Technology Reports
AI Agent Device WebAPI IoT
Special Articles on New AI Agent

IoT Access Control Engine Enabling Control of Various IoT Devices from the Cloud

Service Innovation Department Takashi Yoshikawa Takafumi Yamazoe
Ken Yamashita Yuki Tanaka Shoichi Horiguchi

Although many IoT devices have been released onto the market, the interconnectivity between them is still not very good. NTT DOCOMO developed a "Device WebAPI" as technology for easily connecting such devices and an "IoT access control engine" to handle them through the cloud. This article describes the circumstances leading to these developments, characteristics of the technologies and service examples.

1. Introduction

In recent years, the Internet of Things (IoT) is a word that has begun appearing in a range of situations.

It is about 30 years since Mark Weiser first advocated ubiquitous computing [1]. In that time, differing from the so-called pervasive computing [2], wearable computing [3], mobile computing, and Zen computing, the concept of "ubiquitous computers blending into kinds of places to enrich the lives of people" has been successively continued as a fundamental aim of researchers.

Although various technical issues surrounding IoT have been raised such as data amounts, communications networks, security, data analysis technologies, and costs [4], NTT DOCOMO has taken a particular focus on interconnectivity, and is proceeding with research and development in that area.

To build applications and services using IoT devices, developments have to be done according to

each device. For this reason, device manufacturers respond to these needs by distributing software libraries and Software Development Kits (SDK)*1 in an effort to facilitate easier development. However, since the need to learn about the unique implementation and individual device technologies remains, there has not been much reduction of the workload on developers. Another issue is the difficulty of both libraries and SDKs to support all operating systems and development environments. Furthermore, implementation methods differ for each device and manufacturer, which makes it difficult to support applications and services with different devices once they have been developed - there have already been many cases of redevelopment.

These affect the interconnectivity of IoT devices, which is one issue holding back the rapid spread of IoT applications and services. To solve this issue, NTT DOCOMO abstracted a wide range of IoT devices at the functional level, and developed a "Device WebAPI" to enable access with common RESTful*2 WebAPI*3, and developed an "IoT access control engine" as a cloud-based platform enabling unified handling of IoT devices.

This article describes the technical characteristics of the Device WebAPI and the IoT access control engine, and describes examples of their use with the AI agent platform.

2. Device WebAPI and IoT Access Control Engine

2.1 Overview

One of the values of IoT is that it enables visualization of never-before-seen data, which can lead to the discovery of new value through analysis and hence new solutions. In other words, the real value with IoT comes from interacting with various types of devices and data, which means unified handling of these devices and data is indispensable in spreading IoT services.

However currently, a wide range of IoT device standards exist both in Japan and around the world, and many manufacturers use proprietary specifications for their IoT devices. Therefore, IoT service developers have to understand the specifications and build source code for various devices, which is also a huge impediment to the spread of IoT services. To address this issue, NTT DOCOMO developed its Device WebAPI and IoT access control engine.

The Device WebAPI is an interface abstraction technology that achieves (1) a common device access method using RESTful WebAPI, (2) device abstractions at the functional level, and (3) high versatility and expandability based on plug-in architecture. These three characteristics can help to solve the issue of IoT interconnectivity [5]. This technology has been standardized as a GotAPI by the Open Mobile Alliance (OMA)*4 [6].

Also, the IoT access control engine is a cloud platform that enables remote access to the Device WebAPI, so that in addition to the Device WebAPI technical characteristics, it also provides (4) unified IoT device remote management and (5) multiple permissions*5 management functions.

The IoT access control engine is also one of the engines in the AI agent platform, and can interact with other engines to enable remote control of IoT devices from multipurpose dialogue engine and

*1 SDK: A set of tools and technical documentation required for developing software.
*2 RESTful: The idea of obtaining/providing information by directly pointing to the information to be provided in a stateless manner.
*3 WebAPI: An HTTP-based API.
*4 OMA: An industry standardization organization that aims to standardize service and application technology and achieve interoperability in mobile communications.
*5 Permission: The right of access to a system. In this article, this refers to access rights set in an API to access IoT devices.

proactive support engine and sensor information collection.

