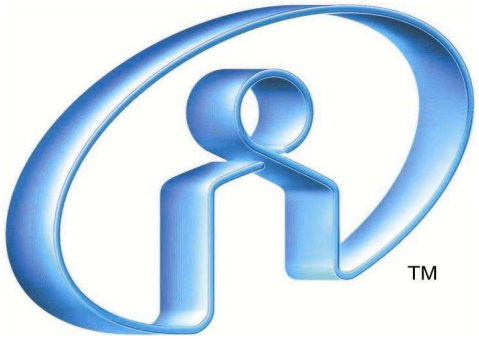
Introduction to ECHONET Lite Web API (EL Web API)



March 15, 2021

Agenda

- Brief Introduction to ECEHONET Lite, the basis of ECHONET Lite Web API
- Overview of ECHONET Lite Web API
- Summary of ECHONET Lite Web API

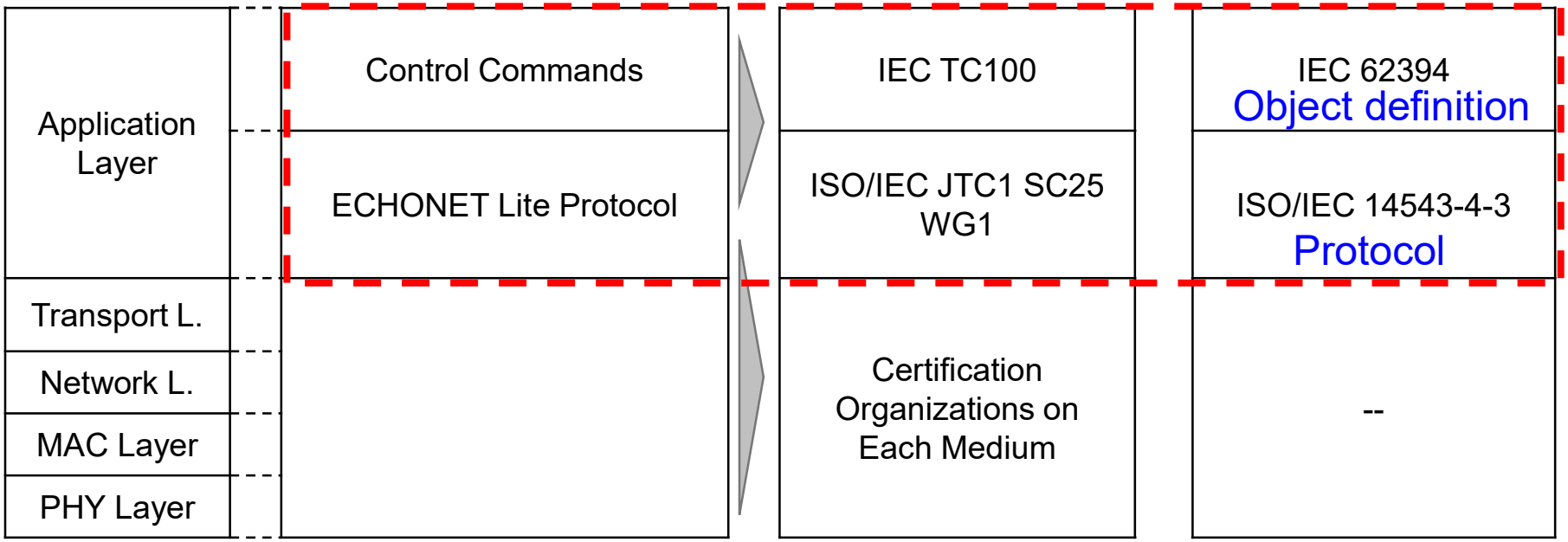
Brief Introduction to ECEHONET Lite, the basis of ECHONET Lite Web API

ECHONET Lite, International Standard

Object definition of “Thing”

||

- Detailed control commands for over a hundred devices are specified by ECHONET Consortium.
- Target devices are conventional home appliances, equipment and sensors.
- “ECHONET Lite” specifies the communication protocol for those commands.



