

WoT-JP CG アウトリーチTF

ECHONET Lite Web APIの紹介 ～これまでのWoT連携状況および課題について～



エコーネットコンソーシアム
技術委員副委員長・WebAPI検討WG主査
寺本 圭一

1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

3 WoTとの連携経緯・連携活動内容

4 WoT とEL Web APIの仕様の違い

5 課題？期待？

1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

3 WoTとの連携経緯・連携活動内容

4 WoT とEL Web APIの仕様の違い

5 課題？期待？

住宅設備機器、エネルギー設備などを監視制御する国際通信規格

ECHONET Lite
とは

スマートホームを実現する公知な標準インターフェース* (2012/2/24 経産省より推奨IF指定)
異なる企業が提供するIoT家電、住宅設備、蓄電池設備の通信を可能にする「共通言葉」

ECHONET Lite
の特徴

- ・ 共通仕様によるマルチベンダ環境を実現
- ・ 各種標準伝送メディアが利用可（軽量通信プロトコル。マルチキャスト対応）
- ・ 機器（100種類以上）に関する多彩なコマンド群を定義
- ・ 国際標準規格（IEC, ISO/IEC準拠）

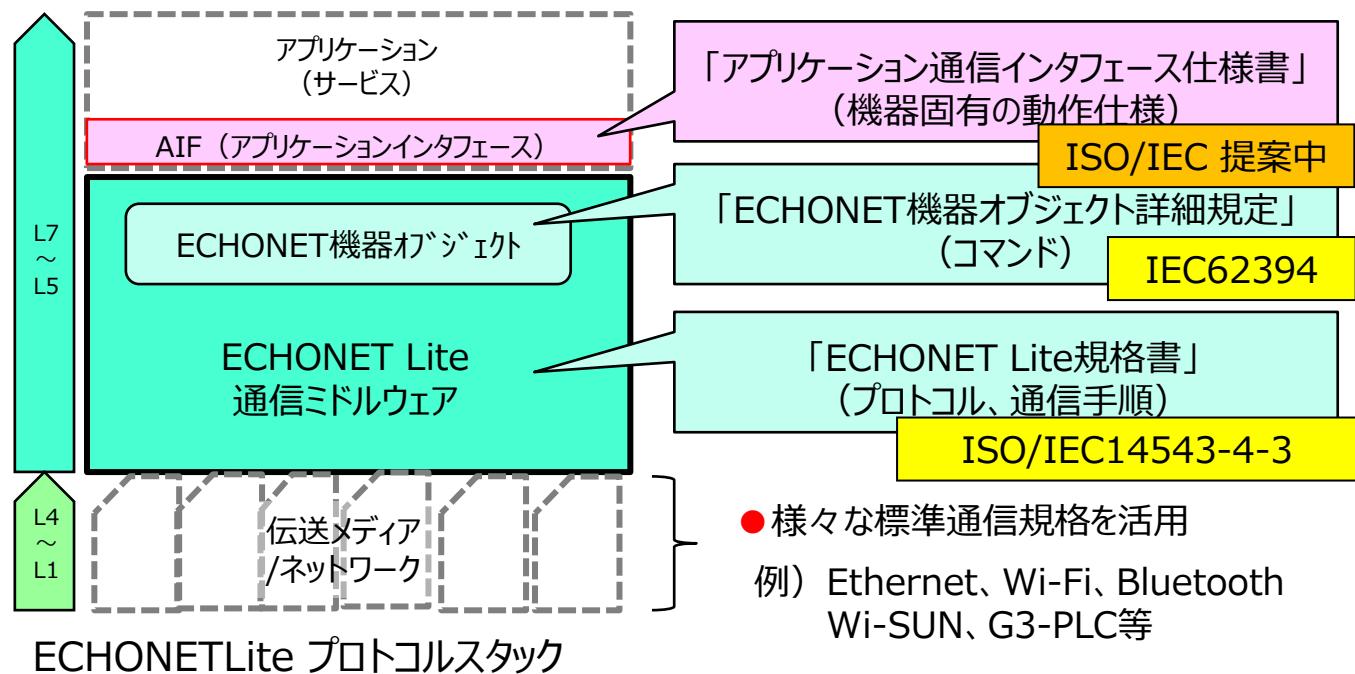
IoT機器の種類毎に制御コマンドを定義



100種類以上の機器仕様を提供

センサ関連機器	火災センサ、人体検知センサ、温度センサ、CO ₂ センサ、電流量センサ、etc.	
空調関連機器	エアコン、扇風機、換気扇、空気清浄機、ホットカーペット、石油ファンヒータ、etc.	
住宅・設備関連機器	電動ブラインド、温水器、電気錠、スマートメーター、太陽光発電、蓄電池、燃料電池、一般照明、非常灯、etc.	
調理・家事関連機器	電子レンジ、食器洗い機、食器乾燥機、洗濯機、衣類乾燥機、etc.	
健康管理関連機器	体重計、体脂肪計、体温計、血圧計、血糖値計、etc.	
管理・操作関連機器	コントローラ、スイッチ（HA機器）、etc.	
AV関連機器	TV、ディスプレイ、etc.	

ECHONET Liteは、IEC、ISO/IECで認定された国際標準規格



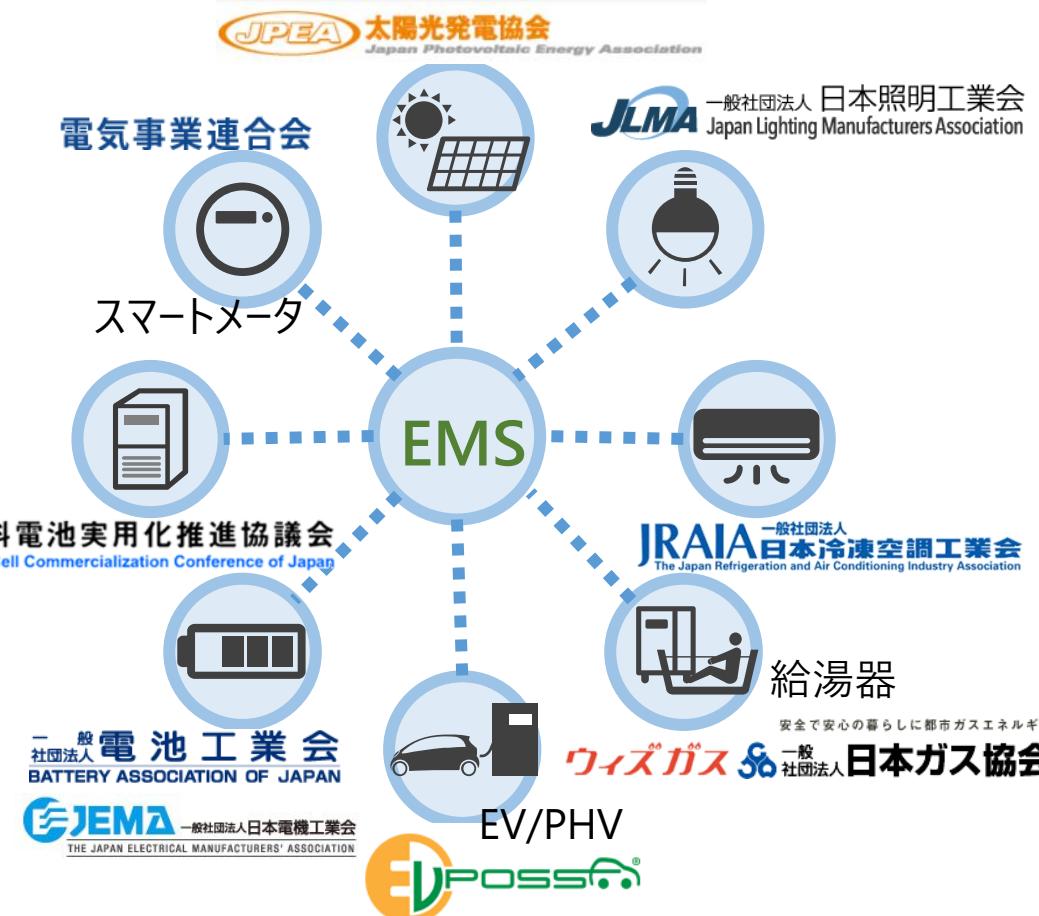
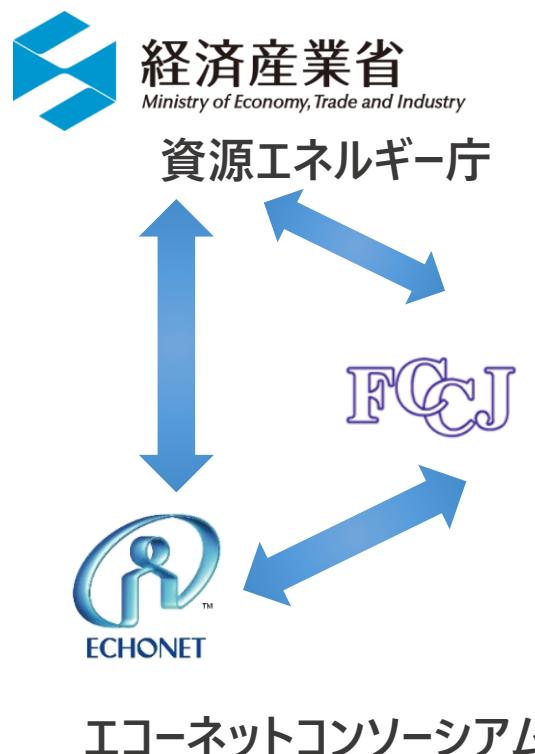
アプリレベルで動作仕様を策定
(第三者認証により相互接続性確保)

機器毎に詳細なコマンドを定義

シンプルなバイナリコマンドにて構成
(基本IPv4/v6, UDPベース。マルチキャスト対応。
TCP対応も可)