Figure 1 shows an overview of the system architecture using the IoT access control engine. An IoT service application accesses the IoT access control engine using a prescribed Application Programming Interface (API) to uniformly authenticate users, control various types of IoT devices and reference accumulated data, etc.

2.2 IoT Access Control Engine API

There are currently three types of API available for the IoT access control engine, as shown below.

(1) Management API: The API for authenticating and authorizing users. It also enables device registration and creation of common users. It is OAuth 2.0[*6] compliant.

(2) Device API: The API for controlling IoT devices. Achieves the same actions with a common interface.

(3) Archive API: The API for acquiring data accumulated from IoT devices.

2.3 Home Gateway

To control IoT devices using short-range radio systems such as Bluetooth®[*7] and Wi-Fi®[*8], a home gateway is required to mediate between the IoT access control engine and IoT devices (Fig. 1).

Agent software using the Device WebAPI operates in the home gateway. Agent software currently supports the Android™[*9] and Node.js[*10] platforms, which means an Android smartphone can be used as the home gateway.

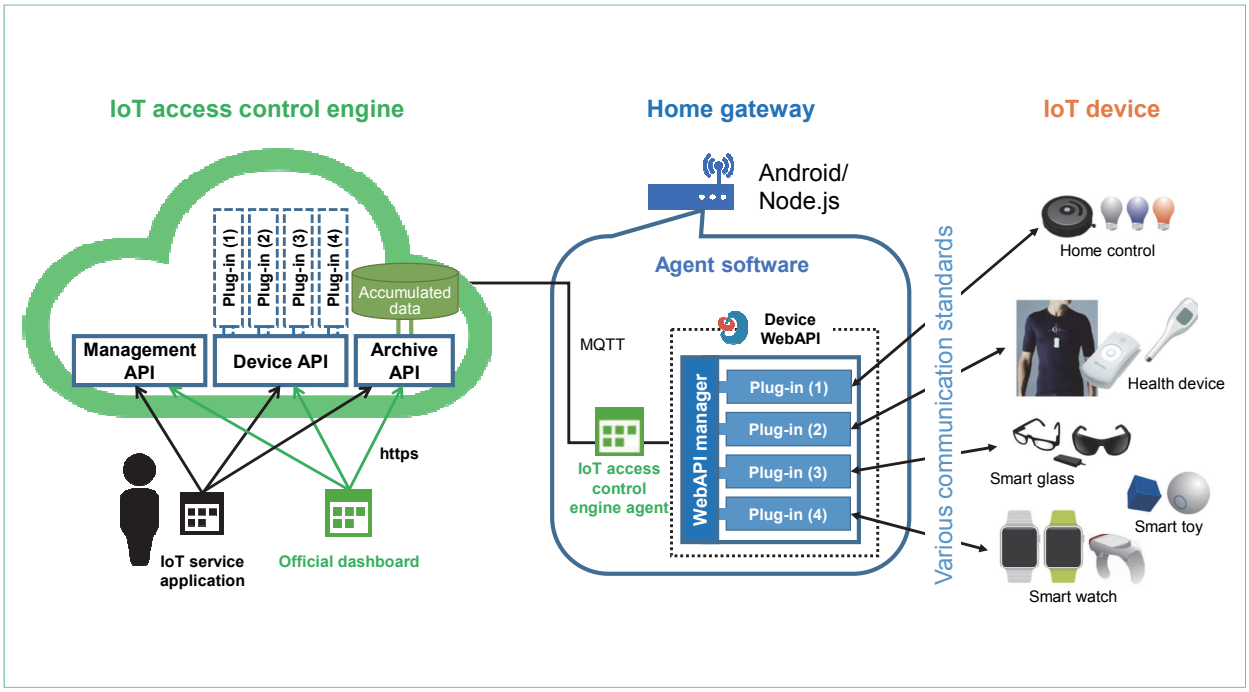

Figure 1 Schematic of the system using IoT access control engine

*6 OAuth 2.0: A mechanism to authorize system operations for a legitimate client. RFC6749.

