WHITE PAPER

Open Connectivity Foundation Universal Cloud Interface

October 20, 2020

OPEN CONNECTIVITY FOUNDATION

CONTENTS

Preface 2

Business Scenarios 2

- Overview 2
- Client-side Cloud 3
- Device-side Cloud 3

Technical Requirements..........3

- Introduction..........3
- Fundamentals..........4
- Linking of Clouds (From the User Perspective)..........5
- Discovery and Retrieval of Cloud Hosted Devices 6
- Device Interaction: Data Model Retrieval and Update 7
- Keeping Up to Date: Events 7

Reference Implementation “plgd” 9

Certification Program..........10

Preface

According to IDC, spending on cloud services and the professional and managed services opportunities around cloud services will surpass $1 trillion in 2024. With 41.6 billion IoT devices predicted to be deployed by 2025, there are a growing number of connected home cloud hubs, and consumers are increasingly looking for an integrated smart ecosystem.

To facilitate the interoperability among IoT devices of different vendors on a large scale, Cloud-to-Cloud (C2C) connection is a simple solution. Vendors only need to develop APIs on the cloud without changing the existing devices and cloud infrastructure.

But this brings new challenges. Vendors find they always need different or custom developed C2C APIs for different partners to meet business requirements. It is often time consuming and a heavy use of resources, when usually there are only slight differences among them.

To fix the problem, Open Connectivity Foundation (OCF) launched the industry's first C2C API standard in August 2020 – the Universal Cloud Interface (UCI), with an open source reference ("plgd", https://github.com/plgd-dev/cloud), and a certification program.

In this white paper, we aim to provide readers with a comprehensive overview of the OCF UCI, including business scenarios (section 2), technical details (section 3), open source reference (section 4), and certification program (section 5).

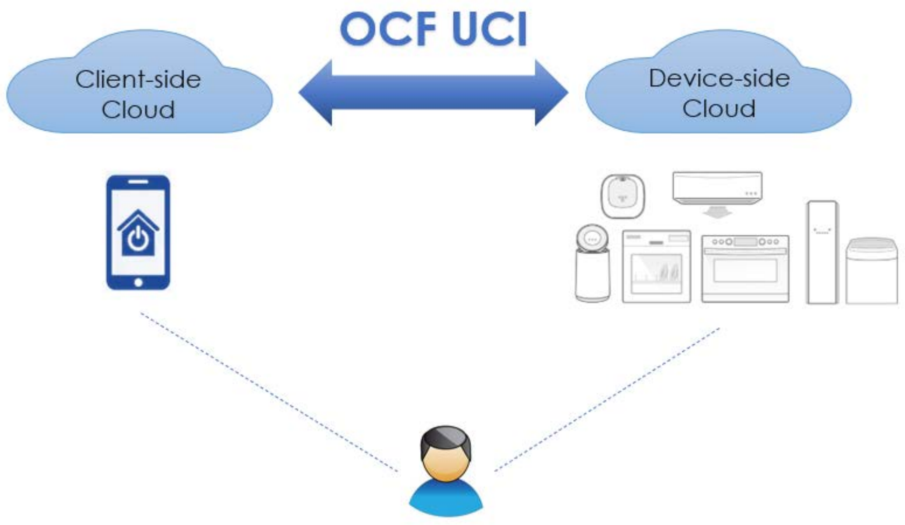
Business Scenarios

Overview

The C2C connection has become a major solution for a variety of B2B collaborations. Compared to others, C2C does not require modification of existing devices or setting up a common cloud platform or gateway. Meanwhile, vendors can manage their data easily, including what data to share with the partner and what not to share.

For example, if a user wants to switch on the air conditioner in a home from the car dashboard, it can be done using a C2C connection between the appliance manufacturer and the car manufacturer.

With the OCF UCI now publicly available, the C2C connection gets even simpler. Clouds that implement the OCF UCI can communicate with each other instantly, meeting all kinds of business requirements and saving on costly custom API investments.



