Technology Reports
AIエージェント　デバイスWebAPI　IoT
新AIエージェント特集

さまざまなIoTデバイスをクラウドから制御可能とするIoTアクセス制御エンジン

サービスイノベーション部　吉川 貴　山添 隆文　山下 顕　田中 祐貴　堀口 賞一

さまざまなIoTデバイスが市販されているが，それらの相互接続性は必ずしも高いとは言えない状況が続いている．ドコモでは，それらを接続しやすくするための技術として「デバイスWebAPI」，またそれをクラウドから扱えるようにした「IoTアクセス制御エンジン」を開発した．本稿ではそれらの開発に至る経緯，そして各技術の特徴やサービス事例について解説する．

1. まえがき

近年，モノのインターネット（IoT：Internet of Things）という言葉をさまざまな場面で目にするようになった．

Mark WeiserがUbiquitous Computingを提唱［1］してから約30年．その間，Pervasive Computing［2］，Wearable Computing［3］，Mobile Computing，はたまたZen Computingと呼び名は違えど，「コンピュータが至る所に遍在して溶け込むことにより，人々の生活を豊かにする」というコンセプトは研究者達の根底に脈々と受け継がれてきた．

IoTを取り巻く技術課題には，データ量，通信網，セキュリティ，データ解析技術，コストなど，さまざまなものが挙げられる［4］が，ドコモでは特に相互接続性の課題に着目し，研究開発を進めてきた．

IoTデバイスを用いてアプリケーションやサービスを構築するためには，各デバイスに応じた開発を行う必要がある．デバイスメーカによってはこのようなニーズに応えるため，ソフトウェアライブラリや開発キット（SDK：Software Development Kit[*1]）を配布し，開発のハードルを下げる努力をしている．

*1　SDK：ソフトウェア開発をするために必要な技術文書やツールなどの一式のこと．

しかし，デバイス固有の実装や技術習得が必要であることは変わらず，開発者の負担が大幅に軽減されることは少ない．また，ライブラリやSDKもすべてのOSや開発環境に対応することは困難であり，その意味でも課題がある．さらに，デバイスやメーカごとに実装手法が異なることから，一度開発したアプリケーションやサービスを別のデバイスに対応させることは難しく，再開発を伴うケースが多い．

これらはすべてIoTデバイスの相互接続性に関する課題であり，IoTのアプリケーションやサービスが爆発的に普及することを妨げる1つの要因となっている．そこでドコモは，さまざまなIoTデバイスを機能レベルで抽象化し，共通のRESTful[*2] WebAPI[*3]でアクセス可能とする「デバイスWebAPI」，およびIoT機器を統一的に扱うことができるプラットフォーム（クラウド基盤）「IoTアクセス制御エンジン」を開発した．

本稿ではデバイスWebAPI，およびIoTアクセス制御エンジンの技術的特徴について解説するとともに，AIエージェント基盤における活用例を述べる．

2. デバイスWebAPIとIoTアクセス制御エンジン

2.1 概　要

IoTの価値の1つは，今まで見えていなかったデータが可視化され，解析によって新たな価値が発見され，解決につなげられる点である．すなわち，IoTでは各種機器やデータとの連携が真価であり，これらを統一的に扱うことがIoTサービス普及の上で欠かせない．

しかし現状は，IoT機器に関する規格は国内外においてさまざまなものが存在しており，メーカ独自仕様のIoT機器も数多く存在する．このような状況のため，IoTサービス開発者はそれぞれの機器仕様について把握した上で，それぞれの機器向けのソースコードを組み立てていく必要があり，IoTサービス普及の上での大きな障壁となっている．そこで，ドコモでは「デバイスWebAPI」および「IoTアクセス制御エンジン」を開発した．

デバイスWebAPIとは，①RESTful WebAPIによるデバイスアクセス手段の共通化，②機能レベルでのデバイスの抽象化，③プラグインアーキテクチャによる高い汎用性／拡張性，の3点を特徴とするインタフェース抽象化技術であり，IoTの相互接続性の課題を解決することができる［5］．なお，本技術はOMA（Open Mobile Alliance）[*4]にてGotAPIとして標準化されている［6］．

また，IoTアクセス制御エンジンはデバイスWebAPIを遠隔地から利用可能なクラウド基盤であり，デバイスWebAPIの技術的特徴に加え，④遠隔からのIoTデバイスの一元管理，⑤多様なパーミッション[*5]管理機能を備える．