*7 Bluetooth®: A short-range wireless communication standard for interconnecting mobile terminals such as mobile phones and notebook computers. A registered trademark of Bluetooth SIG Inc. in the United States.

*8 Wi-Fi®: The name used for devices that interconnect on a wireless LAN using the IEEE802.11 standard specifications, as recognized by the Wi-Fi Alliance. A registered trademark of the Wi-Fi Alliance.

*9 Android™: A software platform for smartphones and tablets consisting of an operating system, middleware and major applications. A trademark of Google LLC.

2.4 Agent Software

Agent software is software that is installed in the home gateway and performs intercommunications between the IoT access control engine and IoT devices. It consists of the IoT access control engine agent, a Device WebAPI manager (virtual server) and plug-ins. The IoT access control engine agent connects using Message Queuing Telemetry Transport (MQTT)*11 protocol between the IoT access control engine and the home gateway, and relays API signals received from the cloud environment*12 to the Device WebAPI.

Handling existing IoT devices with the IoT access control engine is done by including software called plug-ins in the home gateway - no hardware modifications are required. Plug-ins absorb the various differences in IoT device specifications, and the API is defined using these plug-ins. Some of the devices already supported are shown in **Table 1**.

2.5 Official Dashboard

An official dashboard (Web site) is available to access the IoT access control engine API from a Web browser. This dashboard is created using the

Table 1 IoT access engine supported devices

Device type	Manufacturer	Product name
Medical thermometer	A&D	UT-201BLE
Body weight scale	A&D	UC-352BLE
Sphygmomanometer	A&D	UA-651BLE
Activity meter	Fitbit	charge2
Open/close sensor	Ermine	STM250J
Human sensor	Simics	HM92-01WHC
Human sensor	Optex	CPI-J
Human sensor	OMRON	HVC-F
Smart light	Philips	Hue single lamp
Smart light	Philips	Hue go
Smart lock	SESAME	Sesame smart lock
Infrared learning remote control	RATOC Systems	REX-WFIREX1
Infrared learning remote control	RATOC Systems	REX-WFIREX2
Infrared learning remote control	RATOC Systems	REX-WFIREX3
Dust sensor	RATOC Systems	REX-BTPM25
Dust sensor	RATOC Systems	REX-BTPM25V
Environment sensor	Pressac Sensing	CO_2, Temperature and Humidity Sensor

*** Products appearing in the table are trademarks or registered trademarks of their respective manufacturers.**

*10 Node.js: A software execution platform that enables JavaScript to run on various platforms. Node.js and the Node.js logo are trademarks or registered trademarks of Joyent, Inc. Oracle and Java are registered trademarks of Oracle Corporation and its subsidiaries and related companies in the United States and other countries.

*11 MQTT: A Pub/Sub-type light weight message queue protocol. Used for exchanging messages between various IoT devices and servers.

*12 Cloud environment: A virtual computing environment created on a network for use at the required time and in the required amount. Examples include AWS, etc.

IoT access control engine API, and is used for checking operations of developed IoT service applications using the IoT access control engine, etc.

3. Technologies Making Up IoT Access Control Engine

As characteristics of the IoT access control engine, the dynamic API generation, multiple device control and permission management, highly convenient data collection are enabled by the following mechanisms.

3.1 Dynamic API Generation for Individual Functions

The IoT access control engine can upload design information of functions in local environments such as gateway devices to which IoT devices are connected and smartphones to the cloud environment, manage functions in local environments from the cloud environment, and externally open these as APIs. Because no function design information is required in advance at the cloud environment side, it's possible to dynamically expand functions even if they are unknown.