There are two types of clouds in OCF UCI usage scenario: Client-side Cloud and Device-side Cloud.

Client-side Cloud

In the OCF specification, the Client-side Cloud is called the Origin Cloud, which is the cloud hosting the user's clients, e.g. smart phone and smart speaker. Through the UCI, the client can operate and receive notifications from the devices on the Device-side Cloud. Normally, the Client-side Cloud is for the IoT solution or application provider.

Device-side Cloud

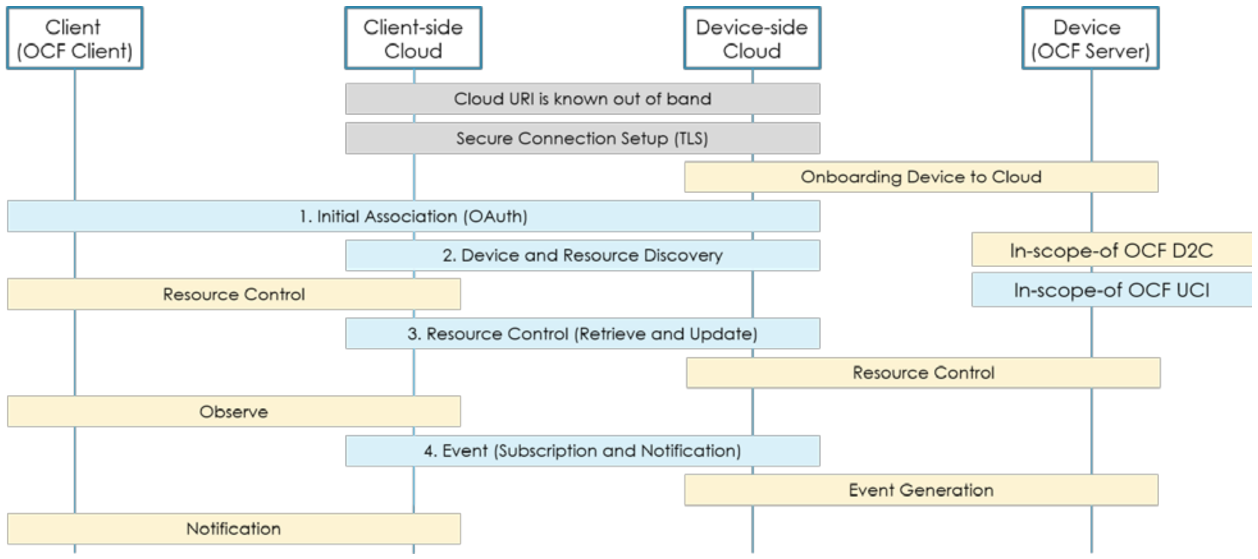
The Device-side Cloud is called the Target Cloud in the OCF specification, which is the cloud hosting the user's devices that are to be controlled, e.g. appliances and sensors. Through the UCI, the devices can expose resources (e.g. switch) and report events to the clients on the Client-side Cloud. Normally, the Device-side Cloud is for the IoT device manufacturer.

Technical Requirements

Introduction

The OCF UCI is an API that helps the IoT industry avoid implementing and maintaining numerous, simultaneously run, proprietary or custom programming interfaces. The OCF UCI is built using secure, industry standardized underlying technologies, with OAuth2.0 providing necessary authentication and HTTPS providing secure connectivity. Well-defined APIs exist for device information retrieval (and update) and event subscription. The APIs are designed to be agnostic of the data models; hence all existing and future data models published by OCF can be used.

The data models describe payloads for the RESTful verbs and when originating from outside of a cloud (for example, when retrieving an end device's information), are passed through unaltered. The media-type used for the payload can be negotiated using existing HTTP mechanisms via use of the Accept header. At a minimum, both JSON and CBOR are supported.