Control commands for many kinds of devices

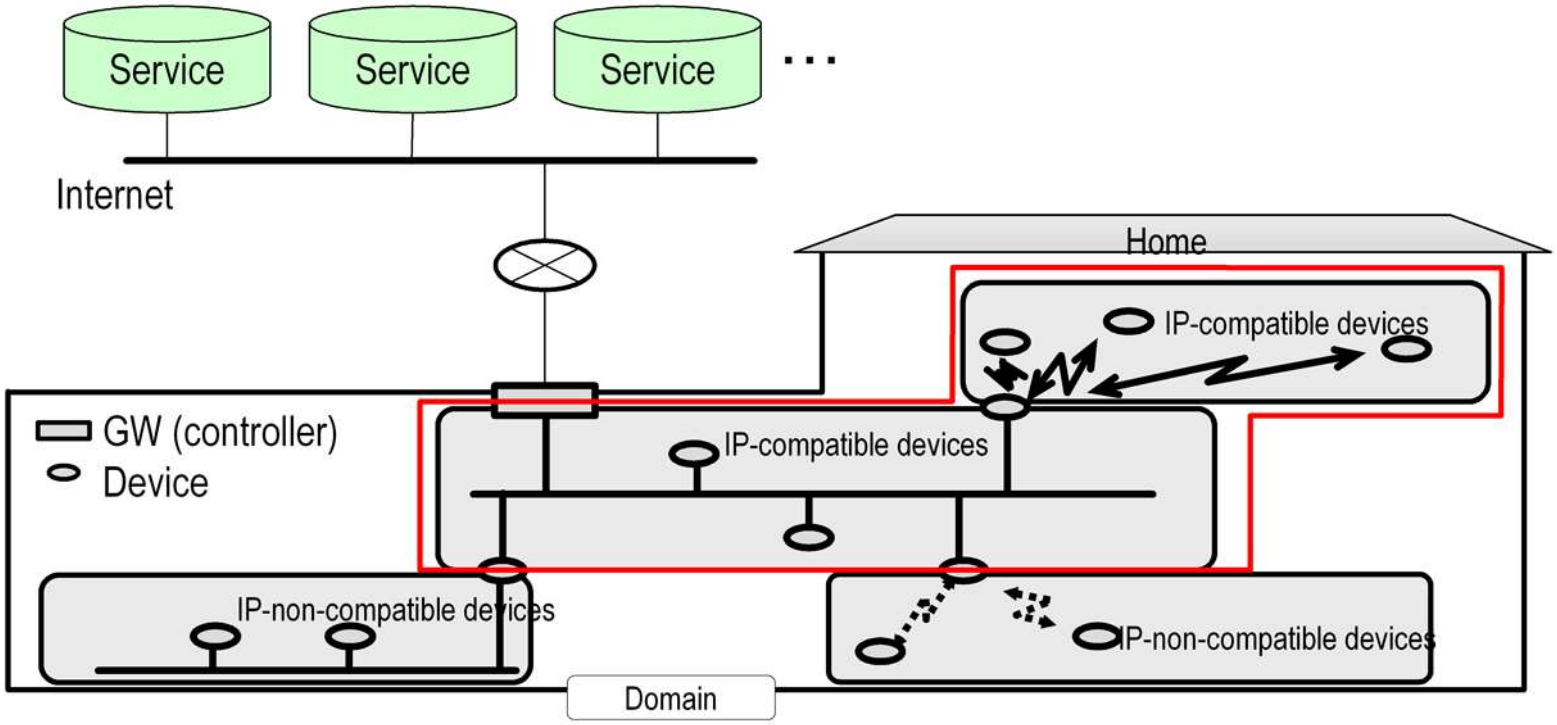
- Seven class groups of devices are defined.

Sensor Related	Security Sensor, Human Detect Sensor, Temperature Sensor, CO2 Sensor, Electricity Sensor, etc.
HVAC Related	Air Conditioner, Fan, Ventilator, Air Cleaner, Carpet, Fan Heater, etc.
Facility Related	Storage battery, Fuel cell, Photovoltaic system, electric water heater, electric vehicle charger / discharger, each smart meter, lighting, etc.
Cooking Related	Refrigerator, Microwave, Washing Machine, Rice Cooker, etc.
Health Related	Weighing scales, Clinical thermometer, Blood Pressure Meter, etc.
Controller Related	Controller, switch, etc.
Audiovisual Related	Television, display, etc.

HVAC: Heating, Ventilation, and Air Conditioning

System architecture of ECHONET

- System architecture of ECHONET Lite is comprised of controller, device and local network.
- Controller may communicate with cloud-based services through the Internet.