さらにIoTアクセス制御エンジンはAIエージェント基盤の中の1エンジンとして他のエンジンと連係動作することにより，多目的対話エンジンや行動先読みエンジンからIoTデバイスを遠隔制御したり，センサ情報を収集したりすることも可能となる．

IoTアクセス制御エンジンを利用したシステムの構成図を**図1**に示す．IoTサービスアプリケーションは，IoTアクセス制御エンジンにて規定されたAPI（Application Programming Interface）にアクセスすることで，ユーザ認証や各種IoT機器の制御，蓄積されたデータの参照などを統一的に行うことができる．

2.2 IoTアクセス制御エンジンのAPI

IoTアクセス制御エンジンでは現在，下記3種類のAPIを用意している．

①Management API：ユーザ認証・認可を行う

＊2　RESTful：提供される情報を直接指し示してステートレスに情報を取得／提供する考え方．
＊3　WebAPI：HTTPベースのAPI．
＊4　OMA：移動通信向けのサービス，アプリケーション実現技術の標準化および相互接続性の確保を目的とした業界標準化団体．
＊5　パーミッション：システムに対するアクセス権限のこと．本稿ではIoTデバイスにアクセスするためのAPIに設定されるアクセス権限を指す．

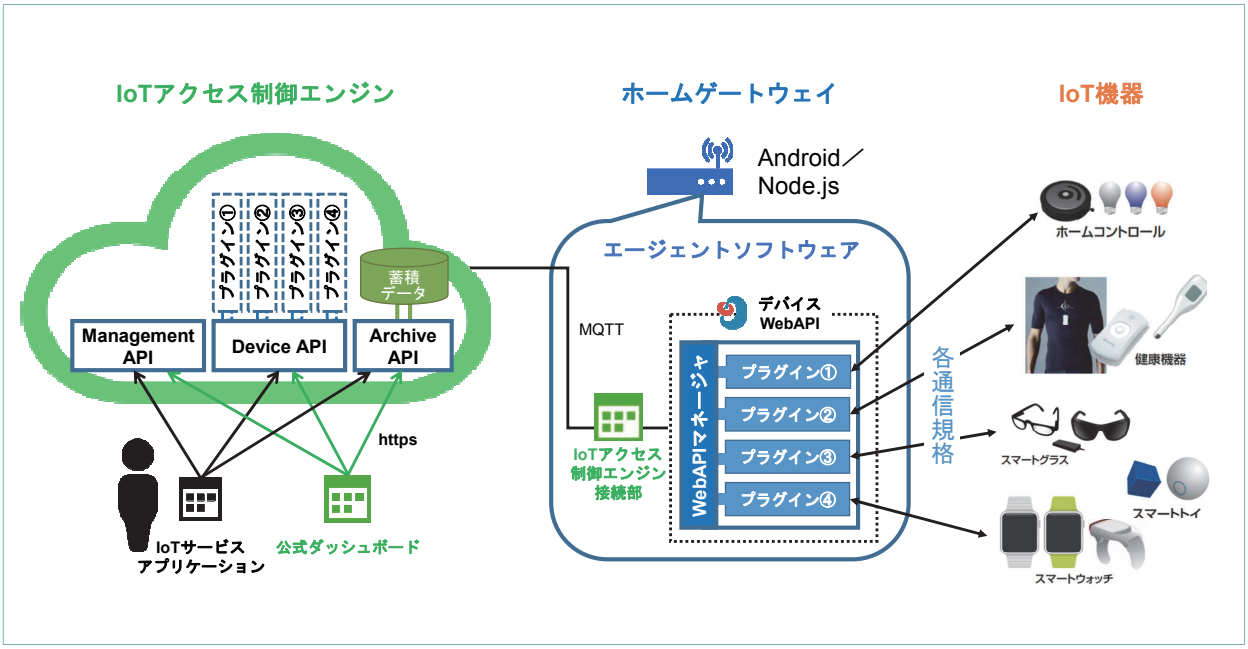

図1　IoTアクセス制御エンジンを利用したシステムの構成

ためのAPI．デバイスの登録や共有ユーザの作成も可．OAuth2.0[*6]準拠．

②Device API：IoT機器の制御を行うためのAPI．共通のインタフェースで同じ動作を実現．

③Archive API：IoT機器から集積したデータを取得するためのAPI．

2.3　ホームゲートウェイ

Bluetooth®[*7]やWi-Fi®[*8]などの近距離無線方式のIoT機器を制御する場合，IoTアクセス制御エンジンとIoT機器を仲介するためのホームゲートウェイが必要となる（図1）．

