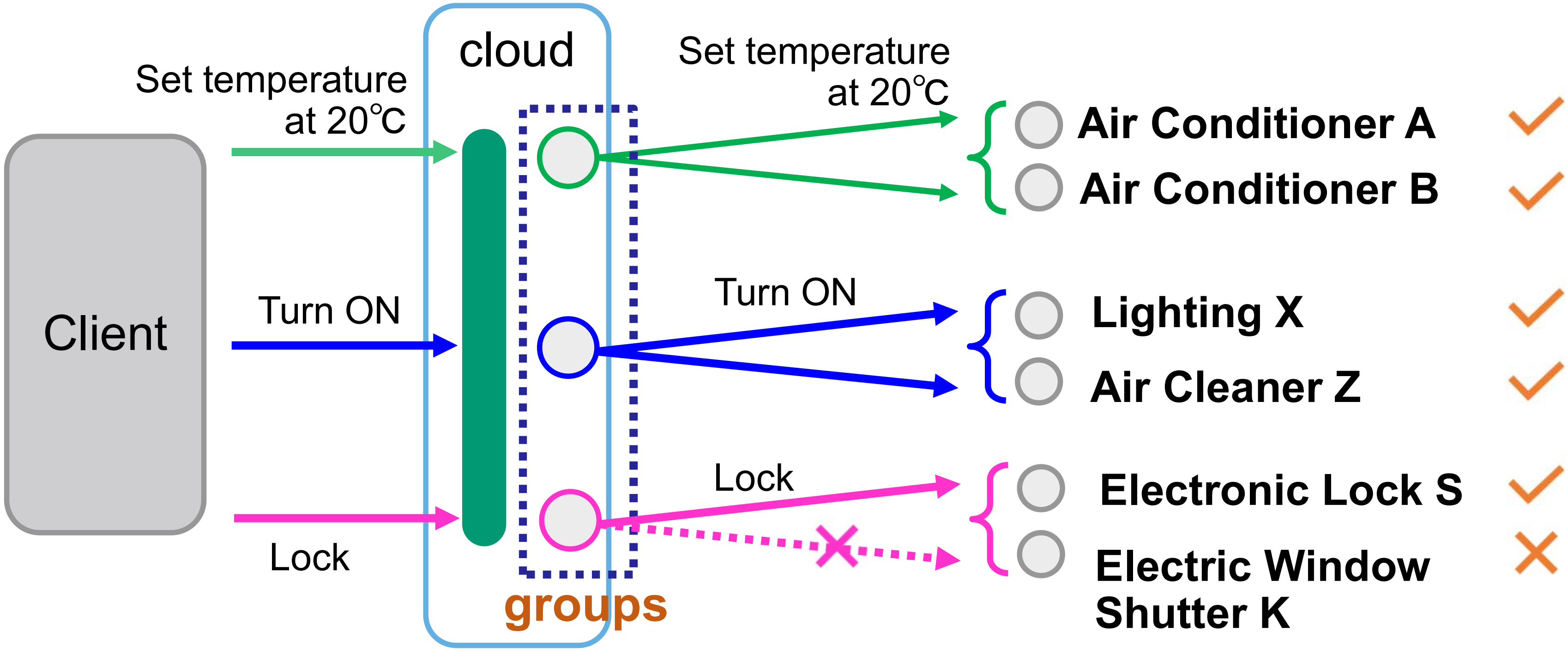
- Register a set of commands
- Invoke action of “bulks”



Functions : Define a group of devices

Create a "groups" object that represents a set of devices and enable a client to issue a command to the set of devices with one API call.