Outline of Control Commands

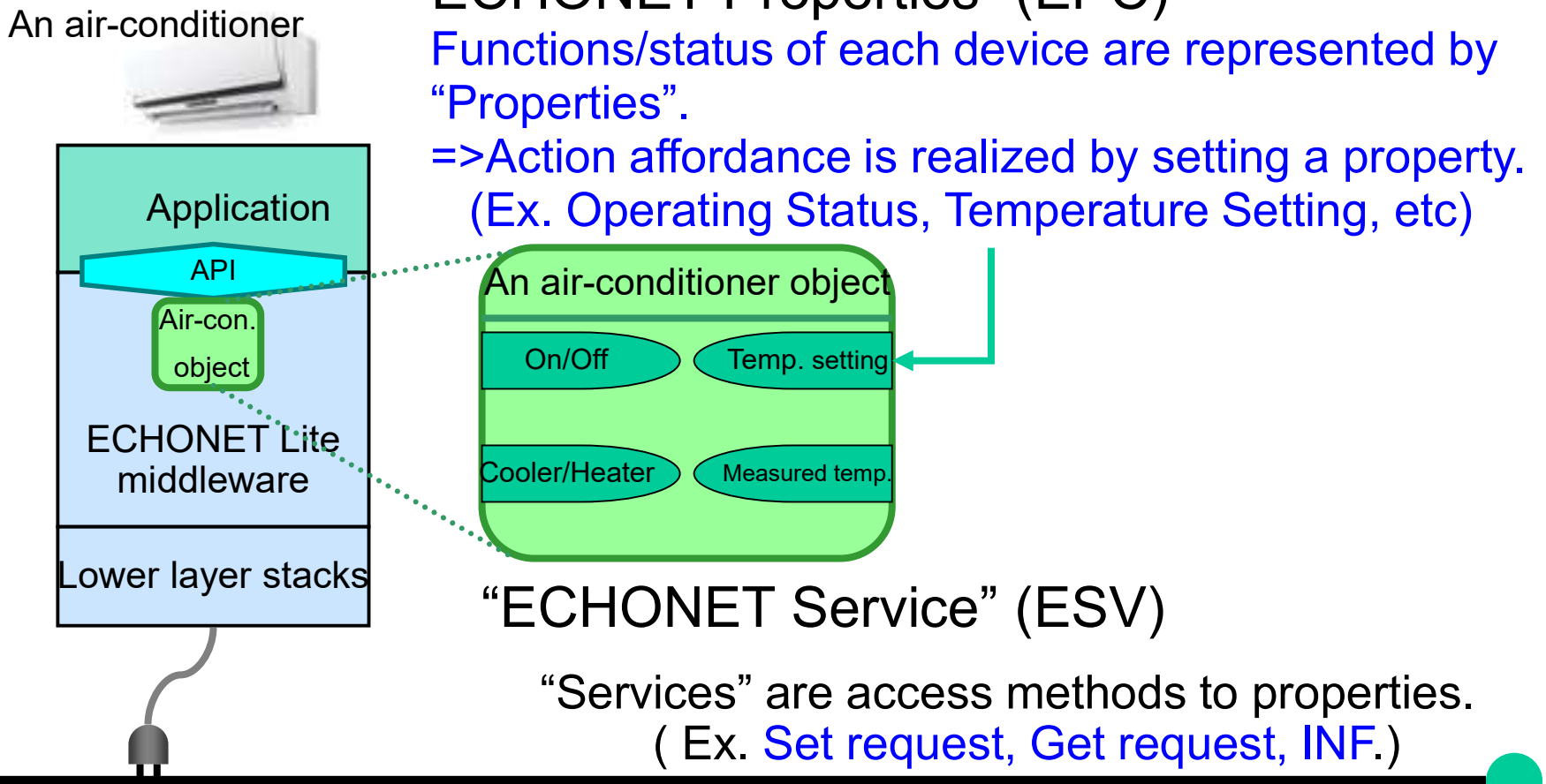
=Object definition of “Thing”

“ECHONET Objects” (EOJ)

Devices/appliances are defined as “Classes”.
(Ex. Air-Conditioner Class, Fire-Sensor Class, etc.)

“ECHONET Properties” (EPC)

Functions/status of each device are represented by “Properties”.
=>Action affordance is realized by setting a property.
(Ex. Operating Status, Temperature Setting, etc)



“ECHONET Service” (ESV)

“Services” are access methods to properties.
(Ex. Set request, Get request, INF.)

* INF is a notification of change of a property value.
(similar to “observable” in WoT)