伝送媒体に依存しない

経産省や主要業界団体と連携し、機器を活用するサービス実現に資する通信規格を策定

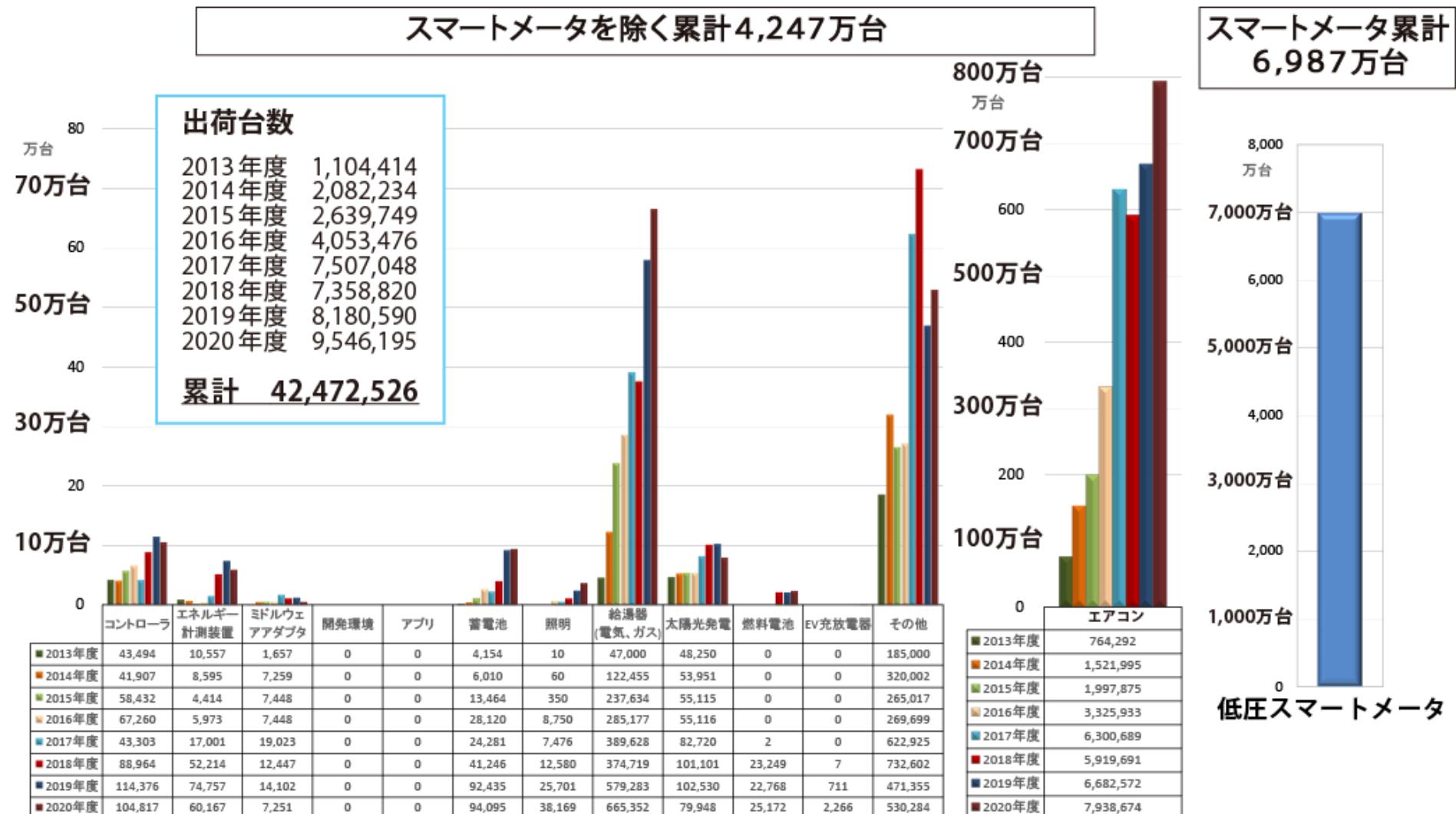


2012年：HEMS重点8機器選定

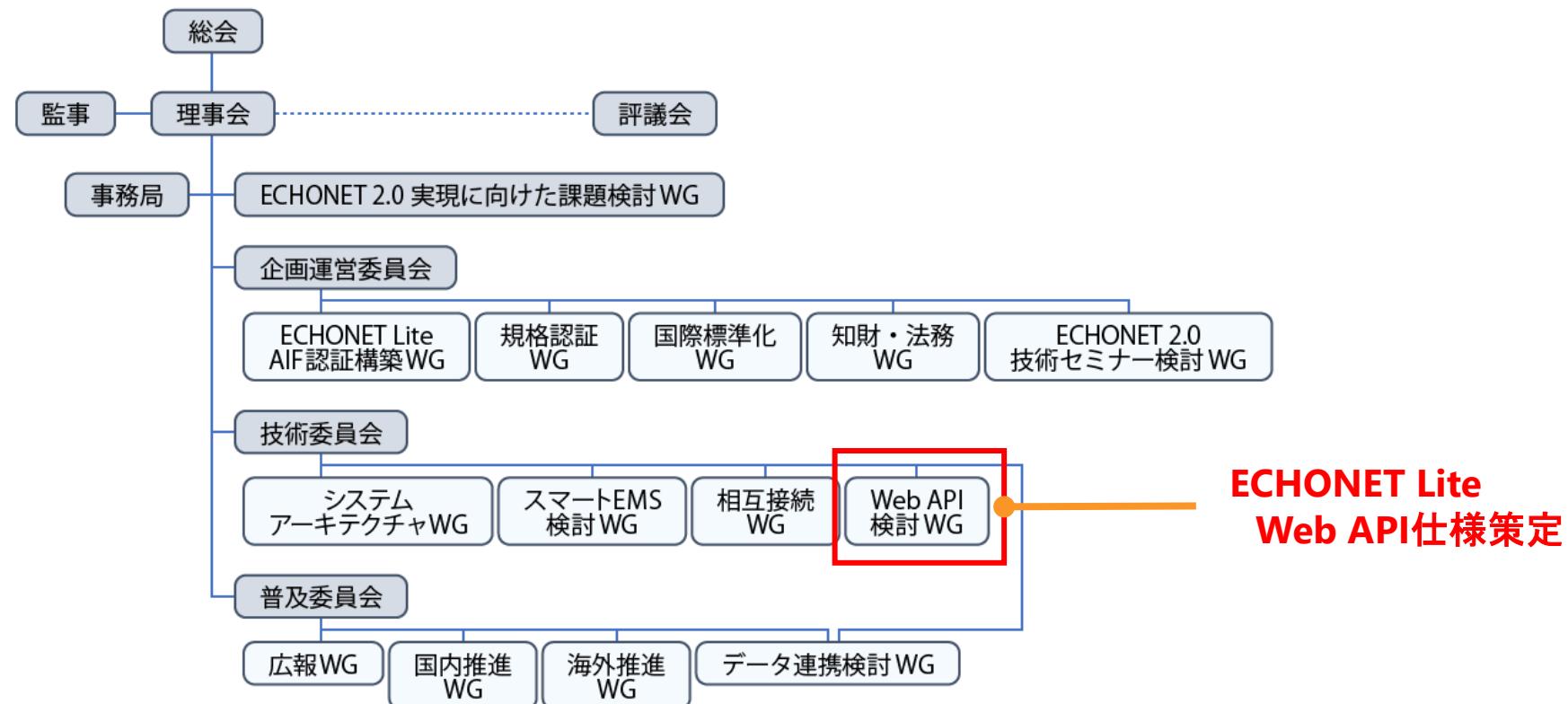
現在：ZEH + などの要件対象に

VPP実証などでも活用中

2020年度に累計1億台突破



参加企業 & 団体	<ul style="list-style-type: none"> ・ 幹事会社 : 6社 (+ 幹事準会員42社) ・ 一般会員 : 159社 (+ 一般準会員42社) ・ 学術会員 : 27会員 	2021/9/24現在
幹事会社	<ul style="list-style-type: none"> ・ 東京電力ホールディングス(株)、(株)東芝、日本電信電話(株)、 パナソニック(株)、(株)日立製作所、三菱電機(株) 	



**ECHONET Lite
Web API仕様策定**

1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

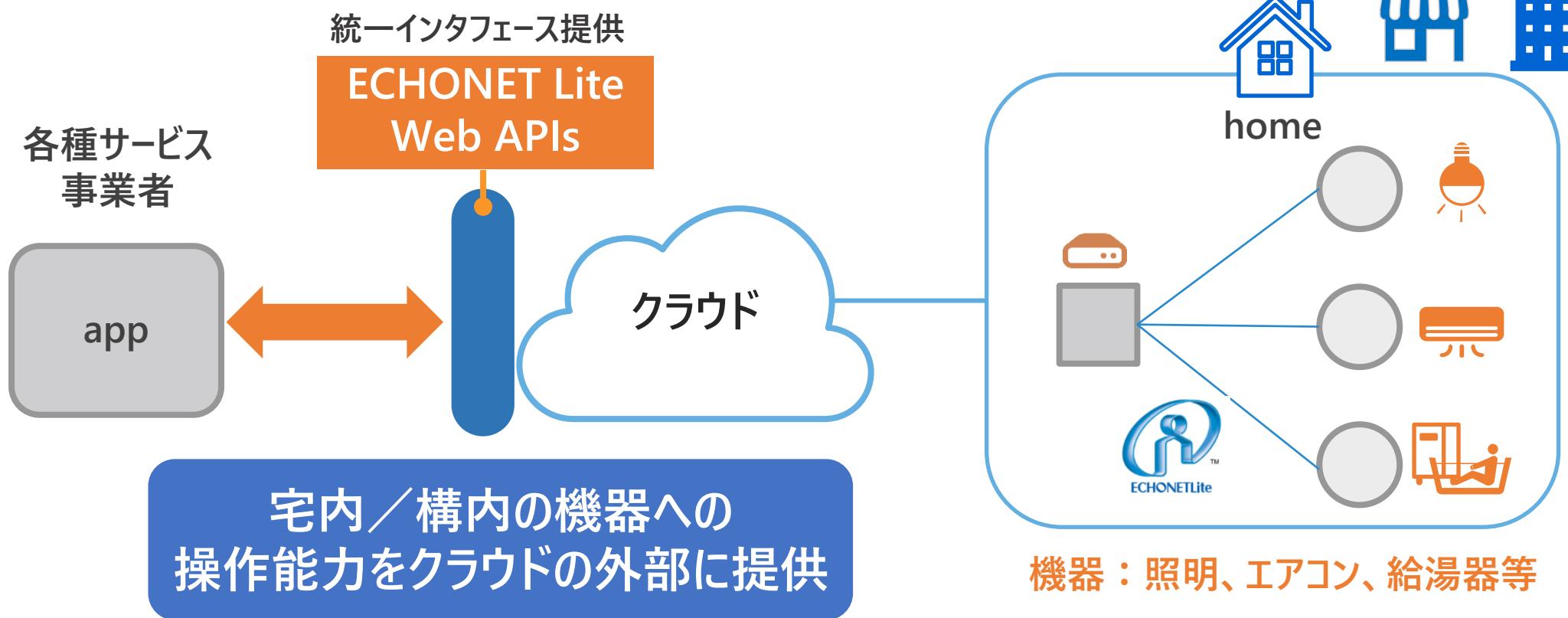
3 WoTとの連携経緯・連携活動内容

4 WoT とEL Web APIの仕様の違い

5 課題？期待？

ECHONET Lite Web APIとは？

ECHONET Lite等の機器をクラウドを介して操作可能とするWeb API



仕様書は一般公開中

教育機関での学習・認定、製品・サービスも順次登場中

<https://echonet.jp>



ダウンロード>Web API>
ECHONET Lite Web APIガイドライン

https://echonet.jp/web_api/



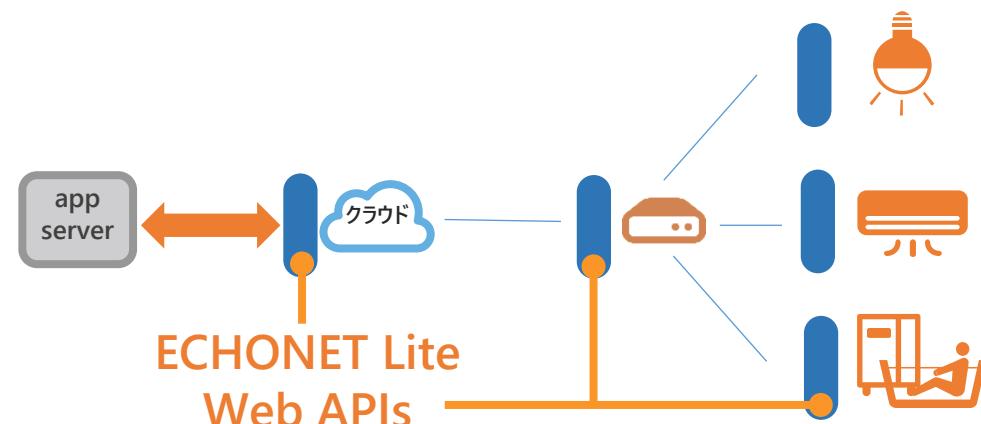
Web API ガイドライン（日本語版）のダウンロード ファイルリスト		
API仕様部 Version 1.1.3 (PDF版)	ECHONET_Lite_Web_API_Specs_v1.1.3.pdf [PDF 2.0MB]	
機器仕様部 Version 1.3.0 (PDF版)	ECHONET_Lite_Web_API_Dev_Specs_v1.3.0.pdf [PDF 2.1MB]	<small>NEW Errata [PDF 41KB]</small>
Device Description(JSON) (機器仕様部 Version 1.3.0 より抜粋) [zip圧縮]	Web_API_device_descriptions_v1.3.0.zip [zip 252KB]	<small>NEW Errata [PDF 40KB]</small>

背景
IoT技術の向上に伴い、クラウド環境を介したサービス提供モデルが進展しています。これらのモデルの中には、ECHONET Lite対応機器をクラウドを介して

- 「ECHONET IoTマスター制度」を開始
教育機関にて教育 & 認定

https://echonet.jp/about_echonet_iot_master_syst/

- EL Web API 製品・サービスも展開開始
クラウド、GW、機器らへも適用可



1 異業種を束ねる 統一APIモデルを提供

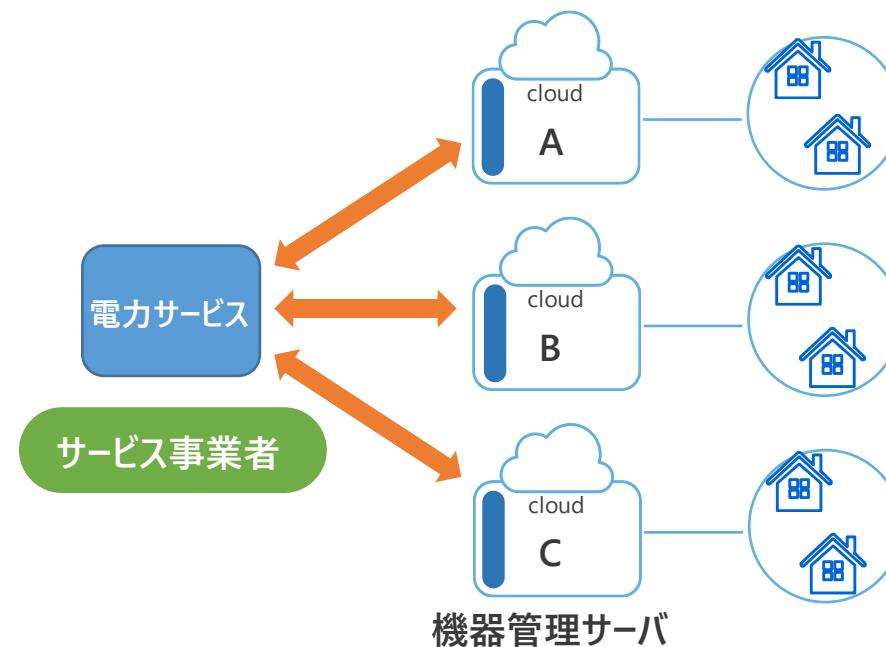
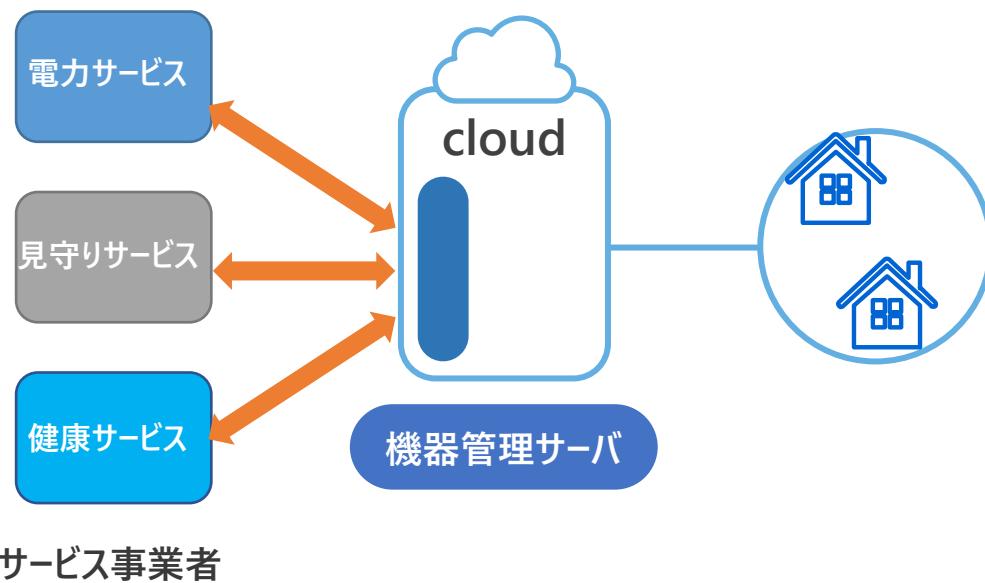
機器管理サーバ視点

機器管理サーバは、統一的な標準EL Web APIを提供することで、様々なサービス・アプリに対して、**統一プログラミングスタイル**に基づく一貫した機器操作・制御モデルが実現可能となり、各種サービス連携が容易となります