The Device WebAPI, a technology that achieves commonality of IoT device control, is used as design information of functions in local environments.

By preparing virtual service in local environments, the Device WebAPI achieves function access without dependence on:

(1) Communications protocols such as wireless LAN or Bluetooth
(2) Operating system or execution environments
(3) Development language or development environments (building SDK environments for individual devices, dependency resolution, etc.)

Also, secure design and free functionality expansion with plug-ins for the virtual server in the local environment used by the Device WebAPI is standardized by OMA [6], and by including API design with the Swagger*13 (OpenAPI Specification) [7] WebAPI standardized specification in plug-ins as function design information, connectivity can be ensured even if the function is unknown.

As an implementation based on the Device WebAPI, NTT DOCOMO has released "DeviceConnect®*14 WebAPI" as MIT licensed*15 open source software on GitHub®*16 [8].

DeviceConnect WebAPI supports the Android, iOS*17 and Node.js environments. To develop plug-ins, source code generation tools are provided for plug-in output for each environment at a time just with API design based on Swagger. For this reason, Device WebAPI can be used in each environment with the least amount of development to correlate APIs and functions. For API design, function granularity, guidelines for describing APIs and function design patterns are prescribed, which makes abstracted design easy and independent of device structure or specifications.

With the IoT access control engine, functions are available from the cloud environment in a similar way from the local environment by generating a configuration for dynamic function access by designing APIs with the Device WebAPI and uploading them to the cloud through MQTT. Also, in development with APIs, functions can be used

*13 Swagger: A framework for building RESTful API or a standard format for describing an interface. Lead by Open API Initiative, also referred to as Open API Specification.
*14 DeviceConnect®: Software for interconnecting various devices through a WebAPI. A registered trademark of NTT DOCOMO, INC.
*15 MIT license: A software license whose license notation enables free and unlimited use, although usage is not covered by guarantee.
*16 GitHub®: A software development platform to promote development through the exchange of source code among multiple developers. A registered trademark of GitHub Inc.
*17 iOS: A trademark or registered trademark of Cisco in the United States and other countries and is used under license.

without any awareness of the local or cloud environments just by changing the API reference destination, if IoT access control engine security authentication is done. This means developers and service users do not need any awareness of the messages exchanged with MQTT.

3.2 Controlling Multiple Devices and Managing Their Permissions

Because the IoT access control engine includes API design for function access as the aforementioned mechanism, it's possible to express function access instructions and results as an API design. Handling these functions structured based on that API design with the IoT access control engine dashboard enables not only operation of various functions and data acquisition, but also settings for the scope of operations and data acquisition for individual functions, and release of those scopes to third parties (particular people or external services) (Figure 2).

Although it's possible with the API to operate the IoT access control engine environment to open functions that don't require user settings for individual functional scopes, or achieve mechanisms to grant permissions for function usage requests from external services, currently, for security reasons, these are restricted.

