IoT Platform
ユーザーガイド
(16.0_0 版)

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2024

＜改版履歴＞

版数	説明	日付
1.0 版	初版	2016/10/7
1.1 版	表の整形 図の番号の修正 注意事項「目安を超えた使い方をする場合の対応」を追加 注意事項「「リソース_Binary」のエラーメッセージについて」を削除 凡例の curl コマンド、JSON の一部誤記修正 凡例のテナント ID について修正	2016/11/1
1.2 版	「ご契約後の通知内容」に関する記載の変更 第 7 章「仕様制限（注意事項)」に 1 つのアクセスコードに紐づけられるリソース数に関する追記	2016/12/1
4_1.0 版	機能拡張に伴う追記 2.1.1.　データ収集　P 権、G 権に関する機能の追記 5.1.　　登録したデータに付与される情報について　新規記載 5.3.　　複数リソースにまたがったデータの一括取得について　新規記載 5.5.　　「CDL」権を使ったリソースの作成頻度について　新規記載 5.8.　　クライアント認証について（2017 年 7 月以降提供）　新規記載 5.9.　　CORS の活用について　新規記載 5.10.　イベント、転送失敗時のログ参照について　新規記載 8.8.　　複数のリソースデータを 1 回の API 操作で参照する例($all)　G 権、P 権で実行できるように記載 8.9.　　リソースデータの特定の name のみを参照する例(select)「name」表記に修正 10.3.　リソース_Binary のデータ参照例　Memo 欄に Range Request に関する記載を追記 13.1.　イベントの登録例　凡例に Content-Type の記載を追記 13.2.　イベントの参照例　ヘッダフィールド値に対する記載誤りを修正	2017/3/27
4_2.0 版	・17 年春のエンハンスに関する記載を削除	2017/8/1
4_2.1 版	機能拡張に伴う追記 5.8.　　クライアント認証の例について記載	2017/9/13
5.0_0 版	機能拡張に伴う追記 5.8.　　クライアント認証の例について一部誤記修正 5.11.　JSON 変換について記載 8.10.　csv/txt/binary/gzip ファイルを「リソース_JSON」へ登録する例、及びその登録された JSON データ参照する例について新規記載 8.11.　配列の要素位置を指定して検索する例について新規記載	2017/11/18
5.1_0 版	一部、記載を追記 5.8.　　クライアント認証の例について一部追記 7.1.2.　トランザクション非保証について一部追記	2018/3/17
7.0_0 版	機能拡張に伴う追記 8.12.　エラー収集用リソースの設定および参照する例について新規記載	2018/9/15
7.0_1 版	一部、記載を修正 8.10.　csv/txt/binary/gzip ファイルを「リソース_JSON」へ登録する例、及びその登録された JSON データ参照する例について一部	2018/12/12

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2024

	誤記 修正 12.1.　アクセスコードの登録例について記載誤りを修正	
9.0_0 版	機能エンハンス 変更なし	2019/8/22
11.0_0 版	機能エンハンス 10 章～13 章 API 実行時レスポンスについて変更	2021/1/16
14.0_0 版	機能エンハンス 変更なし	2022/4/1
16.0_0 版	ダイナミックリソースコントローラー（オンデマンドデータ収集）機能削除に伴う修正　※本書自体の変更なし	2024/6/3

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2024

はじめに

　このたびは「IoT Platform」をご検討いただき、誠にありがとうございます。
この「IoT Platform ユーザーガイド（以下、本書）」は、本サービスをご契約いただいたお客様、またはご採用いただくお客様のための資料です。下記につきましてご了承くださいますよう、よろしくお願いいたします。

1.　ご検討中のお客様は、本書を本サービスご採用可否のご判断のためにのみ、ご参照願います。
2.　本書および本書の内容について、第三者へご開示、ご提供にはならないようにお願いいたします。
3.　発行元の許可なく、本書の記載内容を複写、転写することを禁止します。

本書には本サービスをご採用いただくための重要な情報を記載しています。
ご契約いただいたお客様は、ご利用前に本書をよくお読みの上、本サービスをご利用ください。なお、本書は大切に保管してください。
ご採用いただけないお客様は、お客様の責任で本書をすみやかに廃棄願います。

本書の作成にあたって、細心の注意を払い可能な限り分かりやすい記載に努めていますが、本書の記述に誤りや欠落があっても弊社はいかなる責任も負わないものとします。本書及びその記述内容は予定なく変更される場合があります。

本書の内容の一部または全部を無断で複製・転載・改編しないでください。

Android, Android Studio は、米国 Google Inc. の米国およびその他の国における商標または登録商標です。
Eclipse Paho は米国およびその他の国における Eclipse Foundation, Inc. の商標もしくは登録商標です。
Python は Python Software Foundation の登録商標です。
Ubuntu は Canonical Ltd. の登録商標です。
Linux は Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
Java は、Oracle Corporation 、関連会社の米国およびその他の国における登録商標です。

免責事項について

・本サービスのマニュアルに記載されていない操作を行なったことで思わぬ誤動作や、予期せぬ課金が発生した場合であっても一切の責任を追いかねます。
・本サービスの利用、または利用できなかったことにより万が一損害（業務の中断・データの破損/損失・事故などによる損害や第三者からの賠償請求の可能性を含む）が生じたとしても、責任を一切負いかねます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2024

＜用語＞

用語	説明	備考
IoT	Internet of Things の略 「モノのインターネット」と言われるが、「モノがインターネットプロトコル（インターネット言語）でネットワークされている状態」のこと。	
REST	REpresentational State Transfer　の略 複数のソフトウェア連携に適した設計原則を Web に適用したソフトウェアの設計様式を示す。 ※特定の URL にパラメータを指定して HTTP でアクセスすると XML で記述されたメッセージが送られてくるような、呼び出しインターフェース（「RESTful API」と呼ばれる）。 本サービスでは XML ではなく JSON 形式で返信する。	
MQTT	Message Queuing Telemetry Transport　の略 MQTT とは、TCP/IP ネットワーク上の通信プロトコルで、多数の主体の間で短いメッセージを頻繁に送受信する用途に向いた軽量なプロトコル。M2M ネットワークや IoT（Internet of Things）分野での普及が見込まれている。ヘッダ部分が最小で 2 バイトと小さく、HTTP で同様の通信を行う場合より、通信量や CPU 負荷・電力消費量などを数十分の一に抑える。	
ダイナミック リソース コントローラー	Dynamic Resource Controller 富士通の広域分散技術を用いた分散制御処理。データ収集においてトラフィックの変動に従い、限られたリソースの中で最適化した収集の実現を補佐する機能。	
リソース	リソースデータの収集単位	
リソースデータ	1 つのデータ	
アクセスコード	リソースを対象に設定する認可情報	

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2024

- 目次 -

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第1章　はじめに

1.1. 本書の目的

本書はIoT Platform（以下：本サービス）のご利用に際し、システムを設計する方向けの「ユーザーガイド」です。

1.2. ドキュメント構成

サービスのご利用に際し、お客様を支援するための以下のマニュアルをご用意しております。

マニュアル名	説明
IoT Platform サービス詳細説明書	本サービスのサービス仕様を説明しています。
IoT Platform ユーザーガイド	本サービスを利用したシステムを設計するにあたって、具体的な例を交えて設計指針の説明を行うマニュアルです。（本書）
IoT Platform API リファレンス	本サービスで提供するサービスを利用したアプリケーションを設計するための API のリファレンスマニュアルです。
IoT Platform サービスポータル操作マニュアル	Web インターフェース機能(以下：サービスポータル)に関するマニュアルです。

Memo

リソース、アクセスコードなど本サービス全般の定義・考え方については、サービスポータル操作マニュアル第 3 章をご参照ください。

1.3. 特長

本サービスの API は以下の特長を持っています。API を活用することにより、様々な IoT ビジネスの実現を支援します。

・使いやすいインターフェース

本サービスは、広く利用されている REST(HTTP)や、軽量で負荷の少ない MQTT に対応しています。それぞれのプロトコルに対して、お客様のビジネスに適用しやすいインターフェースを取り揃えています。

・豊富なクエリによる探索

リソースデータ操作のための OData ライクなクエリを数多く用意しています。これらのクエリを組み合わせることにより、複雑な条件における検索を簡単に実現できます。

・SSL/TLS 対応によるセキュアなデータアクセス

API は SSL/TLS にも対応しているため、安全にデータアクセスすることが可能になります。また、各データ群（リソース）に対し、アクセス権限を設定出来る為、お客様協力会社等複数社間でのセキュアなデータ活用が可能です。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第2章　機能概要

2.1.　IoT Platform 機能概要

2.1.1. データ収集

お客様テナント内にデータ登録単位である「リソース」を作ります。リソースには、当該リソースに対する権限である「アクセスコード」を作成し、付与する必要があります。

アクセスコードを API に付与することにより、該当リクエストは認可され、実行できます。

＜権限の種類＞

C：リソースやアクセスコードなど、作成できる権限
R：データを参照できる権限
U：データを登録できる権限
D：リソースやアクセスコード、イベントを削除できる権限
L：リソースやアクセスコード、イベントの一覧を取得できる権限
P：リソースの特定ディレクトリ配下に対してデータを登録できる権限
G：リソースの特定ディレクトリ配下に対してデータを参照できる権限

2.1.2. データ収集準備

サービスポータルからの操作、および API での操作が可能です。

※API での操作対象は「リソース」「アクセスコード」「イベント」のみ。

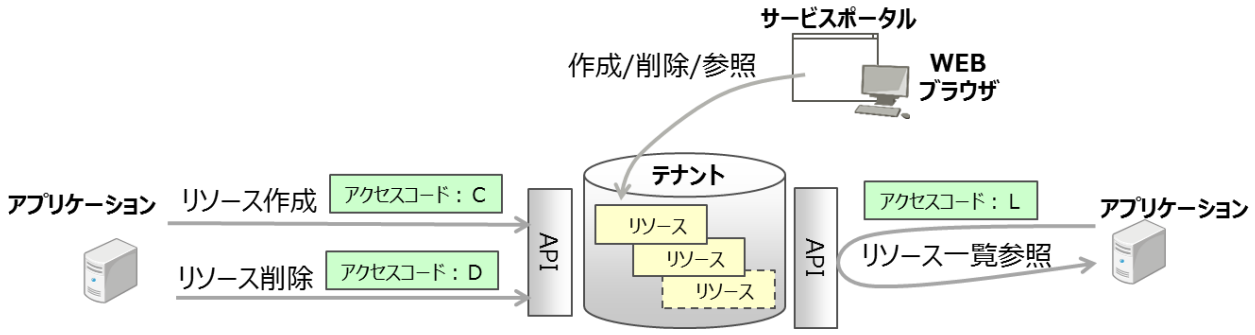


図 1 リソースを管理する

2.1.3. データ収集と活用

データの登録や参照は API で実行できます。

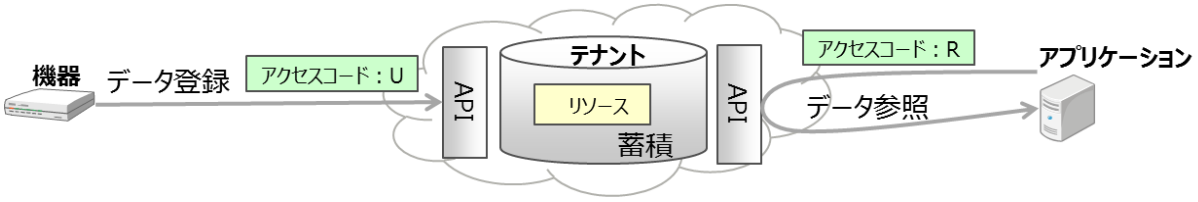


図 2　データを蓄積する

本サービス内にデータを蓄積せず、他サービスへ転送することも可能です。

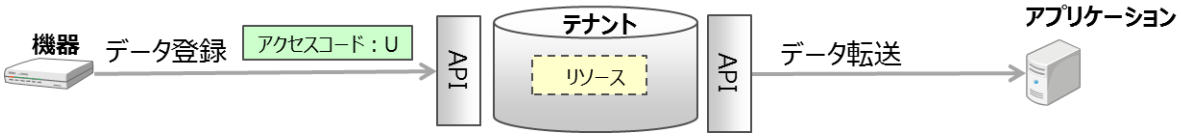


図 3　データを転送する

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

2.1.4. イベント機能

イベントとしてデータの抽出条件を設定することにより、データの登録／更新を契機に抽出条件にマッチする場合にアクションできます。アクションは、Email 通知または指定の API 起動を設定できます。