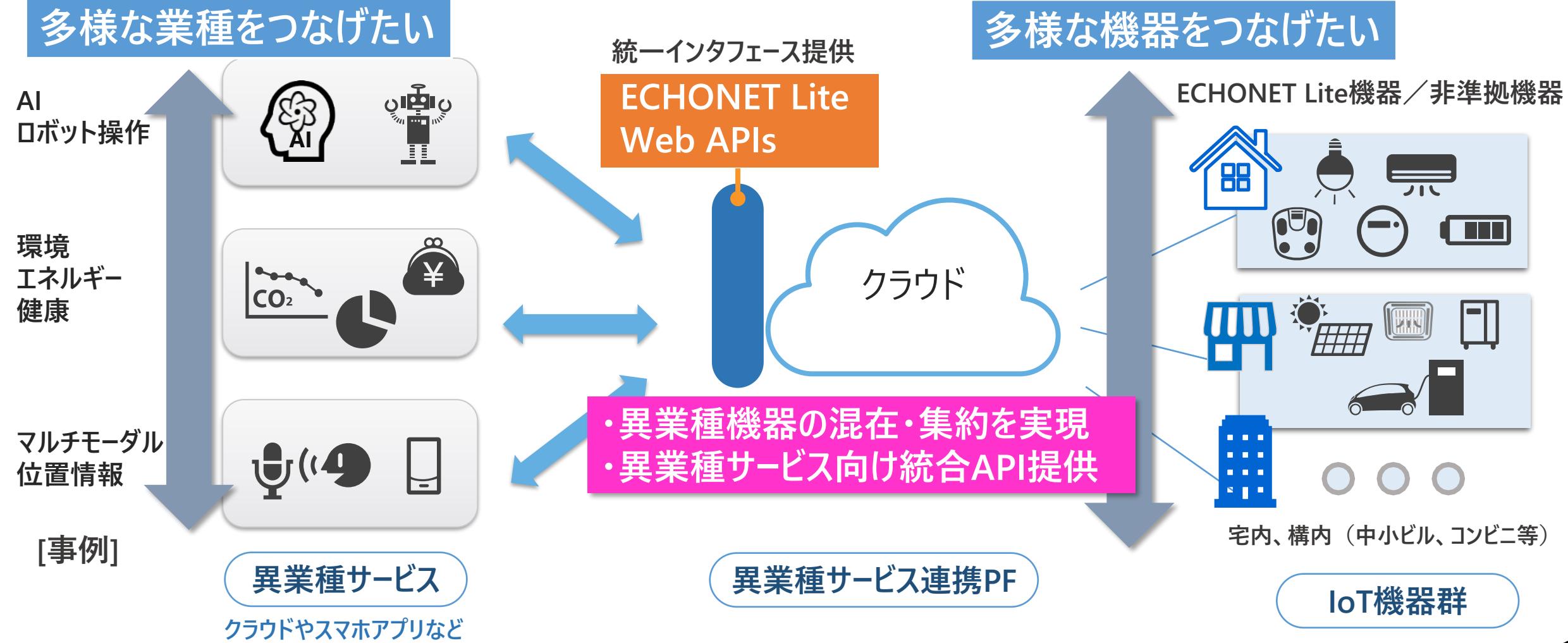
2 複数事業者を束ねる 集約クラウドを実現可能

サービス事業者視点

サービス事業者は、複数の異なるクラウドに対して、同一のWeb APIを用いて、多数機器への操作・制御が可能となります。
クラウドを集約するクラウドも実現可能です。
→より高度なAPIもAPI仕様部で策定・提供中

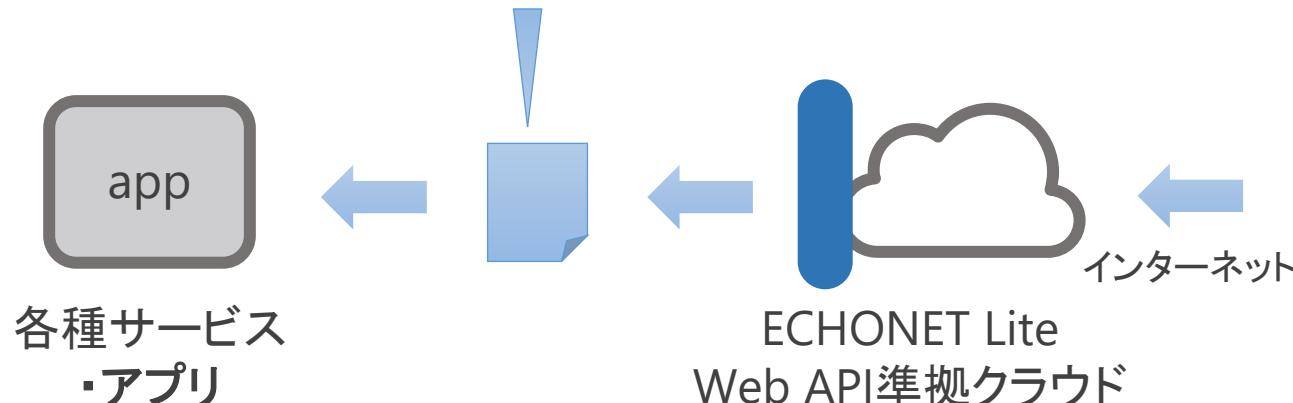


異業種サービスとの連携を促進するIoTクラウドの標準基盤へ

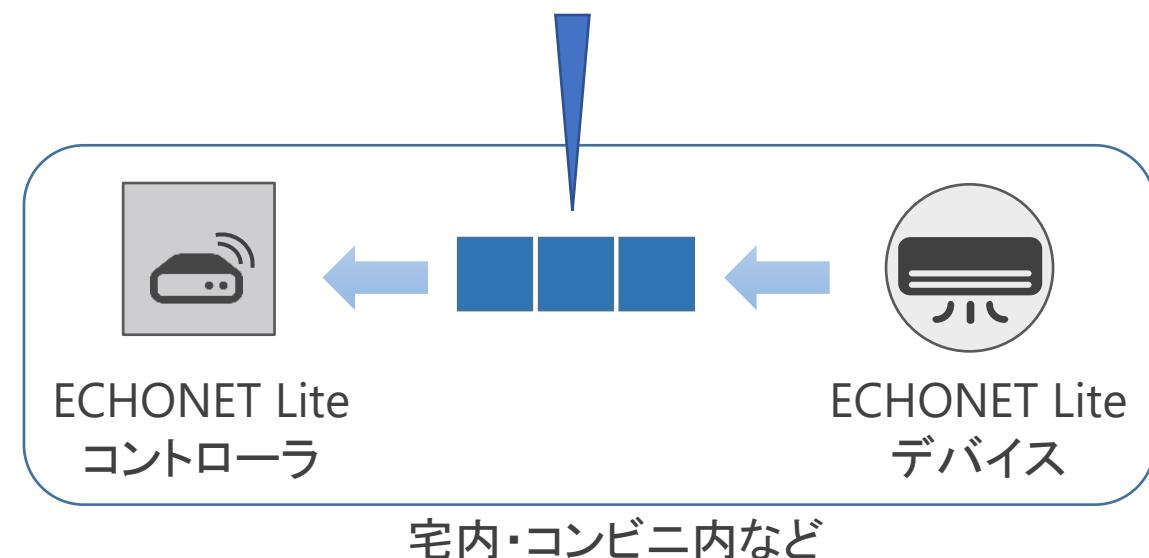


ECHONET LiteコマンドとEL Web APIデータの対応（変換）

■リクエスト：
GET /*/devices/<id>/properties/operationStatus
■レスポンス：
{
 "operationStatus": true
}



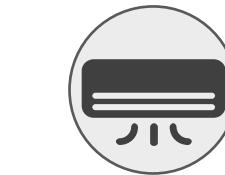
EHD 1 (1B)	EHD 2 (1B)	TID (2B)	SEOJ (3B)	DEOJ (3B)	ESV (1B)	OPC (1B)	EPC1 (1B)	PDC 1 (1B)	EDT 1 (1B)
10	81	81	01300105FF01	72	01	80	01	30	



ELはバイナリ形式、EL Web APIは人間が理解し易い文字列形式

ECHONET Lite Web APIs

```
"deviceType": "homeAirConditioner"  
      + device ID
```

**ECHONET Lite**

ECHONET Lite
家庭用エアコン

EOJ=0x0130**
+ IP addr + portN

リソース指定
/devices/<device id>/properties/**airFlowLevel**
値
{ "airFlowLevel": 5 } or { "airFlowLevel": "auto" }

風量設定

EPC = 0xA0

EDT = 0x35 or 0x41

リソース指定
/devices/<device id>/properties/**operationMode**
値
{ "operationMode": "cooling" }

運転モード設定

EPC = 0xB0

EDT = 0x42

機器オブジェクトやコマンドをWeb APIに即した形式へマッピング

各種指定項目	ECHONET Lite	ECHONET Lite Web API
機器オブジェクト	IP_addr + port_num + EOJ (ECHONET Object) 例： 192.168.0.5 + 3610 + 0x031001	URI, deviceID (システム内ユニークID), deviceType 例： https://xxx/elapi/v1/devices/ <device id> 例：“deviceType”: “generalLighting” (詳細は後述)
コマンド (プロパティ)	EPC (ECHONET Property Code) 例： epc = 0x80 #動作状態	プロパティ・リソース 例： https://xxx/elapi/v1/devices/ <device id>/properties/ operationStatus
アクセスルール (サービス) 要求	ESV (ECHONET Lite Service) 例： esv = Get (0x62) 例： esv = SetC (0x61)	HTTPリクエストのメソッド 例： GET 例： PUT # SET後の値取得含む
アクセスルール (サービス) 応答	ESV + EDT (ECHONET DaTa) 例： esv = Get_Res (0x72), edt = 0x30 #ON 例： esv = Set_Res (0x71)	HTTPレスポンスのステータスコードやボディ 例： HTTP/1.1 200 OK { “operationStatus”: true }

※通知、不可応答のケースは省略

統一的なリソース構成を提供

- ❖ サービス種毎に後続パスが設定可能
- ❖ devicesサービス種などW3CのWoTをベースとしたリソース指定形式を導入

リソース例

Version1

サービス種内
固有ID

<https://webapiechonet.com/elapi/v1/devices/<id>/properties/<property resource name>>

ECHONET Lite
Web API名

サービス種

プロパティ集合

特定プロパティ

機器一覧、機器情報、プロパティ・リソース、などリソース指定

リソース指定例 (versionId=v1)	説明
/elapi/v1/devices	機器一覧
/elapi/v1/devices/<device id>	機器の定義情報 Device Description
/elapi/v1/devices/<device id>/properties	機器の全プロパティ・リソース
/elapi/v1/devices/<device id>/properties/<property resource name>	機器の指定プロパティ・リソース
/elapi/v1/devices/<device id>/actions	機器の全アクション・リソース
/elapi/v1/devices/<device id>/events	機器の全イベント・リソース
/elapi/v1/devices/<device id>/echoCommands	機器へのELコマンド指定 (Opt)

※プロパティ・リソース： プロパティをWeb APIのURIリソースにマッピングしたもの

※アクション・リソース： 操作に時間を要する処理、SETのみ（GET不可）の操作、機器全体のEL規定外の操作

※イベント・リソース： 検討中

リソースパスの詳細化に伴い、機器リソースの詳細部指定が可能

機器一覧

/devices



機器集合

機器情報

/devices/<device id>



全プロパティリソース

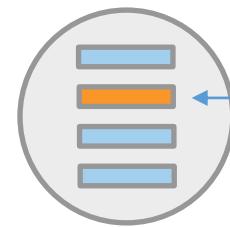
»»» /devices/<device id>/properties



機器内プロパティ集合

指定プロパティリソース

»»» /devices/<device id>/properties/<property resource name>



機器内の
特定プロパティ

アクション実行

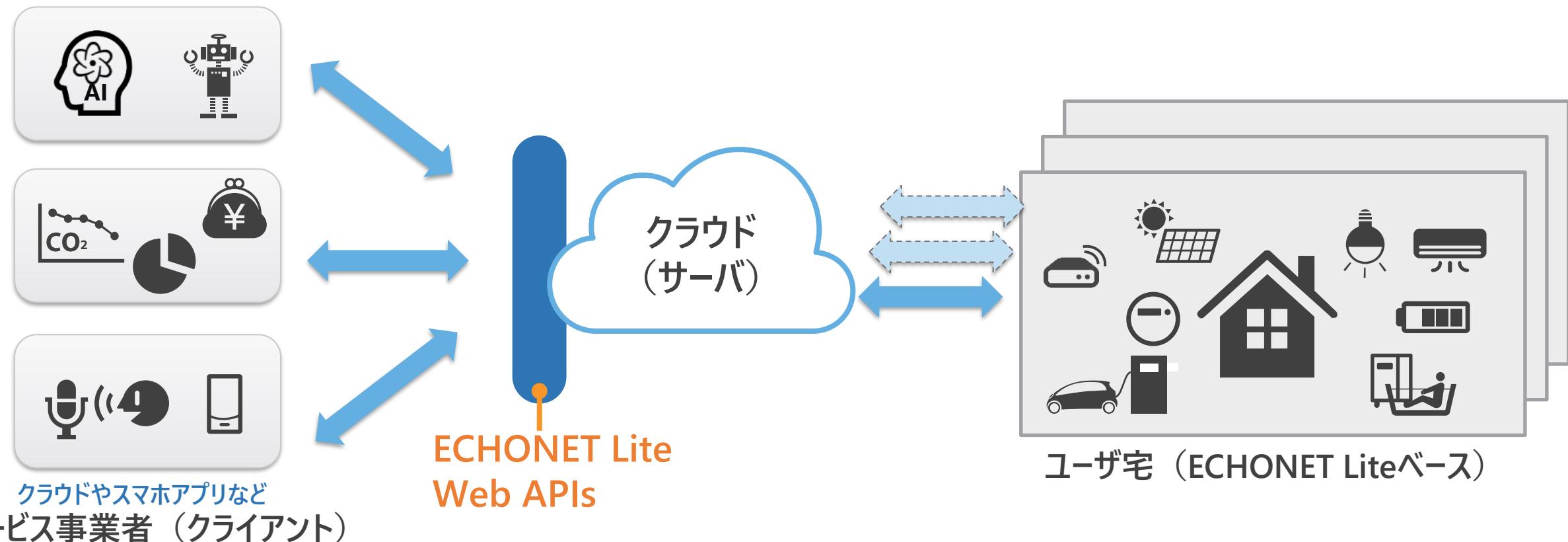
»»» /devices/<device id>/actions/<action name>



イベント処理

»»» /devices/<device id>/events/<event name>

背景 & 要望	IoT技術の向上に伴い、クラウド環境を介したサービス提供モデルが進展 ⇒ ECHONET Lite対応機器をクラウドを介してサービス事業者等へ提供し 提携サービスや応用アプリケーション開発等ビジネス展開へ期待
目的	<u>ECHONET Lite対応機器を利用するクラウドサービスを開発するためのガイドライン</u> （ECHONET Lite Web APIガイドライン）について検討する（WebAPI検討WG：2017年10月設立）。具体的には、「Web APIの仕様書」、「実験用クラウド環境の構築」により、エコーネットおよびその応用分野の更なる普及・拡大を促進