Overview of ECHONET Lite Web API Guideline

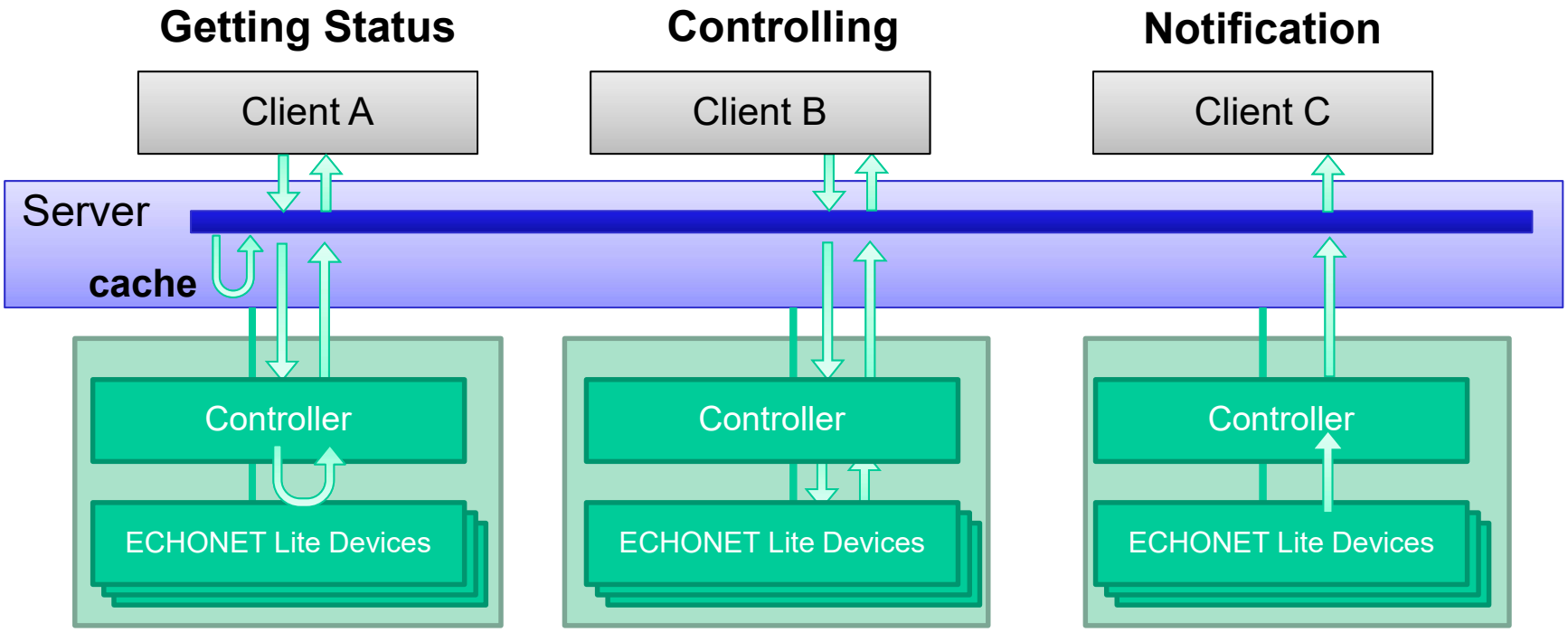
Structure of ECHONET Lite Web API Guideline

- 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.