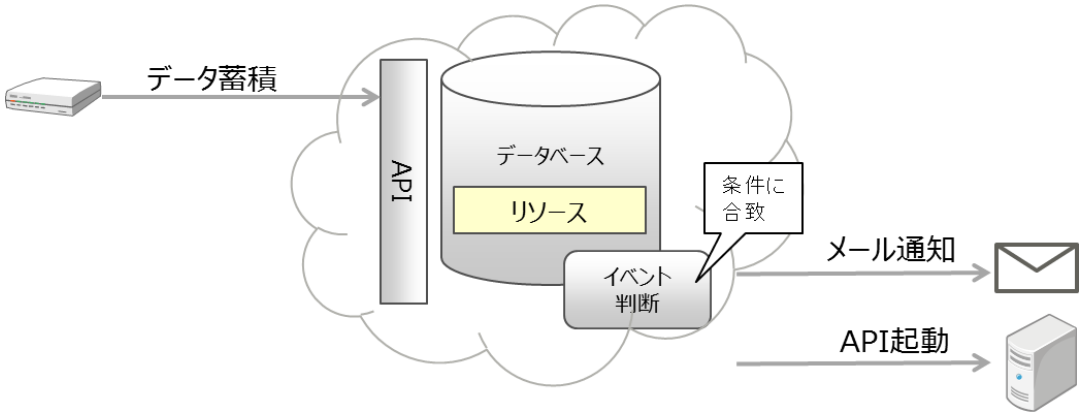


図 4　イベント機能

2.1.5. アクセス制限

本サービスに対するアクセスに IP アドレスの制限をかけることができます。

サービスポータルに対するアクセスおよび、アクセスコードに対して制限を設けることによりリソースに対するアクセスを制限できます。

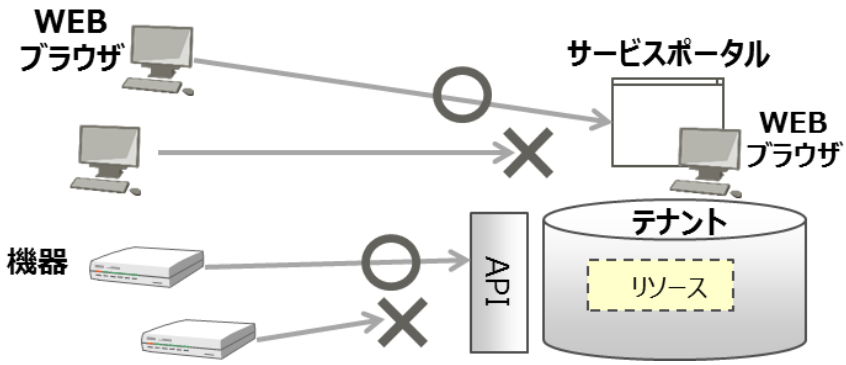


図 5　アクセス制限

2.2.　ダイナミックリソースコントローラー

データ収集において、クラウドにかかる負荷によってデータ収集を制御する指針として利用できるリコメンドリソースを提供します。これによってお客様 IT リソースに合わせたデータ収集を可能とします。

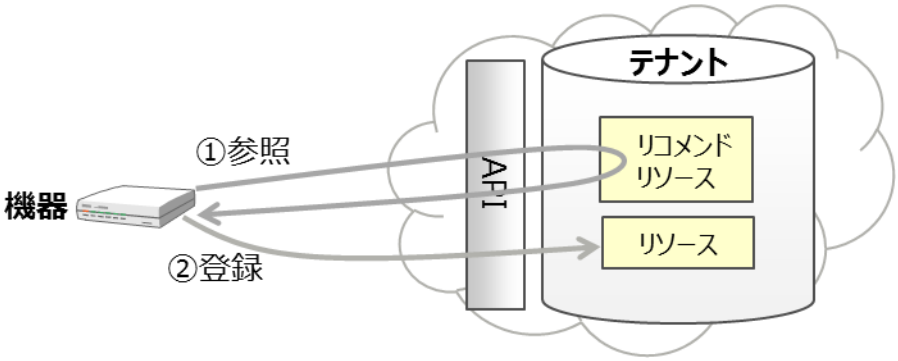


図 6　ダイナミックリソースコントローラー

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

2.2.1. 基本動作

以下にダイナミックリソースコントローラー（以下：DRC）の基本動作を示します。

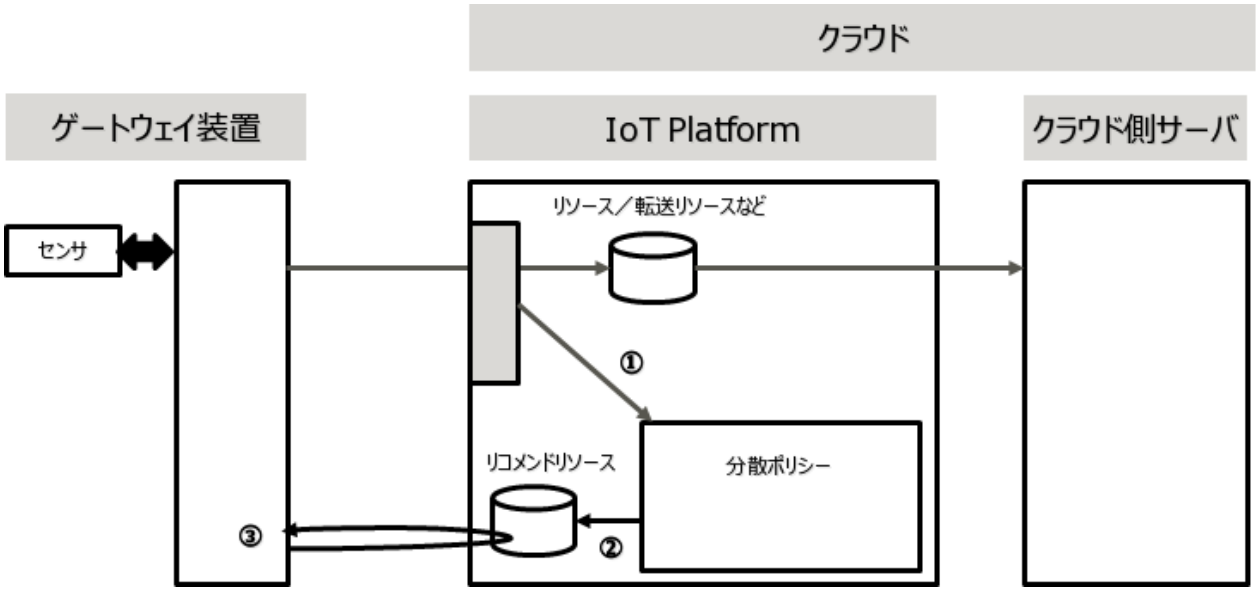


図 7　DRC の基本動作

本サービスでは、本サービスに対する負荷情報を集計し、その値をお客様が設定した条件（エッジコンピューティング条件）と比較してリコメンドとして提供します。お客様はリコメンドを利用することにより、お客様の IT リソース内で最適なデータ収集を行うアプリケーションを開発できます。

DRC の動作の流れは以下の通りです。

① 本サービスが、本サービスにおけるお客様テナントにかかる負荷を監視します。
② 本サービスが、予め設定された分散ポリシーに従い、リコメンドリソースにリコメンドを記載します。
③ ゲートウェイ装置上のアプリケーションが、書き換えられたリコメンドリソースに応じて、処理の内容を変更します。

これにより、負荷に応じて処理の内容を変更できるため、安定したシステム運用が可能になります。

2.2.2. 分散ポリシー

分散ポリシーとは DRC の動作を決定する設定項目です。設定項目には、負荷計測のためのリコメンド周期、算出期間および負荷リソース、リコメンド発行のためのリコメンドリソース、エッジコンピューティング条件およびその解除条件があります。

これらの設定項目を利用し、負荷リソースに記載された負荷情報とお客様が設定したエッジコンピューティング条件を照らし合わせ、条件を満たしていた場合、リコメンドリソースにその結果を記載します。

2.2.3. 負荷計測の動作

負荷計測は、「リコメンド周期」の間隔で、「算出期間」における平均受信量を負荷リソースに書き込みます。算出期間をリコメンド周期よりも大きくすることで、突発的な負荷の増減を平滑化できます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

2.2.4. リコメンドリソース

ゲートウェイ装置に配備するアプリケーションはリコメンドリソースを参照するように開発してください。

参照した際、リコメンド値（recommend_value）が ON であった場合、通常クラウド側サーバで実行する処理をゲートウェイ装置において実施し、送信する通信量を削減してください。これにより、クラウド側サーバへの負荷を低減することができます。

リコメンドリソースに入るリコメンドの JSON データについて説明します。

表 1：リコメンドリソースに入る JSON の Key 名と value の説明

Key 名	value の説明
recommend_value	リコメンド値（"ON"または"OFF"）
recommend_parameter	リコメンド引数（文字列） （サービスポータルにおける分散設定において設定した引数。 但し、本サービスのシステム負荷が極めて大きい場合、"ALM"が入ります）

エッジコンピューティング条件の引数が「recommend is on」である場合において、リコメンド値が「ON」に変化した際にリコメンドリソースに格納される JSON データ例を以下に示します。

```
{
    "recommend_value":"ON"
    "recommend_parameter":"recommend is on"
}
```

2.2.5. 負荷リソース

負荷リソースを参照することにより、本サービスに対してかけている負荷情報を見ることができます。負荷リソースに格納される負荷した JSON データについて説明します。

表 2：負荷リソースに入る JSON の Key 名と value の説明

Key 名	説明
tps	お客様テナントに対する負荷データ（tps)[transaction/second]
bps	お客様テナントに対する負荷データ（bps)[bit/second]

10tps、10,000bps の場合において、負荷リソースに格納される JSON データの例を以下に記載します。

```
{
    "tps":"10",
    "bps":"10000"
}
```

上記 JSON が解釈できるようにアプリケーションを開発して下さい。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第3章　準備するもの

アプリの開発にあたって、以下が必要になります。

3.1. ご契約後の通知内容

ご契約後通知する以下の情報が必要になります。MQTT パスワードはサービスポータルより設定できます。

表 3：ご契約後の通知内容情報一覧

	項目
1	テナントID
2	MQTTブローカーアドレス
3	ベースURI
4	MQTTユーザ名

3.2. アプリ開発環境

データ収集アプリ、データ参照アプリを開発するにあたって、REST（HTTP）ライブラリもしくは、MQTT クライアントライブラリが必要になります。詳細な動作環境については、開発対象となる OS やプログラミング言語の仕様をご確認ください。

本書では、LinuxOS における curl コマンド及び、MQTT クライアントのモジュールを導入した Python 言語のスクリプトにて説明を行います。

Ubuntu 上で curl コマンドを導入する方法を以下に示します。（su 権限にて実行）

```
# apt-get install curl
```

Ubuntu 上で Python 及び MQTT クライアントである Eclipse Paho のモジュールを導入する方法を以下に示します。（su 権限にて実行）

```
# apt-get install python
# apt-get install python-pip
# pip install paho-mqtt
```

3.3. ルート証明書（SSL/TLS 利用時のみ）

本サービスでは、DigiCert 社の「DigiCert SHA2 Secure Server CA」より発行された SSL/TLS サーバ証明書を使用しています。

SSL/TLS 通信を行う場合は同社の「DigiCert Global Root CA」をルート証明書としてご利用下さい。

https://www.digicert.com/digicert-root-certificates.htm#roots

※　URL は 2021 年 1 月現在のものです。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第4章　システム構築の流れ

本サービスを用いて、システム構築をする流れを以下に示します。

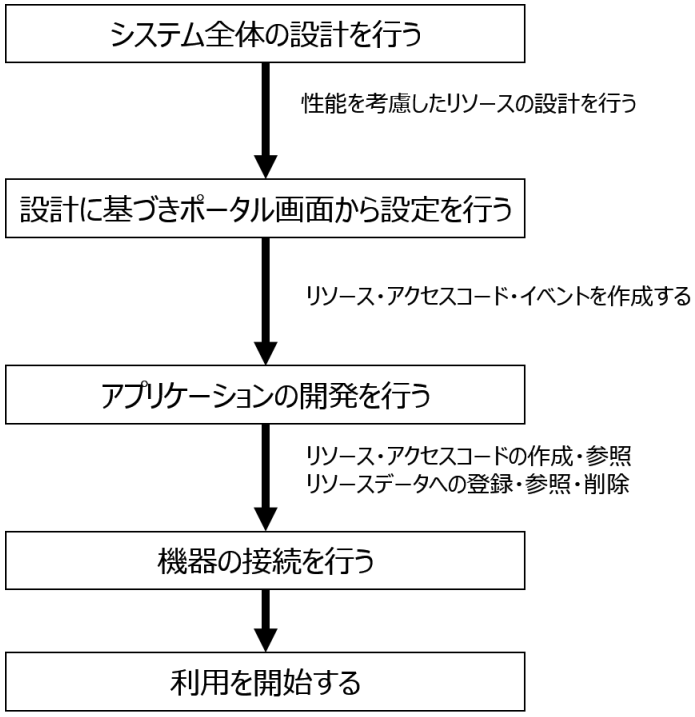


図 8　システム構築の流れ

システムを構築する上で性能やセキュリティを確保するためには、アプリケーションの開発を行う前のシステム全体の設計が重要になります。設計にあたっての考え方については第 5 章をご参照ください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第5章　設計の考え方

本章では、設計時に注意する必要がある項目について記載します。本章をよくお読みの上、リソース、アクセスコード、イベントの設計を行って下さい。

5.1. 登録したデータに付与される情報について

本サービスでは、参照時にデータを分別できるよう、登録した JSON データに必要なパラメータを付与して格納しています。利用する機能によって付与される情報は異なります。本章では具体的な例を用いて、付与される情報を説明します。

[1] JSON による登録

REST にて登録すると GET した際にリソースパスや登録した日付が登録されます。
例を用いて紹介します。まず、以下のリソースとアクセスコードを作成してください。