API仕様部

- ・ユースケース
- ・Web APIモデルの指針
- ・EL 仕様のマッピング指針
- ・応用サービス

機器仕様部

- ・データ型定義
- ・ネーミング指針
- ・搭載プロパティ方針
- ・共通項目（SuperClass）
- ・機器毎のDevice Description



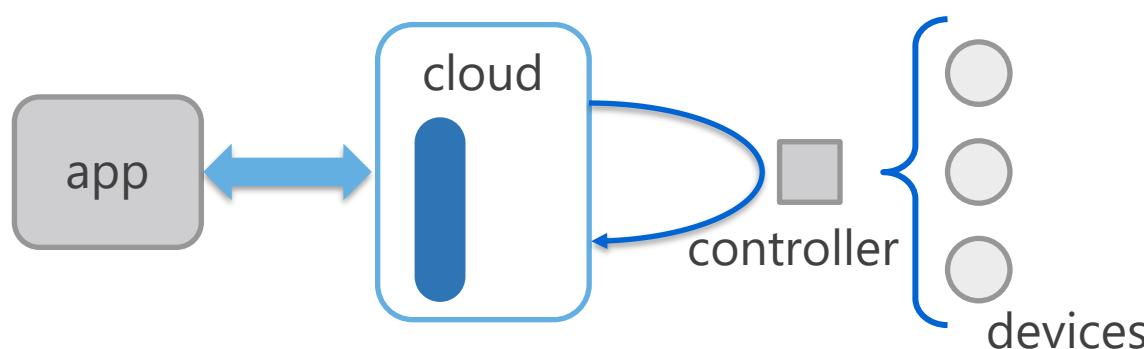
1 機器一覧の取得

EL Web APIを用いて、クラウドから操作・制御できる機器の一覧を取得します。
機器には**デバイスIDと呼ばれる固有ID**が付与されており、このリストが返却されます。

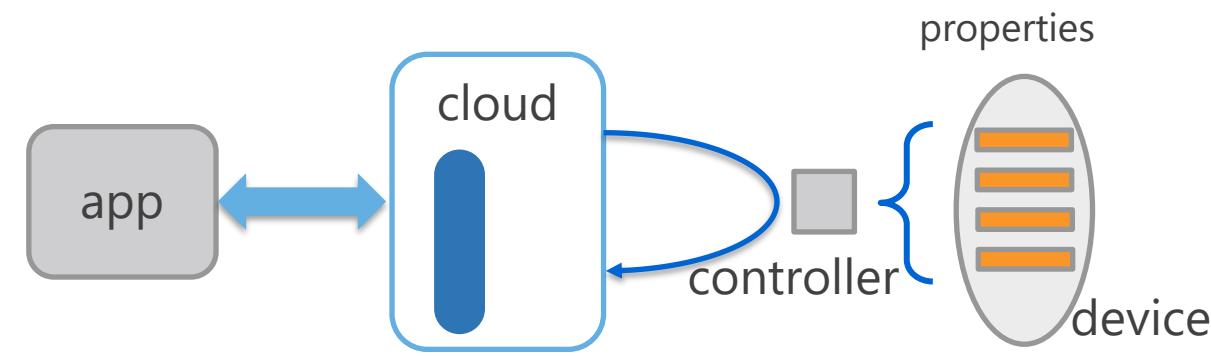
2 クラウドにキャッシュされた機器状態の一括取得

ある機器が保有する各種プロパティ値を一括で取得します。
クラウド上で**直前にキャッシュされた値**が返却されるため、必ずしも最新の値とは限りません。

登録済み機器の一覧をGET



ある機器の状態全てを一度にGET



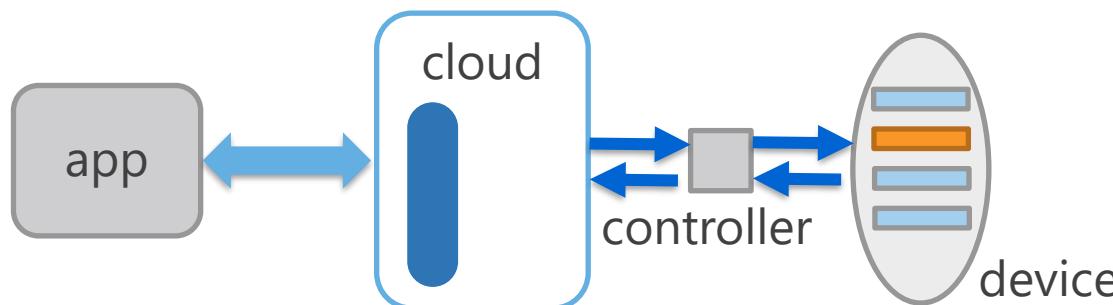
3 機器の監視・制御

EL Web APIを用いて、クラウドから対象機器への**制御(設定)**や**監視(状態取得)**を実施します。具体的には、機器が持つあるプロパティに対してSETまたはGET操作を実施。

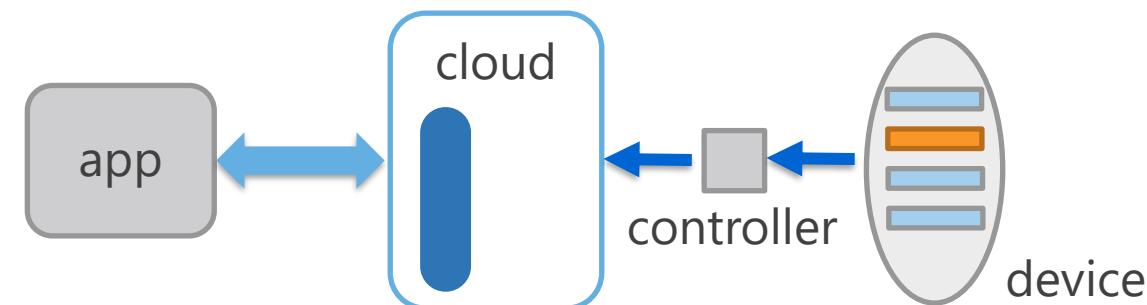
4 機器からの状態通知

ある機器から発信される**通知メッセージ**をクラウドから送信してもらいます。機器の状態変化時や異常状態発生時などのイベントに応じて通知されます。

例) 設定温度の設定／取得



例) 電源On/Offの通知

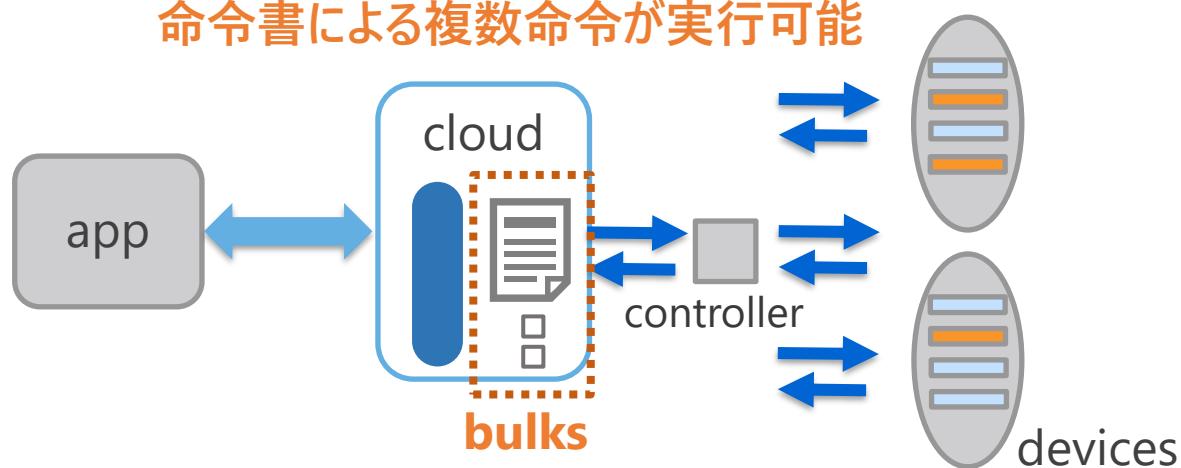


1

複数命令の一括指示

任意の機器、任意のプロパティを対象とした
コマンドを複数列挙した命令セットを作成する
ことができます。この命令セットを用いて、一括
で操作・指示することが可能となります。非同
期・同期実行の指定も可能です。

命令書による複数命令が実行可能

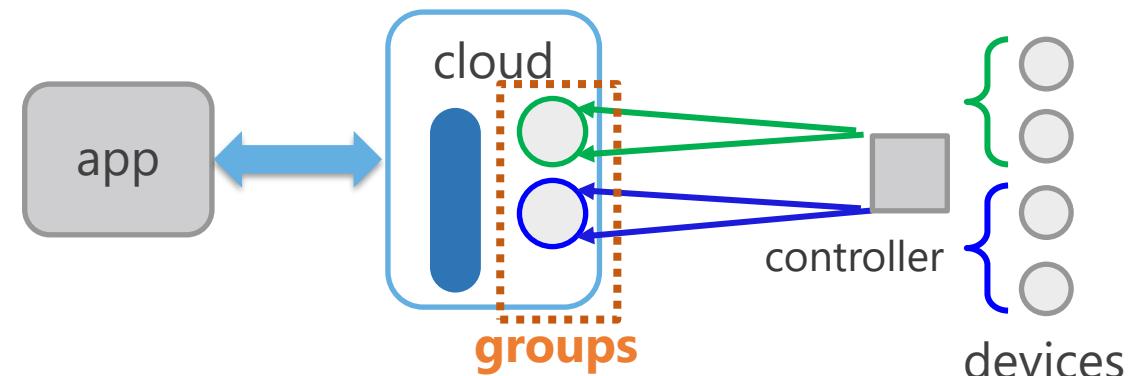


2

機器のグルーピング

複数の機器をまとめてグループ化します。
グループ化した対象は、リソースの分類・整理
や、同種の命令を受け付けるなど、グループ化
により規定される動作も可能となります。仮想
機器化にも利用できます。

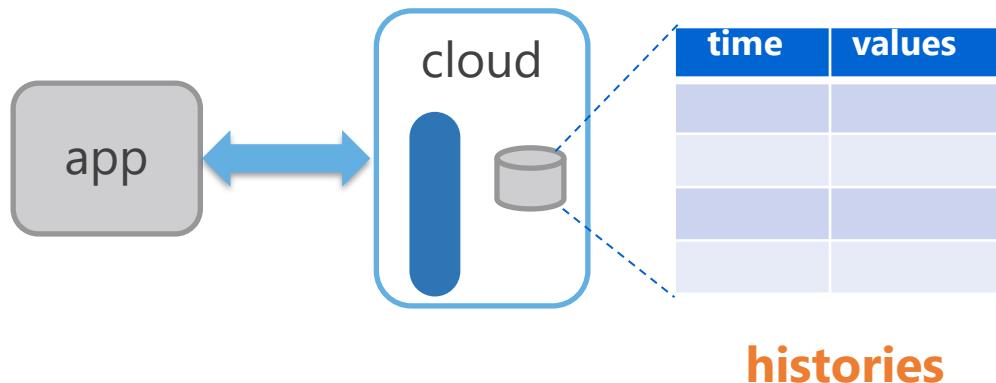
グループ単位での操作が可能



3 履歴データの蓄積・検索

指定した機器に関する取得値を時刻とともに記録し、取得可能にします。取得したい値や時刻の範囲などを指定し、アプリ側でグラフ表示や各種データ加工などに利用できます。

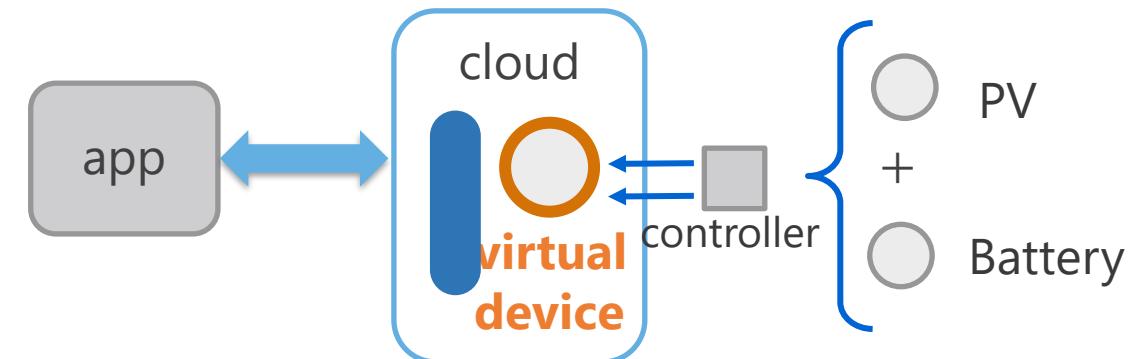
機器の稼働記録・計測値など検索が可能



4 仮想的な機器を定義可能

2種類以上の機器を組み合わせたり、クラウド上で任意の機能を追加することで、あらたな機能を持つ独自機器を定義・操作できます。仕様外の任意の機器やプロパティを定義できます。

複数機器 + 機能を結合した仮想VPP機器など生成できる



Appendix Rel.M対応。今後も継続更新予定

V.1.3.0 時点のサポート機器			
家庭用エアコン(*)	住宅用太陽光発電(*)	拡張照明システム(*)	温度センサ
換気扇	床暖房	冷凍冷蔵庫	電力量センサ
空気清浄機	燃料電池(*)	クッキングヒータ	電流センサ
業務用パッケージエアコン室内機(*)	蓄電池(*)	炊飯器	冷温水熱源機
業務用パッケージエアコン室外機(*)	電気自動車充放電器(*)	業務用ショーケース(*)	電力量メータ
電動雨戸・シャッター	低圧スマート電力メータ(*)	業務用ショーケース室外機(*)	分電盤メータリング
電気温水器(*)	高圧スマート電力メータ(*)	スイッチ (JEM-A/HA端子対応)	空調換気扇
電気錠	一般照明(*)	コントローラ	テレビ
瞬間式給湯器(*)	単機能照明(*)	ハイブリッド給湯機	
浴室暖房乾燥機	電気自動車充電器(*)	洗濯乾燥機	* AIF対応機器