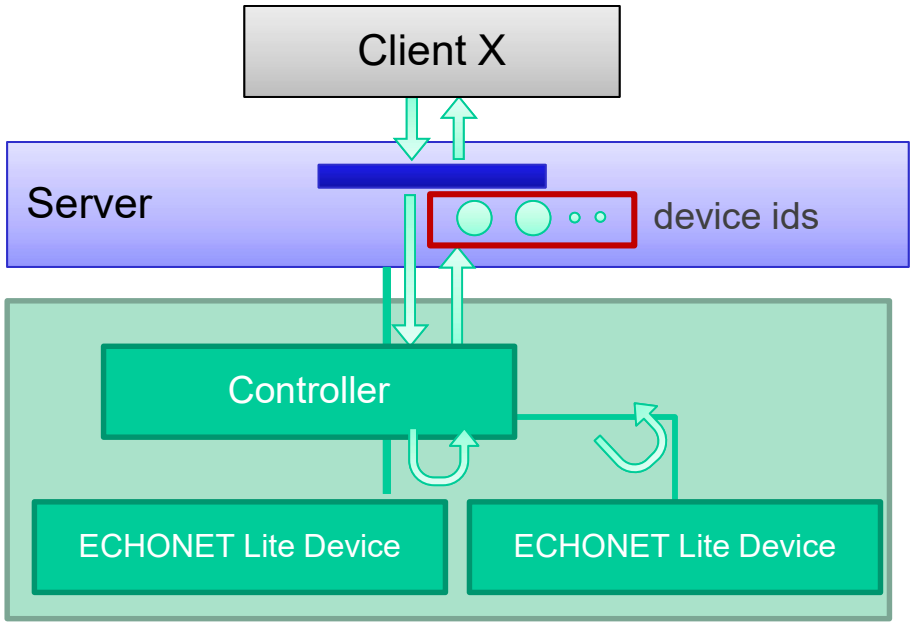


Functions : Get a List of Devices / Get Device Description

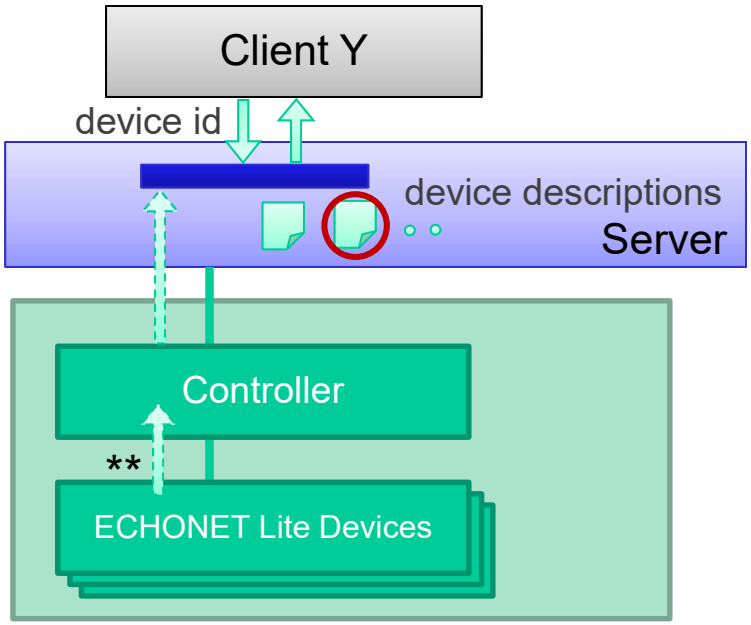
Discovery function is partially supported.

- 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

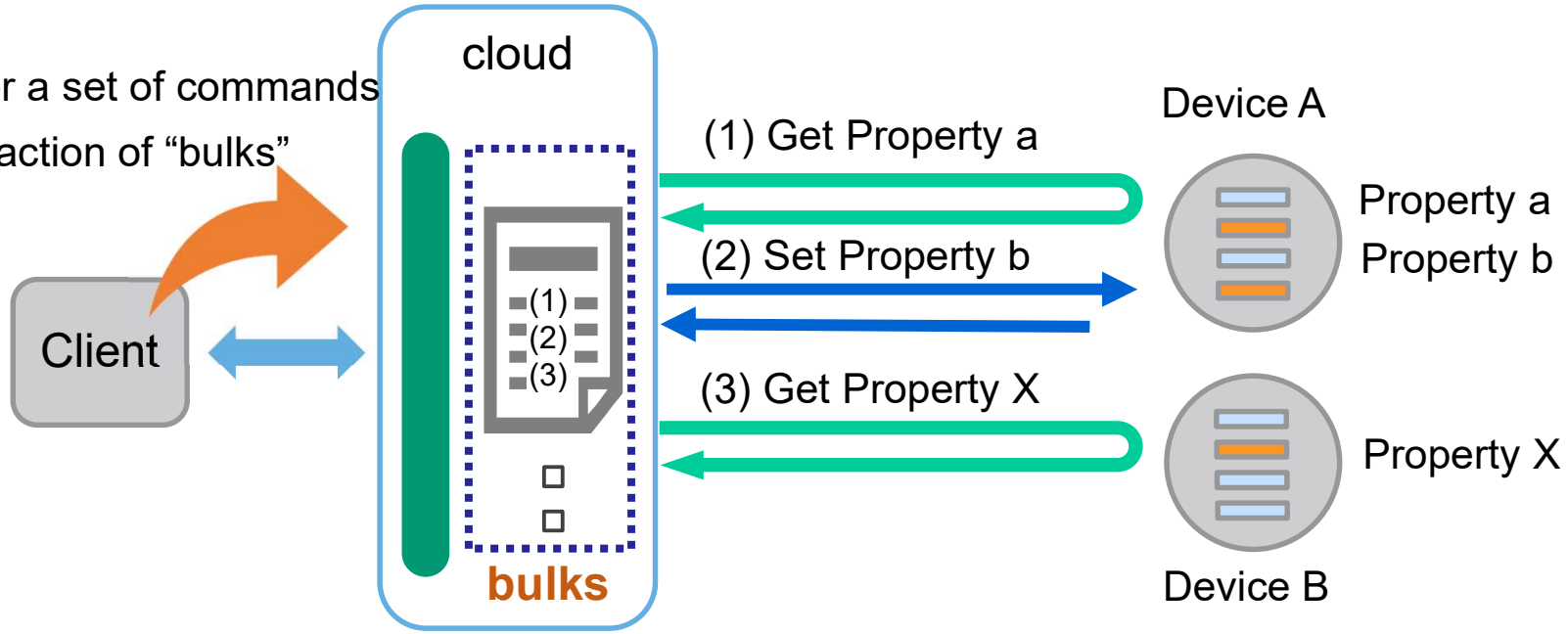
* Device Description is a meta data of a device (similar to TD in WoT)

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).