本ホームゲートウェイではデバイスWebAPIを利用したエージェントソフトウェアを稼働させる．エージェントソフトウェアは現在Android™[*9]およびNode.js[*10]のプラットフォームに対応しており，Androidスマートフォンをホームゲートウェイとして利用することも可能である．

2.4　エージェントソフトウェア

エージェントソフトウェアはホームゲートウェイにインストールされているソフトウェアで，IoTアクセス制御エンジンとIoT機器の仲介を行う．IoTアクセス制御エンジン接続部，デバイスWebAPIマネージャ（仮想サーバ），プラグインで構成される．IoTアクセス制御エンジン接続部は，IoTアクセス制御エンジンとホームゲートウェイ間をMQTT（Message Queuing Telemetry Transport）[*11]プロトコルにより接続し，クラウド環境[*12]から受け取ったAPI信号をデバイスWebAPIに中継している．

既存のIoT機器をIoTアクセス制御エンジンで扱うには，ハードウェア側の改修は不要であり，プラグインと呼ぶソフトウェアをホームゲートウェイに組み込むことにより，対応させることが可能である．各IoT機器の仕様の違いはこのプラグインが吸収しており，APIの定義はこのプラグインで行っている．すでに対応している機器の一部を**表1**に示す．

＊6　OAuth2.0：正当なクライアントに対してシステムの操作を認可する仕組み．RFC6749.

＊7　Bluetooth®：移動端末，ノートPCなどの携帯端末を無線により接続する短距離無線通信規格で，米国Bluetooth SIG Inc.の登録商標．

＊8　Wi-Fi®：IEEE802.11規格を使用した無線LANの規格で，Wi-Fi Allianceによって相互接続が認められたデバイスに用いられる名称．Wi-Fi Allianceの登録商標．

＊9　Android™：スマートフォンやタブレット向けのオペレーティングシステム，ミドルウェア，主要なアプリケーションからなるソフトウェアプラットフォーム．米国Google, Inc.の商標または登録商標．

＊10　Node.js：さまざまなプラットフォーム上でJavaScriptを動作させることのできるソフトウェア実行基盤．Node.jsおよびNode.jsロゴは，Joyent, Inc.の商標または登録商標．OracleとJavaは，Oracle Corporationおよびその子会社，関連会社の米国およびその他の国における登録商標．

表1　IoTアクセス制御エンジン対応機器

機器種別	メーカー	製品名
体温計	A&D	UT-201BLE
体重計	A&D	UC-352BLE
血圧計	A&D	UA-651BLE
活動量計	Fitbit	charge2
開閉センサ	アーミン	STM250J
人感センサ	サイミックス	HM92-01WHC
人感センサ	オプテックス	CPI-J
人感センサ	OMRON	HVC-F
スマートライト	Philips	Hueシングルランプ
スマートライト	Philips	Hue go
スマートロック	SESAME	sesame smart lock
赤外線学習リモコン	ラトックシステム	REX-WFIREX1
赤外線学習リモコン	ラトックシステム	REX-WFIREX2
赤外線学習リモコン	ラトックシステム	REX-WFIREX3
ほこりセンサ	ラトックシステム	REX-BTPM25
ほこりセンサ	ラトックシステム	REX-BTPM25V
環境センサ	Pressac Sensing	CO2，Temperature and Humidity Sensor

※表中の製品名は，それぞれ各メーカーの商標および登録商標．

2.5　公式ダッシュボード

IoTアクセス制御エンジンAPIにWebブラウザからアクセスするための公式ダッシュボード（Webサイト）を用意している．本ダッシュボードはIoTアクセス制御エンジンAPIを利用して作成しており，IoTアクセス制御エンジンを利用したIoTサービスアプリケーションを開発する上での動作確認などで利用する．

3. IoTアクセス制御エンジンの構成技術

IoTアクセス制御エンジンの特徴として，動的なAPI生成，複数機器の制御とその権限管理，利便性が高い形でのデータ集積があり，これらは以下の仕組みにより実現されている．