- 会員によるPlaygroundとして提供中：

- API仕様部Version1.1.3、機器仕様部V.1.3.0(Errata)に準拠（2021年10月現在）
- 神奈川工科大学(KAIT)との共同研究にて実施中

- 利用方法：

- 会員ページ>会員トップ>ECHONET Lite Web API実験クラウド（会員限定）より申請可能
- 会員申請者毎にアカウント発行。詳細は下記より「ユーザーマニュアル」参照のこと



直感的に操作できるGUI。ユースケース、機能を可視化

KAIT殿共同開発



ELWebAPIStudy アイコン

①機器一覧

②Device Description

③各プロパティ設定・値

ECHONET Lite Web API 学習用アプリ v1.0.0

Operation

Device

id: FE000077013072FD6E613E4DB64920B304
deviceType: homeAirConditioner
version: Rel.M
manufacturer: 神奈川工科大学

Request & Response

REQ GET
https://webapiechonet.com/elapi/v1/devices/FE000077013072FD6E613E4DB64920B304/properties/targetTemperature

RES status code: 200

```
{
  "targetTemperature": 18
}
```

Device description

property name	description	writable
onTimerReservation	ONタイマ予約設定	true
timeOfOnTimer	ONタイマ時刻設定値	true
relativeTimeOfOnTimer	ONタイマ相対時間設定値	true
offTimerReservation	OFFタイマ予約設定	true
timeOfOffTimer	OFFタイマ時刻設定値	true

Property

プロパティ名	操作 (値設定)	値取得
動作状態	ON OFF	Get value true
運転モード	冷房 暖房 送風	Get value circulation
温度設定値	18°C 設定	Get value 18

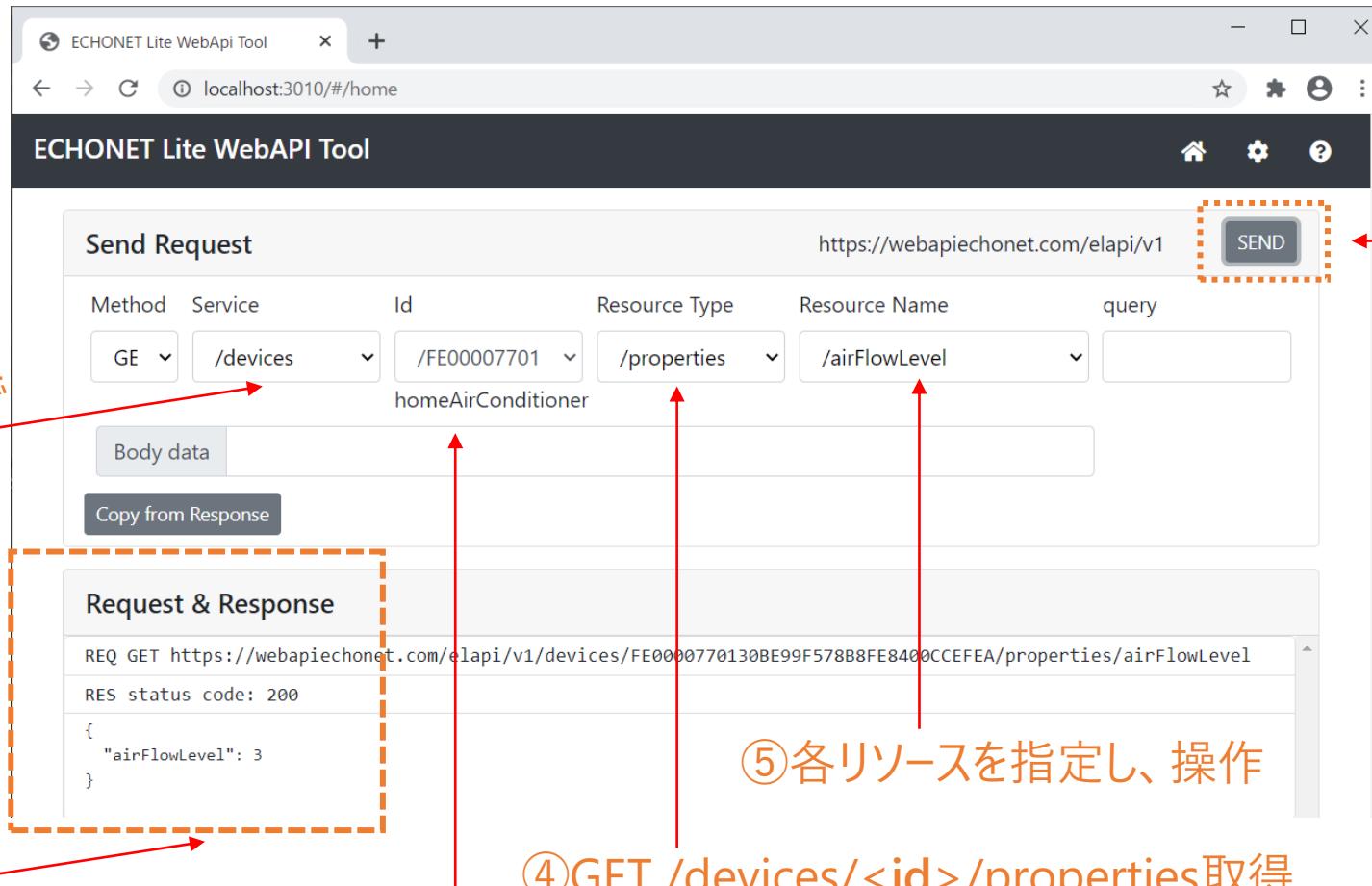
操作ガイド Property の設定や値取得を実行してください

④リクエスト

⑤レスポンス

動的に獲得したデータから、次回送信コマンドの選択・生成を容易に！

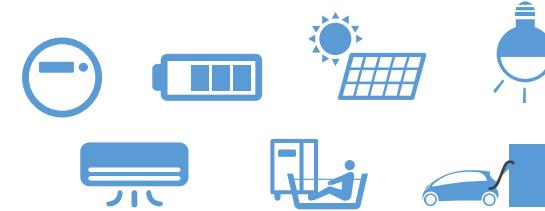
KAIT殿共同開発



- ① <事前に実行>
- ②～⑤リソース設定後、実行

- 実行結果の値を次回選択に活用
- Device Descriptionを利用し動的にリソース選択可能

ECHONET Lite Web API の特徴



既製品(EL)多数に対応

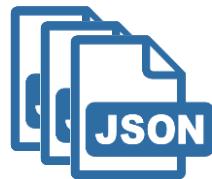
複数のEMS機器含むECHONET Lite製品に対応。全ての機器機能(仕様)をクラウド経由で活用できます



制御 監視 通知

リアルタイム制御・監視

ECHONET Lite機器へのコマンドをクラウド経由で呼び出すAPIを提供。非EL機器も簡単に対応可能です



- Restful API
- 機器用スキーマ (JSON形式)

統一的なWeb APIモデル

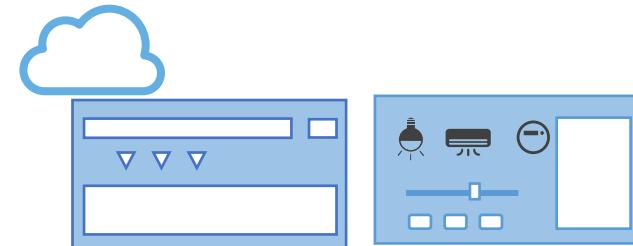
Device Descriptionと呼ぶ機器用スキーマ定義により、機械可読可能な形式でAPI仕様を設計できます



複数命令 グループ化 履歴

応用API機能も充実

複数コマンド一括操作や機器グループ操作、履歴データの検索など高度なAPIを提供します

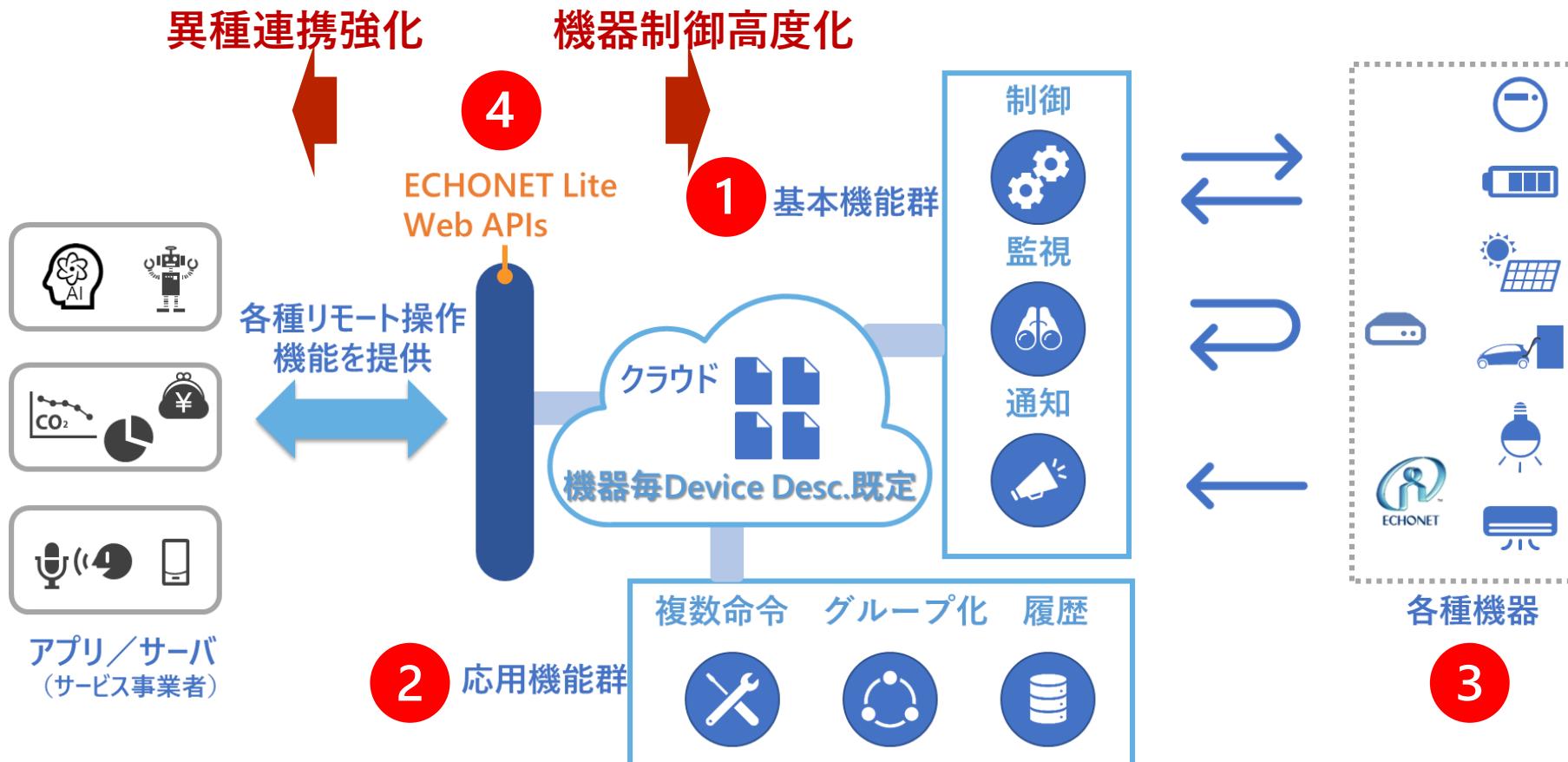


実験クラウド・豊富ツール群

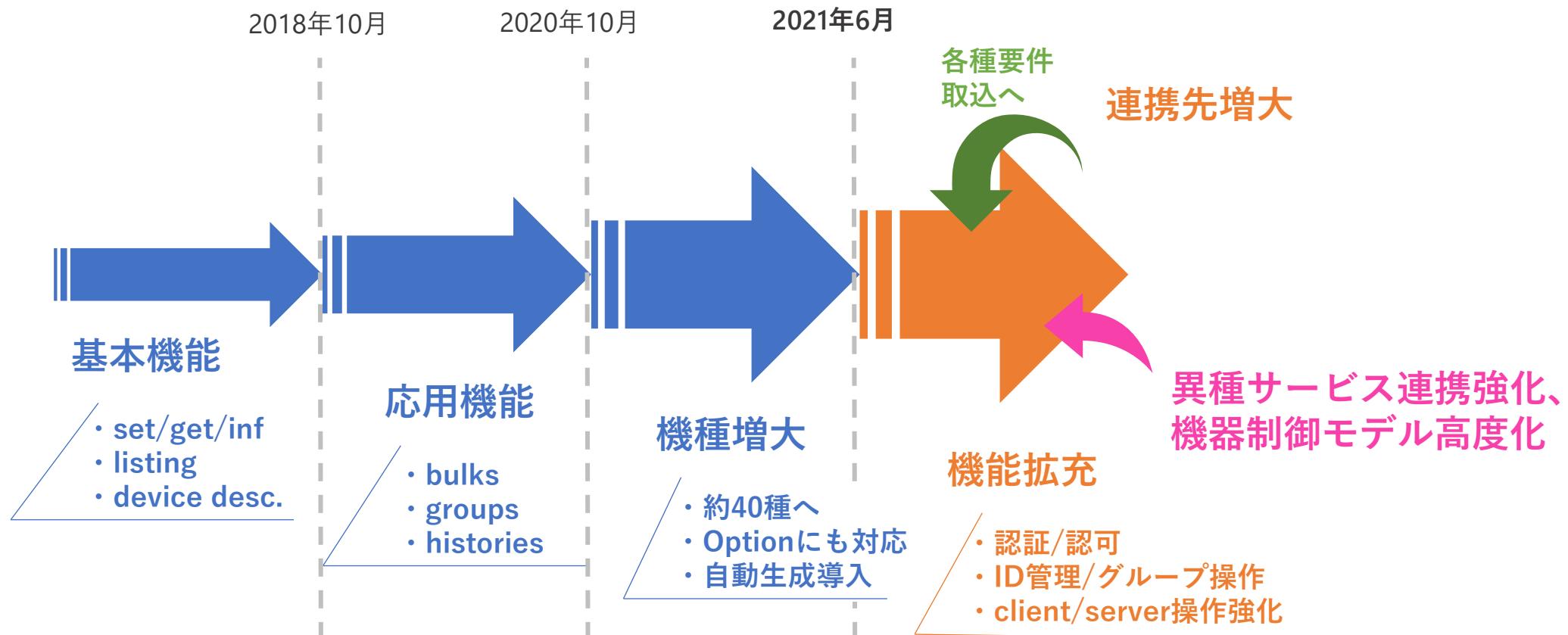
会員向けに実験環境や、開発用・初心者用ツールなど提供。仕様理解が進み、開発の参考となります

