IoT WebAPI Standardization

Technology Reports

Device Connect WebAPI —Web Interface for Variety of Smartphone-linked Devices—

*A variety of devices able to connect with smartphones have entered the market recently. However, the development environments for each product are different, and developing content for each OS environment and individual device is becoming an issue. With device specifications dependent on communication protocols such as Bluetooth®*1, affinity for Web services is also low. As such, we have developed a Web interface technology that operates on smartphones and has strong generality and extensibility. This article describes development of this technology and efforts toward standardization.*

Service Innovation Department *Takafumi Yamazoe*
Hiroaki Hagino

1. Introduction

A variety of devices, such as wrist bands and cameras, that are able to connect with smartphones and exchange information or control them have entered the market.

However, in order to develop services using these devices, original, manufacturer-specific specifications must be supported, which can create difficulty. Standardization of wearable and healthcare devices is advancing, but it is limited in scope, and usually there is no compatibility between specifications. Because of this, implementations specific to individual devices and standards are needed when developing a service using a variety of devices.

One way to implement content that is not dependent on the environment is the Web application (Web content that operates as an application within a Web browser). As part of the standardization of HTML5*2 at the World Wide Web Consortium (W3C)*3, Application Programming Interfaces (APIs)*4 are being consolidated to use devices from Web applications. However, they assume the commoditization of functions and devices, so while they are generic and general purpose, they do not allow the specialized functions provided by individual devices, which differentiate them from each other, to be used. Even hybrid applications, which enable Web applications to operate like native applications, are limited in that they are dependent on the state of device support in the existing framework. Even with hybrid applications, if the user has multiple applications that use varions devises installed on the same terminal, each application will have to have dedicated functions to access the devices. As with native applications, this issue has still not been resolved.

As such, NTT DOCOMO has developed the Device Connect WebAPI, combining standard Web technologies and

©2015 NTT DOCOMO, INC.
Copies of articles may be reproduced only for personal, noncommercial use, provided that the name NTT DOCOMO Technical Journal, the name(s) of the author(s), the title and date of the article appear in the copies.

*1 **Bluetooth®:** A short-range wireless communication specification for wireless connection of mobile terminals, notebook computers, PDAs and other portable terminals. Bluetooth is a registered trademark of Bluetooth SIG Inc. in the United States.
*2 **HTML5:** An enhanced version of HTML formulated by WHATWG and W3C (see*3).

general smartphone functions to create a mechanism that enables any content or service, whether native or Web application, to use any device by accessing the WebAPI. This article describes the features of the Device Connect WebAPI and related initiatives.

2. Features and Mechanism

The main features of the Device Connect WebAPI are a common method for device access, device abstraction through functions, and strong generality and extensibility.

2.1 Common Method for Device Access

Ordinary native applications (applications downloaded from an application store for the OS and running on a smartphone) access devices using methods provided by the OS, as shown in **Figure 1**. This enables any function to be implemented, but also requires applications to be implemented for each OS, and each device. Also, accessing devices from a Web application depends on functions provided by the Web browser, as shown in **Figure 2**. Currently, even though W3C has standardized APIs for accessing basic devices, only some of the functions have been implemented in Web browsers on smartphones. In contrast to this, the Device Connect WebAPI runs as a virtual server on the smartphone and provides a WebAPI to access devices freely, even from a Web browser, so that devices can be used from Web applications. Specifically, access to devices is provided by the following procedure, as shown in **Figure 3**.

(1) The Web application sends a request for device access to the WebAPI on the virtual server through the internal IP network on the smartphone.

(2) The virtual server receives the device access request, and converts it to a control command for the OS and individual device.

