Vol.2019-GN-106 No.1
Vol.2019-CDS-24 No.1
Vol.2019-DCC-21 No.1
2019/1/24

エコーネットコンソーシアムが策定した ECHONET Lite Web API ガイドラインの紹介

藤田裕之[†1]　杉村博[†1]　濱本望絵[†1]　一色正男[†1]

概要：エコーネットコンソーシアムは ECHONET Lite 規格を元にした Web API を策定し，「ECHONET Lite Web API ガイドライン」として 2018 年 10 月に一般公開した．著者らはこの Web API 策定作業に関わってきた．この論文では ECHONET Lite Web API ガイドラインを紹介する．宅内に設置された ECHONET Lite 対応の機器をクラウド経由で基本的な操作を実現するユースケースを主対象として，Web API モデルや機器オブジェクトの JSON モデルについてまとめた．今回策定した仕様は今後発展や変更の可能性があるため，規格ではなく実装のための参考設計指針（ガイドライン）という位置付けである．REST でのアクセス URI を定義し，エアコンやスマートメータなど AIF 重点機器を主とした 12 種類の機器オブジェクトを対象にした具体的な Device Description（JSON 形式で記述した機器仕様）を提供している．

キーワード：エコーネットライト，ECHONET Lite，Web API，REST，IoT，WoT

Introduction of ECHONET Lite Web API Guideline by ECHONET Consortium

HIROYUKI FUJITA[†1]　HIROSHI SUGIMURA[†1]
MOE HAMAMOTO[†1]　MASAO ISSHIKI[†1]

Abstract: ECHONET Consortium has developed ECHONET Lite Web API based on ECHONET Lite protocol and published a guideline document in October 2018. This document describes Web API model and JSON data of device objects considering basic operations to control devices in a house via cloud service.

Keywords: ECHONET Lite，Web API，REST，IoT，WoT

1. はじめに

IoT 技術の進歩に伴い宅内の家電や住宅設備機器をクラウド経由で制御するサービスがすでに実現されている．例えばスマートフォンを利用して宅外からエアコンの電源を入れること（Fig.1）や，スマートスピーカーを利用して音声で照明の明るさを調整すること（Fig.2）が可能である．

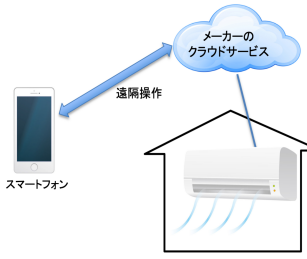

Fig.1 宅外からエアコン制御

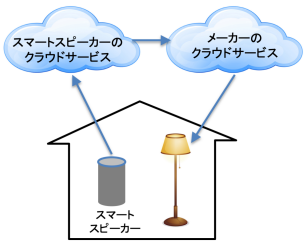

Fig.2 音声で照明制御

しかし現状では製品毎またはメーカー毎にクラウドサービスが存在し，それらが提供する API が異なるため，宅内のさまざまな機器を統一的に制御するサービス（例：デマンドレスポンス）や，機器連携制御のサービス（例：エアコンの室温情報を利用してシーリングファンを動作させる）を実現することは難しい．例えば Fig.3 のようにエアコンと照明がそれぞれ別々のクラウドサービスから制御される構成の場合，この家庭を対象にしたデマンドレスポンスサービスを実現するには，エアコンを制御するためにメーカーA のクラウドサービスに API-A でアクセスし，照明を制御するためにメーカーB のクラウドサービスに API-B でアクセスする必要があり，とても煩雑な制御となる．

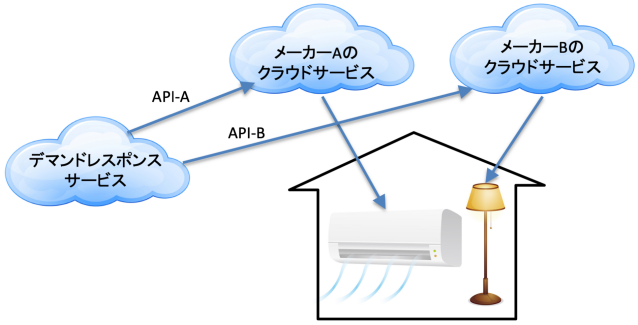

Fig.3 異なるメーカー製のエアコンと照明で構成された家庭を対象とするデマンドレスポンスサービス

エコーネットコンソーシアムはこのような課題を解決するために，宅内の家電や住宅設備を統一的に制御できる ECHONET Lite Web API を策定し，その仕様を「ECHONET Lite Web API ガイドライン」として 2018 年 10 月に一般公