3.1　動的な機能単位のAPI生成

IoTアクセス制御エンジンは，IoT機器が接続されたゲートウェイデバイスやスマートフォンなどのローカル環境にある機能の設計情報をクラウド環境にアップロードすることで，クラウド環境からローカル環境の機能を管理し，外部にAPIとして公開することができる．また，クラウド環境側に，事前に機能の設計情報を必要としないため，未知の機能で

＊11 MQTT：Pub/Sub型の軽量なメッセージキュープロトコル．IoTの各種デバイスとサーバ間でのメッセージ交換に使用される．

＊12 クラウド環境：ネットワーク上に構築され，必要な時に必要なだけ利用できるよう整備された仮想的なコンピューティング環境．AWSなどが挙げられる．

あっても動的に拡張を図ることができる．

ローカル環境にある機能の設計情報としては，IoT機器の制御を共通化する技術である「デバイスWebAPI」を利用している．

デバイスWebAPIは，ローカル環境上に仮想サーバを用意することで，

①無線LANやBluetoothなどといった通信プロトコル

②OSや実行環境

③開発言語や開発環境（個別機器のSDK環境構築，依存関係の解消など）

に依存しない機能アクセスを実現している．

また，デバイスWebAPIでも用いられているローカル環境上の仮想サーバのためのセキュリティ設計とプラグインによる任意の機能拡張が，OMAにて標準化されており［6］，機能の設計情報としてWebAPI標準化仕様のSwagger＊13（OpenAPI Specification）［7］によるAPI設計をプラグインに内包することで未知の機能であっても相互接続性を担保している．

ドコモでは，デバイスWebAPIに基づいた実装として，「デバイスコネクト®＊14 WebAPI」をMITライセンス＊15のオープンソースソフトウェアとして，GitHub®＊16上で公開している［8］．

デバイスコネクトWebAPIはAndroid，iOS＊17，Node.jsの各環境に対応しており，プラグイン開発のためには，Swaggerに基づいたAPI設計をするだけで，各環境用のプラグインが一度に出力できるソースコード生成ツールも提供している．そのため，APIと機能を紐づける最低限の開発のみで，各環境でデバイスWebAPIが利用できるようになっている．API設計としては，機能の粒度，API記述のガイドライン，機能のデザインパターンが規定されており，デバイスの構造や仕様に依存しない抽象化された設計を容易にしている．

IoTアクセス制御エンジンでは，このデバイスWebAPIによるAPI設計をMQTTでクラウド環境に吸い上げて動的に機能アクセスのための構成を生成することで，ローカル環境と同様にクラウド環境からも機能が利用できる．APIを用いた開発においても，IoTアクセス制御エンジンのセキュリティ認証を行えば，APIの参照先を変更するだけでローカル環境・クラウド環境の違いを意識せずに機能が利用できる．開発者・サービス利用者はMQTTでやりとりされるメッセージ処理については意識する必要がない設計となっている．

3.2 複数機器の制御とその権限管理

IoTアクセス制御エンジンでは，前述の仕組みにより機能アクセスのためのAPI設計を内包しているため，機能アクセスの指示とその結果について，任意のAPI設計として表現できるようになっている．そのAPI設計に基づいて構造化された機能をIoTアクセス制御エンジンのダッシュボードが取り扱うことで，各種機能の操作やデータ取得が行えるだけでなく，機能単位で操作可能範囲やデータ取得の範囲を設定して第三者（特定個人や外部サービス）に公開することも可能となっている（**図2**）．

IoTアクセス制御エンジン環境自体に対する操作APIにより，利用者が個別の機能範囲の設定を必要としない機能公開や，外部サービスからの機能利用要求の許諾といった仕組も実現可能であるが，現状ではセキュリティを考慮して制限している．

3.3 利便性が高い形でのデータ集積

IoTアクセス制御エンジンでは統一されたAPI設計により抽象化された機能アクセスを実現しているため，異なるデバイスや環境であっても，目的に応じた収集データの参照や操作ログの確認が可能と

＊13 Swagger：RESTful APIを構築するためのフレームワーク，もしくはインタフェース記述のための標準フォーマットのこと．Open API Initiativeが主導しており，Open API Specificationと呼ばれることもある．