従来：①基本機能策定→②応用機能策定→③機器増大

今後：④異種サービス連携用の機能強化、機器制御モデルの高度化を狙う



外部連携による要件取り込みの上、機能拡充・モデルの高度化など狙う



- 2018/10 ガイドラインV.1.00公開
- 2018/12 実験クラウド会員公開
- 2019/11 機器仕様部V.1.1.0公開
- 2020/08 API仕様部V.1.1.0公開
- 2020/09 機器仕様部V.1.2.0公開

基本機能

- 2020/10 API仕様部V.1.1.1公開
- 2020/10 実験クラウド更新版会員公開
- 2020/10 開発者用GUIツール
「ELWebAPITool」会員公開

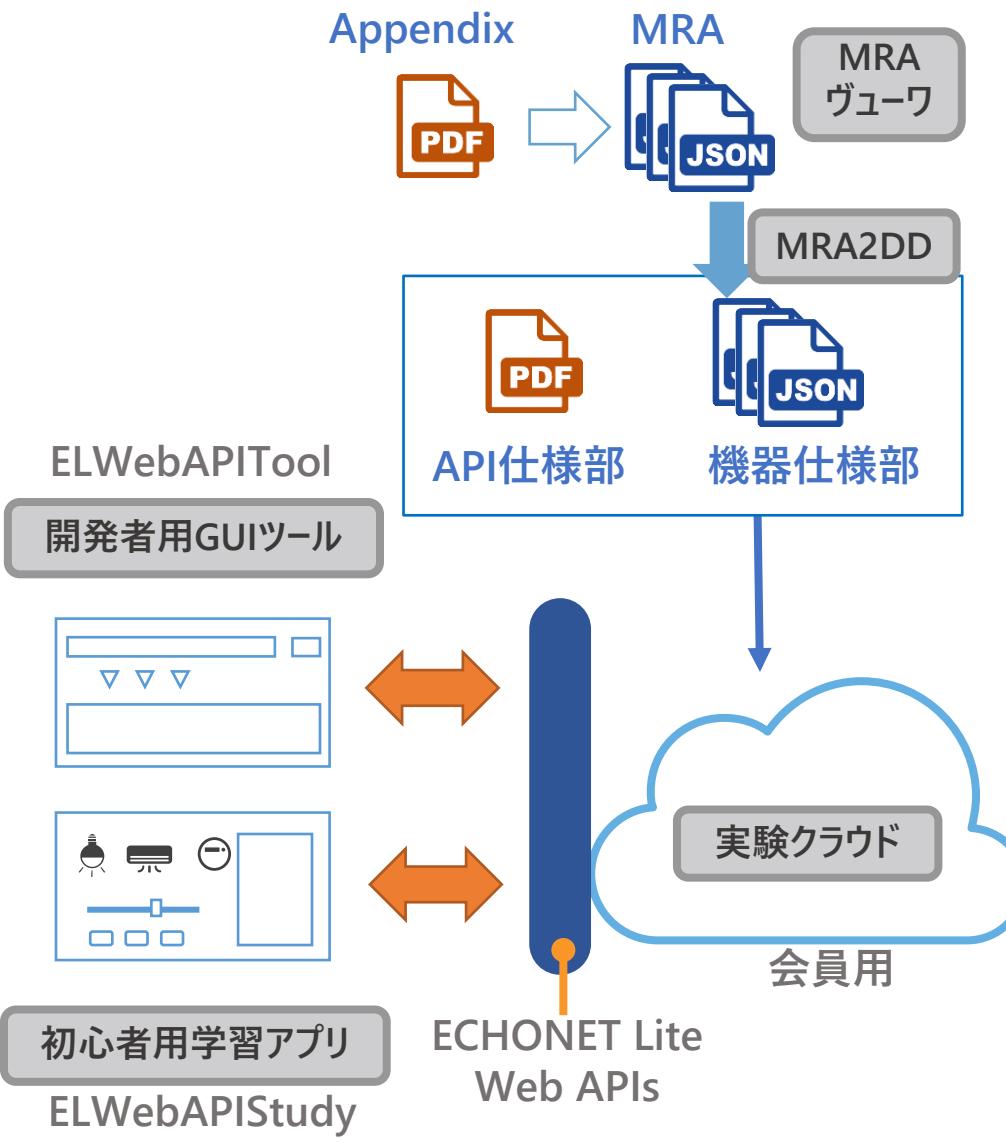
応用機能

- 2021/06 機器仕様部V.1.3.0公開
- 2021/06 API仕様部V.1.1.2公開
- 2021/06 初心者用学習アプリ
「ELWebAPIStudy」会員公開

機器増大

- 2021/07, 08 API仕様部V.1.1.3公開、機器仕様部V.1.3.0 Errata公開
- 2021/08 実験クラウド更新版会員公開
- 2021/11 機械可読機器オブジェクトデータ（MRA）、
MRAビューワ、MRA2DD変換ツール公開

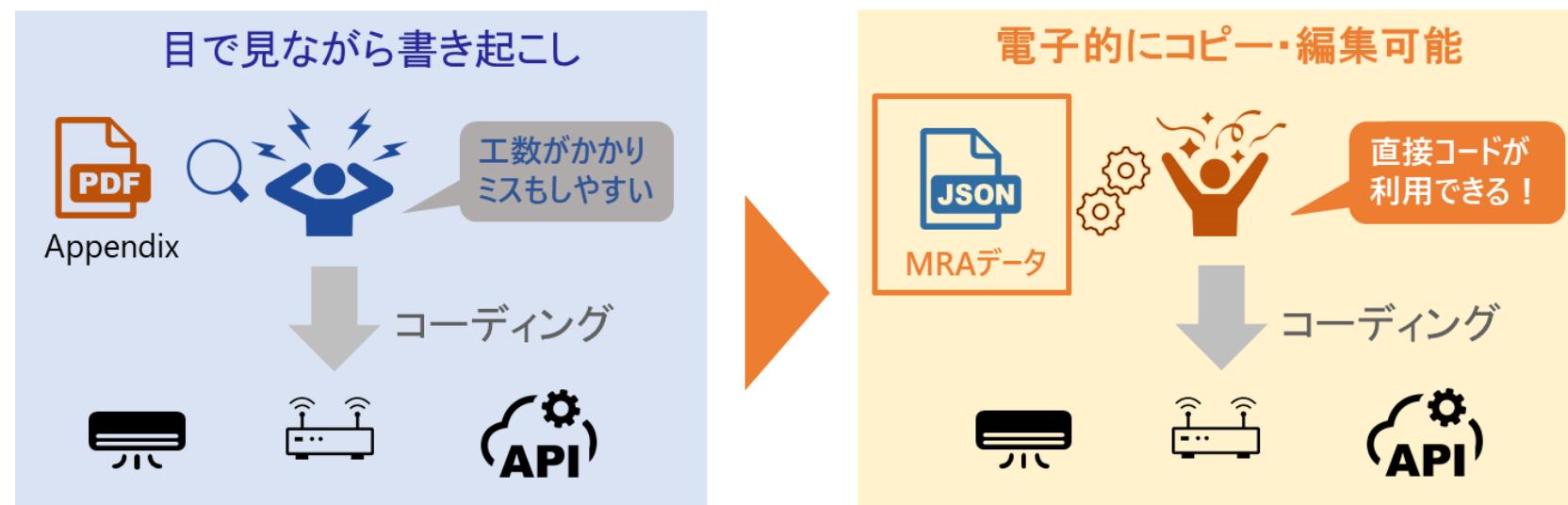
機能拡充





MRA: Appendix 機器オブジェクト詳細規定の内容を、JSON フォーマットで記述したデータファイル

- 機器オブジェクトの仕様を、電子的なプログラム利用などが容易なデータ形式で提供
- 参考データの位置づけにて公開(正式仕様はAppendix本体)
- ECHONET Lite Web APIの「Device Description」形式など多様なデータ形式へ変換可能
- 現状、対象範囲はDevice Description公開機器まで。過去対応バージョンも一部制限有



Appendix (表形式) からMRA (JSON形式) へ

3. 2. 1 家庭用エアコンクラス規定

Appendix

クラスグループコード : 0x01

クラスコード : 0x30

インスタンスコード : 0x01~0x7F (0x00 : 全インスタンス指定コード)

プロパティ名称	EPC	プロパティ内容	データ型	データサイズ	単位	アクセスルール	必須	状変時アナウンス	備考
		値域(10進表記)							
動作状態	0x80	ON/OFFの状態を示す。	unsigned char	1 Byte	—	Set	○	○	
		ON=0x30, OFF=0x31				Get	○		
節電動作設定	0x8F	機器の節電動作を設定し、状態を取得する。	unsigned char	1 Byte	—	Set/Get	○	○	
		節電動作中=0x41 通常動作中=0x42							
運転モード設定	0xB0	自動／冷房／暖房／除湿／送風／その他の運転モードを設定し、設定状態を取得する。	unsigned char	1 Byte	—	Set/Get	○	○	
		順番に以下のコードが対応。 0x41/0x42/0x43/0x44/0x45/0x40							
温度自動設定	0xB1	AUTO／非 AUTO を設定し、設定状態を取得する。	unsigned char	1 Byte	—	Set/Get			
		AUTO=0x41, 非 AUTO=0x42							
急速動作モード設定	0xB2	通常運転／急速／静音を設定し、設定状態を取得する。	unsigned char	1 Byte	—	Set/Get			

https://echonet.jp/spec_mra_rm/ にて一般公開中

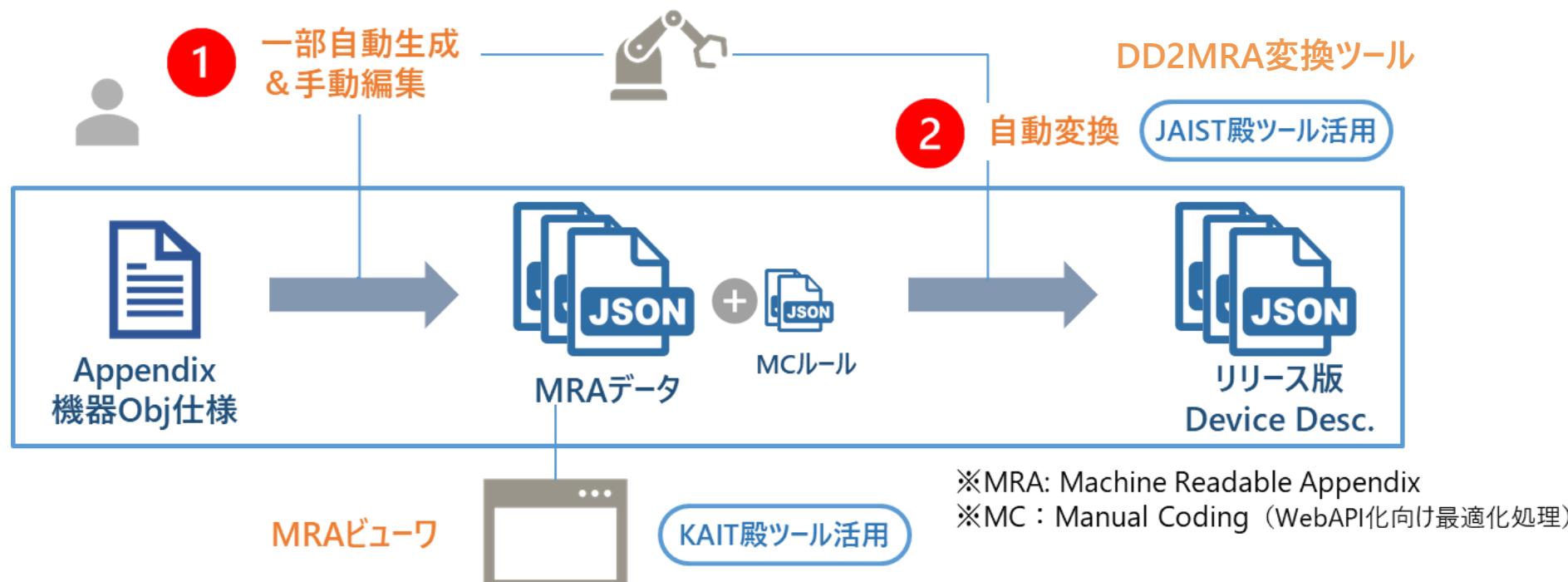
MRA

```
{
    "eoj": "0x0130",
    "validRelease": {
        "from": "A",
        "to": "latest"
    },
    "className": {
        "ja": "家庭用エアコン",
        "en": "Home air conditioner"
    },
    "shortName": "homeAirConditioner",
    "elProperties": [
        {
            "epc": "0x80",
            "validRelease": {
                "from": "A",
                "to": "latest"
            },
            "propertyName": {
                "ja": "動作状態",
                "en": "Operation status"
            },
            "shortName": "operationStatus",
            "accessRule": {
                "get": "required",
                "set": "required",
                "inf": "required"
            },
            "descriptions": {
                "ja": "ON/OFFの状態を示す",
                "en": "This property indicates the ON/OFF status."
            },
            "data": {
                "$ref": "#/definitions/state_ON-OFF_3031"
            }
        }
    ]
}
```



オブジェクト仕様のWeb API版機器仕様化ツール(JAIST殿提供)