†1 神奈川工科大学
Kanagawa Institute of Technology

Vol.2019-GN-106 No.1
Vol.2019-CDS-24 No.1
Vol.2019-DCC-21 No.1
2019/1/24

開した[1]．今回策定した仕様は今後発展や変更の可能性があるため，規格ではなく実装のための参考設計指針（ガイドライン）という位置付けである．API の実装は REST(a) を利用する標準的な Web API である．さまざまな機器の仕様を Device Description という machine readable なデータ（そのままプログラムで利用できる構造化されたデータ）として提供することを特徴としている．

これまで ECHONET Lite を Web API で扱う試みがいくつか存在した．いずれもバイナリデータで定義された ECHONET Lite protocol を扱いやすい Web API とすることが課題となっている．

経済産業省のスマートハウス・ビル標準・事業促進検討会は HEMS 情報基盤-HEMS データ利活用事業者間 API 標準仕様書[2]を提案した．大規模 HEMS 実証事業において ECHONET Lite のコマンドログを取得するためである．コマンドログはバイナリーデータのまま利用している．

大和田茂（Sony CSL）は ECHONET Lite 機器も含めたネットワーク上の家電をコントロールするプラットフォーム Kadecot[3][4]を開発した．機器名は文字列で指定するがプロパティやデータはバイナリーデータのまま扱っている．

大和ハウス工業株式会社は Sony CSL と共同で住宅 API[5]を提案した．ECHONET Lite のプロパティやデータを文字列で扱っている．ただし ECHONET Lite のプロパティやデータを一対一に WebAPI にマッピングしているため，デバイス寄りの実装がそのまま引き継がれている．

著者らは ECHONET Lite 機器を容易に制御できる REST を利用した実装の Web API を提案した [6]．ビットマップにエンコードされたステータスも含めプロパティ名やステータス名を定義した．ECHONET Lite では複数のプロパティの操作が必要な機能を Web API では一つのプロパティにまとめた．機器の仕様を Device Description という JSON 形式のデータで提供することで，ECHONET Lite の機器仕様を容易に扱えるようにした．この Web API は今回のガイドラインの基本設計として引き継がれている．

一方，World Wide Web Consortium [7]は Internet of Things (IoT)の上位概念として Web of Things (WoT) を定義し，Alljoyn by Open Connectivity Foundation[8]や oneM2M[9]や GotAPI by Open Mobile Alliance[10]などの既存 IoT platform を汎用的に扱う規格の標準化を策定している[11]．Things のメタデータを記述する Things Description，操作を記述する Scripting API，各種プロトコルを標準的なメッセージに変換する Binding Templates などで構成される．Things との interactions として Property（データの read/write で表現できる機能）, Action（Action はリセットやフェードインといった read/write での表現が難しい機能）, Event（状態の通知機能）が定義され，Things Description に具体的内容を記述する．今回のガイドライン策定においては WoT Working Group とコミュニケーションを取り，標準化を注視しながら作業を進めた．特に「ECHONET Lite 機器オブジェクト詳細規定」をマシンリーダブルなデータである Device Description として記述する際に，WoT の Things Description を参考にした．

a RErepresentational State Transfer の略．Web サービス設計モデルのひとつ

2. システム構成

想定するシステム構成を Fig.4 に示す．ECHONET Lite クラウドサービス（EL クラウド）はクラウドサービス事業者対して ECHONET Lite Web API を提供する．EL クラウドは ECHONET Lite Web API で命令を受信すると，宅内の HEMS コントローラ経由で宅内の ECHONET Lite 対応機器を制御する．EL クラウドは通常 HEMS コントローラのベンダーが提供するクラウドサービスである．EL クラウドと HEMS コントローラ間の通信方式は特に規定しない．