Generally, WebAPIs are used through an IP network, so the Device Connect WebAPI can be accessed from either native or Web applications. Also, it is not dependent on the OS development

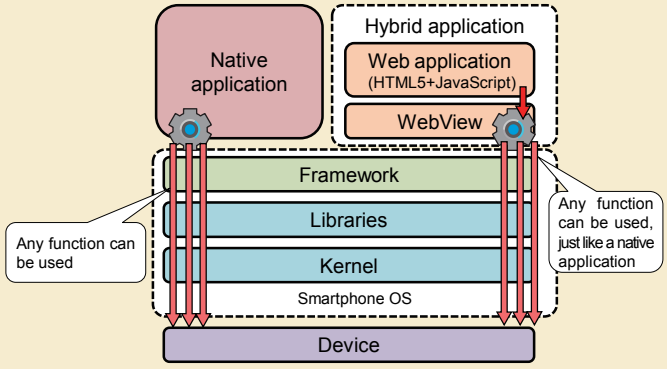

Figure 1 Using a device from a native or hybrid application

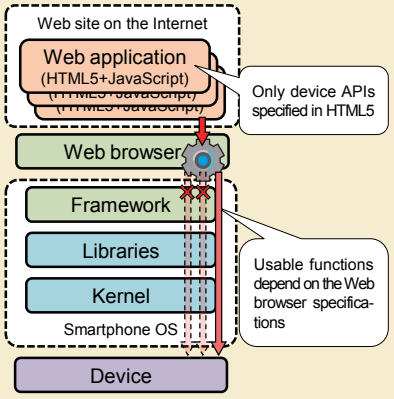

Figure 2 Using a device from a general Web application (Web browser)

*3 **W3C:** An international organization that promotes the standardization of technologies used on the WWW.

*4 **API:** General-purpose interfaces for using functions and data.

environment, run-time environment or device being connected to, and general-purpose Web or native application implementations can be used.

2.2 Device Abstraction by Function

The Device Connect WebAPI abstracts individual devices, with original functions specified by each manufacturer, using the set of functions provided by each device. For example, devices with a function to turn on a light, whether a light switch, a camera, or a toy, all perform a common operation with a common code description such as "turn on the light." **Figure 4** shows an example of controlling devices with different purposes in similar ways from a Web browser.

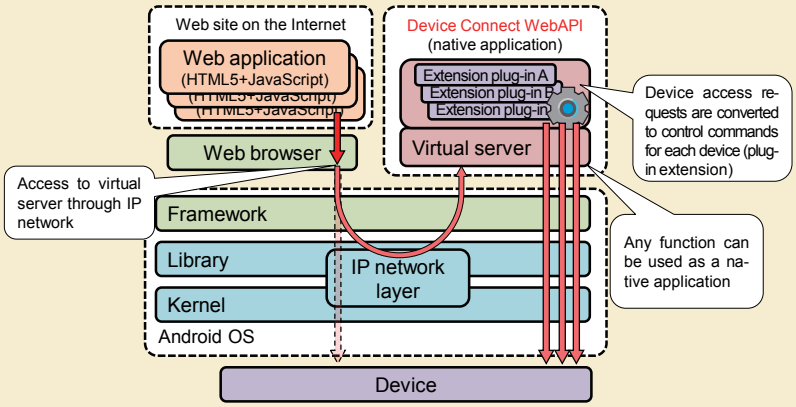

Figure 3 Using a device from the Device Connect WebAPI (Android OS example)

2.3 Strong Generality and Extensibility

The Device Connect WebAPI is composed of the core, which operates as a virtual server, and plug-ins, which connect to and control devices, as shown in **Figure 5**.

