IoT Platform DRC(オンデマンドデータ収集)
ユーザーガイド
(11.0_0 版)

＜改版履歴＞

版数	説明	日付
9.0_0	初版登録	2019/08/22
9.0_1	誤記訂正	2019/12/11
11.0_0	7.2　使用リソース一覧に関する誤記訂正 7.5　API 汎用例に関する記載の注意書きを追加 7.5　登録データの汎用例に関する誤記訂正 7.5-7.9 API 実行時のレスポンスを 200 OK→200 に修正	2021/1/16

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 DRC(オンデマンドデータ収集) 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。

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。

＜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。	
エッジ	現場のデータを蓄積している様々な機器	
DRC(オンデマンドデータ収集)	エッジに蓄積された実データを効率よく利活用できる機能	
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	

- 目次 -

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）に含まれるDRC(オンデマンドデータ収集)（以下：本機能）のご利用に際し、システムを設計する方向けの「DRC(オンデマンドデータ収集)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。
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。
IoT Platform DRC(オンデマンドデータ収集)ユーザーガイド	本機能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 DRC(オンデマンドデータ収集)インターフェースリファレンス	本機能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インターフェースのリファレンスマニュアルです。

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

本サービスは以下の特長を持っています。これらの特長により、過去に遡って必要なデータを効率よく取り出すことができるため、分散された大量のエッジに蓄積された大容量のデータ(実データ)から必要な情報を必要なときに素早く取り出すことができます。

・メタデータ登録機能

エッジにおいて発生したデータの特徴（メタデータ）を収集・蓄積し、あとから検索しやすいように分解・整理する機能です。これにより、実データをすべて上げるときと比較して、伝送量を低減できます。

・データ検索機能

エッジに分散されて蓄積された実データを検索し収集します。検索要求に基づき、事前に収集しておいたメタデータの中から実データを保有しているエッジを選別し指示することができます。
これにより、大容量のデータから必要な情報を必要なときに素早く取り出すことができます。

第2章　DRC(オンデマンドデータ収集)とは

2.1. DRC(オンデマンドデータ収集)

DRC(オンデマンドデータ収集)（以下、本機能）を利用することにより、エッジに蓄積された実データを効率よく利活用できます。
センサや動画像を多数のエッジより収集する IoT ビジネスにおいて、予めエッジからデータの特徴のみを抽出したメタデータを登録しておくことにより、メタデータを元にエッジに対して実データを抽出することができます。これにより、エッジとクラウドの通信量を必要最小限にしつつ、エッジで生成された実データを効率よく収集することができます。