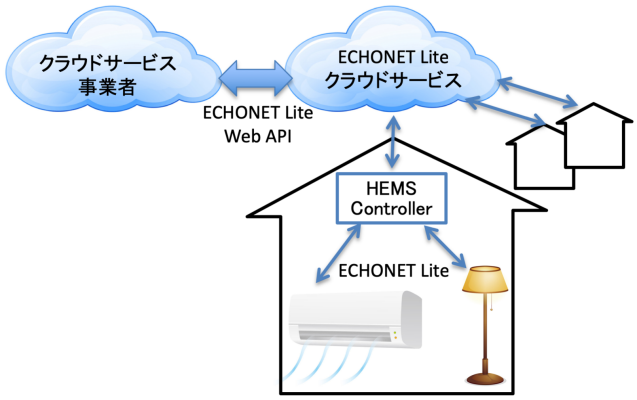

Fig.4　システム構成

3. WebAPI

3.1 ユースケース

ガイドライン V1.00 では，機器の一覧取得・管理情報取得や状態取得・制御といった基本的ユースケースを対象に API を定義している．機器の一覧取得では，あらかじめ HEMS コントローラが ECHONET Lite コマンドを用いて収集し EL クラウド内で保持している宅内の機器リストを利用する．機器の状態取得では，HEMS コントローラ経由で ECHONET Lite 機器のプロパティ値を取得する．特定の機器の全てのプロパティ値を一括して取得することもできる．機器の状態制御では，HEMS コントローラ経由で ECHONET Lite 機器のプロパティの値を変更することで状態制御を行う．

ECHONET Lite には値の変化を通知するプロパティ（状変通知プロパティ）がある．サーバー（EL クラウド）からクライアント（クラウドサービス事業者）への通知機能を Web API に実装するには REST 以外の PUSH 通信（例：long-polling, websocket, MQTT）の実装が必要となる．ガイ

Vol.2019-GN-106 No.1
Vol.2019-CDS-24 No.1
Vol.2019-DCC-21 No.1
2019/1/24

ドライン V1.00 ではこれらを紹介するだけにとどめ，実装依存としている．

認証・認可に関しては OAuth などを紹介し，実装依存としている．

3.2 Web API モデルの指針（基本方針）

本ガイドラインでは，Web API のモデルとして，現在実質的な Web API の標準である REST を採用する．操作対象となるリソースを URI で記述し，それに対して HTTP メソッドでアクセスすることで機器の制御を行う．データ形式は現在主流である JSON を採用する．REST の 4 原則である，①HTTP によるステートレス性，②URI によるアドレス可視性，③HTTP メソッドによる統一インタフェース，④JSON によるリソース間の接続性に留意しつつ，API を規定する．

リソースの識別方法は，下記を基本とする．

形式 スキーム://ホスト名/リソースへのパス
例 https://www.example.com/elapi/v1/devices/12345678

この例では, HTTPS スキームにてホスト www.example.com が保有する ID が 12345678 の機器データのリソースを指定している．このようにパスで階層化してリソースを指定する．リソースに対するアクションは，HTTP メソッドの GET, PUT, POST を利用する．リソースは WoT の Things Description を参考にして，Property, Action, Event を定義した. ガイドライン V1.00 では Property と Action を利用している．

前述以外の基本方針として，下記を定める．

- HTTP のバージョンは HTTP/1.1 を採用する．
- JSON 形式は，RFC 8259 に従う．Content-Type として「application/json」を指定し，文字コードは UTF-8 を使用する．クライアント側では，Accept というリクエストヘッダでメディアタイプ「application/json」を指定する．
- 日時形式には，RFC 3339（ISO 8601）を採用し，タイムゾーンを考慮した下記表記に従う．

例：2018-01-02T12:34:56+09:00

ただし，年月日，時，分，秒など個別に指定すべきケースは個々の記述に従う．

- リソースに指定されるプロパティ名称など属性名の命名規則として，単一語の場合は小文字を使用するが，複合語の場合は（ローワー）キャメルケースを使用する．本ガイドラインでは説明の都合，任意の値や記述を表現する場合には山括弧「<>」を使用する．
- ECHONET Lite の機器オブジェクトの Device Description（後述）は，W3C で策定中の WoT モデルの Things Description を参考として記述する．

以下に ECHONET Lite Web API 一覧を示す．

Table.1 ECHONET Lite Web API 一覧

Method	URI	リソース
GET	/elapi	バージョン一覧
GET	/elapi/<versionId>	対象リソース種一覧
GET	/elapi/<versionId>/devices	機器リスト
GET	/elapi/<versionId>/devices/<deviceId>	機器情報
GET	/elapi/<versionId>/devices/<deviceId>/properties	すべてのプロパティ値
GET/PUT	/elapi/<versionId>/devices/<deviceId>/properties/<propertyResourceName>	プロパティ値
POST	/elapi/<versionId>/devices/<deviceId>/actions/<actionResourceName>	アクション