＊14 デバイスコネクト®：さまざまなデバイスをWebAPIにより相互接続するためのソフトウェア．NTTドコモの登録商標．

＊15 MITライセンス：無保証だが，ライセンス表記のみで無償・無制限で利用できるソフトウェアライセンス．

＊16 GitHub®：複数の開発者が相互にソースコードをやり取りし開発を促進させるための，ソフトウェア開発プラットフォーム．GitHub Inc.の登録商標．

＊17 iOS：米国およびその他の国におけるCisco社の商標または登録商標であり，ライセンスに基づき使用されている．

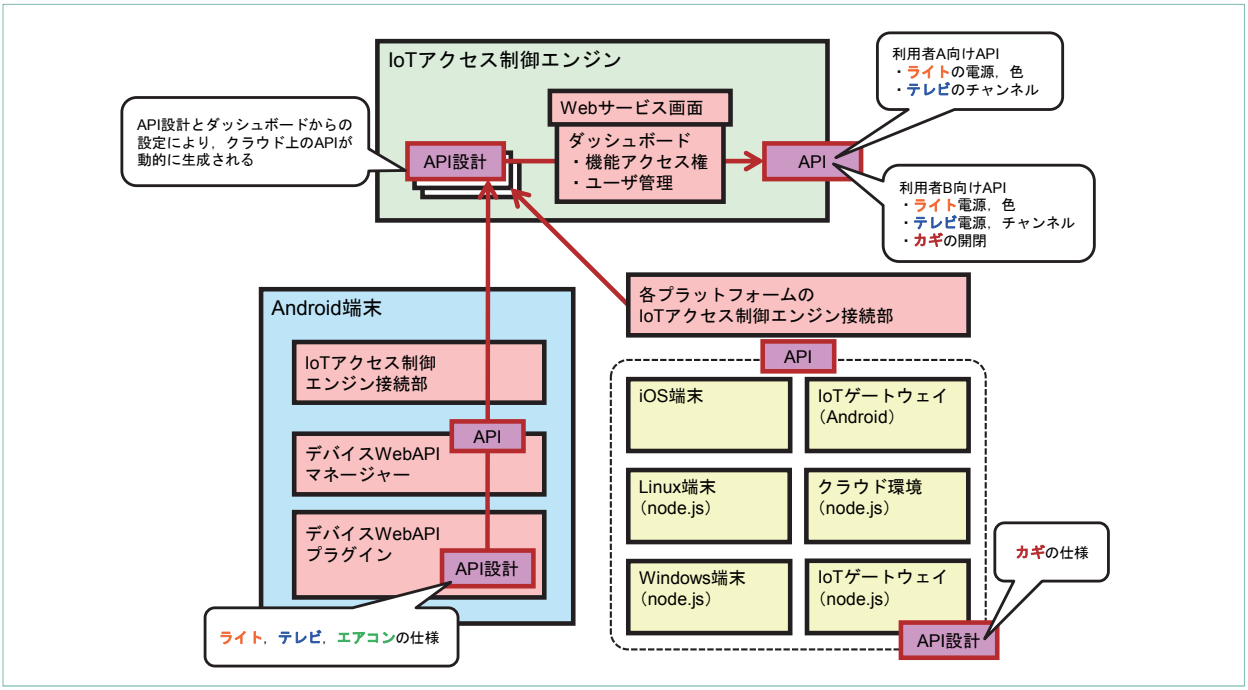

図2　動的な機能単位のAPI生成

なっている．

IoTアクセス制御エンジンは，ローカル環境からクラウド環境，デバイスの仕様の差異をなくしたデータ集積までをも一貫したアーキテクチャで実現しているため，従来は幅広いシステム設計の知識が要求されたAIを活用するユースケースであっても容易に実現できる．

具体的には，IoTアクセス制御エンジン内の，機械学習[*18]のためのクラウドサービスを，特定のプラットフォームに依存しない形でクラウドプラグイン機能として扱い，また，機械学習のための学習データおよび正解データとして，IoTアクセス制御エンジンに蓄積されたデータを機械学習サービスに入力，生成された学習モデルをローカル環境でのルールとして特定の推論エンジンに依存しない形で実行する，といった利用の仕方が見込まれる．

4. AIエージェント基盤における利用

これまでに述べた「IoTアクセス制御エンジン」のAIエージェント基盤内での活用方法を解説する．また，その一例として，「多目的対話エンジン（ユーザの自然言語を解釈するエンジン）」と連携させ，音声対話による家電操作を実現した家電制御サービス「家電くん」についても簡単に解説する．

4.1　遠隔制御

IoTアクセス制御エンジンの利用方法の1つはIoTデバイスの遠隔制御（下り）である．「家電くん」を使った最もポピュラーな対話による家電操作の利用例を**図3**に示す．ユーザが対話デバイス／アプリに対して「テレビをつけて」「ライトを消して」などと発話すると，多目的対話エンジンが動作し，音声認識処理，自然対話処理が行われて，外部サービ