リソース種別	リソース
リソースパス	area1/device1
データ形式	JSON
保存期間	1 日

アクセスコード	DeviceAccessCode
リソースパス	area1/device1
アクセス権限	「R」と「U」に権限を付与

次に、以下の API を発行し JSON を登録してください。

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DeviceAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/area1/device1.json
BODY	{"sensor1": 100}

200 OK が返却されたことを確認し、以下の API を呼び出し、JSON をご参照ください。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DeviceAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/ area1/device1/_past

レスポンス BODY 部に記載される JSON は以下の通りです。登録したオリジナルの JSON データがネーム”_data”に対する値となり、リソースパス(_resource_path)と登録した日付（_date）がネームとして追加されています。

```
[
 {
  "_data":{
    "sensor1":100
  },
   "_date":"20170214T072509.987Z",
   "_resource_path":"area1/device1"
 }
]
```

レスポンス BODY 部の書式については、「IoT Platform API リファレンス」3.3 章をご参照ください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

[2] イベント（Mail）による登録

イベントを使ってメールを送った場合、メールの添付ファイルに登録された JSON データと操作内容、日付、イベント ID、リソースパスが「eventinfo.txt」というファイルで添付されます。

先述のリソースに対して、メールを飛ばすようイベントを設定した場合、メールに以下のテキストファイル「eventinfo.txt」が添付されます。「eventinfo.txt」の中身は以下の通りです。登録したオリジナルの JSON データがネーム”body”の値となり、他に操作内容(operation)、イベントを発行した日付(date)、イベント ID(event_id)、リソースパス(resource_path)が追加されています。

```
{
  "operation":"create",
  "date":"20170214T072509.99Z",
  "body":{
           "sensor1":100
         },
  "event_id":"58a2b0bc0341",
  "resource_path":"area1/device1"
}
```

メール通知の設定については、「IoT Platform API リファレンス」10.2 章をご参照ください。

[3] リソース転送による登録

リソース（転送）使って外部 API を呼び出した場合、API 受信先に対して、元の JSON データのみが送付されます。例を用いて紹介します。まず、以下のリソースとアクセスコードを作成してください。外部 API についてはお客様自身でご用意願います。

リソース種別	リソース（転送）
リソースパス	_fwd/area1/device1
データ形式	JSON
URI	http://<外部 API>
メソッド	PUT

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	FwdDeviceAccessCode
リソースパス	_fwd/area1/device1
アクセス権限	「R」と「U」に権限を付与

次に、以下の API を発行し JSON を登録してください。

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DeviceAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_fwd/area1/device1.json
BODY	{"sensor1":100}

200 OK が返却されたことを確認し、転送先の外部 API のサーバから受信した JSON をご参照ください。外部 API のあるサーバにて受信したレスポンス BODY 部に記載される JSON は以下の通りです。これはリクエストした JSON と同じです。

```
{"sensor1":100}
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

5.2. リソース_JSON の性能について

「リソース_JSON」の設計を誤ると応答時間が長くなったり、データを取り出しにくくなる場合があります。快適にご利用頂けるようリソースを設計する方法について記載します。

下図のように、複数の端末のデータを 1 つのリソースに集約する場合、データ取得の上限値を超えてしまい取得できない、検索に時間がかかる等、使いにくくなる場合があります。

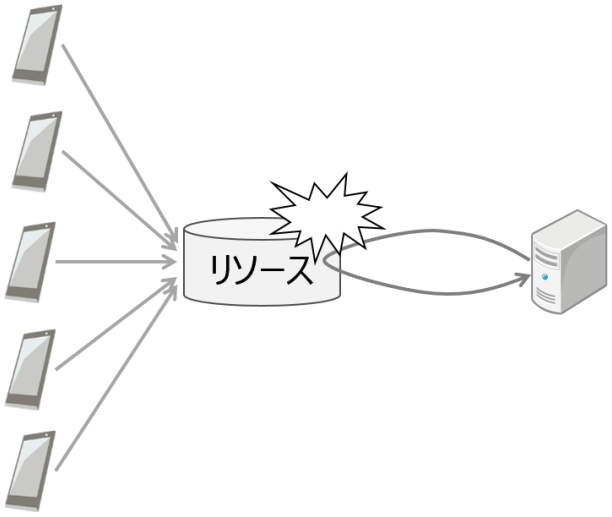


図 9　非推奨の例

下図のように、端末やセンサー 1 つにつき 1 つリソースを作成することで、上記問題を回避できます。システムとして求める要件を考慮し設計して下さい。また、「リソース_JSON」の場合、複数のリソースデータを一回の API 操作によって参照することも可能です。詳細は 8.8 章及び「IoT Platform API リファレンス」3.3 章をご参照ください。

また、filter 条件を用いた検索 API を用いる場合は_date の検索条件を and で追加して時刻範囲を限定した検索条件とすることにで応答時間を高速にすることができます。

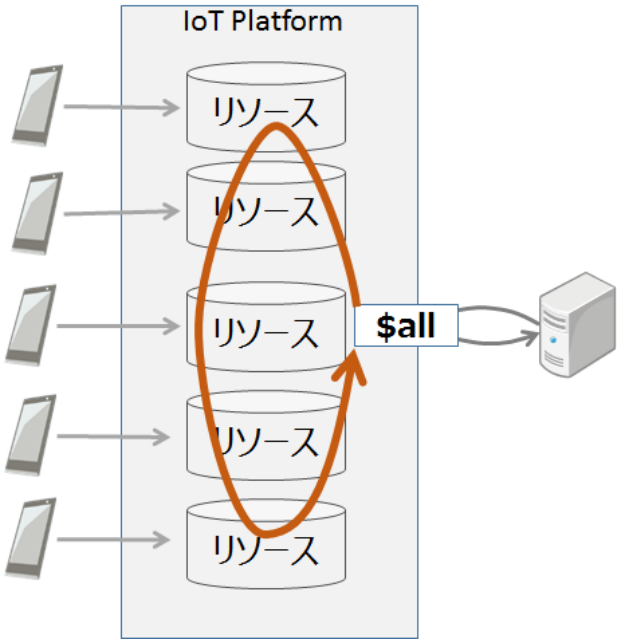


図 10　推奨例(複数のリソースを 1 回の API 操作によって参照する)

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

5.3. 複数リソースにまたがったデータの一括取得について

5.2 章のように、複数のリソースから一度に取得する場合において、アクセスコード「R」権を使うと、該当するすべてのリソースに対して同一のアクセスコードを付与する必要があり、ひとつでもアクセスコードの設定がなされていないとデータを取得できません。

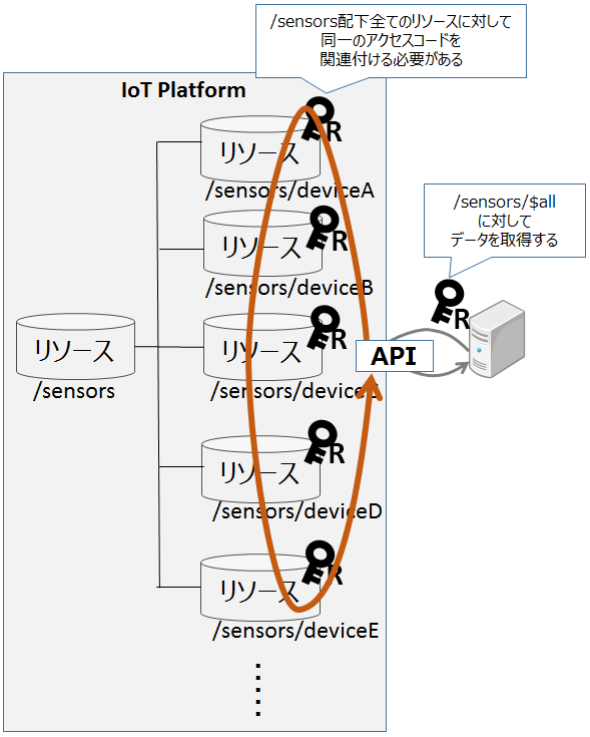


図 11　アクセスコード「R」権を使って複数リソースを参照する場合

アクセスコード「G」権を使うことにより、この問題を解決することができます。以下の図の通り、リソースにアクセスコード「G」権を設定すると、設定されたリソース配下のすべてのリソースデータを参照できます。詳細は、8.8 章をご参照ください。

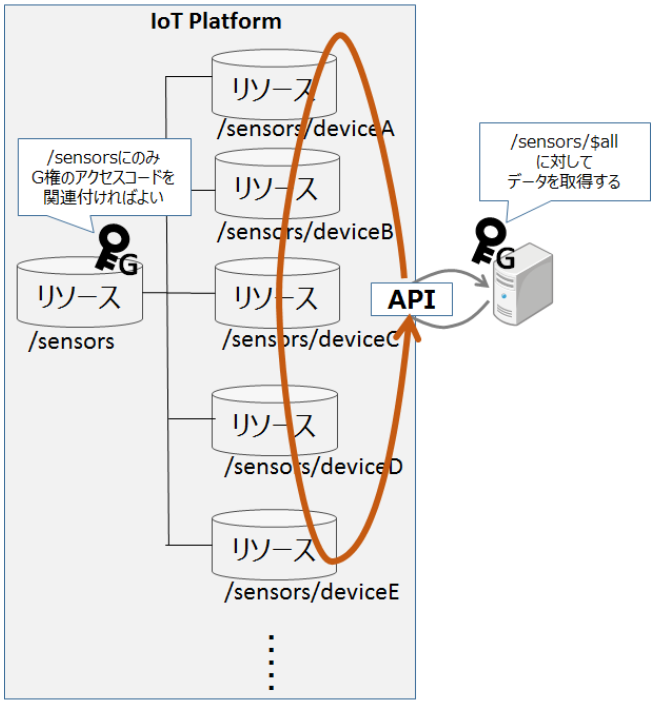


図 12　アクセスコード「G」権を使って複数リソースを参照する場合

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

5.4. アクセスコードにおける「CDL」権の設定について

アクセスコードの設定において、多数のリソースに対して「CDL」権を付与し、リソースのメタデータ参照を行った場合、指定したすべてのリソースのリソース配下に対して検索するため、レスポンスが遅くなる場合があります。

1 つのアクセスコードの設定において、多数のリソースに対して「CDL」権を付与しないようにしてください。

5.5. 「CDL」権を使ったリソースの作成頻度について

「CDL」権をもつアクセスコードによりリソースを作成・削除できますが、1 日に何度も作成・削除を繰り返すことは避けてください。システム全体の性能劣化の原因になる場合があります。

5.6. 転送／イベント時のサーバの設定について

転送機能、イベント機能を用い、本サービスと異なる API を設計する場合、API 受信元のサーバの KeepAlive は ON にしてください。

KeepAlive が OFF になっていた場合、転送／イベントが途中で終了したり、欠損したりする場合があります。

5.7. リソースの公開範囲について

リソースとアクセスコードは公開範囲を意識して設計してください。例えば、1 つのリソースに様々なユーザのセンサーデータを格納した場合、ユーザが別のユーザのセンサーデータを閲覧できてしまい、セキュリティの観点から望ましくありません。

設計時点で公開範囲を意識してリソースを設計してください。

5.8. クライアント認証について

クライアント認証を行うことにより、デバイス単位で認証できるため、より堅牢なアクセス制御が可能となります。クライアント認証は以下の流れで実施します。

1) クライアント証明書の生成

↓クライアント秘密鍵、クライアント証明書が 1 セットになったファイルを生成します。
↓サービスポータル操作マニュアル 4.13 章をご参照ください。
↓

2) クライアント証明書の分離

↓ファイルからクライアント秘密鍵、クライアント証明書を分離します。
↓

3) アクセスコードの設定

↓アクセスコードに予め用意したクライアント証明書を設定します。
↓設定方法についてはサービスポータル操作マニュアル 4.7 章をご参照ください。
↓

4) デバイスにクライアント証明書を設定

↓データを登録・参照するデバイスにクライアント秘密鍵、クライアント証明書を格納します。
↓

5) API を発行

API を発行します。発行時に秘密鍵、クライアント証明書をご参照ください。

実際に、サービスポータルから取得した PKCS#12 形式ファイル（client.p12）からクライアント秘密鍵、クライアント証明書を分離する手順について実例を交えて紹介します。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

PKCS#12 ファイルには、本サービスの CA 証明書、クライアント証明書、クライアント秘密鍵がセットで格納されています。

アクセスコードにクライアント認証の設定をするためには、クライアント証明書が必要になります。また、そのアクセスコードを使って API を発行するためには、クライアント証明書とクライアント秘密鍵が必要になります。それぞれ、実施するためには元の PKCS#12 形式ファイルを分離する必要があります。（但し REST を実行するアプリケーションによっては PKCS#12 形式ファイルのままでも分離せず実行できる場合があります。）

ここでは、PKCS#12 形式ファイルから client.crt（クライアント証明書）と、client.key（クライアント秘密鍵）を分離し生成する方法について例を示します。

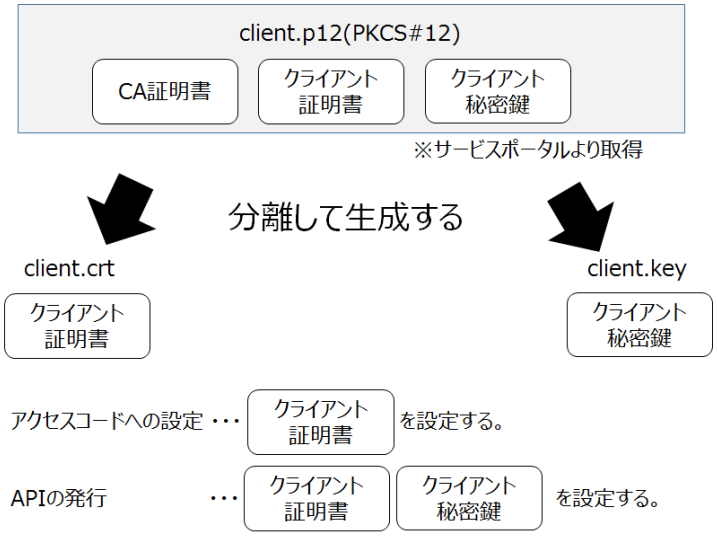


図 13　PKCS#12 を分離する

事前準備として OpenSSL のコマンドをインストールしてください。OpenSSL のインストール方法については公式サイトをご確認下さい。本章では Ubuntu 上で実行する例を使って説明します。

まず、以下のコマンドを実行することにより、PKCS #12 ファイル（例としてここでは 20171011000000.p12 とします。）からクライアント証明書（client.crt)を生成します。