3.3 Web API の詳細

以下，Table.1 に示した API の内容について説明する．

GET /elapi

サービス：EL クラウドが対応するバージョン一覧の取得
解説：現在は V1.00 であるが，将来を見越して複数のバージョン対応のためのコマンドである．

Sample:

■ リクエスト

```
GET /elapi
```

■ レスポンス

```
{
  "versions":[
    {
      "id":"v1",
      "status":"CURRENT",
      "updated":"2018-01-01T12:34:56Z+09:00"
    },
    {
      "id":"v2",
      "status":"EXPERIMENTAL",
      "updated":"2018-01-02T01:02:03Z+09:00"
    }
  ]
}
```

GET /elapi/<versionId>

サービス：EL クラウドが対応するリソース種一覧の取得
解説：ガイドライン V1.00 では devices, controllers, sites, users, groups の 5 種類のリソースを提案している．devices は必須．devices 以外のリソース種を提供する場合，そのリソース種の配下のパスには devices を配置する．ガイドライン V1.00 では devices のケースのみ規定する．

Sample:

■ リクエスト

Vol.2019-GN-106 No.1
Vol.2019-CDS-24 No.1
Vol.2019-DCC-21 No.1
2019/1/24

```
GET /elapi/v1
■ レスポンス
{
  "v1":[
    {
      "name":"devices",
      "descriptions":{
        "ja":"device resource",
        "en":"device resource"
      },
      "total":10
    },
    {
      "name":"controllers",
      "descriptions":{
        "ja":"controller resource",
        "en":"controller resource"
      },
      "total":1
    }
  ]
}
```

GET /elapi/<versionId>/devices

サービス：制御対象機器リストの取得

解説:EL クラウドは HEMS コントローラ経由で ECHONET Lite のコマンド(b)を利用して，宅内の ECHONET Lite 機器のリストをあらかじめ取得する．EL クラウドが Web API で本コマンドを受信すると，該当するユーザーの機器リストをレスポンスする．

Sample:

```
■ リクエスト
GET /elapi/v1/devices
■ レスポンス
{
  "devices":[
    {
      "id":"0xFE000006123456789ABCDEF1234",
      "deviceType":"generalLighting",
      "protocol":{
        "type":"ECHONET_Lite v1.13",
        "version":"Rel.J"
      },
      "manufacturer":{
        "code":"0x000077",
        "descriptions":{
          "ja":"神奈川工科大学",
          "en":"KAIT"
        }
      }
    },
    {...}
  ]
}
```

機器リストには機器を識別するための id や機器種別を表す deviceType などが記述されている．以下に deviceType の例を示す．

Table.2 Device Type

機器名	deviceType
家庭用エアコン	homeAirConditioner
電気温水器	electricWaterHeater
瞬間式給湯器	instantaneousWaterHeater
燃料電池	fuelCell
蓄電池	storageBattery
電気自動車充放電器	evChargerDischarger
低圧スマート電力量メータ	lvSmartElectricEnergyMeter
高圧スマート電力量メータ	hvSmartElectricEnergyMeter
一般照明	generalLighting
電気自動車充電器	evCharger
拡張照明システム	enhancedLightingSystem
コントローラ	controller

GET /elapi/<versionId>/devices/<deviceId>

サービス：指定した機器の Device Description の取得

解説： Device Description とは，機器が実装している機能（Properties, Actions, Events）を JSON 形式で記述したものである．詳細は後述する．EL クラウドは HEMS コントローラ経由で ECHONET Lite のコマンドを利用して，宅内の ECHONET Lite 機器のプロパティマップ(c)をあらかじめ取得する．EL クラウドが本コマンドを受信すると，該当する機器のプロパティマップをもとに Device Description をレスポンスする．