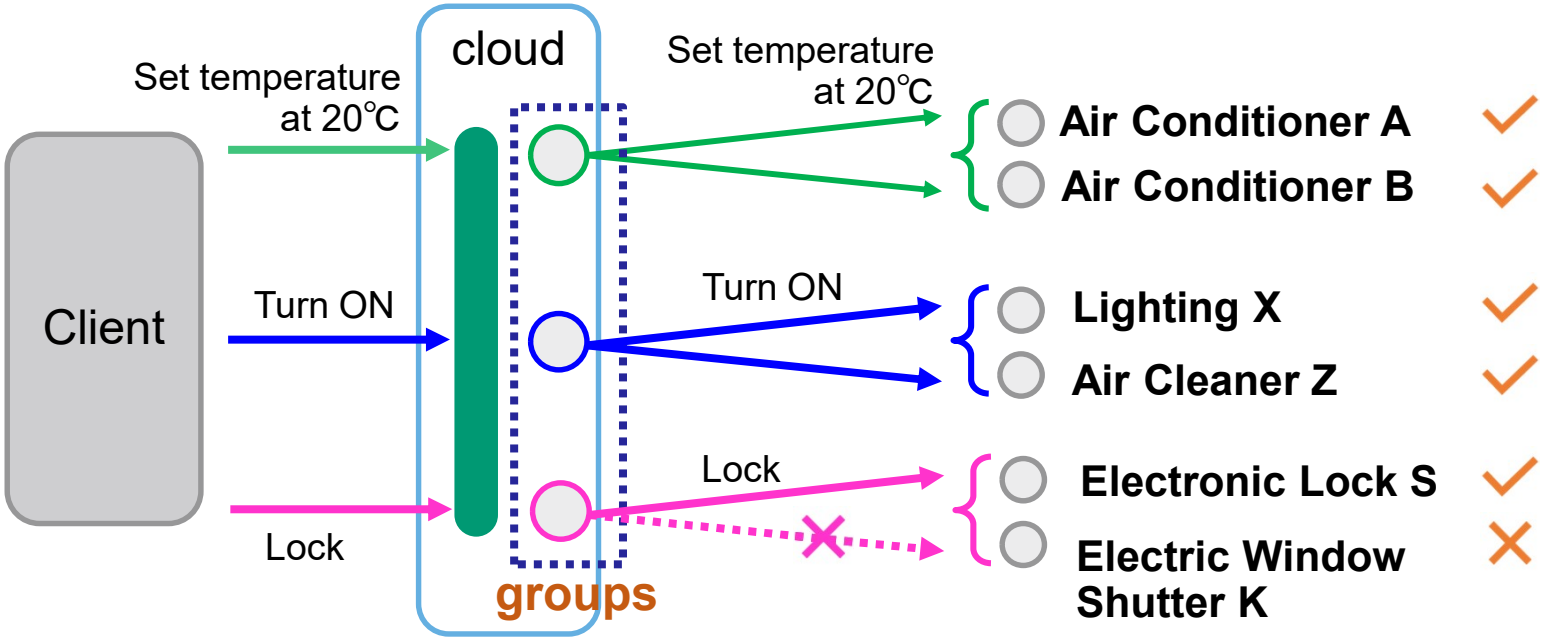
- Register a set of commands
- Invoke action of “bulks”



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.

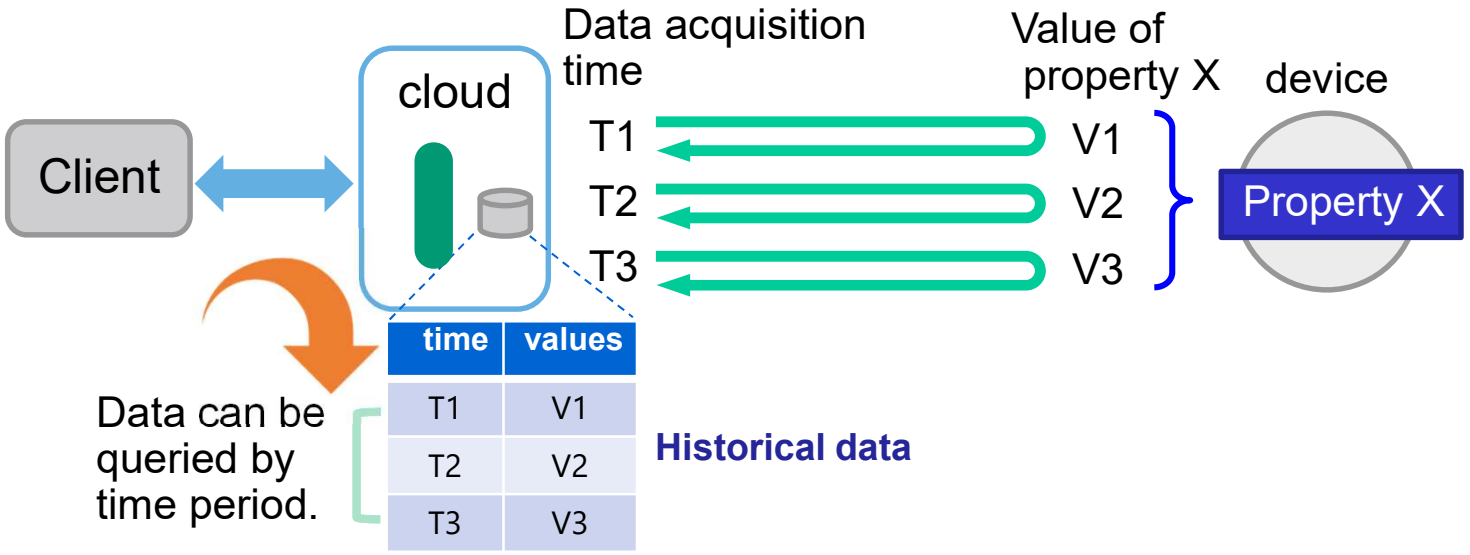
- Client can issue a command to multiple devices with one API call.
- Only devices that implement the command execute the command.