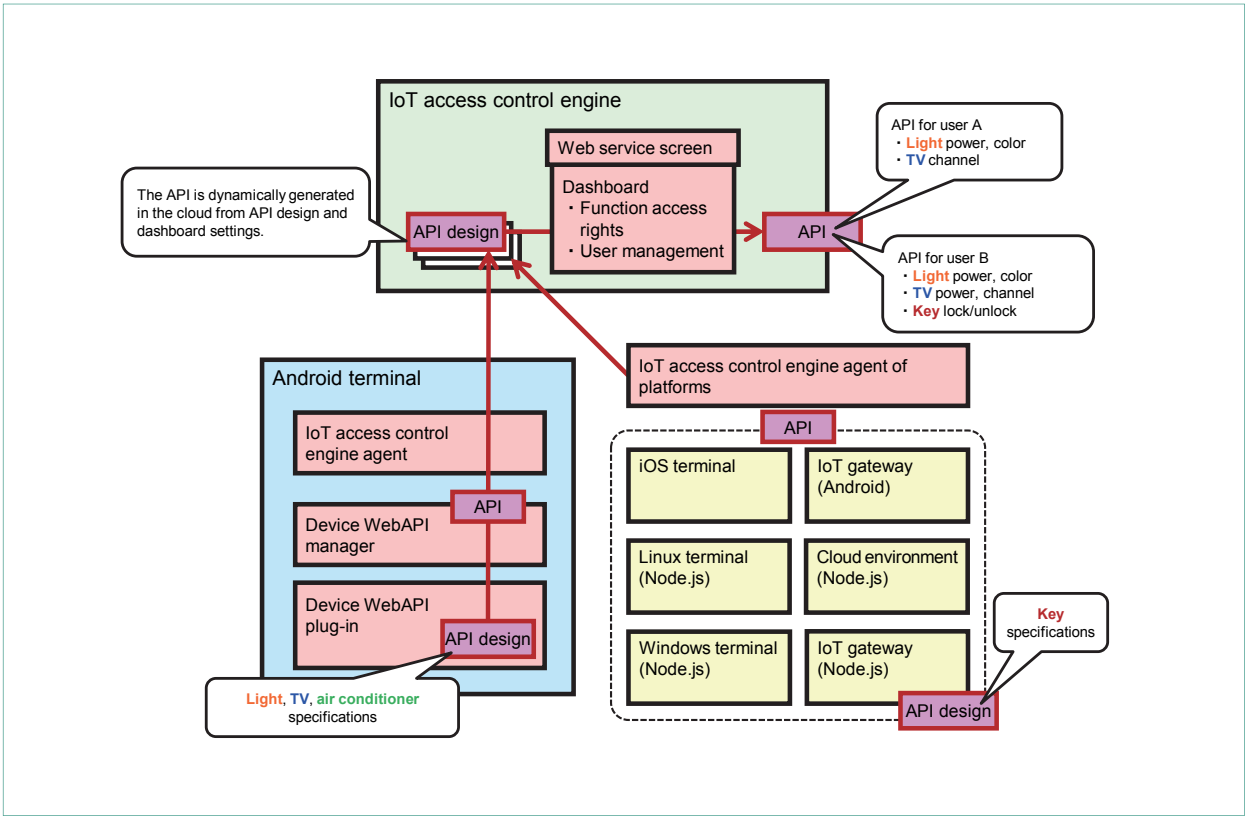

Figure 2 Dynamic API generation for individual functions

3.3 Data Collection in a Highly Convenient Form

The IoT access control engine has achieved abstracted function access with unified API design. Therefore, it's possible to reference data collected for specific purposes or check operations logs even with different devices or environments.

With the IoT access control engine, it's possible to easily achieve AI usage cases that would normally require substantial system design knowledge, because the architecture is consistent from the local through to the cloud environments and data collection regardless of differences in device specifications.

Specifically, there are prospects for usage methods, for example, using cloud services for machine learning*18 in the IoT access control engine as cloud plug-ins that don't depend on a particular platform, or inputting data accumulated in the IoT access control engine into machine learning services as learning data and correct data for machine learning and then executing the generated learning model as rules in the local environment without dependence on a particular inference engine.

4. Interaction in AI Agent Platform

This chapter describes usage methods in the AI agent platform of the "IoT access control engine" discussed above. Also, as an example, this chapter briefly describes the "Kaden-kun" appliance control service that enables appliance operation with voice by interacting with "multipurpose dialogue engine (the engine for interpreting users' natural language)."

4.1 Remote Control

One usage of the IoT access control engine is IoT device remote control (downlink). **Figure 3** describes the most popular example, controlling appliances with dialogue, using "Kaden-kun." When the user makes utterances such as "turn on the TV" or "turn off the lights" for dialogue-enabled devices/applications, the multipurpose dialogue engine executes voice recognition and natural dialogue processing, and generates a request to an external service. Then, through the action of the appliance operation dialogue scenario, an API request processed in the IoT access control engine is generated.