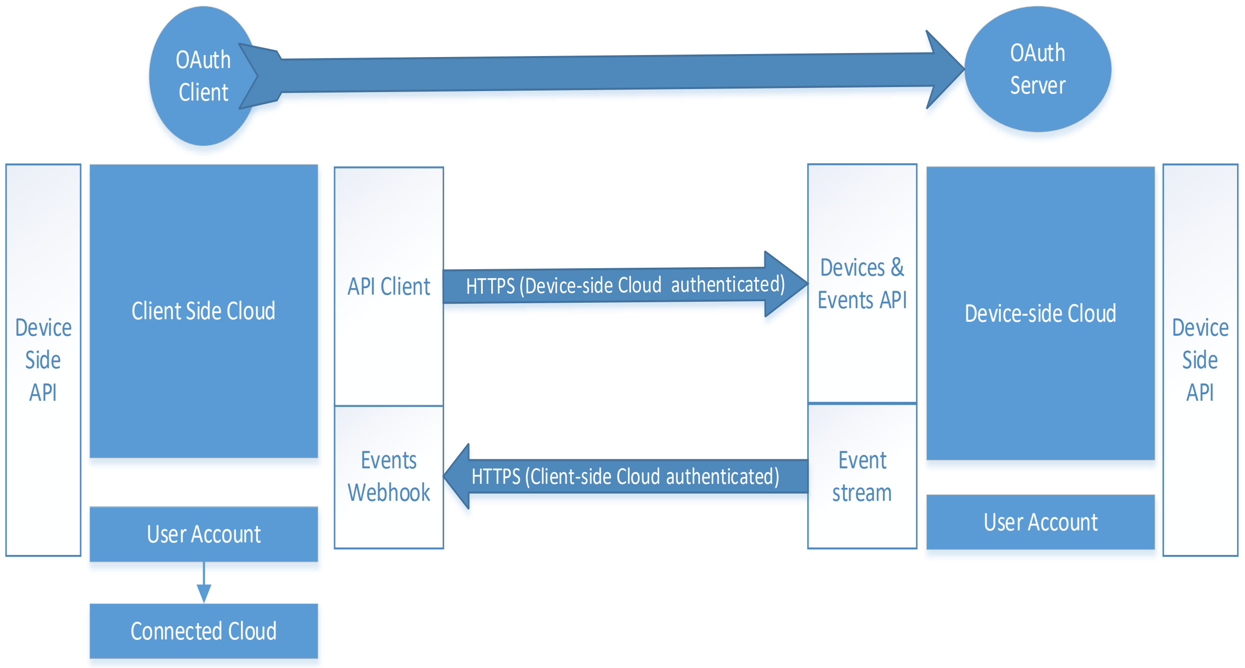

Client (OCF Client)
Client-side Cloud
Device-side Cloud
Device (OCF Server)
Cloud URI is known out of band
Secure Connection Setup (TLS)
Onboarding Device to Cloud
1. Initial Association (OAuth)
2. Device and Resource Discovery
In-scope-of OCF D2C
In-scope-of OCF UCI
Resource Control
3. Resource Control (Retrieve and Update)
Resource Control
Observe
4. Event (Subscription and Notification)
Event Generation
Notification

Fundamentals

As previously noted, the UCI is a RESTful API over HTTPS. A machine-readable definition of the API is defined using OpenAPI 2.0.

The Client-side Cloud communicates with the Device-side Cloud, using the information that was provided as part of the OAuth 2.0-based linking of the user's accounts that exist on both of the clouds. All requests between the two clouds are over HTTPS and include the OAuth2.0 provided Bearer Token (for identity).

The format of the payload that is conveyed between clouds using UCI, is negotiated using standard HTTP mechanisms. The device representations and resources are defined by the OCF specifications, which themselves leverage Open API 2.0 as part of the definitions. As such, the payloads may be (at a minimum) either JSON or CBOR encoded.