＊18 **機械学習**：事例を基にした統計処理により，計算機に入力と出力の関係を学習させる枠組み．

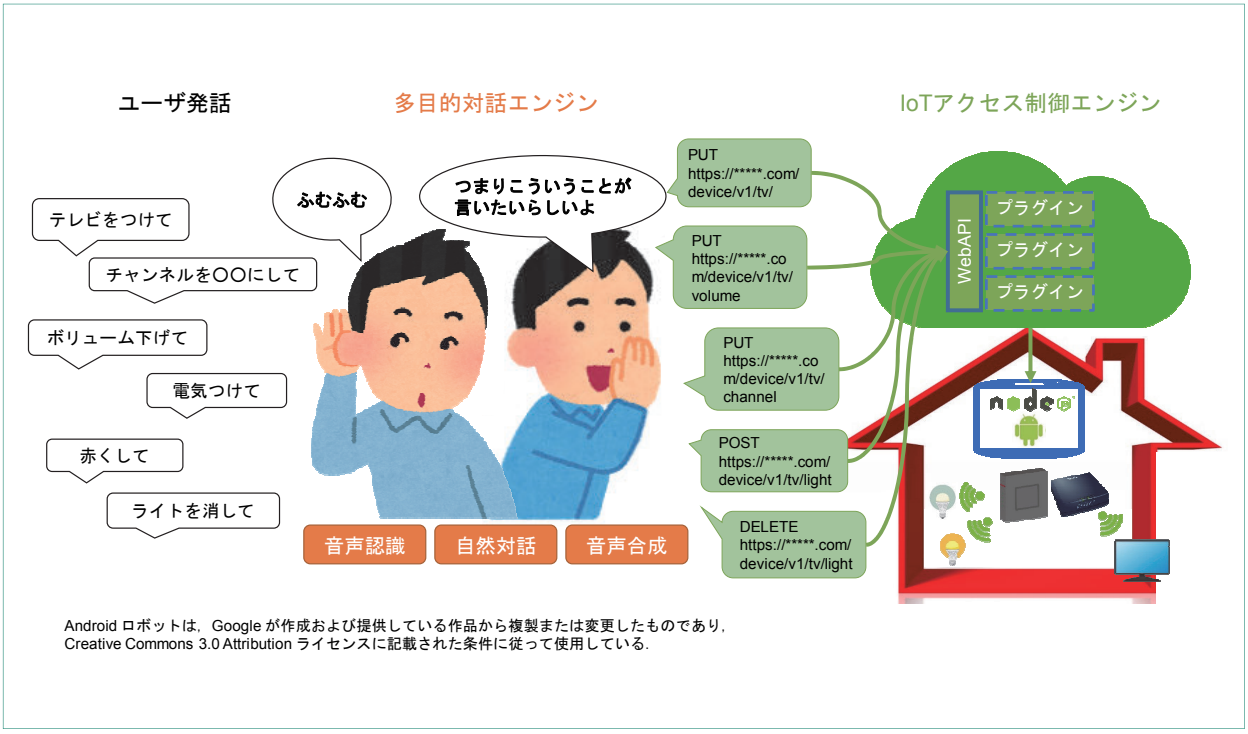

図3　対話による家電操作の利用例

スへのリクエストが生成される．ここで，家電操作用対話シナリオを動作させることで，IoTアクセス制御エンジンで動作するAPIリクエストが生成される．

例えば「テレビを消して」であれば「DELETE/device/tv/」であり，「ライトの色を赤くして」であれば「PUT/device/light?color="FF0000"」となる．

IoTアクセス制御エンジンのエンドポイントにREST API[*19]がリクエストされると，エンジン内で当該マネージャおよびプラグインへとルーチングがされ，家電制御が実行され，レスポンスが返る．最終的には多目的対話エンジン上で家電操作用対話シナリオがシステム発話を生成し，音声合成[*20]処理を経てユーザに対して「テレビを消しました」「ライトを赤くしました」といった発話が行われることで処理を終了する．