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.



Guiding Principle for ECHONET Lite Web API

- Use JSON (RFC 8259) as our primary representation format
  Not JSON-LD.
- Follow a consistent RESTful style and syntax
- Handle CRUD actions using HTTP methods

Other basic policies :

- All API access : over HTTPS
- HTTP : HTTP/1.1
- Content-Type : application/json
- Unicode : UTF-8
- Use of HTTPS
- Date and time formats : RFC 3339 (ISO 8601)
  e.g., 2018-01-02T12:34:56+09:00
  The format of date (yyyy, mm, dd), hours, minutes and seconds should be followed as particular case.
- Naming convention: lower camel case for property names.
- Device Description : We referred to the early draft of WoT TD and simplified it. Currently, there are several differences.

Path and HTTP Method to Access Device Information on Server

- Client can access device list, Device Description, properties, actions, events on a server.
- Paths used to access them are defined in EL Web API guideline.
- Paths under "devices" are not allowed to be changed.

Path (versionId=v1)	Description
/elapi/v1/devices	list of devices (GET)
/elapi/v1/devices/<deviceId>	Device Description (GET)
/elapi/v1/devices/<deviceId>/properties	all property values (GET)
/elapi/v1/devices/<deviceId>/properties/<property name>	specified property value (GET, PUT)
/elapi/v1/devices/<deviceId>/actions/<action name>	action (POST)
/elapi/v1/devices/<deviceId>/events/<event name>	event resources (Reserved for future use)
/elapi/v1/devices/<deviceId>/echoCommands	Issue ECHONET Lite Commands (Opt)

Security considerations

- Confidentiality and Integrity of data
  - Basic policy is to use HTTPS.
- Authentication / Authorization
  - In many cases, authentication is required for reasons such as identifying API users and restricting the access to devices.
  - ECHONET Lite Web API guideline refers to OAuth 2.0 as an example of authentication and authorization method.

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>
```

Read Status (Read Property)

- Get the latest property value of the specified device.
- On success, the property value is returned in the response.

Example – General Lighting

■ Request

```
GET /elapi/v1/devices/<deviceId>/properties/operatingMode HTTP/1.1
```

■ Response

```
{
    "operatingMode": "color"
}
```

■ Request – in case of object

```
GET /elapi/v1/devices/<deviceId>/properties/rgb HTTP/1.1
```

■ Response

```
{
    "rgb": {
        "r": 20, "g": 255, "b": 0
    }
}
```

Control (Set Property, basic method for control)

- Set the target property value of the specified device
- On success, new property value is returned in the response.

Example – General Lighting

■ Request

```
PUT /elapi/v1/devices/<deviceId>/properties/operatingMode HTTP/1.1
{
    "operatingMode": "color"
}
```

■ Response

```
{
    "operatingMode": "color"
}
```

■ Request – in case of object

```
PUT /elapi/v1/devices/<deviceId>/properties/rgb HTTP/1.1
{
    "rgb": {
        "r": 20, "g": 255, "b": 0
    } }
```

■ Response