For example, this will be "DELETE/device/tv" for the utterance "turn off the TV," or "PUT/device/light?color="FF0000"" for "change the light color to red."

When a request is sent to the IoT access control engine endpoint using the REST API*19, routing is done to the relevant manager and plug-in in the engine, appliance control is executed and a response is returned. Finally, the appliance control dialogue scenario generates a system utterance in the multipurpose dialogue engine, the utterances "I've turned off the TV" or "I've made the light turn red" are made to the user with speech synthesis*20, and processing finishes.

4.2 Information Collection

Another use of the IoT access control engine is remote information collection from various sensors and IoT devices (uplink). This is also done through interaction with the multipurpose dialogue engine. For example, when the user asks "What is the

*18 Machine learning: A mechanism allowing a computer to learn the relationship between inputs and outputs, through statistical processing of example data.

*19 REST API: A style of software architecture used on the Web.

*20 Speech synthesis: Technology for artificially creating speech data from text and verbally reading out text.

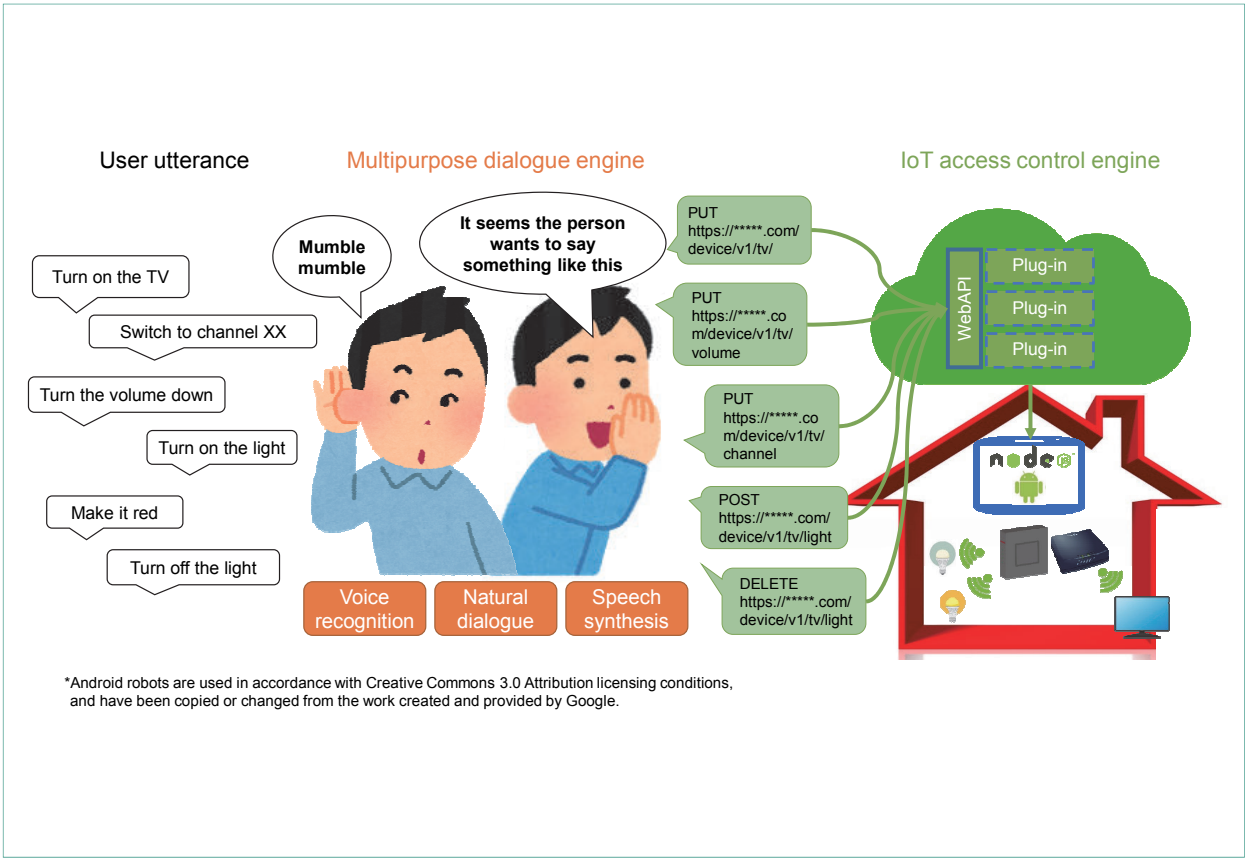

Figure 3 An example of operating appliances with dialogue

temperature of the room?" or "Is the house locked?", the IoT access control engine makes "GET/device/temperature/" and "GET/device/lock/" requests respectively, and calls the endpoint*21 in HyperText Transfer Protocol (HTTP). Then the system returns the utterances of "the current temperature is 26 ℃,"or "the house is unlocked."

4.3 Interacting with Multipurpose Dialogue Engine

Although it's possible to control appliances with natural dialogue by interacting with the multipurpose dialogue engine as discussed above, there are issues with combining natural dialogue processing and IoT. The following describes "Kaden-kun" solutions to these issues.

The first issue is control target determination. Since usage cases can entail users wanting to control more than one appliance with voice, or wanting to operate multiple devices at once (for example turning multiple lights on or off), determining targets for control is an issue.