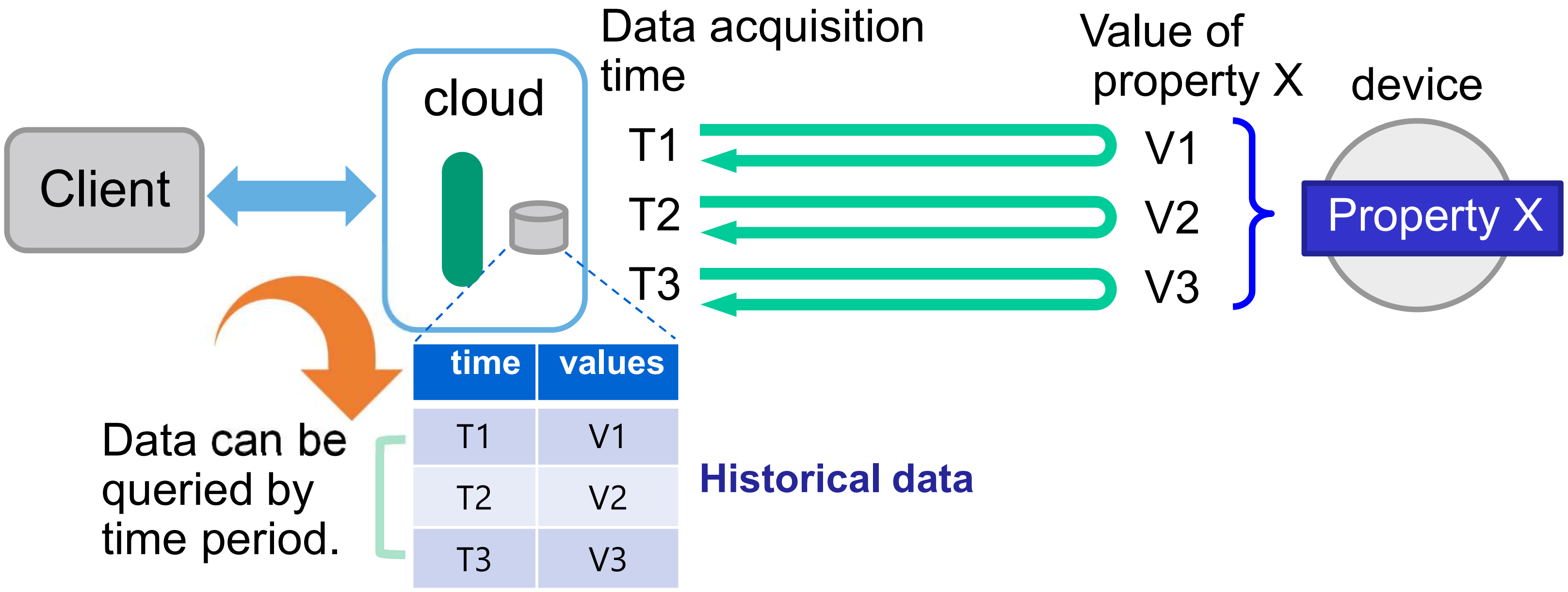
- ➢ Client can issue a command to multiple devices with one API call.
- ➢ Only devices that implement the command execute the command.



Functions : Query historical data of property, action or event

Server can implement "histories" objects to provide clients with access to historical data of device properties, action executions and events.

- ➢ Client can retrieve historical data by invoking actions of a "histories" object.
- ➢ Currently, a client cannot request a server to record historical data. Namely, only historical data that are recorded by the server voluntarily can be accessed.
- ➢ Client can query historical data by specifying time period, device and property / action / event.



time	values
T1	V1
T2	V2
T3	V3

Functions : Additional objects

- Two additional objects (resHistories and groupHistories) are defined for historical data access.
  - (original) histories: access history of one property in one device.
  - resHistories: access history of multiple properties in one device.
  - groupHistories: access histories of one property in multiple devices. (useful in cases such as obtaining consumed electric energy value from multiple devices)
- Object for demand response control of electric power and object for personal health data access are defined.

Device Description - Example

Example – General Lighting

Property Resource Name	Access Method	Data Type	EPC (EL)	Property Name in ECHONET Lite
brightness	GET, PUT	number	0x80	Illuminance level
operationMode	GET, PUT	enum	0xB6	Lighting mode setting
rgb	GET, PUT	object	0xC0	RGB setting for color lighting

```
{
    "deviceType": "generalLighting",
    "eoj": "0x0290",
    "description": {"ja": "一般照明","en": "General Lighting"},
    "properties": [
        {   "name": "brightness","epc": "0xB0",
            "description": { "ja": "照度レベル設定","en": "Illuminance level"},
            "writable": true,"observable": false,
            "schema": {"type": "number","unit": "%","minimum": 0,"maximum": 100} },
        {   "name": "operationMode","epc": "0xB6",
            "description": {"ja": "点灯モード設定","en": "Lighting mode setting"},
            "writable": true,"observable": true,
            "schema":  {"type": “string", “enum": ["auto", "normal", "night“, "color“],  "values": [
                    {"value": "auto","ja": "自動","en": "Auto Lighting","edt": "0x41"},
                    {"value": "normal","ja": "通常灯","en": "Normal Lighting","edt": "0x42"},
                    {"value": "night","ja": "常夜灯","en": "Night Lighting","edt": "0x43"},
                    {"value": "color","ja": "カラー灯","en": "Color Lighting","edt": "0x45"}
                ]}
        }, <the following is omitted>
```

Summary of ECHONET Lite Web API

Summary of ECHONET Lite Web API

- **EHONET Lite Web API is developed based on the operations and device objects of ECHONET Lite.**
  - Device Description, which is similar to TD, is the metadata of "Thing".
  - Get (HTTP GET), Set (HTTP PUT) and INF (observable) operations for a property are the basic operations of EHONET Lite Web API.
    Action (HTTP POST) is also defined. Event is for future use.
  - ECHONET Lite device is basically controlled by setting its property in Web API.
  - Logical object such as "bulks" and "groups" is controlled by invoking action or by setting its property in Web API.

- **Differences from the current specification of WoT**
  - Syntax of Device Description is a little different from the syntax of TD.
    e.g. JSON instead of JSON-LD. Thus, no ontology support at present.
  - No support of binding template concept for simplification.
    - ✓ Path used to access a property is defined in ECHONET Lite Web API guideline.
    - ✓ Only HTTPS protocol is supported.
    - ✓ Device Description does not describe authentication / authorization method, which is treated as an implementation matter.
  - No scripting specification.
  - Discovery function is partially supported.