As described in section 2.1, the core

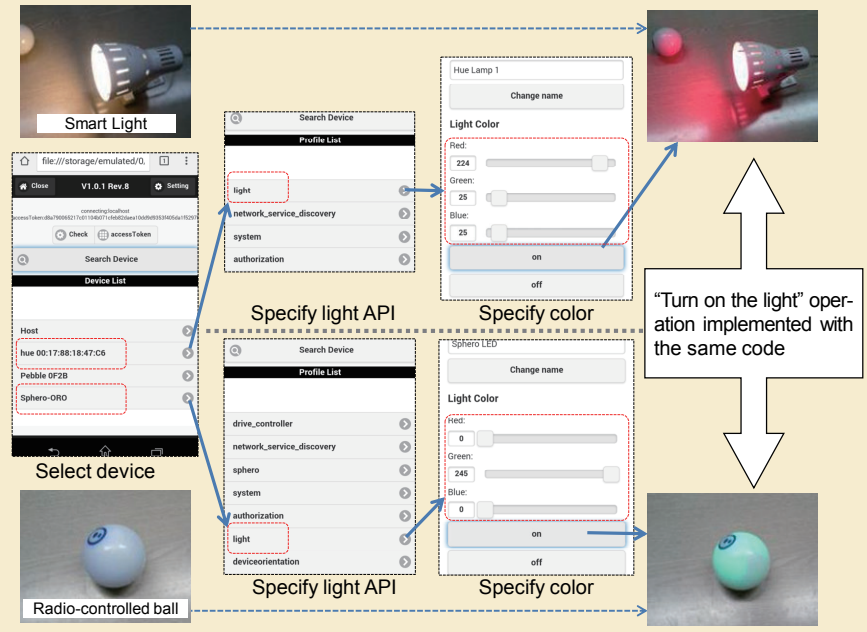

Figure 4 Example of operation from a Web browser

relays API requests received through the IP network to the plug-ins. As such, the core does not have any real functionality besides system administration, and does not have individual API specifications.

Conversely, the plug-ins have the actual functions for accessing the device, and provide API specifications for using the various functions of the device.

Since the core does not implement any functionality or specify APIs, any new device can be supported by adding a new plug-in. For the convenience of developers, general and generic functions are pre-defined in the API specifications, but any additional APIs can be defined in the plug-in, so that special and unique functionality in each device can be used. This enables the Device Connect WebAPI to achieve strong generality and extensibility.

3. Security Measures

The Device Connect WebAPI is designed to extend functionality, allowing access to various device functions from the smartphone OS or Web browser, so ensuring security is an issue (**Figure 6**).

As such, it implements various measures to protect against malicious applications, interception, tampering or impersonation (spoofing) of the virtual server.

3.1 User Consent for Use of Functions by Services

When a user installs a native application on a smartphone OS, the user is presented with a screen to authorize the application to use certain functions, to prevent the application from behaving in ways unintended by the user. However, applications using the Device Connect WebAPI are installed separately from the Device Connect WebAPI, and the functions are used through the IP network, so the confirmation screen cannot be displayed for the Device Connect WebAPI, which has already received permission to use the functions.

Thus, to prevent unintended access to functions, the Device Connect WebAPI virtual server uses OAuth authentication*5, which is a security model used widely on the Internet. When a service first accesses the virtual server, it checks a list of functions that it will use with the user, preventing access to functions not intended by the user, as with conventional applications.

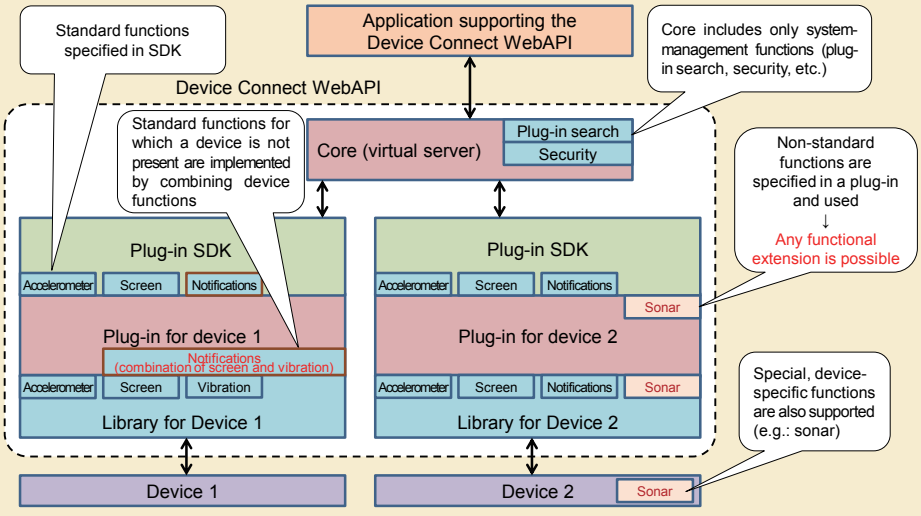

Figure 5 Device Connect WebAPI architecture

*5 **OAuth authentication:** A standard method for certifying secure APIs used in Web services and other applications.