- MRAデータから自動変換ツールを用いてDevice Descriptionを自動生成
- 自動変換処理の導入により、エディトリアルなミスなど大幅低減



1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

3 WoTとの連携経緯・連携活動内容

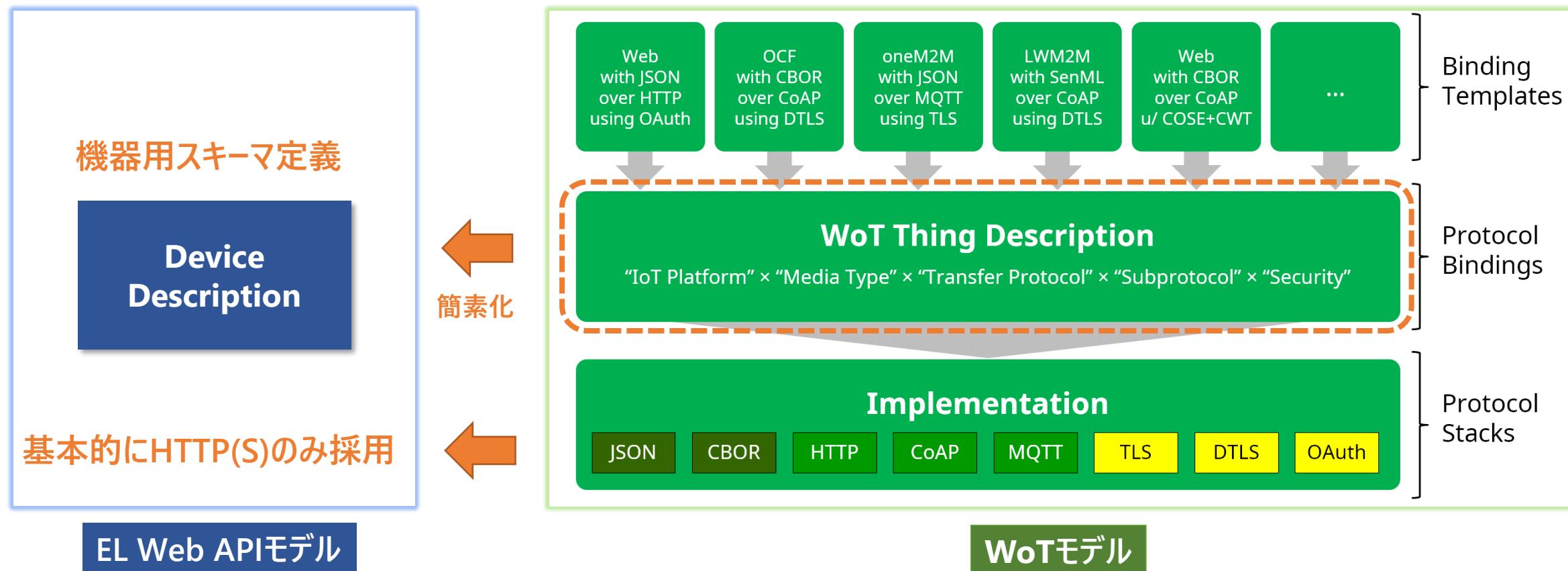
4 WoT とEL Web APIの仕様の違い

5 課題？期待？

Webの標準化団体W3Cで策定中のWoTモデルを参考に定義（W3Cと提携中）

❖ WoTの**Things Description**を簡素化した機器定義用**Device Description**を設計

WoT Binding Templates (<https://www.w3.org/TR/wot-architecture/>)より



日・英
表記

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

ECHONET Lite Device Description
(DD)

Thing Description

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

From
previous
page

ECHONET Lite Device Description
(DD)

Thing Description

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

- **連携目標：**

- WoTとECHONET Lite Web APIとの連携モデル確立（JAIST殿協力）

- **活動成果：**

- 7月：EL Web API Device DescriptionからWoT Thing Descriptionへのマッピング検討
- 9月：当方Web APIとの連携ユースケースをWoTへ提案
- 10月：WoT, EL Web API連携サーバモデルを試作し、WoTプラグフェストにて接続確認
- (3月予定：エコーネットHPにて活動詳細について公開)

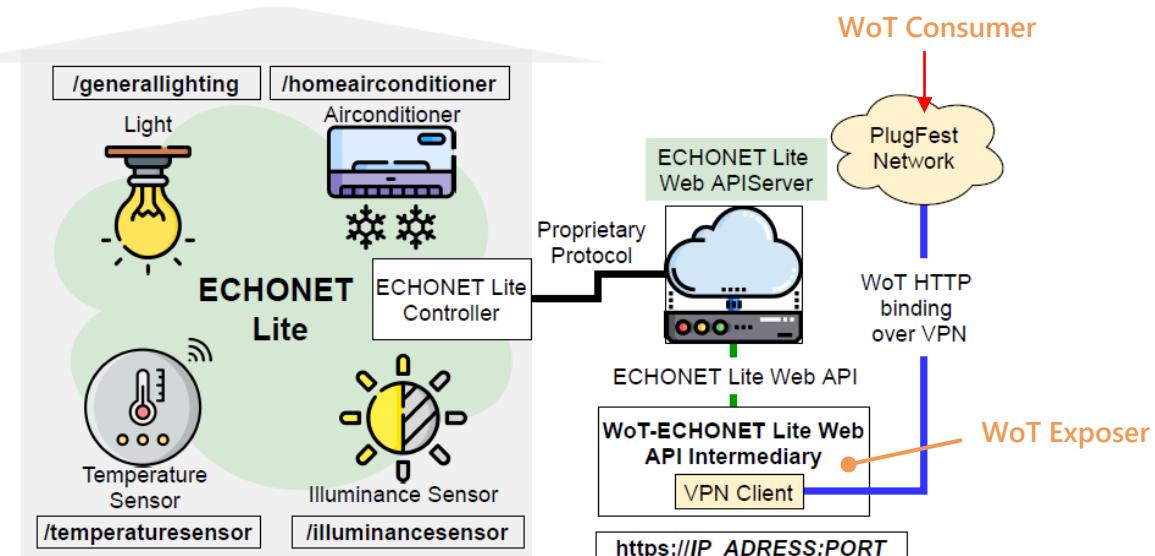
「WoT Sept 2021 Plugfest/Testfest」（9月～10月）

接続相手：日立、富士通、ミュンヘン工科大学

検証結果：上記接続相手WoT Consumer（アプリ）から、当方のWoT Exposer（デバイス）配下の物理デバイスが制御できることを確認

補足：Exposerは、ECHONET Lite Web API対応サーバ機能およびWoTから同APIへの自動変換機能を搭載

Plugfestでのシステム構成



外出・帰宅時の室内機器制御例を提案

❖ Title: Leaving and Coming Home

❖ Target Users:

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

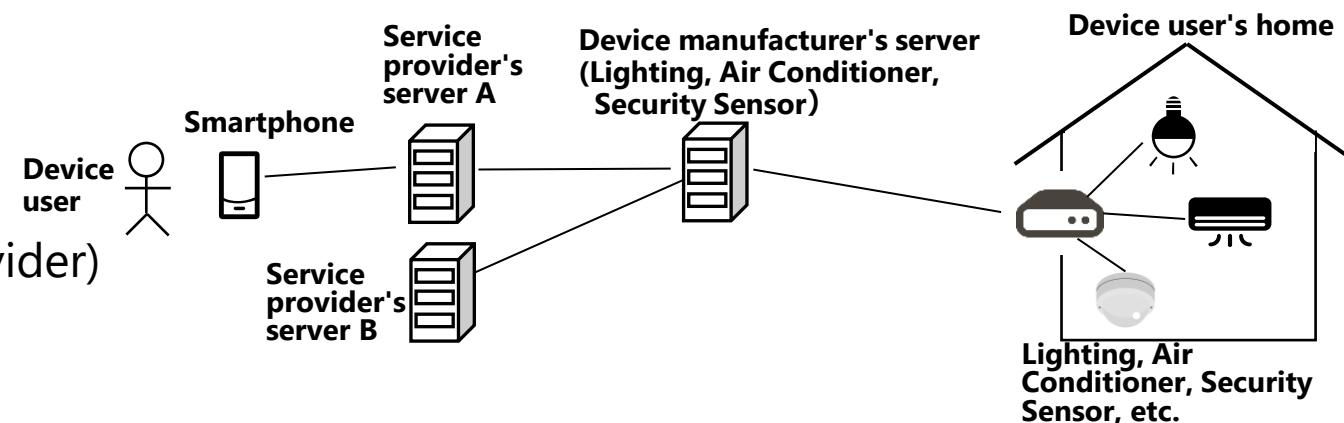


Figure: example system structure

❖ 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.

1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

3 WoTとの連携経緯・連携活動内容

4 WoT とEL Web APIの仕様の違い

5 課題？期待？

状態取得応答形式の違い：EL Web APIはembedded (nested) JSON形式が基本

例) 照明のカラー灯モード時RGB設定取得を行う場合

GET .../properties/rgb HTTP/1.1

WoTの応答形式

```
{  
    "r": 29,  
    "g": 255,  
    "b": 0  
}
```

例) 照明の明るさ取得を行う場合

GET .../properties/brightness HTTP/1.1

WoTの応答形式

60

EL Web APIの応答形式

```
{  
    "rgb": {  
        "r": 29,  
        "g": 255,  
        "b": 0  
    }  
}
```

EL Web APIの応答形式

```
{  
    "brightness": 60  
}
```

制御要求・応答形式の違い：EL Web APIはembedded (nested) JSON形式が基本

例) 照明の明るさ設定を行う場合

WoTの要求形式

PUT /light/properties/brightness HTTP/1.1
80

EL Web APIの要求形式

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

WoTの応答形式

- Empty HTTP body

HTTP/1.1 200 OK

EL Web APIの応答形式

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

※設定後、再度値を取得した結果を返却
(実機動作の反映を確認するため。遅延も考慮)

1 ECHONET Liteとは？

2 ECHONET Lite Web APIとは？

3 WoTとの連携経緯・連携活動内容

4 WoT とEL Web APIの仕様の違い

5 課題？期待？

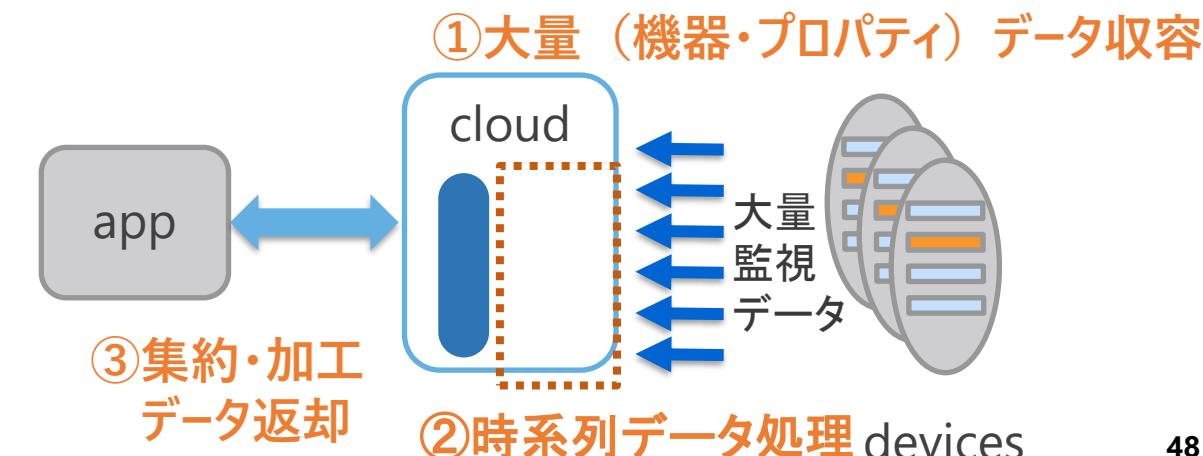
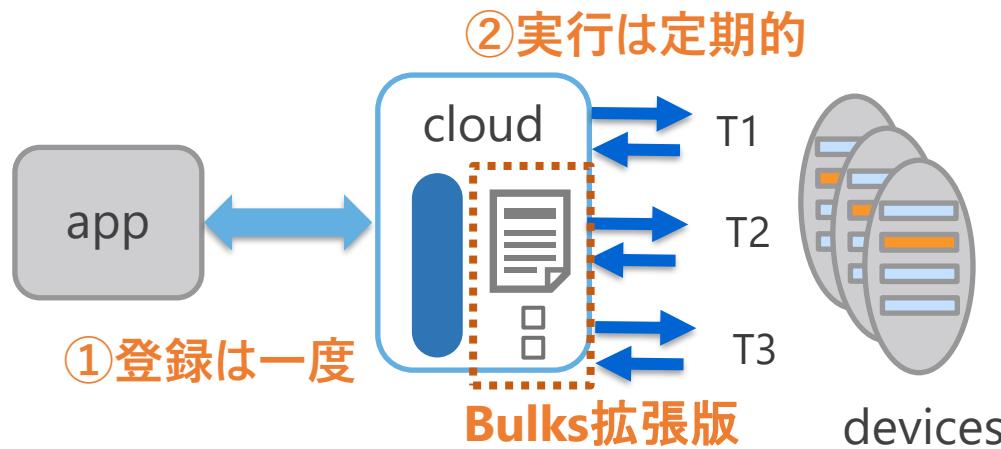
抽象度の高いAPIの策定・活用へ期待あり

❖ 複数命令セットの一括実行

- パターン化された複数命令セットをサーバに登録して実行したい
 - 一度登録した命令セットは、再利用可能したい（EL Web APIのbulksで実現済）
 - 一定間隔で定期的に実行したい