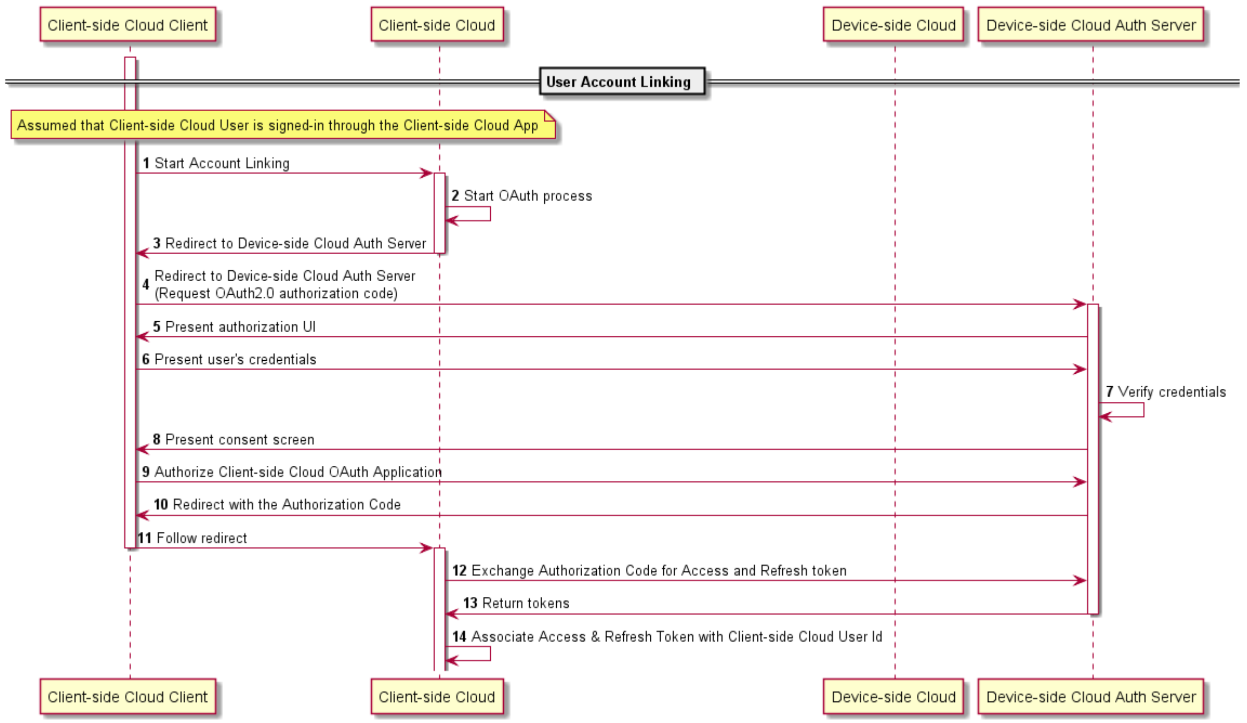
Linking of Clouds (From the User Perspective)

Consider a user who has accounts on two distinct, separately owned clouds supporting individual apps and devices associated with each of those accounts on those clouds. The user wants to have a unified view for all of their devices from a single client or app rather than having a client or app per cloud. The user selects their preferred application and then via that app, simply adds devices to that app or account. This is done by directly connecting their preferred Client-side Cloud or Application Cloud with an account connected to the devices of the Device-side Cloud. This initiates a standard OAuth2.0 authorization code grant type flow.

The account linking API is the mechanism by which devices hosted (on behalf of a user) by the Device-side Cloud are linked with a user identity on the Client-side Cloud. Account linking is established solely between the Client-side Cloud and the Device-side Cloud; information is not proxied to another third-party cloud.