3.2 Interception or Falsification of Communication

Since the application and virtual server communicate through the IP network, there is a risk of interception or falsification. Ideally, communication should be encrypted using Secure Sockets Layer (SSL)[*6] or other means, but the virtual server is installed and used on the smartphone, so encryption key information cannot be protected from reverse engineering[*7], and dynamically generated keys cannot be used from the Web browser. Thus, for our smartphone security model, we verified the secrecy of HTTP[*8] communication, and confirmed that various information regarding HTTP communication within the terminal cannot be obtained without having root permissions.

Communication outside the terminal, which requires a separate mechanism for secrecy, is designed to be handled by plug-ins, so it is outside of the scope of the Device Connect WebAPI. Plug-ins operate as ordinary native applications, so they are able to use dynamically generated key information, unlike the Web browser, and they can be used for encrypting communication outside of the terminal. For information passed between the virtual Web server core and plug-ins, security can be preserved by various security models provided by the smartphone OS.

3.3 Impersonation of Virtual Servers

Virtual servers are implemented as ordinary native applications, so the possibility of an application terminating the virtual server and then impersonating the virtual server must be handled.

Thus, in addition to monitoring the state of the virtual server, the virtual server is explicitly launched from the Web layer, and during processing, an Intent[*9] URL scheme is used as a mechanism to protect against impersonation. This is the mechanism that enables native applications to be launched from the standard

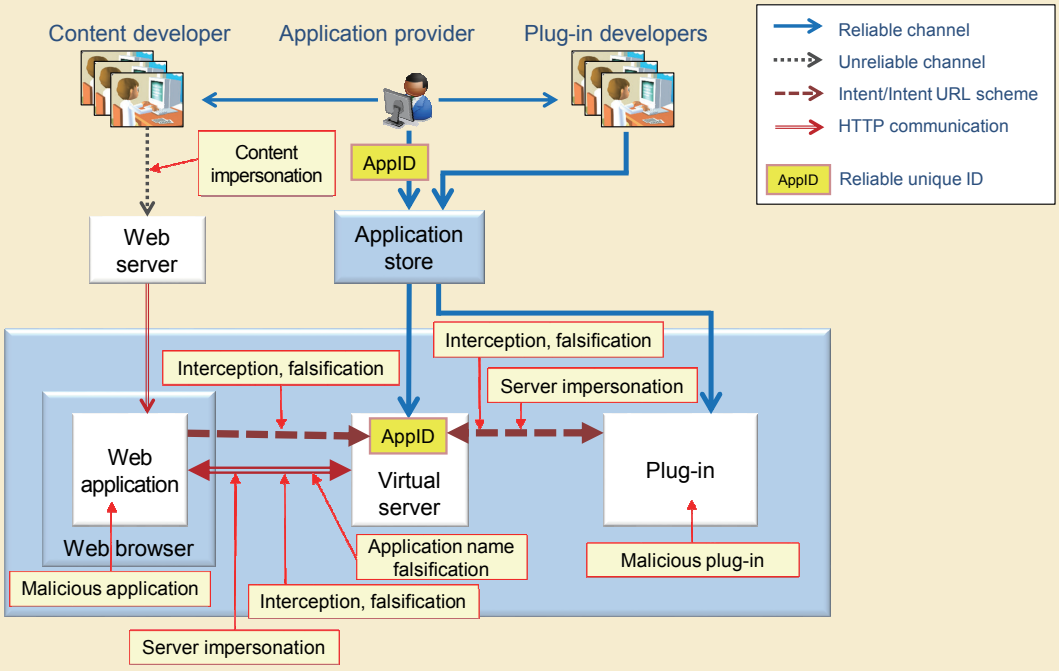

Figure 6 Security measure perspective

*6 **SSL:** A protocol for encrypted communications.