4.2　情報収集

もう一方のIoTアクセス制御エンジンの利用方法は，遠隔からの各種センサやIoTデバイスの情報収集（上り）である．これも多目的対話エンジンと連携することにより，例えば「部屋の温度は何度？」「家の鍵閉めたかな？」などといったユーザ発話に対して，IoTアクセス制御エンジンの「GET/device/temperature/」「GET/device/lock/」といったリクエストで，エンドポイント[*21]をHTTP（HyperText Transfer Protocol）で呼び出すことにより，「現在の温度は26℃です」「鍵は開いています」といったシステム発話を返すことが可能となる．

4.3　多目的対話エンジンとの連携

前述のように，多目的対話エンジンと連携することにより，自然対話による家電制御が可能となるが，

＊19 REST API：ウェブで用いられるソフトウェアアーキテクチャのスタイルの1つ．
＊20 音声合成：テキストから人工的に音声データを作り出し，テキストを読上げできるようにする技術．
＊21 エンドポイント：APIにアクセスするためのURI．

自然対話処理とIoTの組合せならではの課題がある．これらについて「家電くん」での対策を以下に示す．

1つめの課題は制御対象をどう特定するか，という課題である．ユーザが音声によって制御する家電は1つとは限らず，また複数の家電を一度に操作したい（例えば複数のライトを一度に点灯／消灯させるなど）というユースケースもあるため，どのように制御対象を特定するかという課題がある．

これについてはユーザにニックネームやグループ名を設定させ，自ら発話させることで解決を図った．「家電くん」の設定サイトのイメージを**図4**に示す．このように，ユーザに自らニックネームやグループ名を設定させ，対話シナリオとIoTアクセス制御エンジンとの連携により，ニックネームやグループ名を最長一致で抽出するという処理を実装した．ユーザが「（ニックネーム）をつけて」「（グループ名）を消して」といった発話を行うことで，制御対象を正しく特定し，制御することが可能となった．

2つめに，毎回動作対象を発話せずに動作させたい，という課題（要望）がある．日本語に顕著な特徴として，主語を省略するということがあるが，IoTアクセス制御エンジンはその特性上，必ず動作対象をサービスID[*22]として特定してAPIをリクエストする必要があるため，毎回主語を発話する必要があった．

これに対しては，制御対象をキャッシュ[*23]する処理を対話シナリオ上で実装することで解決した．具体的には，一度制御対象のスロット[*24]を埋めれば，制御対象を言い直したり，キャンセルを発話したりするまでは，制御対象のスロットをキャッシュし続けるという処理を入れることで課題を解決している（**図5**）．

図4 「家電くん」の設定サイト

＊22 サービスID：IoTアクセス制御エンジン内で，特定のデバイスの特定の機能を識別するためのユニークな識別子．
＊23 キャッシュ：配信データの一時的な保存．
＊24 スロット：音声対話の結果，アクションを起こすために必要な情報を補完するためのデータモデル．例えば家電操作の場合は「制御対象」と「操作内容」であり，天気予報であれば「場所」と「時間」が必要なスロットとなる．