```
$ openssl pkcs12 -in 20171011000000.p12 -clcerts -nokeys -out client.crt
```

ここで生成された client.crt をアクセスコードに設定してください。アクセスコードの設定方法についてはサービスポータル操作マニュアル 4.7 章をご参照ください。

次に以下のコマンドを実行することにより、PKCS #12 ファイル（20171011000000.p12）からクライアント秘密鍵（client.key)を生成します。

```
$ openssl pkcs12 -in 20171011000000.p12 -nocerts -nodes -out client.key
```

API を発行する際、client.key 及び client.crt を使って API を発行してください。curl で実行する場合、引数”--key”,"-E"を使うことで実行できます。以下の例ではアクセスコード ClientAccessCode にクライアント証明書を設定しており、このアクセスコードを使って API を発行しています。

```
$ curl -X PUT -H 'Authorization:Bearer ClientAccessCode' 'https://<zone>.fujitsu.com/v1/<テナント ID>/clientResource.json' -d '{"sensor":10}' -E <client.crt を格納しているファイルパス>/client.crt --key <client.key を格納しているファイルパス>/client.key
```

クライアント証明書の管理（配布の管理や期限切れの管理・対応）については、お客様責任となりますのでご注意ください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

5.9. CORS の活用について

通常、JavaScript 等の機能を使うとブラウザから API アクセスを行うことができません。

CORS(Cross-Origin Resource Sharing)の設定を有効にすることにより、JavaScript による通信が可能となります。設定方法については、サービスポータル操作マニュアル 4.14.2 章をご参照ください。

5.10. イベント、転送失敗時のログ参照について

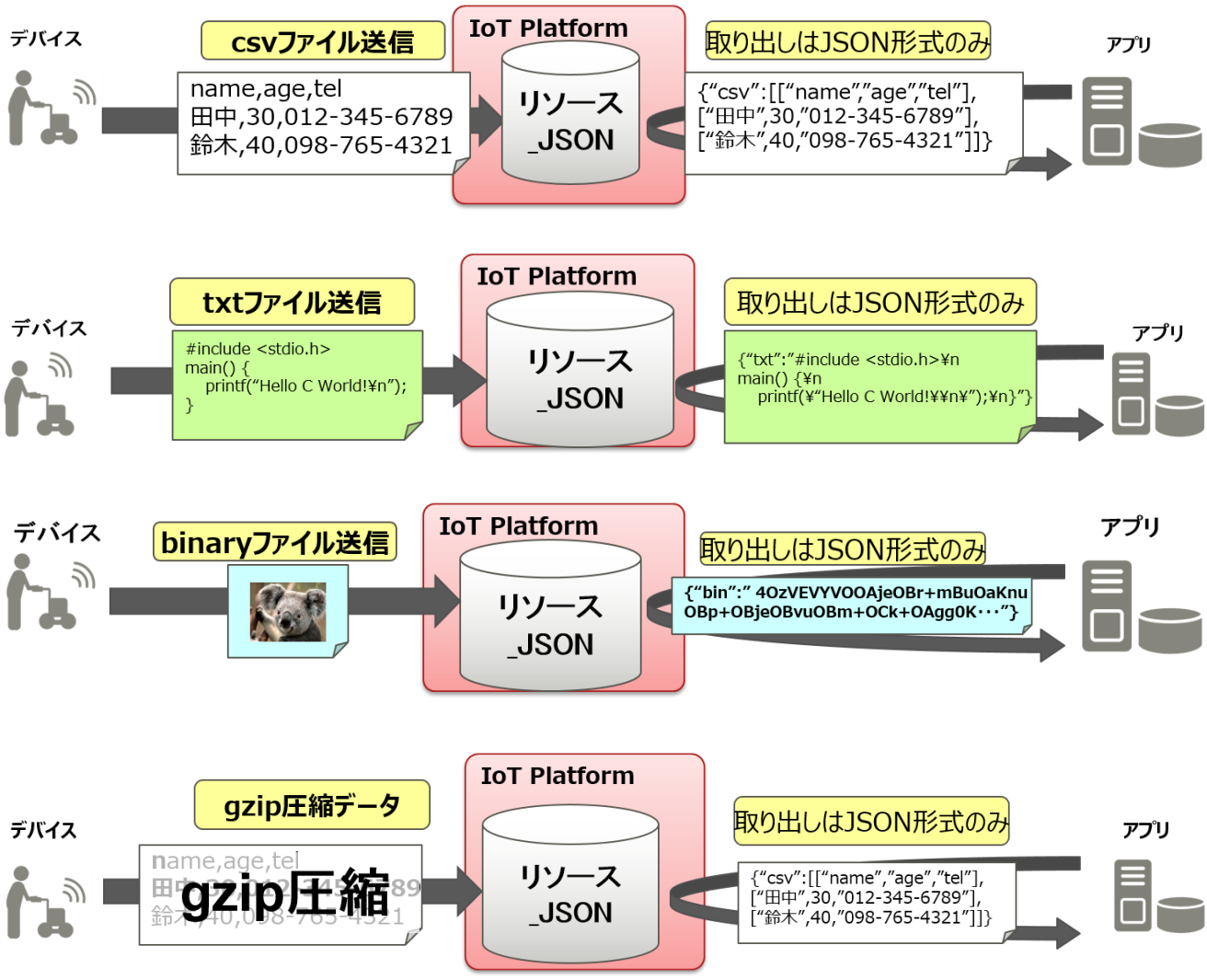
イベントやリソース（転送）、MQTT を使うことにより、お客様が使用する別サーバと連携することができます。この機能を使って何らかの事情でお客様サーバとの通信が実行できなかった場合、エラー収集用リソース「_error」を参照することにより、エラーの原因を確認することができます。

エラー収集用リソース「_error」に対してはアクセスコードの設定が必要です。設定方法については、サービスポータル操作マニュアルをご参照ください。また、エラー収集用リソースに格納されるデータの詳細については 8.12 章及び「IoT Platform API リファレンス」をご参照ください。

5.11. JSON 変換について

本サービスでは csv ファイル、txt ファイル、binary ファイル、gzip ファイルをリソース_JSON に登録することができます。登録されたデータは本サービスにて JSON 形式へ変換後、格納されます。csv ファイルなどを圧縮した gzip ファイルについては解凍後 JSON 形式に変換され格納されます。

登録されたデータを参照・検索する際は JSON 形式でのみ取得可能となりますので、設計の際はご注意ください。



All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第6章　システム設計の具体例

6.1. 多数のセンサーから収集、参照する場合

東京蒲田地区にある複数のセンサーを収集する例を用いて設計方法を紹介します。また、状況によってセンサーの数は動的に増減する場合があり、API によってリソースを随時追加・削除したいものとします。

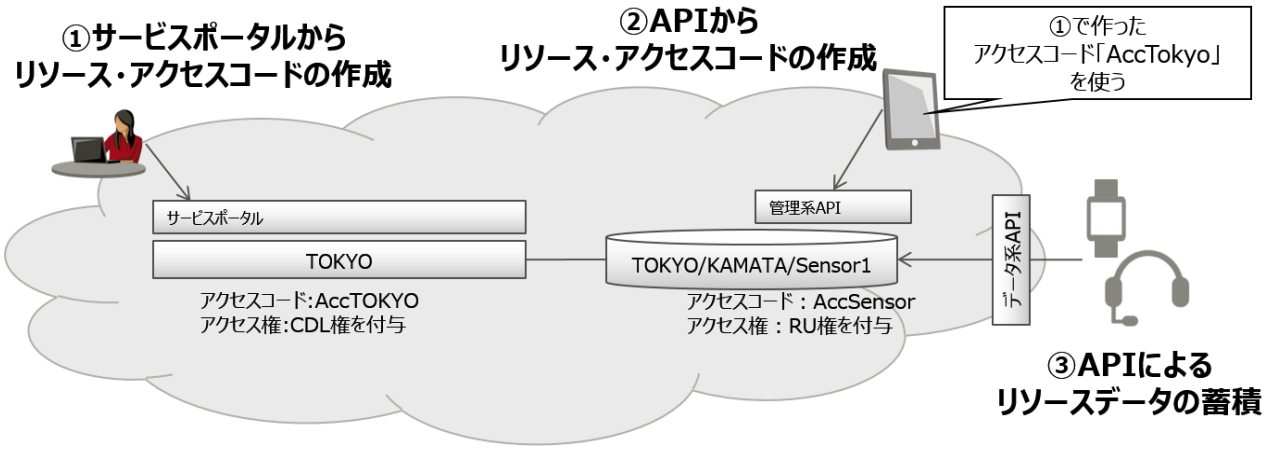


図 14　システム設計の具体例

[1]サービスポータルによるリソース、及びそのアクセスコードの作成

まず、はじめにリソースとリソースに対して CDL 兼を有するアクセスコードをサービスポータルより作成します。ここでは以下を作成します。

作成する親リソースのリソースパス	TOKYO

作成するアクセスコード	AccTOKYO
設定するリソース１	TOKYO
設定するリソース１への アクセス権限	「CDL」権を付与

[2]API によるリソース及びアクセスコードの作成

次に API を用いて、蒲田にあるセンサー情報を格納する以下のリソースとアクセスコードを作成します。

作成するリソース	TOKYO/KAMATA/Sensor1

作成するアクセスコード	AccSensor
設定するリソース１	TOKYO/KAMATA/Sensor1
設定するリソース１への アクセス権限	「R」「U」に権限を付与

リソースを作成する場合の API のパラメータを以下に記載します。
BODY 文に記載する内容は 11.1 章及び「IoT Platform API リファレンス」第 7 章をご参照ください。

メソッド	POST
ヘッダフィールド名	Authorization:Bearer AccTOKYO
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/ TOKYO/KAMATA/Sensor1

正常にリソースが作成できた場合、201 が返却されます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

このように親リソースに対して CDL 権を持っているアクセスコード「AccTOKYO」を指定する必要があります。

同様にアクセスコードを作成する場合の API のパラメータを以下に記載します。
BODY 文に記載する内容は 12.1 章及び「IoT Platform API リファレンス」第 8 章を参照ください。

メソッド	POST
ヘッダ	Authorization:Bearer AccTOKYO
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccSensor

正常にアクセスコードが作成できた場合、201 が返却されます。
このように親リソースに対して CDL 権を持っているアクセスコード「AccTOKYO」を指定する必要があります。

[3]リソースに対するリソースデータの蓄積

これにより、「TOKYO/KAMATA/Sensor1」に対して以下のようにリソースデータを登録できるようになります。BODY 文に記載する内容は 8.1 章または「IoT Platform API リファレンス」3.1 章をご参照ください。

メソッド	PUT
ヘッダ	Authorization:Bearer AccSensor
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/TOKYO/KAMATA/Sensor1

正常にリソースデータが登録できた場合、200 OK が返却されます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第7章　仕様制限（注意事項）

7.1. API について

7.1.1. 対応プロトコル

本サービスでは、データの登録・転送・更新・参照・削除（以下、操作）のためのインターフェースとして、REST（HTTP）と MQTT をサポートしています。REST（HTTP）は、従来から広く用いられているプロトコルのため、既に稼働しているシステムに簡便に適用することが可能になります。MQTT は、REST と比較し軽量なプロトコルです。センサーデータを送信する頻度が高い場合、MQTT を用いることにより、機器（スマートフォン、ゲートウェイ装置など）にかかる電力消費や CPU 負荷を小さくすることが可能になります。クライアントとして Eclipse Paho などがあります。

また MQTT は PUBLISH／SUBSCRIBE 機能を有しています。SUBSCRIBE を活用することにより、機器に対して通知や制御を行うアプリを作ることができます。

本サービスは API を使用する際 SSL/TLS 通信を使用することが可能です。SSL/TLS を用いて通信することで、セキュアかつ安全にデータアクセスすることができます[1]。

7.1.2. トランザクション非保証

本サービスでは高性能を実現するために非同期なアーキテクチャーを採用しており、リソースデータ等のユーザデータに関しての操作は、トランザクションを保証しません。そのため、強い ACID 属性（トランザクション処理において必要とされる Atomicity（原子性）、Consistency（一貫性）、Isolation（独立性）、Durability（永続性）のこと。）が必要な場合は、上位アプリケーションでの確認再送等の対応が必要です。

- トランザクションの順序の非保証
  例）連続的にデータを受信した場合、データ受信順序と異なる順序で蓄積記録される場合があります。
  例）ユーザデータの受信を契機にイベントが発生する場合、データ受信順序と異なる順序でイベントが発生する場合があります。

- トランザクション完了の非保証
  例）障害発生タイミングによっては、受信完了したデータであっても、蓄積記録されない場合があります。
  例）障害発生タイミングによっては、受信完了したデータが複数回蓄積記録される場合があります。

7.1.3. 同時アクセス

- 本サービスの全ての API はスレッドセーフであり同時並行的なアクセスが可能です。
- ただしその場合、前項に従い実行順序は保証されません。
- 同一の「リソース_JSON」のデータに対して、全く同時刻に複数の登録があった場合、すべてのアクセスが記録されます。上書きはされません。
- 同一の「リソース_Binary」のデータに対して、全く同時刻に複数の登録があった場合、1 つのみ保持され、他データは上書きされます。

[1] SSL 利用料は API 利用料と比べ、高価になっていますのでご注意下さい。また、API 使用料、SSL 使用料、イベント利用料は従量課金制となっています。各種料金プランに無料利用分を用意しております。使用状況が大きく変わる場合、料金プランの見直し・変更をご検討下さい。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

7.1.4. API で使用する文字コード

UTF-8 のみサポートしています。

7.1.5. 時刻について

世界標準時　+00:00(UTC)を採用しています。

7.1.6. クラウド側サーバの証明書について

転送リソースやイベントを用いて外部サーバに対して HTTPS 通信を行う場合、外部サーバの内部にサーバ証明書を配備する必要があります。この時、サーバ証明書として公的 CA の署名がない証明書を使うことはできません。

7.1.7. データの到達性

本サービスではリクエスト受信応答を返した後に、サービス内で欠損する場合があります。特に転送やイベント送信において、成功のリクエスト応答後転送先に送信を試みますが、欠損する可能性があります。

7.1.8. クラウド側サーバにおける API のセッション数について

転送やイベント送信では、同じリソースであっても、毎回別のセッションを使って送信します。転送先の API 側は転送された分だけセッションを作成できるように実装してください。

7.1.9. 同一リソースに対する REST／MQTT の両立について

「リソース_JSON」では、REST で登録したリソースデータを MQTT にて SUBSCRIBE により参照することができます。但し、「リソース_Binary」においては REST で登録したリソースデータを MQTT にて SUBSCRIBE により参照することができません。

7.1.10. 転送・イベント機能利用時の外部サーバの設定について

転送機能またはイベント機能において外部サーバの API を呼び出す場合、連携先サーバには HTTP の Response ヘッダに「Content-Length」を設定する必要があります。

7.1.11. 目安を超えた使い方をする場合の対応

データの登録や参照頻度について、以下の各要素に対する目安を 1 つでも超えた使い方を想定される場合は、お客様の利用シーンが実現できずご利用いただけない場合があります。当社営業経由で事前にご相談ください。

＜目安＞

1 通信におけるデータ量		256KByte
通信頻度（ピーク）	データ系 API	100 回／秒
	管理系 API※	1 回／秒
登録リソース数		10,000 個
1 リソース内データ蓄積容量		100Mbyte
同時接続数		100

※リソースやアクセスコード、イベントを作成・更新・削除・参照する API を指します。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

7.2. REST

7.2.1. 取得データ数の制限について

取得データが以下のいずれかの条件を満たしていた場合、取得できません。

・取得データ件数が 1001 件以上の場合
・「リソース_JSON」の取得データ量が 16MB より大きい場合
・「リソース_Binary」の取得データ量が 100MB より大きい場合

上記の条件を満たし取得できなかった場合、レスポンス BODY 文に取得可能な件数が記載されます。件数を top にて指定し再度 API を発行してください。レスポンス BODY 文の詳細については「IoT Platform API リファレンス」の付録 1 を参照して下さい。

7.3. MQTT

7.3.1. アクセスコードへのアクセス制限設定時の MQTTS について

アクセスコードに IP アドレスによるアクセス制限の設定を行った場合、そのアクセスコードを使った MQTTS による通信はできません。

7.3.2. QoS について

MQTT の QoS に対応していますが、受信後のシステム内処理中のエラーによっては、保障されないことがあります。

7.3.3. コネクション断時の振る舞い

MQTT コネクションはサーバの保守等で切れた場合、retain データが消える場合があることを考慮してください。

7.3.4. retain フラグ付与について

REST および MQTT のそれぞれから retain フラグを立て登録すると、それぞれ独立して保持します。

この場合、MQTT クライアントが該当トピックに対し SUBSCRIBE を行うと、両方のメッセージが PUBLISH されます。

7.3.5. keepalive について

MQTT の keepalive により接続を維持する場合において、通信網によっては通信網の設定により切断される可能性があります。

7.3.6. 再接続時のクライアント ID について

MQTT 通信時にコネクションが切れる場合がありますが、この後即座に再接続を行う場合は切断前と同じクライアント ID を利用して接続をおこなうよう MQTT クライアントアプリケーションを設計してください。

7.4. その他

7.4.1. 1 つのアクセスコードに紐づけられるリソース数

1 つのアクセスコードに紐づけられるリソースの数は 1,000 件までです。

7.4.2. 本誌に記載がない場合について

現状仕様が優先されます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第8章　リソースデータの制御（REST）

本サービスのリソースデータを API で制御する例をご紹介します。実際のコマンド入力やプログラムの実行を通して、リソースデータの操作を行います。

ご契約後の通知内容をご用意ください。本章では以下で説明しますが、必要に応じて読み替えて実行して下さい。

テナント ID	<テナント ID>
ベース URI	http://<zone>.fujitsu.com/

8.1. リソースデータの登録例

「リソース_JSON」に対して、JSON データを登録することができます。

ここでは「出勤管理（富士通花子）」のリソースに対して出勤情報のデータを登録する API を呼び出し、そのデータが実際に転送されることを確認するケースを紹介します。今回のケースにおいて、出退勤はリソースデータの変数「diligence」で管理され、「1」が出勤、「0」が休みを意味するように管理するものとします。今回の例では、2015 年 8 月 3 日 UTC 午前 9 時に富士通花子さんが出勤したことを登録するものとします。

事前に以下のリソースをサービスポータル画面から作成してください。

リソース種別	リソース
リソースパス	diligence/status/hanako
データ形式	JSON
保存期間	1 日

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	DiligenceManagement22222
リソースパス	diligence/status/hanako
アクセス権限	「R」と「U」に権限を付与

アクセス権限の R は読み込み権限、U は書き込み権限を示します。

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako.json?$date=20150803T090000Z
BODY	{"diligence":1}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer DiligenceManagement22222' -d '{"diligence":1}' '
http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako.json?$date=20150803T090
000Z'
```