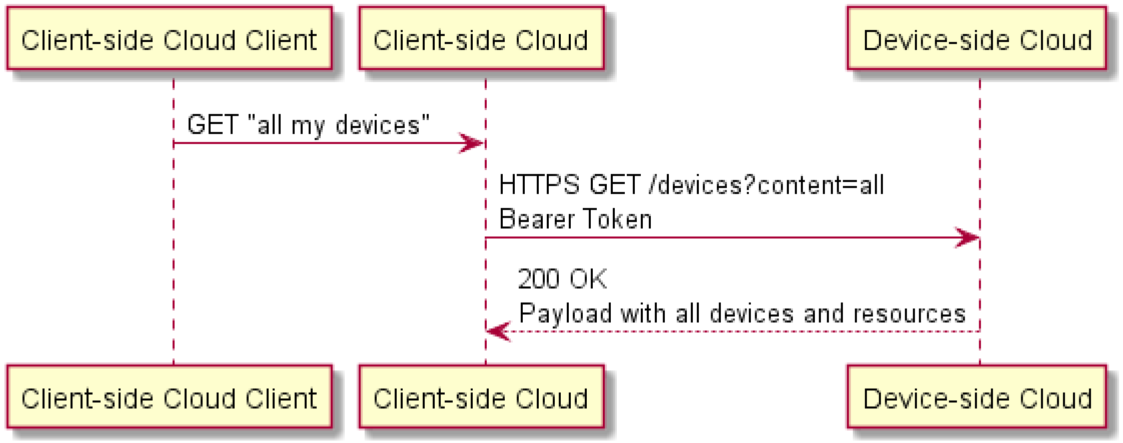
Successful registration of the OAuth 2.0 Client-side Cloud Client in the Device-side Cloud relies on the two entities establishing trust and obtaining the required client parameters and OAuth2.0 Token Endpoints (e.g. client ID, client secret, allowed redirect URIs). This is fully defined by the OAuth specifications (IETF RFC 6749). The linking is then achieved via the use of an OAuth2.0 authorization code grant type. Part of the linking process is the end-user consent, which is very important in cross-domain identity federation, ensuring that a malicious OAuth 2.0 Client-side Cloud Client cannot obtain authorization without the awareness and explicit consent of the resource owner (that is the user) of the Device-side Cloud.

Once the accounts are linked, the user is able to request all their devices through the Client-side Cloud, because the Client-side Cloud can make requests to the Device-side Cloud on behalf of the Device-side Cloud user account.


Client-side Cloud Client
Client-side Cloud
Device-side Cloud
Device-side Cloud Auth Server
User Account Linking
Assumed that Client-side Cloud User is signed-in through the Client-side Cloud App
1 Start Account Linking
2 Start OAuth process
3 Redirect to Device-side Cloud Auth Server
4 Redirect to Device-side Cloud Auth Server (Request OAuth2.0 authorization code)
5 Present authorization UI
6 Present user's credentials
7 Verify credentials
8 Present consent screen
9 Authorize Client-side Cloud OAuth Application
10 Redirect with the Authorization Code
11 Follow redirect
12 Exchange Authorization Code for Access and Refresh token
13 Return tokens
14 Associate Access & Refresh Token with Client-side Cloud User Id

Discovery and Retrieval of Cloud Hosted Devices

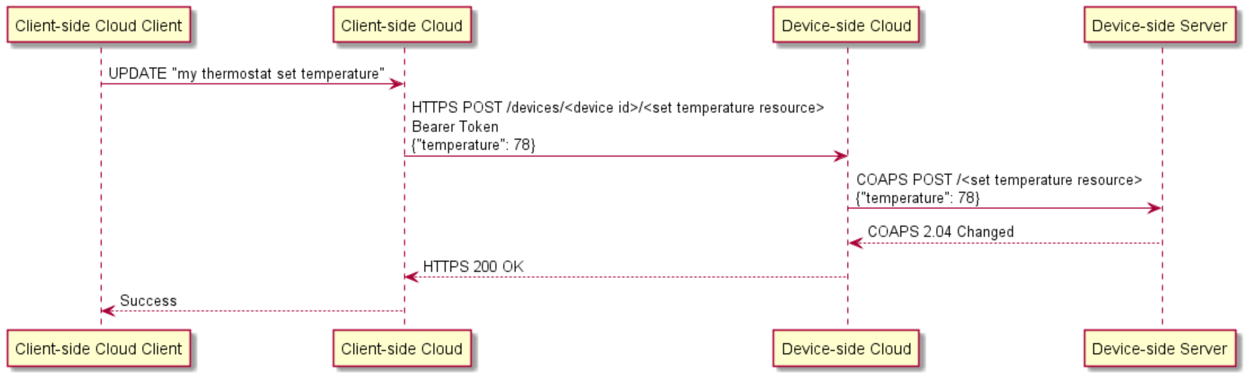
Once accounts are linked, the Client-side Cloud may invoke the UCI defined Device APIs offered by the Device-side Cloud to retrieve a complete set of device information. A single HTTP GET request is sufficient to provide a response with the entire set of hosted devices (that are associated with the user) and the resources hosted by those devices, as well as the current state of those resources. This allows the UX associated with the Client-side Client to easily provide a complete experience equivalent to that provided for Client-side Cloud hosted devices in a simple manner (single request).