Sample:

```
■ リクエスト
GET /elapi/v1/devices/1234
■ レスポンス
{
  "deviceType":generalLighting,
  "eoj":< eoj in Hex string >,
  "descriptions":{
    "ja":"一般照明",
    "en":"General Lighting"
  },
  "properties":{
    "operationStatus":< property object >,
    "Brightness":< property object >,
     ...
  },
  "actions":{ },
  "events":{ }
}
```

b Node profile のインスタンスリストの GET コマンド

c プロパティのリストに対応するもの

Vol.2019-GN-106 No.1
Vol.2019-CDS-24 No.1
Vol.2019-DCC-21 No.1
2019/1/24

GET /elapi/<versionId>/devices/<deviceId>/properties

サービス：指定した機器の全てのプロパティ値の取得

解説：<deviceId> で指定した機器の全てのプロパティ値を取得する．値は EL クラウドにキャッシュされたものである．一括して Static な値を取得するのに便利な機能である．最新のプロパティ値が必要な場合は次の API を参照のこと．

Sample:

■ リクエスト

```
GET /elapi/v1/devices/1234/properties
```

■ レスポンス

```
{
  "operationStatus":true,
  "faultStatus":false,
  "brightness":50,
  "operationMode":"color",
  "rgb":{
    "r":20,
    "g":255,
    "b":0
  }
}
```

GET /elapi/<versionId>/devices/<deviceId>/properties/<propertyResourceName>

サービス：指定したプロパティ値の取得

解説：EL クラウドが本コマンドを受信すると，HEMS コントローラ経由で ECHONET Lite の GET のコマンドを対象機器に送信し，受信した ECHONET Lite データをもとに，プロパティ値を Web API でレスポンスする．

Sample:

■ リクエスト

```
GET /elapi/v1/devices/1234/properties
/operationMode
```

■ レスポンス

```
{
  "operationMode": "color"
}
```

Sample:（レスポンスが object の場合）

■ リクエスト

```
GET /elapi/v1/devices/1234/properties
/rgb
```

■ レスポンス

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

Sample:（リクエストに query がある場合）

ECHONET Lite プロトコルの GET コマンドには引数が存在しない．GET で引数が必要な場合は引数を設定するための別のプロパティが用意され，あらかじめ値を SET する．Web API ではこのような Atomic な操作を一つの API にまとめ，引数は GET の query として実装する．例えばスマートメータの履歴値を Web API で取得する場合，日にちを query で指定しプロパティ値を取得する．EL クラウドはそのコマンドを受信すると ECHONET Lite コマンドでスマートメータに対して日にちを SET した後に履歴値を GET し，そのデータをクライアントへレスポンスする．

■ リクエスト

```
GET /elapi/v1/devices/1234/properties
/normalDirectionIntegralElectricEnergyLog1?
day=0
```

■ レスポンス

```
{
  "normalDirectionIntegralElectricEnergyLog1":{
    "day":0,
    "energy":[
      20,
      34,
      59,
      109
    ]
  }
}
```

PUT /elapi/<versionId>/devices/<deviceId>/properties/<propertyResourceName>

サービス：プロパティ値の設定

解説：EL クラウドがこのコマンドを受信すると，対象機器の Device Description の内容に基づき ECHONET Lite の SET のコマンドを作成し，HEMS コントローラ経由で宅内の機器のプロパティ値を設定する．その後 ECHONET Lite の GET のコマンドでプロパティ値を取得し，Web API でレスポンスする．

Sample:

■ リクエスト

```
PUT /elapi/v1/devices/1234/properties
/operationMode
{
  "operationMode": "color"
}
```

■ レスポンス

```
{
  "operationMode": "color"
}
```

Sample:（リクエストが object の場合）

■ リクエスト

```
PUT /elapi/v1/devices/1234/properties
/rgb
{
```

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

■ レスポンス

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

POST /elapi/<versionId>/devices/<deviceId> /actions/<actionResourceName>

サービス：アクションの実行

解説：ECHONET Lite で SET only のプロパティは Web API では Action というカテゴリーに記述され POST メソッドで実行される．Action によっては実行時にデータを渡すものもある．以下の例の自動車充放電器の積算充電電力量をリセットする機能では渡すデータがない．

Sample:

■ リクエスト

```
POST /elapi/v1/devices/1234/actions
/resetCumulativeChargingElectricEnergy
```

4. Device Description

ECHONET Lite 規格では機器種別：EOJ(d)，プロパティ種別： EPC(e)や状態を表すプロパティデータ：EDT(f)はバイナリーデータで定義されており，「APPENDIX ECHONET 機器オブジェクト詳細規定」として公開されている．例えばエアコンの運転モードの冷房に対応するコードは