正常完了できた場合、レスポンスとして「200 OK」が返ります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

8.2. リソースデータの転送例

「リソース（転送）_JSON」を用いることに寄って JSON データを転送先へ転送することができます。

ここでは「出勤管理（富士通太郎）」のリソース（転送用）に対して出勤情報のデータを登録する API「http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako」を呼び出し、そのデータが実際に登録されていることを確認するケースを紹介します。今回のケースにおいて、出退勤はリソースデータの変数「diligence」で管理され、「1」が出勤、「0」が休みを意味するように管理するものとします。今回の例では、2015 年 8 月 3 日日本時間午前 9 時に富士通太郎さんが出勤したことを登録するものとします。

事前に以下のリソースを作成してください。転送先として便宜上、本サービスの 8.1 章のリソースに向けて転送をしていますが、他のサーバをお使いいただいても構いません。また、MQTT Subscribe を行うクライアントに対してのみ転送したい場合 URI の指定をする必要はありません。

リソース種別	リソース（転送）
リソースパス	_fwd/diligence/status/taro
データ形式	JSON
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako
メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	DiligenceManagement11111
リソースパス	_fwd/diligence/status/taro
アクセス権限	「R」と「U」に権限を付与

アクセス権限の R は読み込み権限、U は書き込み権限を示します。
API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement11111
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_fwd/diligence/status/taro.json
BODY	{"diligence":1}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X PUT -H 'Authorization:Bearer DiligenceManagement11111' -d '{"diligence":1}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_fwd/diligence/status/taro.json'
```

正常完了できた場合、レスポンスとして「200 OK」が返ります。

Memo

リソース（転送用）を用いた転送におけるレスポンスは、REST を発行したアプリケーションから本サービスへ到達した事を示すものであり、本サービスから転送先への到達を保証するものではありません。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

8.3. リソースデータの更新例

「リソース_JSON」に対して JSON データを更新することができます。

ここでは「出勤管理（富士通花子）」のリソースに対して出勤情報のデータを更新する API を呼び出し、既に登録されていたデータが実際に更新されていることを確認するケースを紹介します。

リソースパスの後に「_past(20150803T090000Z).json」を追記することにより、8.1 章で登録したリソースデータを更新できます。

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past(20150803T090000Z).json
BODY	{"diligence":0}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter 以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer DiligenceManagement22222' -d '{"diligence":0}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past(20150803T090000Z).json?$newdate=20150803T100000Z'
```

登録が正常完了できた場合、レスポンスとして「200 OK」が返ります。

Memo

既に同じ時刻において、リソースデータが複数登録されていた場合、複数のうちの 1 つのリソースデータのみが更新されますので、ご注意下さい。

8.4. リソースデータの参照例

「リソース_JSON」における JSON データを参照することができます。

ここでは、出勤管理（富士通花子）」のリソースに対して出勤情報のデータを参照する API を呼び出し、最後に登録したデータを参照するケースを紹介します。

事前に 8.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。API を呼び出すのに必要なパラメータは以下のとおりです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_present.json

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X GET -H 'Authorization:Bearer DiligenceManagement22222' 'http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_present.json'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

結果は整形されません。

```
[
  {
    "_data":{
      "diligence":0
    },
    "_date":"20150803T090000.000Z" ,
    "_resource_path":"diligence/status/hanako"
  }
]
```

参照時の JSON データは以下の構造になっています。

Key	Value
_data	<拡張子>に従った形式の登録済みのデータ
_date	対象データの登録日時

8.5. リソースデータの検索例

「リソース_JSON」における JSON データを検索することができます。

ここでは「出勤管理（富士通花子）」のリソースに対して出勤情報のデータを検索する API を呼び出し、8 月の出勤状況を確認するケースを紹介します。

事前に 8.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。

API を呼び出すのに必要なパラメータは以下の通りです。filter 条件の演算子を用いることにより、リソースデータの範囲指定が可能です。ここでは、2015 年 8 月 1 日～8 月 31 日までの出勤状況を検索しています。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past?$filter=_date gt 20150801T000000Z and _date lt 20150901T000000Z

LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X GET -H 'Authorization:Bearer DiligenceManagement22222' 'http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past?$filter=_date%20gt%2020150801T000000Z%20and%20_date%20lt%2020150901T000000Z'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

```
[
 {
  "_data":{
  "diligence":1
  },
  "_date":"20150803T100000.000Z" ,
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
  "_resource_path":"diligence/status/hanako"

 }
]
```

8.6. リソースデータの削除例

「リソース_JSON」に登録した JSON データを削除することができます。

ここでは、2015 年 8 月 1 日～8 月 31 日までの出勤状況を削除しています。事前に 8.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	DELETE
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer DiligenceManagement22222
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past?$filter=_date gt 20150801T000000Z and _date lt 20150901T000000Z

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X DELETE -H 'Authorization:Bearer DiligenceManagement22222' 'http://<zone>.fujitsu.com/v1/<テナント ID>/diligence/status/hanako/_past?$filter=_date%20gt%20150801T000000Z%20and%20_date%20lt%20150901T000000Z'
```

リソースデータの削除が正常に行えた場合、レスポンスとしてヘッダに「200 OK」が返ります。

8.7. リソースデータの一括登録例(bulk insert)

bulk insert 機能を使うことにより、「リソース_JSON」における JSON データを一括で登録することができます。

ここでは身に着けているデバイスの温湿度データを登録する API を呼び出すケースを紹介します。今回の例では、2016 年 10 月 1 日午前 9 時～午前 10 時において 1 分毎に取得したユーザ ID 0001 が身につけていたデバイスの温湿度を一括で送信します。(煩雑になるため、以下の説明では午前 9 時～午前 9 時 5 分までとしています。)

事前に以下のリソースをサービスポータル画面から作成してください。

リソース種別	リソース
リソースパス	users/0001/temperature
データ形式	JSON
保存期間	1 日

事前に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	TemperatureAccessCode
リソースパス	users/0001/temperature
アクセス権限	「R」と「U」に権限を付与

アクセス権限の R は読み込み権限、U は書き込み権限を示します。

API を呼び出すのに必要なパラメータは以下の通りです。bulk=single_resource_path が Bulk Insert を使用するためのフラグになります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer TemperatureAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/users/0001/temperature.json?**$bulk=single_resource_path**

BODY	[
	{
	"_date":"20161001T090000Z",
	"_data":{
	"temperature":25.0,
	"moisture":10.0
	}
	},{
	"_date":"20161001T090100Z",
	"_data":{
	"temperature":25.3,
	"moisture":9.0
	}
	},{
	"_date":"20161001T090200Z",
	"_data":{
	"temperature":26.0,
	"moisture":12.0
	}
	},{
	"_date":"20161001T090300Z",
	"_data":{
	"temperature":27.0,
	"moisture":17.0
	}
	},{
	"_date":"20161001T090400Z",
	"_data":{
	"temperature":29.0,
	"moisture":19.0
	}
	}
	]

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer TemperatureAccessCode' -d '<上記 BODY 文>' 'http://<zone>.fujitsu.com/v1/<テナント ID>/users/0001/temperature.json?$bulk=single_resource_path'
```