❖ 集約・加工データ処理

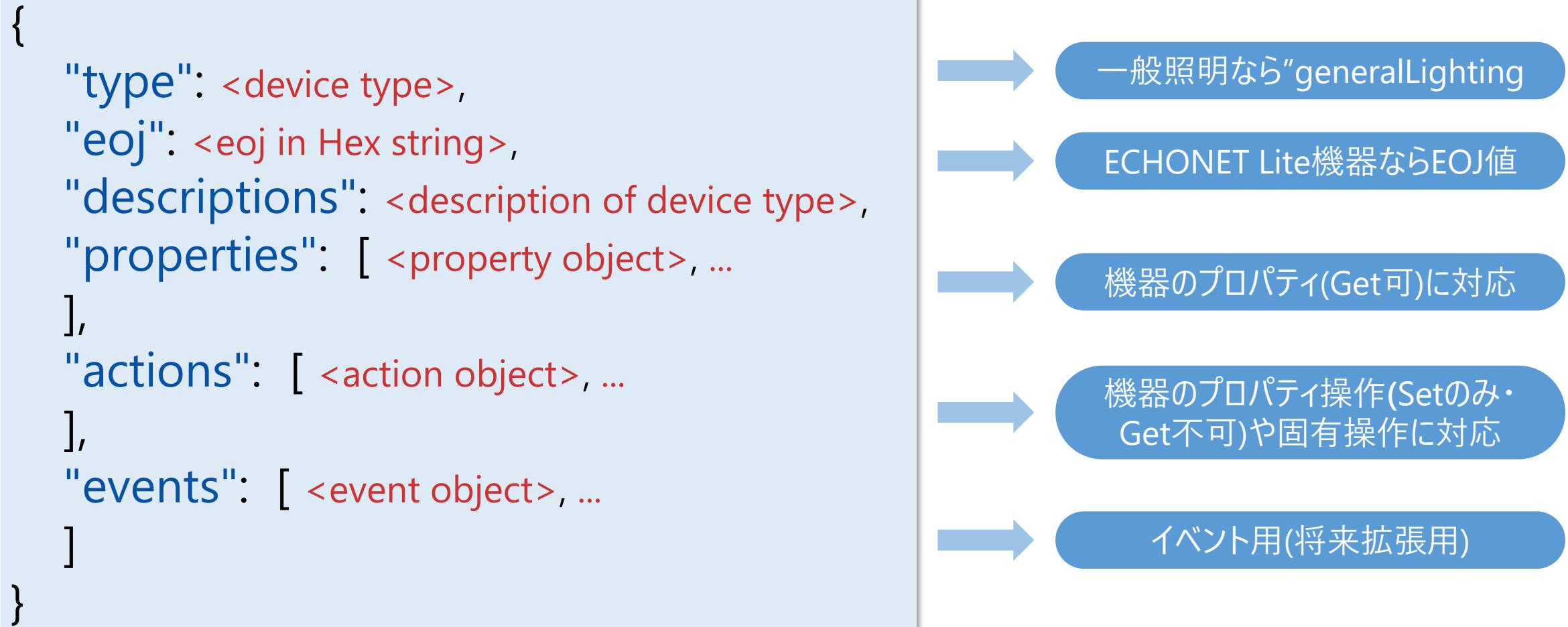
- 大量デバイス、大量データを集約して、必要なデータに加工（絞り込んで）クライアントへ返却したい



- ❖ **ECHONET Lite, ECHONET Lite Web APIの概要について解説**
- ❖ **EL Web APIはWoTモデルを参考に仕様化。独自に応用APIを規定**
- ❖ **WoTとEL Web APIのコラボ状況、技術差異について紹介**
- ❖ **単純なAPIから、より高度なAPIの対応について期待**

以下、付録

機器仕様に相当。GET /elapi/v1/devices/<device id> にて取得可能



Property Resource Name	Access Method	Data Type	EPC(EL)	プロパティ名(EL)
brightness	GET, PUT	number	0x80	照度レベル設定, Illuminance level
operationMode	GET, PUT	enum	0xB6	点灯モード設定, Lighting mode setting
rgb	GET, PUT	object	0xC0	カラー灯モード時RGB設定, RGB setting for color lighting

```
{
    "type": "generalLighting",
    "eoj": "0x0290",
    "descriptions": {"ja": "一般照明", "en": "General Lighting"},
    "properties": [
        {
            "name": "brightness", "epc": "0xB0",
            "descriptions": { "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",
            "descriptions": { "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" }
            ] }
        }
    ], <以下、省略>
}
```

各プロパティリソース用定義を記述

brightness用スキーマ

部屋の明るさを設定できる

operationMode用スキーマ

照明の点灯モード
(自動、常夜灯など)
を設定できる

後続ページで解説①

後続ページで解説②

```
{ "name": "brightness", "epc": "0xB0",
  "descriptions": { "ja": "照度レベル設定", "en": "Illuminance level" },
  "writable": true, "observable": false,
  "schema": { "type": "number", "unit": "%", "minimum": 0, "maximum": 100 }
},
```

❖ 解説：

- “name”より、エンドポイント（リソース）は下記に相当。properties配下のためGET操作対象
 - <https://webapiechonet.com/elapi/v1/devices/<id>/properties/brightness>
- “writable”がtrue、“observable”がfalseのため、
 - SET操作対象、状態変化通知は未対応を示す
- “schema”より、データ型は数字、単位は%で最小値0から最大値100までの値（multipleOf省略のため、刻み幅は1）。よって、リクエスト時（Set）またはレスポンス時（Get/Set）でのデータ値の例は下記の通り（照度23%を指定）：

```
{
  "brightness": 23
}
```
- 例えば、刻み幅multipleを10とすれば、10%の倍数値のみ指定可能となる（上記の値(23)をSetする場合エラーとなる）

```
{ "name": "operationMode", "epc": "0xB6",
  "descriptions": {"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"} ]}
```

❖ 解説：

- “name”より、エンドポイント（リソース）は下記に相当。properties配下のためGET操作対象
 - <https://webapiechonet.com/elapi/v1/devices/<id>/properties/operationMode>
- “writable”がtrue、“observable”がtrueのため、
 - SET操作対象、状態変化通知は対応を示す
- “schema”より、データ型は文字列、列挙型で“auto”以降の4種から選択となる。“values”では各値の説明（日本語、英語）とECHONET Liteでのデータ値が示されている。よって、リクエスト時（Set）またはレスポンス時（Get/Set）でのデータ値の例は下記の通り（点灯モードとして自動を指定）：

```
{
  "operationMode": "auto"
}
```
- 上記例では、点灯モードが4種列挙されているが、実機の搭載モードに合わせてカスタマイズ可能

機器オブジェクトにおけるプロパティを扱うためのデータ型、キーワード

Data Type of JSON	Description	Keyword
boolean	true / false	values
string	ASCII data	
string	列挙型	enum
string	日時	format: date-time
string	年月日	format: date
string	時刻	format: time
number	数値	unit, minimum, maximum, multipleOf
null		
array	同一data typeの要素の配列	minItems, maxItems, items
object	オブジェクト型	properties

EL Web APIではDevice Descriptionにより入手可能

Appendix

スーパークラスのプロパティマップ
を活用 (Set, Get, Inf別)。
EPCを指定して個別にGET

Web API機器仕様部

機器毎のDevice Descriptionを
活用 (Set+Get+Inf一括)。
GET /devices/<id> で取得

Setプロパティマップ
(EPC=0x9E)

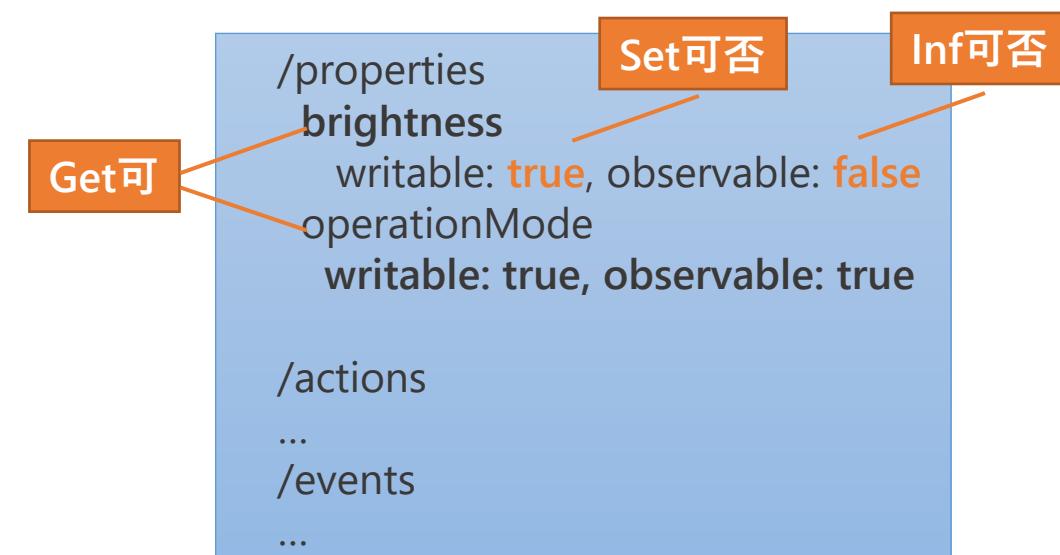
0x80 | 0x88 | 0x8A |

Getプロパティマップ
(EPC=0x9F)

0x80 | 0x8F | 0xB3 |

状変アナウンス
プロパティマップ
(EPC=0x9D)

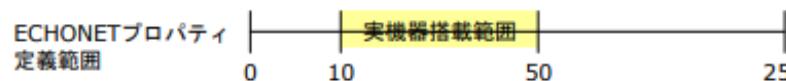
0x80 | 0x88 | 0x8F |



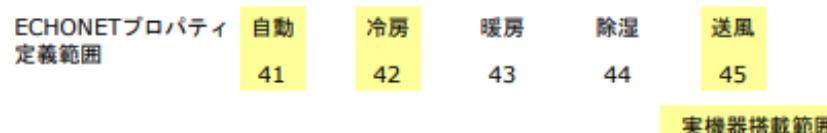
Device Description内スキーマにて事前定義可能

Appendix

値の搭載範囲（連続値、離散値）
は事前に実機仕様を入手するか実
機でのキャリブレーションが必要



- 0～10を設定された場合：プロパティ値を10とすることを推奨する。
- 10～50を設定された場合：プロパティ値は設定値とする。
- 50～253を設定された場合：プロパティ値は50とすることを推奨する。



- 41,42,45を設定された場合：プロパティ値は設定値とする。
- 43,44を設定された場合：設定した値を無視し、プロパティ値はそのままとする。

Web API機器仕様部

Device Descriptionにて事前定義可
(個別にカスタマイズ可能)。
標準値指定値（ままの使用）も可能

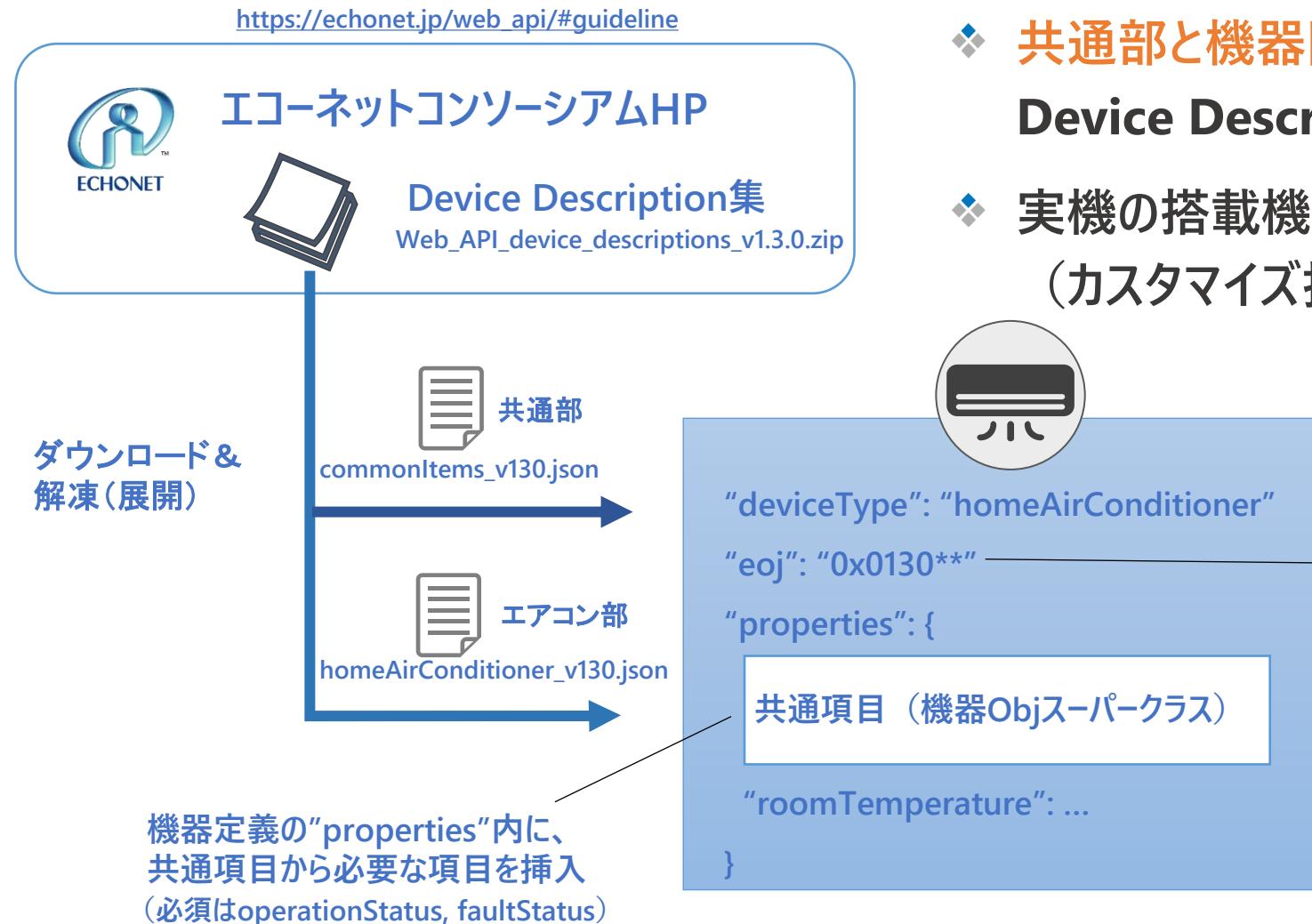
対応範囲を指定可能
刻み幅(multipleOf)も指定可

```

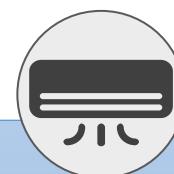
"properties": [
    {
        "name": "brightness", "epc": "0xB0",
        "writable": true, "observable": false,
        "schema": {"type": "number", "unit": "%", "minimum": 0, "maximum": 100}
    },
    {
        "name": "operationMode", "epc": "0xB6",
        "writable": true, "observable": true,
        "schema": {"type": "string", "enum": ["auto", "normal", "night", "color"]},
        "values": [
            {"value": "auto", "ja": "自動"}, {"value": "normal", "ja": "通常"}, {"value": "night", "ja": "常夜灯"}, {"value": "color", "ja": "カラー灯"}]
    }
]
    
```

対応する値のみ列举すれば良い

コンソーシアムHP掲載のDevice Description集（アーカイブ）が活用可能



- ❖ 共通部と機器固有部を組み合わせて対象機器の Device Descriptionを作成
- ❖ 実機の搭載機能に応じてカスタマイズ可能
(カスタマイズ指針は仕様書を参考に)



“deviceType”: “homeAirConditioner”
“eoj”: “0x0130**”
“properties”: {
 共通項目（機器Objスーパークラス）
 “roomTemperature”: ...
}

* * にインスタンス番号を記入

エアコンのDevice Description