We attempted to solve this issue by setting nicknames or group names by users and having users utter them. **Figure 4** describes an image of the "Kaden-kun" settings site. In this way, we

*21 Endpoint: URI to access to API.

Figure 4　The "Kaden-kun" settings site

implemented processing in which users set nicknames or group names, and then nicknames or group names with longest matching are extracted by linking the dialogue scenario with the IoT access control engine. When the user utters "turn on (nickname)" or "turn off (group name)," it's possible to correctly determine the target for control and control it.

Secondly, there is the issue of the desire to operate a target without specifying it, instead of specifying a target every time. The subject of sentences is characteristically omitted from Japanese language, but with the IoT access control engine, the subject had to be uttered to make a request using the API by determining the target for operation as a service ID[*22].

We solved this issue by implementing cache[*23] processing for the control targets in dialogue scenarios. Specifically, we solved the issue by inserting processing to continue caching the control target slot until the control target is uttered again or cancel is uttered, once the control target is input into the slot[*24] for the control target (**Figure 5**).

5. Conclusion

This article has described a Device WebAPI and IoT access control engine, technologies to solve issues with IoT device interconnectivity.

The Device WebAPI abstracts IoT devices at

*22 Service ID: A unique identifier in the IoT access control engine for identifying particular functions in particular devices.
*23 Cache: Temporarily stored data to be distributed.
*24 Slot: A data model that complements information required to launch actions as a result of speech dialogue. For example, the slots required to operate an appliance are "control target" and "operation details," or for a weather forecast are "location" and "time."