正常完了できた場合、レスポンスとして「200 OK」が返ります。
Bulk Insert はイベント処理に対応していません。また、JSON 配列の中に格納できる JSON の数は 1000 個までです。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

8.8. 複数のリソースデータを 1 回の API 操作で参照する例($all)

クエリに$all を用いることにより、「リソース_JSON」における複数の JSON データの一回の API 操作で参照することができます。ここでは身に着けているデバイスの温湿度データの中から 10 月 1 日午前 9 時～午前 9 時半の情報のみを抽出する API を呼び出すケースを紹介します。今回の例では、8.7 章で定義したリソースに加え、以下のようなリソースが定義され、それぞれに温湿度のリソースデータが格納されているものとします。

事前に以下のリソースをサービスポータル画面から作成してください。

リソース種別	リソース
リソースパス	users/0001/temperature
データ形式	JSON
保存期間	1 日

リソース種別	リソース
リソースパス	users/0002/temperature
データ形式	JSON
保存期間	1 日

リソース種別	リソース
リソースパス	users/0003/temperature
データ形式	JSON
保存期間	1 日

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。
また転送先 URI についてはサービスポータルにて登録してください。

アクセスコード	TemperatureAccessCode
設定するリソース	Users
設定するリソースへのアクセス権限	「P」と「G」に権限を付与

アクセス権限の G はリソース配下全ての読み込み権限、P はリソース配下全ての書き込み権限を示します。上記の通り設定したアクセスコードを使うと users 配下は読み書き可能となるため、users/0001/temperature, users/0002/temperature, users/0003/temperature はすべて読み書きが可能となります。

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer TemperatureAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/users/**$all**/_past? $filter=_date gt 20161001T090000Z and _date lt 20161001T093000Z
BODY	なし

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X GET -H 'Authorization:Bearer TemperatureAccessCode' 'http://<zone>.fujitsu.com/v1/<テナント ID>/users/$all/_past?$filter=_date%20gt%2020161001T090000Z%20and%20_date%20lt%20020161001T093000Z'
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

正常完了できた場合、レスポンスとして「200 OK」が返り以下のような JSON が返却されます。


```
[
  {
  "_data":{
    "temperature":25.0,
    "moisture":10.0
   },
  "_date":"20161001T090000Z",
   "_resource_path":"users/0001/temperature"
  },{
  "_data":{
    "temperature":39.0,
    "moisture":30.0
   },
  "_date":"20161001T090000Z",
   "_resource_path":"users/0002/temperature"
  },{
  "_data":{
    "temperature":0.0,
    "moisture":10.0
    },
  "_date":"20161001T090000Z",
  "_resource_path":"users/0003/temperature"
 },{
  "_data":{
  "temperature":25.3,
   "moisture":9.0
   },
   "_date":"20161001T090100Z",
   "_resource_path":"users/0001/temperature"
 },{
  "_data":{
    "temperature":40.1,
    "moisture":10.0
   },
  "_date":"20161001T090100Z",
   "_resource_path":"users/0002/temperature"
 },{
  "_data":{
    "temperature":-3.0,
    "moisture":10.0
   }
  "_date":"20161001T090100Z",
    "_resource_path":"users/0003/temperature"
 },{
(以下略)
 }
]
```


All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

8.9. リソースデータの特定の name のみを参照する例(select)

「リソース_JSON」に格納した複数の name の中から特定の name のみを参照することができます。8.7 章で登録したデータの中から、温度のみを参照するケースを紹介します。

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer TemperatureAccessCode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/users/0001/temperature/_past?$select=temperature
BODY	なし

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」(半角スペース)は「%20」、「+」は「%2b」、「'」(シングルクォーテーション)は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X GET -H 'Authorization:Bearer TemperatureAccessCode' 'http://<zone>.fujitsu.com/v1/<テナント ID>/users/0001/temperature/_past?$select=temperature'
```

正常完了できた場合、レスポンスとして「200 OK」が返り以下のような JSON が返却されます。

```
[
 {
  {
    "_data":{
    "temperature":25.0
   },
   "_date":"20161001T090000Z",
   "_resource_path":"users/0001/temperature"
  },{
  "_data":{
  "temperature":25.3
   },
     "_date":"20161001T090100Z",
     "_resource_path":"users/0001/temperature"
 },{
(以下略)
 }
]
```

8.10. csv/txt/binary/gzip ファイルを「リソース_JSON」へ登録する例、及びその登録された JSON データ参照する例

csv ファイル、txt ファイル、binary ファイル、gzip ファイルを「リソース_JSON」に登録することができます。登録されたデータを参照・検索する際は JSON 形式でのみ取得可能となります。ここでは各ファイルの登録及び参照方法について紹介します。

事前に以下のリソースをサービスポータル画面から作成してください。

リソース種別	リソース
リソースパス	convert
データ形式	JSON

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

保存期間	1 日

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	ConvertAcc
リソースパス	convert
アクセス権限	「R」と「U」に権限を付与

アクセス権限の R は読み込み権限、U は書き込み権限を示します。
API を呼び出すのに必要なパラメータは以下の通りです。

[1] csv ファイルを登録及び参照する場合

＜登録＞

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert.csv
BODY	登録する CSV ファイル(file.csv) ＜file.csv の内容＞ name,age,tel 田中,30,012-345-6789 鈴木,40,098-765-4321

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer ConvertAcc' --data-binary @file.csv 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert.csv'
```

リソースデータの登録が正常に行えた場合、レスポンスとしてヘッダに「200 OK」が返ります。

＜参照＞

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json
BODY	なし

```
$ curl -i -X GET -H 'Authorization:Bearer ConvertAcc' 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
[
  {
    "_data":{
      {"csv":[["name","age","tel"],
        ["田中",30,"012-345-6789"],
        ["鈴木",40,"098-765-4321"]]
      },
    "_date":"20170901T095721.680Z",
    "_resource_path":"convert"
  }
]
```

[2] txt ファイルを登録及び参照する場合

＜登録＞

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert.txt

BODY	登録するテキストファイル(file.txt)
	
	＜file.txt の内容＞
	#include <stdio.h>
	main() {
	printf("Hello C World!¥¥n");
	}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「’」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -I -X PUT -H 'Authorization:Bearer ConvertAc' --data-binary @file.txt 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert.txt'
```

リソースデータの登録が正常に行えた場合、レスポンスとしてヘッダに「200 OK」が返ります。

＜参照＞

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json
BODY	なし

```
$ curl -i -X GET -H 'Authorization:Bearer ConvertAcc' 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
[
  {
    "_data":{
      {"txt":"#include <stdio.h>¥nmain() {¥nprintf(¥"Hello CWorld!¥¥n¥");¥n}"
      },
    "_date":"20170901T105721.680Z",
    "_resource_path":"convert"
  }
]
```


[3] binary ファイルを登録及び参照する場合

＜登録＞

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert.bin
BODY	登録する JPEG ファイル(koala.jpg)

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer ConvertAcc' --data-binary @./koala.jpg 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert.bin'
```

リソースデータの登録が正常に行えた場合、レスポンスとしてヘッダに「200」が返ります。

＜参照＞

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json
BODY	なし

```
$ curl -i -X GET -H 'Authorization:Bearer ConvertAcc' 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。


```
[
  {
    "_data":{
      {"bin":"4OzVEVYVOOAjeOBr+mBuOaKnuOBp+OBjeOBvuOBm+OCk+OAggOK・・・"
      },
    "_date":"20170901T115721.680Z",
    "_resource_path":"convert"
  }
```


All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
]
```

[4] gzip ファイルを登録及び参照する場合

＜登録＞

HTTP メソッド	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert.json.gz
BODY	登録する gzip 圧縮データ(file.gz)

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「’」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X PUT -H 'Authorization:Bearer ConvertAcc' --data-binary @./file.gz 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert.json.gz'
```

リソースデータの登録が正常に行えた場合、レスポンスとしてヘッダに「200 OK」が返ります。

＜参照＞

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ConvertAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json
BODY	なし

```
$ curl -i -X GET -H 'Authorization:Bearer ConvertAcc' 'http://<zone>.fujitsu.com/v1/<テナント ID>/convert/_present.json'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

```
[
  {
    "_data":{
      {"csv":[["name","age","tel"],
        ["田中",30,"012-345-6789"],
        ["鈴木",40,"098-765-4321"]]
      },
    "_date":"20170901T095721.680Z",
    "_resource_path":"convert"
  }
]
```

その他、JSON 変換の詳細については「IoT Platform API リファレンス」をご参照ください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

8.11. 配列の要素位置を指定して検索する例

配列の要素位置を指定して検索することが可能です。ここでは、その検索方法について紹介します。まず、事前に以下のリソースをサービスポータル画面から作成してください。

リソース種別	リソース
リソースパス	Search
データ形式	JSON
保存期間	1 日

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	SearchAcc
リソースパス	Search
アクセス権限	「R」と「U」に権限を付与

リソースとアクセスコードの作成が完了したら、①～④のリソースデータ（配列表記の JSON データ）を登録してください。

```
① {"data":[["name","age"],["田中",45]]}

② {"data":[["name","age"],["山口",26]]}

③ {"data":[["name","age"],["佐藤",37]]}

④ {"data":[["name","age"],["野地",26]]}
```

Memo

＜配列の考え方及び表記方法＞
上記リソースデータにおいて”age”は「配列名”data”の 1 番目の行（0 行目）、2 番目の列（1 列目）」に位置することになります。また、今回のような 2 次元配列では”age”の場所は下記のような表記となります。

data.0.1

登録したリソースデータは「1 行目に項目（"name","age"）、2 行目にその項目に対するデータ」を配列表記の JSON データとして表したものです。ここでは 26 歳の人のリソースデータのみを検索したいとします。filter 条件で要素の位置を JSONPath 表記で指定することにより、該当するリソースデータの検索ができます。その場合の API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer SearchAcc
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/search/_past?$filter=data.1.1 eq 26’

LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X GET -H ’Authorization: Bearer SearchAcc ’ ’http://<zone>.fujitsu.com/v1/<テナント ID>/search/_past?$filter=data.1.1%20eq%2026’
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

```
[
  {
    "_data":{
      "data":[["name","age"],
        ["野地",26]]
    },
    "_date":"20170901T095721.680Z",
    "_resource_path":" search"
  },{
    "_data":{
      "data":[["name","age"],
        ["山口",26]]
    },
    "_date":"20170901T095647.245Z",
    "_resource_path":" search"
  }
]
```

8.12. エラー収集用リソースの設定および参照する例

イベントやリソース（転送）、MQTT を使うことにより、お客様が使用する別サーバと連携することができます。この機能を使って何らかの事情でお客様サーバとの通信が実行できなかった場合、エラー収集用リソース「_error」を参照することにより、エラーの原因を確認することができます。

ここではエラーの原因を確認する方法について紹介します。まず、事前にエラー収集用リソースに対して、以下のアクセスコードとアクセス権限を付与してください。

アクセスコード	ErrorAccRU
リソースパス	_error
アクセス権限	「R」と「U」に権限を付与

次にポータル画面の「共通設定」タブ→「エラー収集」タブを選択し、下記設定を行います。

エラー収集の有効/無効		有効
収集保存先リソース	リソースパス	_error
	アクセスコード	ErrorAccRU

更新ボタンをクリックし、設定は完了です。設定完了後、エラーが発生するとその情報がエラー収集用リソースに格納されるようになります。

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer ErrorAccRU
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_error/_past

LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「=」は「%3d」に置き換えてください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
$ curl -i -X GET -H 'Authorization:Bearer ErrorAccRU' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_error/_past'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200 OK」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

```
[
  {
    "_data":{
      "collection_first_time":"<エラー収集期間の開始時刻>",
      "collection_last_time":"<エラー収集期間の終了時刻>",
      "errors": [
        {
          "error_factor_message":"<失敗要因メッセージ>",
          "number_of_times":<エラー収集期間内で失敗要因と失敗要因メッセージが同一であったものの回数>,
          "resource_path":"<エラーが発生したリソースパス>",
          "trigger":"<失敗要因>"
        }
      ]
    },
    "_date":"20180901T235959.000Z",
    "_resource_path":"_error"
  }
(以下略)
]
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第9章　リソースデータの制御（MQTT）

本サービスの MQTT を用いた API の使用例をご紹介します。リソースの登録、参照をご体験頂けます。ご契約後の通知内容をご用意ください。本章では以下で説明しますが、必要に応じて読み替えて実行して下さい[2]。MQTT のユーザ名はテナント毎に固定となっています。また MQTT のパスワードはサービスポータルにて設定する必要があります。

	項目	情報
1	テナントID	<テナントID>
2	MQTTブローカーアドレス	<zone>.fujitsu.com
3	IoTのベースURI	http://<zone>.fujitsu.com/
4	MQTTのユーザ名	<テナントID>

9.1. リソースデータの登録例

ここでは 8.1 章と同様に「出勤管理（富士通花子）」のリソースに対して出勤情報のデータを登録する API を呼び出し、そのデータが実際に登録されていることを参照するケースを紹介します。

今回のケースにおいて、出退勤はリソースデータの変数「diligence」で管理され、「1」が出勤、「0」が休みを意味するように管理するものとします。ここでは、現在時刻に富士通花子さんが出勤したことを登録するものとします。事前に 8.1 章に従い、アクセスコードの取得を行って下さい。

API を呼び出すのに必要なパラメータは以下の通りです。

MQTT ブローカーアドレス	<zone>.fujitsu.com
ポート番号	1883/TCP　(SSL/TLS の場合、8883/TCP)
メッセージタイプ	PUBLISH
MQTT ユーザ名	<テナント ID>
MQTT パスワード	<MQTT パスワード>
トピック	DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako
ヘッダ：DUP フラグ	0
ヘッダ：QoS フラグ	0
ヘッダ：RETAIN フラグ	True

ペイロード	{
	　"diligence":1
	}

Eclipse Paho のライブラリが導入された Python 上で動作するソースコードを以下に示します。ファイル名を pub.py とします。

```
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    if str(rc)=='0':
        print('[CONNACK]:client can connect :' + str(rc));
    else:
        print('[CONNACK]:client can¥'t connect :' + str(rc));
    client.publish(topic,payload,qos=0,retain=False)
```

[2] MQTT では、登録・参照ができますが、更新・検索・削除ができません。更新・検索・削除を行いたい場合は REST を使用してください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
def on_publish(client,userdata,mid):
    print("client is published ")
    client.disconnect()

if __name__ == '__main__':
    is_connect = True

    username = ' <テナント ID>'
    password = '<MQTT パスワード>'
    host = '<zone>.fujitsu.com'
    topic = 'DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako'
    port = 1883
    payload = '{"diligence":1}'
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_publish = on_publish
    client.username_pw_set(username,password)
    client.connect(host,port,keepalive=1800)
    client.loop_forever()
```

Eclipse Paho のライブラリが導入された Python 上で動作するソースコードを以下に示します。以下を実行します。

```
$python pub.py
```

以下が出力されます。

```
[CONNACK]:client can connect :0
client is published
```

MQTT では JSON のデータフォーマットが誤った状態で登録しても正常応答してしまうため、本サービスへのデータ格納が成功したかどうか判別できません。正しく登録されたか確認するためには REST で参照し、確認してください。

9.2. リソースデータの参照例

ここでは、MQTT を使って登録されたリソースデータを参照します。事前に 8.1 章に従い、アクセスコードの取得を行って下さい。API を呼び出すのに必要なパラメータは以下の通りです。

MQTT ブローカーアドレス	<zone>.fujitsu.com
ポート番号	1883/TCP　(SSL/TLS の場合、8883/TCP)
メッセージタイプ	SUBSCRIBE
Topic	DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako
Keep-alive	1800 秒
ヘッダ：DUP フラグ	0
ヘッダ：QoS フラグ	0
ヘッダ：RETAIN フラグ	0
ペイロード	DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako

ファイル名を sub.py とします。

```
import paho.mqtt.client as mqtt
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
def on_connect(client, userdata, flags, rc):
    if str(rc)=='0':
        print('[CONNACK]:client can connect :' + str(rc));
    else:
        print('[CONNACK]:client can¥'t connect :' + str(rc));
    client.subscribe(topic,qos=0)


def on_message(client, userdata, msg):
    print('Message is recieved.¥r¥nTOPIC:' + msg.topic + '¥r¥npayload:¥r¥n' + str(msg.payloa
d))
    client.disconnect();

if __name__ == '__main__':
    username = '<テナント ID>'
    password = '< MQTT パスワード>'
    host = '<zone>.fujitsu.com'
    topic='DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako'
    port = 1883

    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    client.username_pw_set(username, password)
    client.connect(host, port=port, keepalive=1800)
    client.loop_forever()
```

以下を実行します。

```
$python sub.py
```

別のターミナルなどから 4.1 章で作成した以下を実行します。

```
$python pub.py
```

実行後データ登録時に以下が返ってきます。

```
topic: DiligenceManagement22222/v1/<テナント ID>/diligence/status/hanako

payload:
{ "diligence":1}
```

Memo

CONNECT、SUBSCRIBE 状態の維持には制限があります。
CONNECT 時に指定できる KeepAlive 値は、1～1800 秒です。
このため、データ通信状態によってはセッション維持ができない場合があります。PINGREQ を利用してセッション維持するか、または切断後に再接続してください。
(PUBLISH 時の retain を True にすることにより、切断中の情報を後から受け取ることができます。)
※但し、CONNECT のみとなる接続が繰り返された場合は、接続を拒否する場合があります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第10章 リソース_Binary のデータ制御

本サービスのリソース_Binary におけるリソースデータを API で制御する例をご紹介します。実際のコマンド入力やプログラムの実行を通して、リソースデータの操作を行います。ここでは、「東京・蒲田工場」においてライン A、ライン B の状況をカメラで定期的に撮影し、本サービスに登録、検索、参照、削除できることを確認するケースを紹介します。

ご契約後の通知内容をご用意ください。本章では以下で説明しますが、必要に応じて読み替えて実行して下さい。

テナント ID	<テナント ID>
ベース URI	http://<zone>.fujitsu.com/

予めサービスポータルより以下の通り、リソース及びアクセスコードを作成して下さい。作成方法については「IoT Platform サービスポータル操作マニュアル」をご参照ください。

リソース種別	リソース
リソースパス	_bin/Tokyo/Kamata/LineA/normal
データ形式	バイナリ
保存期間	1 日

アクセスコード	AccTokyoLineAllRU
リソースパス	_bin/Tokyo/Kamata/LineA/normal
アクセス権限	「R」と「U」に権限を付与

アクセス権限の「R」はリソースデータに対する読み込み権限、「U」はリソースデータに対する書き込み権限を意味します。

10.1. リソース_Binary のデータ登録例

「リソース_Binary」に対して、Binary データを登録することができます。

ここでは東京・蒲田工場における製造ライン A において正常動作している際の画像を登録する例を使って紹介します。具体的にはリソース_Binary「_bin/Tokyo/Kamata/LineA/normal」に対してバイナリである画像を登録します。今回のケースにおいて、画像は JPG ファイル形式にて圧縮され、管理するものとします。今回の例では、2016 年 9 月 1 日日本時間午前 10 時 01 分に製造ライン A が正常動作している画像(lineA1000.jpg)を登録するものとします。画像に対するメタデータとして「productID12345」という文字列も登録することとします。

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	PUT
ヘッダフィールド名１	Authorization
ヘッダフィールド値１	Bearer AccTokyoLineAllRU
ヘッダフィールド名２	x-iotpf-meta-data1
ヘッダフィールド値２	productID12345
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal?$date=20160901T100100Z
BODY	登録する JPEG ファイル(lineA1000.jpg)

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」(半角スペース)は「%20」、「+」は「%2b」、「'」(シングルクォーテーション)は「%27」、「=」は「%3d」に置き換えてください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
$ curl -i -X PUT -H 'Authorization:Bearer AccTokyoLineAllRU' -H 'x-iotpf-meta-data1: product ID12345' --data-binary @./lineA1000.jpg 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal?$date=20160901T100100Z'
```

正常完了できた場合、レスポンスとして「200」が返ります。

10.2. リソース_Binary のデータ検索例

ここでは東京・蒲田工場における製造ライン A において正常動作している際の画像を検索する例を使って紹介します。具体的にはリソース_Binary「_bin/Tokyo/Kamata/LineA/normal」に対して Binary である画像を検索します。

事前に 10.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。

今回のケースにおいて、画像は JPG ファイル形式にて圧縮され、管理されているものとします。今回の例では、2016 年 9 月 1 日日本時間午前 10 時から午前 11 時の間に製造ライン A が正常動作していた画像を検索するものとします。データは予め午前 10 時 01 分、10 時 21 分、10 時 41 分に登録されているものとします。

API を呼び出すのに必要なパラメータは以下の通りです。filter 条件の演算子を用いることにより、リソースデータの範囲指定が可能です。ここでは、2016 年 9 月 1 日日本時間午前 10 時～午前 11 時までの出勤状況を検索しています。Filter 条件の演算子については「IoT Platform API リファレンス」をご参照ください。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoLineAllRU
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past?$filter=_date gt 20160901T100000Z and _date lt 20160901T110000Z

LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「=」は「%3d」に置き換えてください。

```
$ curl -i -X GET -H 'Authorization:Bearer AccTokyoLineAllRU' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past?$filter=_date%20gt%2020160901T100000Z%20and%20_date%20lt%2020160901T110000Z'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200」が返ります。以下の出力例ではわかりやすくするために適宜改行を挿入し整形していますが、実際の出力結果は整形されません。

```
[
 {
  "content_type":"application/x-www-form-urlencoded",
  "name":"_bin/Tokyo/Kamata/LineA/normal/_past(20160901T100100.000Z)"
 },
 {
  "content_type":"application/x-www-form-urlencoded",
  "name":"_bin/Tokyo/Kamata/LineA/normal/_past(20160901T102100.000Z)"
 },
 {
  "content_type":"application/x-www-form-urlencoded",
  "name":"_bin/Tokyo/Kamata/LineA/normal/_past(20160901T104100.000Z)"
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
  }
]
```

リソース_Binary において、リソースデータを検索した際、実際のバイナリデータはレスポンスに含まれません。実際のバイナリデータを取得したい場合、1 つずつ参照し取得する必要があります。
（10.3 章参照）

10.3. リソース_Binary のデータ参照例

ここでは、東京・蒲田工場における製造ライン A において正常動作している際の画像を参照する例を使って紹介します。具体的にはリソース_Binary「_bin/Tokyo/Kamata/LineA/normal」に対して 2016 年 9 月 1 日日本時間午前 10 時 01 分に登録された画像を参照します。

事前に 10.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。API を呼び出すのに必要なパラメータは以下のとおりです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoLineAllRU
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past(20160901T100100.000Z)

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「+」は「%2b」に置き換えてください。

```
$ curl -X GET -H 'Authorization:Bearer AccTokyoLineAllRU' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past(20160901T100100.000Z )' > outputLineA.jpg
```

リソースデータの参照が正常に行えた場合、レスポンスとして対象時刻のバイナリデータ（今回の場合 10.1 章で登録した JPG ファイル）が取得できます。

Memo

バイナリの参照において、バイナリのデータ量が大きい場合、途中で通信が切断される場合があります。この時 HTTP Range Request を使うことにより、切断されたあとも続きのデータをダウンロードできます。

206 が返却された場合、Header の Content-Range をご参照ください。その後、Header に Range を指定し API を呼ぶことで続きのデータを取得できます。

10.4. リソース_Binary のデータ削除例

ここでは、東京・蒲田工場における製造ライン A において正常動作している際の画像を削除する例を使って紹介します。具体的にはリソース_Binary「_bin/Tokyo/Kamata/LineA/normal」に対して 2016 年 9 月 1 日日本時間午前 10 時 1 分に登録された画像を削除します。

事前に 10.1 章に従い、アクセスコードの取得、及びデータの登録を行って下さい。API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	DELETE
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoLineAllRU

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past?$filter=_date eq 20160901T100100.000Z

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X DELETE -H 'Authorization:Bearer AccTokyoLineAllRU' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineA/normal/_past?$filter=_date%20eq%2020160901T100100.000Z'
```

リソースデータの削除が正常に行えた場合、レスポンスとしてヘッダに「200」が返ります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第11章 リソースの制御

本サービスのリソースを API で制御する例をご紹介します。実際のコマンド入力やプログラムの実行を通して、リソースの操作を行います。

ここでは、第 10 章で示した「東京・蒲田工場」においてライン A、ライン B の状況をカメラで定期的に撮影するために事前に必要なリソース、アクセスコード、イベントを登録、参照、削除するケースを紹介します。

ご契約後の通知内容をご用意ください。本章では以下で説明しますが、必要に応じて読み替えて実行して下さい。

テナント ID	<テナント ID>
ベース URI	http://<zone>.fujitsu.com/

事前に以下のリソースをサービスポータル画面から作成してください。作成方法については「IoT Platform サービスポータル操作マニュアル」をご参照ください。

リソース種別	リソース
リソースパス	_bin/Tokyo
データ形式	バイナリ
保存期間	1 日

次に以下のリソースパスに対して、以下のアクセスコードとアクセス権限を付与してください。
また、転送先 URI についてはサービスポータルにて登録してください。

リソースパス	_bin/Tokyo
アクセスコード	AccTokyoCDL
アクセス権限	「CDL」に権限を付与

アクセス権限の「CDL」はリソースパス配下に対して、API よりリソースの作成 (C)、削除 (D)、参照 (L) ができることを意味します。上記のようにリソース及びアクセスコードを事前に作成することにより、「_bin/Tokyo」配下のリソースパスに対して API を用いてリソースを作成したり削除したりすることが可能になります。

これにより、利用するデバイスが増えた際、その都度サービスポータルからの操作を行う必要がなくなります。

11.1. リソースの登録例

ここでは東京・蒲田工場における製造ライン B において画像を登録するリソースを作成します。具体的にはリソース_Binary「_bin/Tokyo/Kamata/LineB/normal」を API 経由にて登録します。登録するリソースは以下の通りです。

リソース種別	リソース
リソースパス	_bin/Tokyo/Kamata/LineB/normal
データ形式	バイナリ
保存期間	3 日

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	POST
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineB/

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

	normal

BODY	{
	"resource":{
	"retention_period":3
	}
	}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X POST -H 'Authorization:Bearer AccTokyoCDL' -d '{"resource":{"retention_period":3}}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineB/normal'
```