Device Interaction: Data Model Retrieval and Update

From the perspective of the client connected to the Client-side Cloud, there is no distinction between devices and their resources hosted by the Client-side Cloud itself, and devices and their resources that are hosted by a Device-side Cloud reached via the UCI. As stated, all resources are provided following the OCF specifications, and follow the APIs defined in those specifications with regard to the information provided when retrieved, support for updates, and which properties with a resource may be updated. As these are all defined using machine readable OpenAPI 2.0, this lends itself to rich automated tool chains and code generation techniques for supporting and processing the information provided.

For example, see the following for a representative flow showing how a resource is updated on a Device-side Cloud hosted server.



Keeping Up to Date: Events

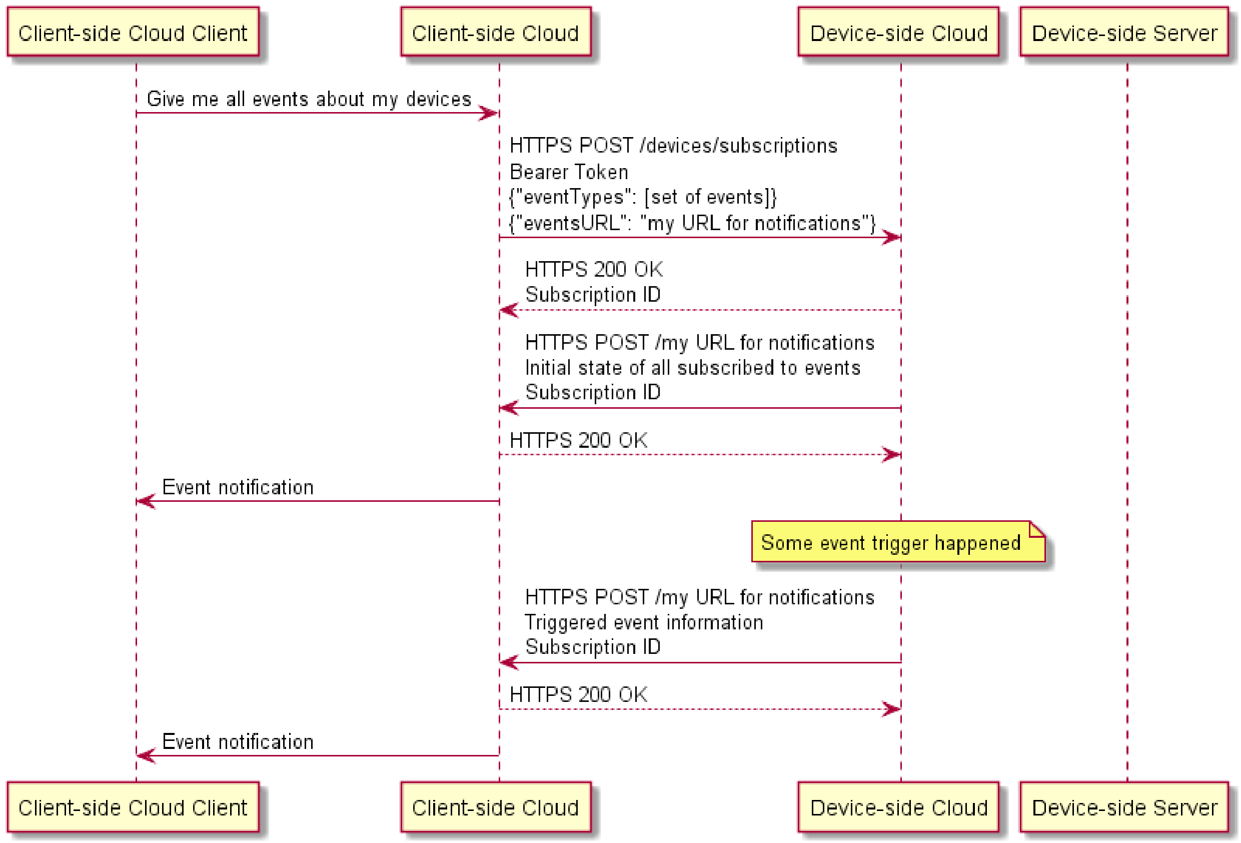
The UCI supports a set of events to which a Client-side Cloud can subscribe in order to provide to a connected client dynamic information with regard to the state of the user's connected devices; this removes the need for a client having to repeatedly ask for the information via device or resource retrieval.