*7 **Reverse engineering:** A process of analyzing the configuration and operation of software or hardware to clarity manufacturing methods and operating principles.

*8 **HTTP:** A communications protocol used to send and receive HTML and other content between Web browsers and Web servers.

*9 **Intent:** A mechanism provided by the Android OS for programs to exchange parameters. Used between components within an application and between applications to exchange information.

Android™*10 Web browser, by specifying a package name*11, allowing information to be passed explicitly from the Web layer to the native application layer. However, the opposite, passing information from the native application layer to the Web layer, is not possible. Information verifying that impersonation is not occurring is sent as a parameter to the native application layer the first time only, and for all subsequent interaction, information proving that it is not fake is exchanged in the Web layer only in order to detect any impersonation.

3.4 Malicious Plug-ins

Plug-ins contain the implementations of functions, and can be considered the same as ordinary native applications. Thus, using the smartphone security model, it is possible to check whether a malicious application is using functions not intended when the plug-in is installed or on the settings screen.

4. Initiatives for Development of the Device Connect WebAPI

The source code for the overall architecture of Device Connect WebAPI, including a virtual server implementation, has already been published on GitHub*12 as open source software [1]. A Web interface specification as a REpresentational State Transfer (REST) API*13, as well as Software Development Kits (SDK)*14 for Android, iOS*15, and JavaScript*16 environments to make content development easier, and SDKs for Android and iOS plug-in development are provided.

As described in section 2, many features of the Device Connect WebAPI, such as generality through a Web interface and extensibility through plug-ins, are based on the architecture. Standardization of device access has been done with various goals in the past, but they have focused on building a closed model with vertically-integrated protocol stack, or conversely, specified only a limited part of the communication protocol. Very little effort has emphasized generality and extensibility of the architecture. For example, in setting near-field communication protocols, several organizations have their own specifications, as a means of creating exclusive lock-in, but they effectively lose any compatibility between devices, and reduce convenience for users.

On the other hand, by not specifying the protocol stack there is no lock-in with the Device Connect WebAPI, and by using a Web interface that abstracts functionality, the architecture emphasizes incorporation of other specifications. Thus, even with a flood of different specifications, it can be used as an interface for integrating all of them. NTT DOCOMO has already contributed the Device Connect WebAPI specification to the Open Mobile Alliance (OMA)*17 Generic open terminal API (GotAPI)*18 specification, and it will be released as a standard in March 2015.

5. Conclusion

We have developed the Device Connect WebAPI as an architecture enabling use of all functions of a variety of devices linked to smartphones from content. The technology developed has been released as open source software on GitHub using the MIT license*19, and is being standardized at OMA as GotAPI. Publishing details of the technology contributes to expanding its use and improving security, and will promote use in a variety of content and support for more devices.

Reference

[1] GitHub: "Device Connect." https://github.com/DeviceConnect

*10 **Android™:** A software platform for smartphones and tablets consisting of an operating system, middleware and major applications. A trademark or registered trademark of Google Inc., in the United States.

*11 **Package name:** Information identifying an application. Uniqueness is guaranteed by smartphone application stores.

*12 **GitHub:** A sharing Web service for software development projects.

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

*14 **SDK:** A tool or set of tools used for software development.

*15 **iOS:** A trademark or registered trademark of Cisco Corp. in the U.S.A. and other countries.

*16 **JavaScript:** A script language appropriate for use in Web browsers. JavaScript is a registered trademark or trademark of Oracle Corporation, its subsidiaries and affiliates in the United States and other countries.

*17 **OMA:** An organization that promotes standardization of mobile-related applications.

*18 **GotAPI:** A general-purpose Web interface specified by the OMA and based on a study of the DeviceConnect WebAPI implementation.

*19 **MIT License:** A software license providing no guarantee, but permitting unlimited use, free of charge, by just adding the license descriptor.