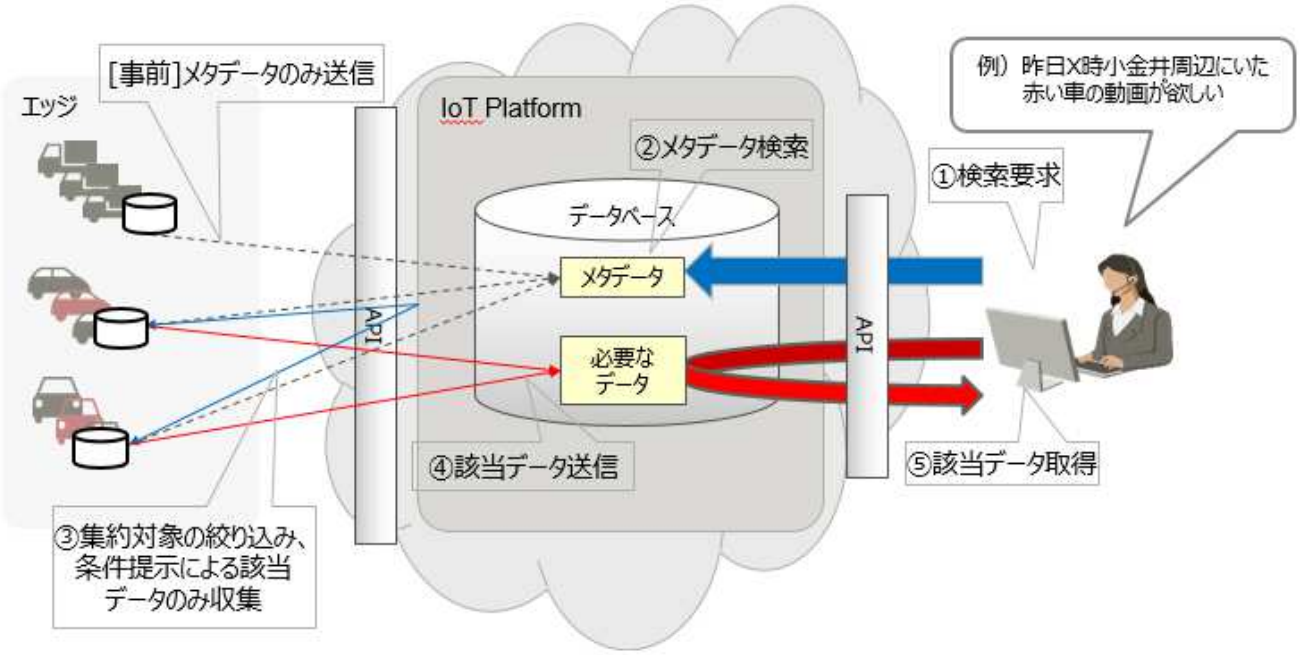


図　DRC(オンデマンドデータ収集)の利用イメージ

第3章　機能概要

3.1. 機能概要

本サービスは以下の機能を有しています。

3.1.1. メタデータ収集機能

本機能はメタデータを収集し蓄積します。メタデータとは実データ（動画像、センサ、機器稼動ログなど）の特徴のみを抽出したデータ（時刻、位置情報、タグなど）を指します。収集されたメタデータは後から検索しやすいように再整理されます。また、整理内容について結果を確認することができます。

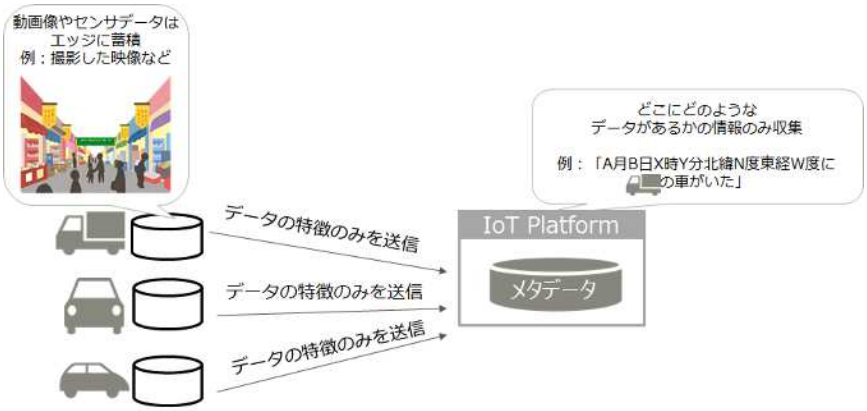


図　　メタデータ収集機能のイメージ図

3.1.2. 検索要求機能

本機能はお客様が設計した検索クエリを元にエッジに蓄積されたデータを検索することができます。
以下に、データを検索し取得するまでの流れの例を示します

① データ検索クエリの登録
　お客様がいつどこにあったどのようなデータが欲しいかを記載したメタデータのクエリを登録します。本サービスはこのクエリを元に検索を行います。