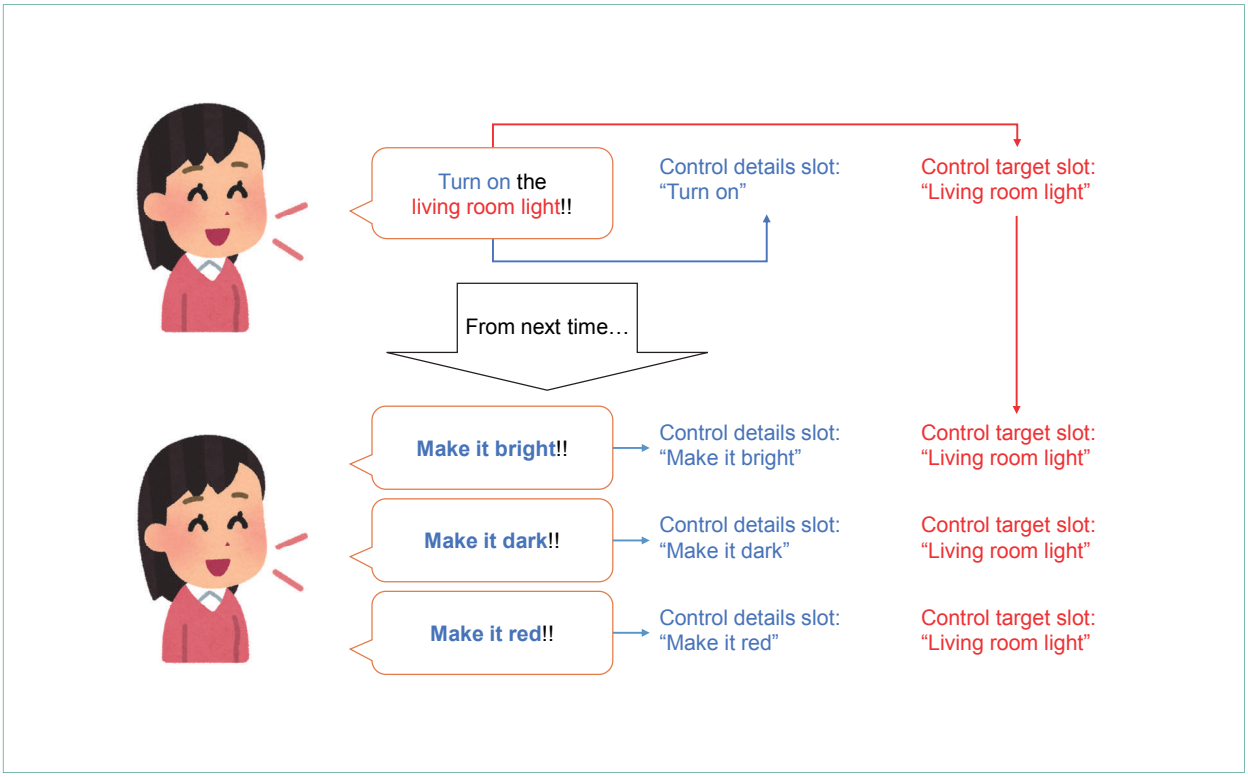

Figure 5 Control target slot caching

the functional level, and enables access with the common RESTful WebAPI. The IoT access control engine provides remote control and diverse permissions management functions. Moreover, as one of the engines on the AI agent platform, the IoT access control engine enables both remote control of IoT devices and information collection from them by interacting with other engines.

In future, we plan to provide cloud plug-in functions and a rule engine to expand functionality.

With cloud plug-in functions, the Device WebAPI is not held in the local environment, but in the cloud environment. By controlling it, it's possible to directly use devices that require interaction between cloud services.

Also, having a similar design to function access using the Device WebAPI, the rule engine is achieved with plug-ins to generate and control rules, which enables remote handling of local environment rules from the IoT access control engine. In addition, we plan to offer technologies to make rule description easy across multiple gateway devices and cloud services with the IoT access control engine.

To further advance the AI agent platform, we will continue research and development into more efficient protocols specialized for IoT, support for various communications networks such as Low Power, Wide Area (LPWA)[*25], and log data analysis and machine learning specialized for IoT.

*25 LPWA: Wireless communications technology that can support a wide communications area at the kilometer level with low power consumption.

REFERENCES

[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] T. Yamazoe et al.: "Device Connect WebAPI – Web Interface for Variety of Smartphone-linked Devices –," NTT DOCOMO Technical Journal, Vol.17, No.1, pp.4-9, Jul. 2015.
[6] OMA SpecWorks: "GotAPI." https://www.omaspecworks.org/what-is-omaspecworks/iot/gotapi/
[7] Swagger: "OpenAPI Specification." https://swagger.io/specification/
[8] GitHub: "DeviceConnect." https://github.com/DeviceConnect