正常完了できた場合、レスポンスとして「201」が返ります。またヘッダには Location として登録したリソースパス（http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineB/normal）が返却されます。

11.2. リソースのメタデータ参照例

ここでは、「_bin/Tokyo」配下において登録されている全てのリソースを参照します。11.1 章を参照し、予めリソースを登録してください。API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/$all/_resources

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X GET -H 'Authorization:Bearer AccTokyoCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/$all/_resources'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200」が返ります。

```
{
 "resources":[{
  "resource_path":"_bin/Tokyo/Kamata/LineA/normal",
  "retention_period":1,
  "last_modified":20160901T100100Z
  }
 },
  {
  "resource_path":"_bin/Tokyo/Kamata/LineA/fault",
  "retention_period":1,
  "last_modified":20160901T100000Z
  }
 },
 {
  "resource_path":"_bin/Tokyo/Kamata/LineB/normal",
  "retention_period":3,
  "last_modified":20160901T120100Z
  }
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
  },{
   "resource_path":"_bin/Tokyo/Kamata/LineB/fault",
   "retention_period":1,
   "last_modified":20160901T120000Z
   }
  }
]
```

今回の場合、「_bin/Tokyo」配下に「_bin/Tokyo/Kamata/LineA/normal」、「_bin/Tokyo/Kamata/LineA/fault」、「_bin/Tokyo/Kamata/LineB/normal」、「_bin/Tokyo/Kamata/LineB/fault」の 4 つのリソースがあることが確認できます。

11.3. リソースの削除例

ここでは東京・蒲田工場における製造ライン C において画像を登録するリソース（_bin/Tokyo/Kamata/LineC/normal）を削除したいとします。11.1 章を参照し、予めリソースを登録してください。

リソース種別	リソース_Binary
リソースパス	_bin/Tokyo/Kamata/LineC/normal
保存期間（日）	3

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	DELETE
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineC/normal

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X DELETE -H 'Authorization:Bearer AccTokyoCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/Tokyo/Kamata/LineC/normal'
```

リソースの削除が正常に行えた場合、レスポンスとしてヘッダに「204」が返ります。リソースに紐づくアクセスコードが存在する場合、「423」を返します。12.3 章を参照の上、アクセスコードによる紐付けを解除してから、リソースを削除してください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第12章 アクセスコードの制御

12.1. アクセスコードの登録例

11.1 章において登録したリソースに対して、アクセスコードを登録します。これにより、登録したアクセスコードを使ってリソースデータの登録を行うことができます。登録するアクセスコードは以下の通りです。

アクセスコード	AccTokyoLineAllRU
リソースパス	_bin/Tokyo/Kamata/LineB/normal
アクセス権限	「R」「U」に権限を付与

アクセス権限の「R」(read)はリソースデータに対する読み込み権限、「U」(update)はリソースデータに対する書き込み権限を意味します。

HTTP メソッド	POST
ヘッダフィールド名 1	Authorization
ヘッダフィールド値 1	Bearer AccTokyoCDL
ヘッダフィールド名 2	Content-Type
ヘッダフィールド値 2	application/json; charset=UTF-8
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU
BODY	{ "access_code":{ "permissions":{ "resource_operations":[{ "operations":["update", "read"], "resource_path":"_bin/Tokyo/Kamata/LineB/normal" }] } } }

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X POST -H 'Authorization:Bearer AccTokyoCDL' -H 'Content-Type: application/json; charset=UTF-8' -d '＜上記 BODY 部＞' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU'
```

正常完了できた場合、レスポンスとして「201」が返ります。またヘッダには Location として登録したアクセスコード（http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU）が返却されます。

12.2. アクセスコードの参照例

ここでは、12.1 章において登録したアクセスコードの詳細情報（リソースパスとそのリソースパスへの権限）を参照します。予め第 11 章、12.1 章を参照しアクセスコードを登録してください。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X GET -H 'Authorization:Bearer AccTokyoCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200」が返ります。

```
{
  "access_codes":[{
    "permissions":{
      "ip_filter":[ ],
      "resource_operations":[{
        "operations":[ "read", "update" ],
        "resource_path":"_bin/Tokyo/Kamata/LineA/normal"
      } ]
    },
    "access_code":"AccTokyoLineAllRU"
  } ]
}
```

12.3. アクセスコードの削除例

ここでは 12.1 章で登録したアクセスコードを削除します。12.1 章を参照し、予めリソースを登録してください。

アクセスコード	AccTokyoLineAllRU
リソースパス	_bin/Tokyo/Kamata/LineB/normal
アクセス権限	「R」「U」に権限を付与

API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	DELETE
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer AccTokyoCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X DELETE -H 'Authorization:Bearer AccTokyoCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_access_codes/AccTokyoLineAllRU'
```

リソースの削除が正常に行えた場合、レスポンスとしてヘッダに「204」が返ります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

第13章 イベントの制御

13.1. イベントの登録例

ここでは、A 地点における温度センサーの値が 30 度を超えた場合、Email を送信するイベントを作成します。予め以下のリソースを作成してください。

リソース種別	リソース
リソースパス 1	sensors
データ形式	JSON
保存期間（日）	1 日

リソース種別	リソース
リソースパス 2	sensors/temperature
データ形式	JSON
保存期間（日）	1 日

また、以下のアクセスコードを作成してください。

アクセスコード	TemperatureCDL
リソースパス 1	sensors
リソースパス 1 への アクセス権限	「CDL」に権限を付与

アクセスコード	TemperatureRU
リソースパス 2	sensors/temperature
リソースパス 2 への アクセス権限	「R」「U」に権限を付与

アクセス権限の「CDL」はリソースパス配下に対して、API よりリソースの作成 (C)、削除 (D)、参照 (L) ができることを意味します。アクセス権限の「R」(read)はリソースデータに対する読み込み権限、「U」(update)はリソースデータに対する書き込み権限を意味します。

HTTP メソッド	POST
ヘッダフィールド名 1	Authorization
ヘッダフィールド値 1	Bearer TemperatureCDL
ヘッダフィールド名 2	Content-Type
ヘッダフィールド値 2	application/json; charset=UTF-8
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_events

BODY	{
	"event":{
	"notification":{
	"smtp":{
	"subject":"温度が 30 度超えました",
	"body":"温度が 30 度を超えています。確認してください。",
	"send_to":"hoge@piyo.fujitsu.com"
	}
	},
	"conditions":{
	"targets":{
	"read_access_code":"TemperatureRU",
	"operations":["create","update"],
	"resource_path":"sensors/temperature"

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

	}, "notification_condition":{ "body_conditions":{ "path":"sensors/temperature", "path_type":"JSONPath", "comparing_operator":"ge", "value":30.00000 } } } }}

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X POST -H 'Authorization:Bearer TemperatureCDL' -H 'Content-Type:application/json; charset=UTF-8' -d '＜上記 BODY 部＞' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_events'
```

正常完了できた場合、レスポンスとして「201」が返ります。またヘッダには Location として登録したイベントのイベント番号（http://<zone>.fujitsu.com/v1/<テナント ID>/_events/1234567890）が返却されます。この時、1234567890 がイベント ID となるので控えておいてください。

13.2. イベントの参照例

ここでは、13.1 章で登録したイベントを参照します。予め 13.1 章を参照しイベントを登録してください。ここでは例としてイベント ID は「1234567890」とします。

HTTP メソッド	GET
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer TemperatureCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_events/1234567890

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X GET -H 'Authorization:Bearer TemparatureCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_events/1234567890'
```

リソースデータの参照が正常に行えた場合、レスポンスとして以下の JSON データとヘッダに「200」が返ります。

```
{
  "events":[{
    "conditions":{
      "targets":{
        "operations":["create","update"],
        "resource_path":"sensors/temperature",
        "read_access_code":"TemperatureRU"
      },
      "notification_condition":{
        "body_conditions":{
          "path":"sensors/temperature",
          "path_type":"JSONPath",
```

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

```
          "comparing_operator":"ge",
          "value":30.00000
        }
      }
    },
    "notification":{
      "smtp":{
        "subject":"温度が 30 度超えました",
        "body":"温度が 30 度を超えています。確認してください。",
        "send_to":" hoge@piyo.fujitsu.com"
      }
    },
    "event_id":"1234567890"
  } ]
}
```

13.3. イベントの削除例

ここでは、13.1 章で登録したイベントを削除します。13.1 章を参照し、予めイベントを削除してください。API を呼び出すのに必要なパラメータは以下の通りです。

HTTP メソッド	DELETE
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer TemperatureCDL
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_events/1234567890

例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X DELETE -H 'Authorization:Bearer TemperatureCDL' 'http://<zone>.fujitsu.com/v1/<テナント ID>/_events/1234567890'
```

リソースの削除が正常に行えた場合、レスポンスとしてヘッダに「204」が返ります。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

付録 1 REST アプリ(Android)

リソースデータへの操作とログ取得が行える Android アプリのプロジェクト一式をご用意しています。
ファイルの取得方法については、担当の営業にお問い合わせ下さい。

ライセンス

本アプリには、以下に記載されたライセンスが適用されます。
本アプリをご使用になる前に必ずこれらのライセンス条件をお読み下さい。

Copyright (c) 2018 Fujitsu Limited

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software only for the activities in connection with "FUJITSU Cloud Service IoT Platform" service of Fujitsu Limited, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

動作環境

「SampleAppREST.zip」を展開し、Android Studio へインポートしてお使いください。インポート及びビルド手順につきましては、Android Developer の公式サイトをご参照ください。

本アプリは以下の環境で動作確認を行っております。

・ARROWS NX F-05F(Android 4.4)

また、アプリケーションが動作する API レベルは以下の通りです。アプリケーションをビルドするにあたり、ビルド対象 SDK バージョンと同じ SDK が必要になります。

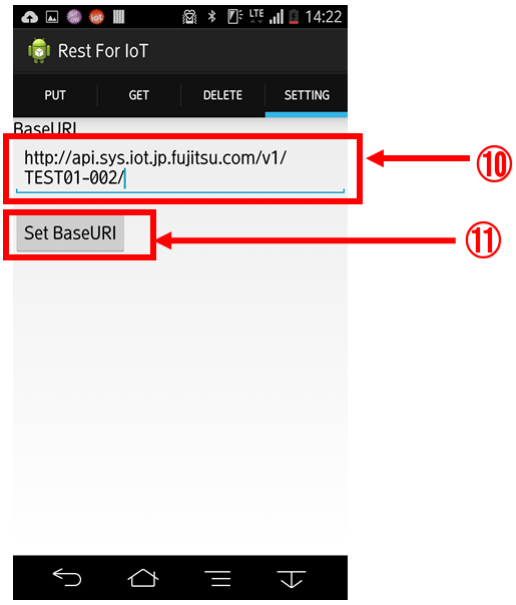
	API レベル	Android バージョン
最小 SDK バージョン	19	Android 4.4 KitKat
ビルド対象 SDK バージョン	21	Android 5.0 Lollipop

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022

概要説明

	項目	概要
①	メソッドタブ	実行したいメソッドを PUT/GET/DELETE からタップ操作によって選びます。 またベース URI を設定したい場合、SETTING タブから設定します。
②	ベース URI	SETTING タブにおいて設定したベース URI が設定されています。
③	リソースパス	リソースパス及びクエリを文字列にて設定します。 クエリに対する文字列はパーセントエンコードする必要があります REST 実行ボタンを押す時に入力したリソースパスが読み込まれます。
④	アクセスコード	アクセスコードを文字列にて設定します。 REST 実行ボタンを押す時に入力したアクセスコードが読み込まれます。
⑤	BODY 文	BODY 文を文字列にて設定します。 REST 実行ボタンを押す時に入力した BODY 文が読み込まれます。
⑥	REST 発行ボタン	押すと、[1]～[5]に設定された内容の REST を実行します。実行した結果はレスポンス表示部に表示されます。
⑦	クリアボタン	レスポンス表示部の内容をクリアします。
⑧	トースト	実行ログの出力先を表示します。 本トーストは REST 実行ボタンを押した後のみ一定時間表示されます。 実行ログは、Android 端末内のフォルダ配下に実行した REST の内容とレスポンスが表示されます。1 回の API 実行につき、1 ファイル出力されます。 ファイルの格納先は以下の通りです。 /storage/emulated/0/Download
⑨	レスポンス表示部	REST 発行ボタンにて実行された結果を表示します。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022



	項目	概要
⑩	ベース URI 設定	ベース URI を文字列にて設定します。
⑪	ベース URI 設定ボタン	押すと、ベース URI 設定に入力されている文字列が設定されます。 PUT/GET/DELETE すべてのタブに反映されます。

All Rights Reserved, Copyright©FUJITSU LIMITED 2016-2022