```
{
    "rgb": {
        "r": 20, "g": 255, "b": 0
    } }
```

Control (Invoke Action)

- For devices defined by ECHONET Lite,
  only "set only" properties are modeled by action.
- For logical objects such as "bulks", "groups" and "histories",
  some control functions are modeled by action.

■ Request

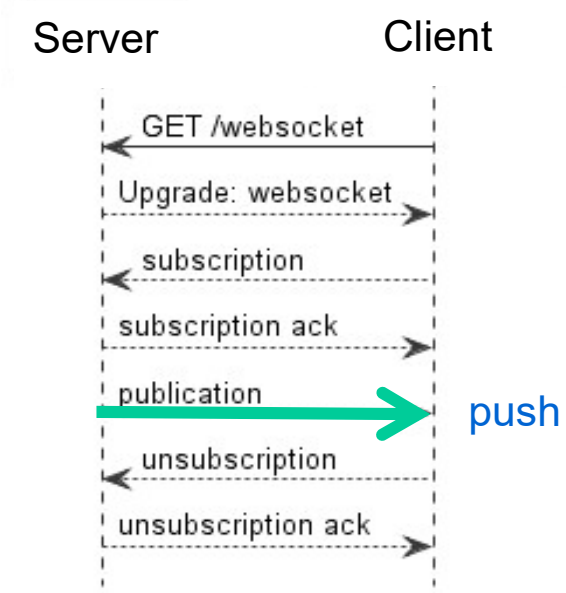
```
POST /elapi/v1/devices/<deviceId>/actions/exampleAction HTTP/1.1
{
    "exampleParamter1": 1
}
```

■ Response

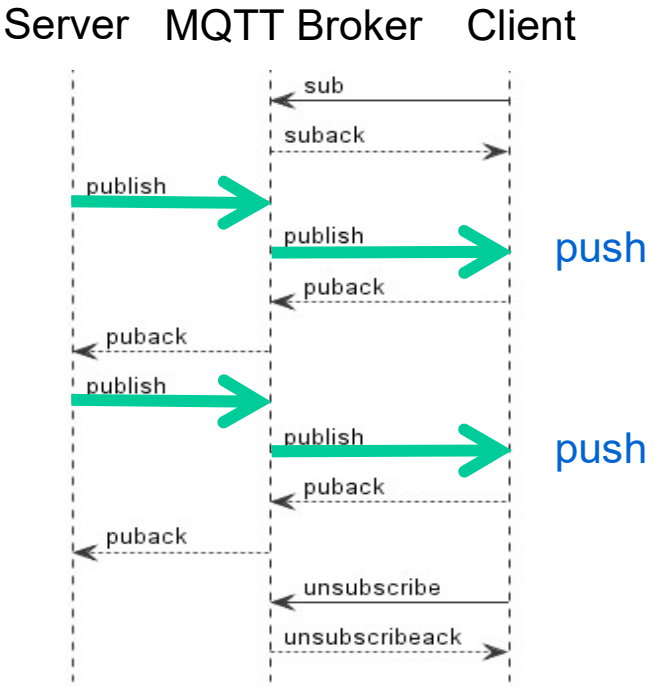
```
{
    "exampleResponse": 10
}
```

Notification of Property Value (INF)

- When dealing with ECHONET Lite INFs, it is also desirable to support push notification type communication methods for sending messages from a server to registered clients.
- ECHONET Lite Web API Guideline explains some examples : Long-polling, WebSocket, MQTT, Webhook



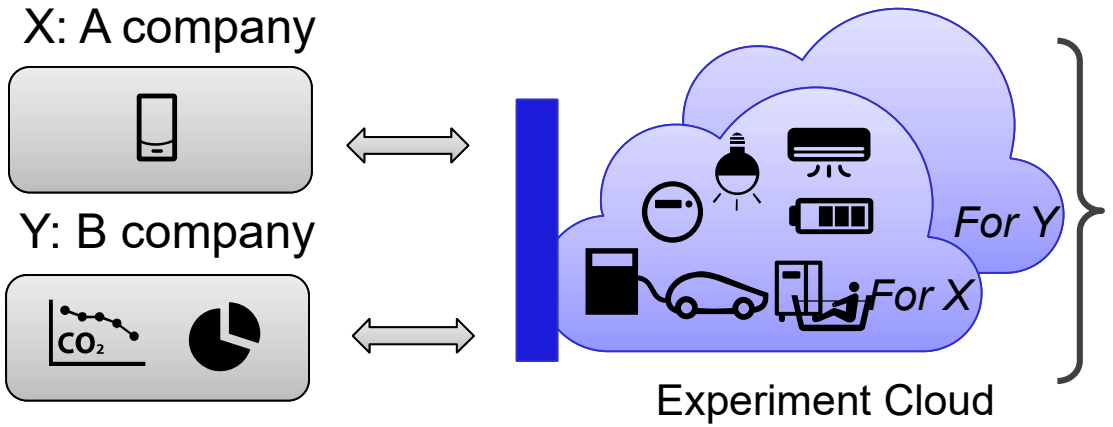
Example of WebSocket



Example of MQTT

EL Web API Experiment Cloud

- EL Web API Cloud complies with the latest Web API guideline API specification and device specifications.
- Users of EL Web API Experiment Cloud can understand the specifications and verify the operation of each Web API.



- Open to ECHOENT Consortium members.
- Enable users to operate emulated devices on the cloud.
- Each user can build his/her own system configuration.

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.
    Exception: set only property in ECHONET Lite is mapped to action 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.