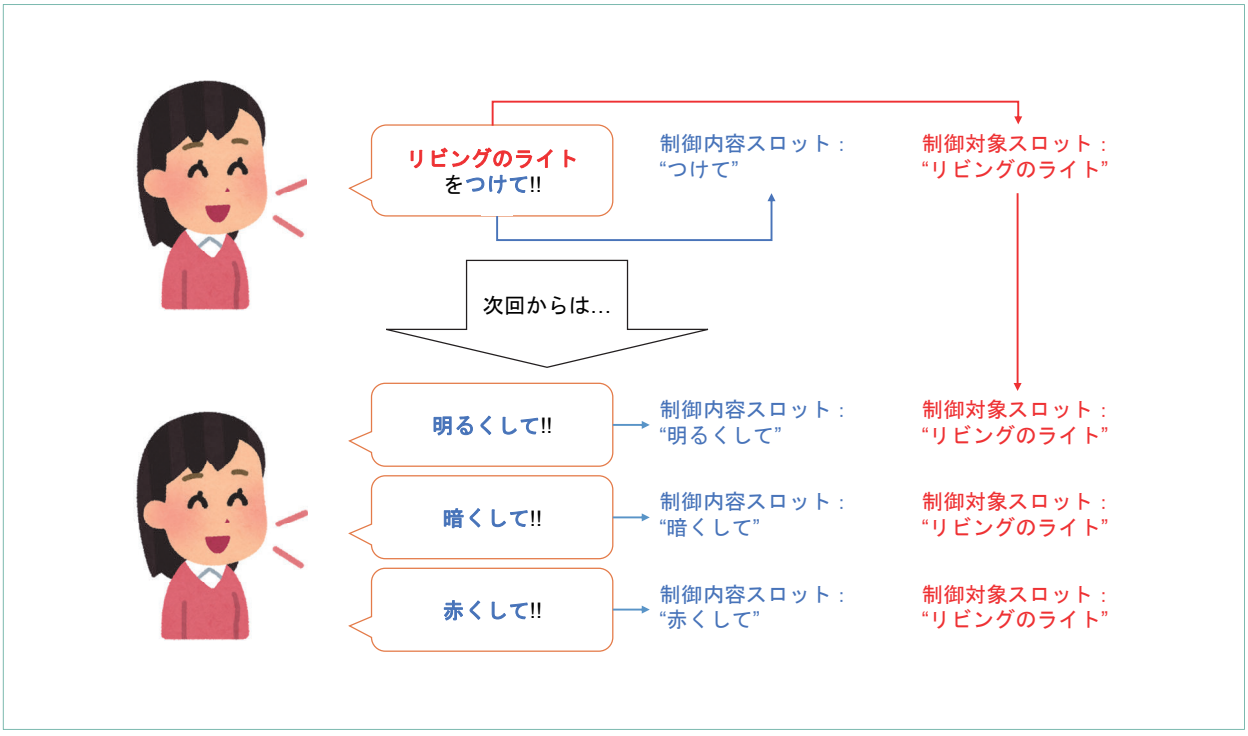

図5　制御対象スロットのキャッシュ

5. あとがき

本稿では，IoTデバイスの相互接続性の課題を解決するための技術としてデバイスWebAPIとIoTアクセス制御エンジンについて解説した．

デバイスWebAPIはIoTデバイスを機能レベルで抽象化し，共通のRESTful WebAPIでアクセスすることを可能とする．また，IoTアクセス制御エンジンにより遠隔制御や多彩なパーミッション管理機能が提供される．さらに，AIエージェント基盤の1エンジンとして連係動作することにより，IoTデバイスの遠隔制御や情報収集を行うことが可能となった．

今後の機能拡張としては，クラウドプラグイン機能やルールエンジンの提供を計画している．

クラウドプラグイン機能では，デバイスWebAPIをローカル環境ではなく，クラウド環境にもち，その制御を行うことで，クラウドサービス間連携を前提とするデバイスも直接利用することができる．

また，ルールエンジンは，デバイスWebAPIでの機能アクセスと同様の設計による，ルールの生成や制御を行うプラグインを用意することで実現しており，ローカル環境のルールのハンドリングをIoTアクセス制御エンジンから遠隔で行うこともできる．さらに，IoTアクセス制御エンジン自体としても，複数のゲートウェイデバイスやクラウドサービスにまたがったルール記述を容易にする技術を提供予定である．

また，IoTに特化した通信プロトコルの効率化や，LPWA（Low Power，Wide Area）[*25]に代表されるようなさまざまな通信ネットワークへの対応，IoTに特化したログデータの解析や機械学習といった研究開発を継続し，AIエージェント基盤のさらなる

＊25 LPWA：低消費電力でキロメートルレベルの広い領域を通信範囲にできる無線通信技術．

高度化を進めていく．

文　献

［1］ M.Weiser：“The Computer for the 21st Century,” Scientific American 265, No. 3, pp.94-104, Sep. 1991.

［2］ M. Satyanarayanan：“Pervasive computing：Vision and challenges,” IEEE Personal communications Vol.8, No.4, pp.10-17, Aug. 2001.

［3］ T. Starner：“Human-powered wearable computing,” IBM Systems Journal, Vol.35, Issue 3.4, pp.618–629, 1996.

［4］ G. D. Abowd：“Software engineering issues for ubiquitous computing,” Proc. of the 21st international conference on Software engineering, ACM, 1999.

［5］ 山添，ほか：“デバイスコネクトWebAPI　―多種多様なスマートフォン連携デバイスのためのWebインタフェース―,”本誌，Vol.23，No.1，pp.8-13，Apr. 2015.

［6］ OMA SpecWorks：“GotAPI.” https://www.omaspecworks.org/what-is-oma-specworks/iot/gotapi/

［7］ Swagger：“OpenAPI Specification.” https://swagger.io/specification/

［8］ GitHub：“DeviceConnect.” https://github.com/DeviceConnect