The events supported are across three different levels: at the level of all devices, at the level of a single device, and at the level of a resource itself.

Taking each in turn, the UCI provides, at the level of all devices, the ability to subscribe for device online and offline, and device registered and unregistered events. At the level of a single device, the UCI provides the ability to subscribe for resource published and unpublished events. At the level of a specific resource, the UCI provides the ability to subscribe to all content changes for that resource. When the Device-side Cloud detects that a subscribed to event has occurred, it sends a notification for that event to a URL that was provided by the Client-side Cloud in the original subscription.

All notifications are protected via an HMAC signature to avoid spoofing by a malicious actor.

The following flow illustrates this model in a general sense.


Client-side Cloud Client
Client-side Cloud
Device-side Cloud
Device-side Server
Give me all events about my devices
HTTPS POST /devices/subscriptions
Bearer Token
{"eventTypes": [set of events]}
{"eventsURL": "my URL for notifications"}
HTTPS 200 OK
Subscription ID
HTTPS POST /my URL for notifications
Initial state of all subscribed to events
Subscription ID
HTTPS 200 OK
Event notification
Some event trigger happened
HTTPS POST /my URL for notifications
Triggered event information
Subscription ID
HTTPS 200 OK
Event notification
Client-side Cloud Client
Client-side Cloud
Device-side Cloud
Device-side Server

Reference Implementation “plgd”

An interoperable IoT platform needs to support multiple communication protocols and be secure by default, as well as resilient, scalable and support multitenancy. Reference implementation "plgd" (pronounced "plugged"), available at https://github.com/plgd-dev/cloud, is a cloud native IoT platform which fulfils the above needs and more.

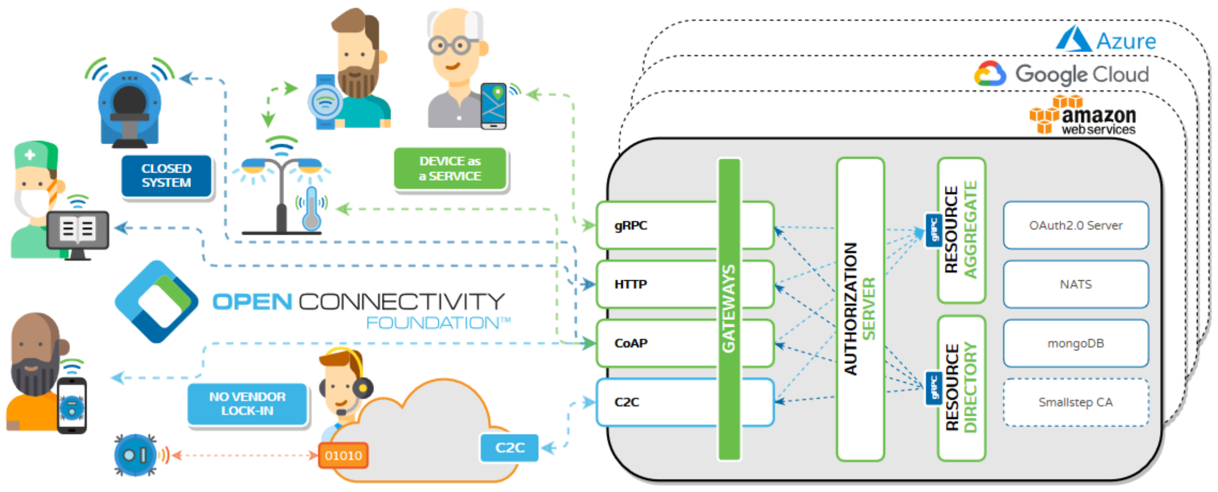
By default, the plgd cloud supports both OCF UCI scenarios as well as Device-to-Cloud connectivity using standardized OCF protocol. In addition to these standardized IoT connectivity APIs, plgd cloud exposes its gRPC gateway and offers a plgd client library available for all platforms, including the mobile ones. This allows companies to build end-to-end solutions between:

- device and app,
- device over cloud and app,
- or even device over 3rd party cloud to your cloud and app.

Possibilities of how this IoT platform can be deployed are unlimited. Low memory and CPU footprint are keeping operational costs low, which is beneficial not only for the managed deployments but also enables deployment on a gateway based on RaspberryPi for home usage. This extends to standard deployment scenarios like "public" deployment at any of the big cloud providers using k8s or "offline" deployment in your own infrastructure.

Vendor lock-in for IoT platforms is now a thing of the past, as OCF coupled with the plgd cloud reference implementation provides an end-to-end solution fulfilling cross-paltform and cross-domain secure IoT connectivity. This allows adopters to bypass choosing a specific deployment model or even hosting provider.

The combination of standardized OCF D2C and C2C APIs with wise selection of technologies (go-lang, MongoDB) and design patterns (Event Sourcing, CQRS) created this mature and interoperable IoT platform. The times when IoT vendors were creating more and more fragmentations with their closed ecosystems is hopefully gone as supporting heterogenous interfaces of all diverse platforms was costly for many companies. Additionally, there is a possibility to integrate standardized IoT using the plgd cloud and generate the economic value through interconnectedness.



Certification Program

To ensure compliance and interoperability of UCI-Certified Clouds, OCF expanded the Device-based Certification Program to include UCI-Certified Clouds. In leveraging the existing program, many aspects were already defined, including the general certification procedure, a conformance test tool (CTT) framework, and the availability of ready-to-test Authorized Test Labs. This ensured a quick deployment once the technical details were completed and published.

As OCF worked to define the UCI Specifications, OCF also worked to translate the UCI requirements into Test Cases. The Test Cases were authored to ensure complete test coverage of all the mandatory requirements of a UCI implementation for both Origin and Target Clouds. As the Test Cases progressed through their development process, the OCF's CTT developer was brought in to integrate the needed engine updates to support UCI Testing. This was a complicated task as to test each cloud type, the CTT had to be able to fully simulate the other side of the cloud interface. After significant development and internal testing, it was time to expand the test coverage and begin preliminary evaluation of Vendor-implemented UCI Clouds.

OCF held the first Multi-Vendor Interoperability Test Event (MVITE) with a focus on cloud testing in December 2019. Since both the CTT and the Vendor implementations were still in the early stages of development there was a lot of troubleshooting and adjustments needed to get through this first test event. Many items were identified for correction in both the Specification and the CTT's implementation. A few months later, in May 2020 the OCF hosted its first-ever virtual MVITE with the primary focus on validating the UCI API and associated CTT Test Cases.

Following the successful MVITE in May, the final remaining issues were known, tracked, and resolved before the launch of the UCI Certification in August. OCF Members wishing to certify their UCI implementation can log into the OCF Certification Management System (https://cms.openconnectivity.org) and submit their implementation to a Test Lab for evaluation and eventual listing on the OCF Certified Product Listing (https://www.openconnectivity.org/certified-products).