EOJ: 0x0130, EPC: 0xB0, EDT: 0x42

である．

ECHONET Lite Web API ではこれらに対応する情報を Device Description と呼び「ECHONET Lite Web API Appendix」として公開している．Device Description は W3C の WoT の Things Description を参考に全体の framework は共通とし，ECHONET Lite 固有の情報を追加した．Device Description はプログラムで容易に利用できるように以下の特徴がある

- Device Description 全体は JSON 形式で記述
- 機器名，プロパティ名や状態名は文字列で定義
- プロパティ毎に writable かどうかの記述が可能
- プロパティのデータフォーマットは JSON Schema で記述
- プロパティのデータタイプは JSON Schema に準拠（number, boolean, string, array, object）

JSON Schema とは，データのフォーマットを JSON 形式で記述したものである．これを利用するとデータの Validation を容易に行うことができる．以下に JSON Schema の例を示す．

例：数値データで最小値が１，最大値が８

```
"schema":{
  {
    "type":"number",
    "minimum":1,
    "maximum":8
  }
}
```

例：文字データで取りうる値が cooling と heating のみ

```
"schema":{
  {
    "type":"string",
    "enum":["cooling","heating"]
  }
}
```

例：３つの数値データ（最小値 0，最大値 255）から構成され，それぞれ red, green, blue の名前がある場合

```
"schema":{
  {
    "type":"object",
    "properties":{
      "red":{
        "type":"number",
        "minimum":0,
        "maximum":255
      },
      "green":{
        "type":"number",
        "minimum":0,
        "maximum":255
      },
      "blue":{
        "type":"number",
        "minimum":0,
        "maximum":255
      }
    }
  }
}
```

以下に家庭用エアコンの Device Description の抜粋を示す．

```
{
  "deviceType":"homeAirConditioner",
  "eoj":"0x0130",
  "descriptions":{
    "ja":"家庭用エアコン",
    "en":"Home Air Conditioner"
  },
  "properties":{
    "operationMode":{
```

d ECHONET Object
e ECHONET Property Code
f ECHONET Data

```
      "epc":"0xB0",
      "descriptions":{
        "ja":"運転モード設定",
        "en":"Operation mode setting"
      },
      "writable":true,
      "observable":true,
      "schema":{
        "type":"string",
        "enum":[
          "cooling",
          "heating"
        ],
        "values":[
          {
            "value":"cooling",
            "descriptions":{
              "ja":"冷房",
              "en":"Cooling"
            },
            "edt":"0x42"
          },
          {
            "value":"heating",
            "descriptions":{
              "ja":"暖房",
              "en":"Heating"
            },
            "edt":"0x43"
          }
        ]
      }
    }
  },
  ...
  "actions":{
    ...
  }
}
```


5. 実験サーバー

ECHONET Lite Web API の仕様理解の支援やガイドラインの動作検証を目的に，著者らとエコーネットコンソーシアムは共同で実験サーバーを開発し，エコーネットコンソーシアム会員向けに公開予定である．複数ユーザーからの同時アクセスも可能で，ユーザーごとに設定した API key をもとにユーザーを識別する．ECHONET Lite 機器は実験サーバー上でエミュレートする．各ユーザーは ECHONET Lite 機器の構成（機器種別や台数）を自由に設定できる．また，read only の property の値を変更するための Web I/F も用意した．Device Description の JSON データを利用してプロパティデータの生成やバリデーションを行なっている．

6. 結論

ECHONET Lite Web API のガイドライン V1.00 が策定されたことにより，ECHONET Lite 対応 HEMS システムの将来性がさらに広がった．各社の ECHONET Lite 対応クラウドが ECHONET Lite Web API をサポートし，デマンドレスポンスやバーチャルパワープラントなどのクラウドサービスの開発が促進されることを期待する．

参考文献

1 https://echonet.jp/web_api/#guideline
2 http://www.meti.go.jp/committee/kenkyukai/shoujo/smart_house/pdf/009_s10_00.pdf
3 https://www.sonycsl.co.jp/tokyo/347/
4 https://github.com/SonyCSL/Kadecot-JS
5 https://github.com/KAIT-HEMS/PicoGW
6 藤田 裕之, 杉村 博, 村上 隆史, 一色 正男 : ECHONET Lite Web API と Protocol Bridge，情報処理学会研究報告 Voi.2018-CDS-21 No.23
7 https://www.w3.org
8 https://openconnectivity.org
9 http://www.onem2m.org
10 https://device-webapi.org/gotapi.html
11 https://www.w3.org/WoT/