② メタデータの検索
　本サービスが事前に集めたメタデータを元にその実データがどのエッジにあるか判断します。

③ 前処理の指示
　本サービスが検索条件に該当するエッジに対して指示を行います。

④ 該当する実データのみ収集
　エッジが本サービスに対して検索条件に該当する実データを送信します。

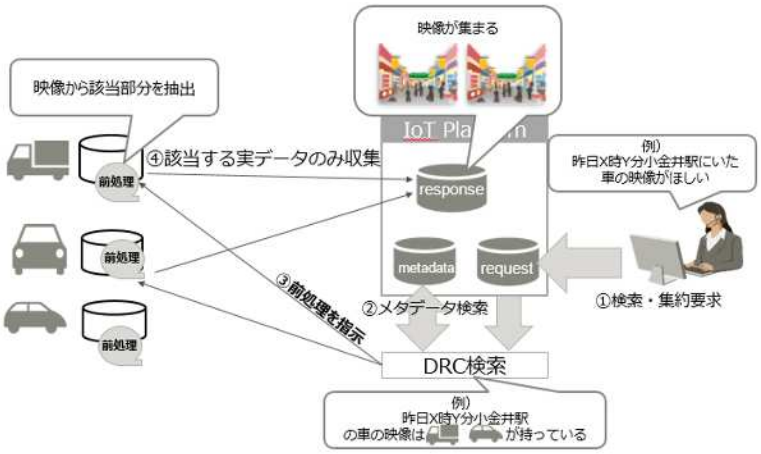


図　　検索要求機能のイメージ図

3.2. 特長

本サービスは以下２つの特長を持っています。

3.2.1. エッジへの検索指示を絞り込める Graceful 機能

本サービスは、エッジに蓄積されたデータを収集する際、収集されるエッジの数や量を調整することができます。
予め収集したメタデータによっては、検索指示を行った結果、対象となるエッジの数や量が膨大になってしまい、データを収集しきれない問題が生じます。これを解決するために、対象となるエッジの数や量が膨大になった場合、指定したエッジの数や量だけ収集するようエッジへの指示を絞りこむことができます。これにより、意図せず膨大な収集を行ってしまうことがなくなるため、短期間で効率よくデータを収集することができます。

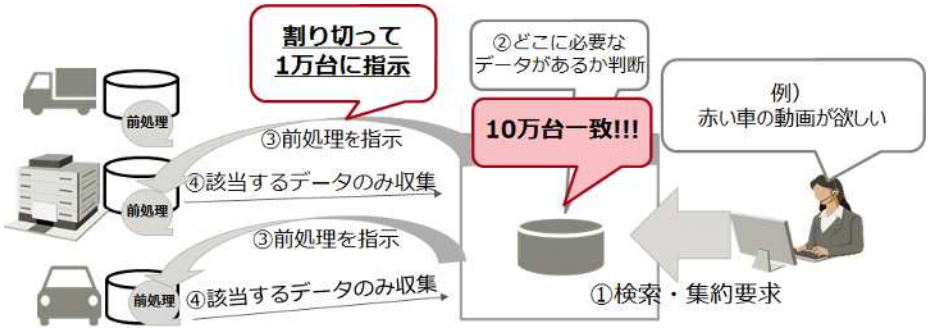


図　Gracful の利用イメージ

3.2.2. 詳細なログを表示する Verbose 機能

エッジに検索指示を出した場合、エッジに電源が入っていなかった場合などにおいて、データを収集することができなくなる場合があります。Verbose 機能を有効にすると、詳細な経過ログを表示できるため、収集できなかった場合の原因を特定することができます。

第4章　ユースケース

4.1. インシデント深掘

インシデントが発生した際に原因分析／影響見極めを行うにあたって周辺に存在したセンサ群から効率よくデータを収集するケースです。例えば、点在する車載器におけるドライブレコーダーから過去に発生した事故の動画を収集することができます。

4.2. 予測用データ探索型

特定事象や特定行動に対する予測精度向上のために類似する関連データを収集することができます。これを使って、エッジ AI などエッジにある機械学習モデルの最適化を行ったり、渋滞状況や気象状況を判別したりすることができます。

第5章　準備するもの

本章では、本機能を使用するために必要なものについて説明します。

5.1. ご契約後の通知内容

ご契約後弊社から通知する以下の情報が必要になります。

表 1：ご契約後の通知内容情報一覧

	項目
1	テナント ID
2	ベース URI

5.2. アプリ開発環境

各種アプリを開発するにあたって、REST（HTTP）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。（root 権限にて実行）

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

5.3. ルート証明書（SSL/TLS 利用時のみ）

本サービスでは、SSL/TLS サーバ証明書として SSL 通信のためのサーバ証明書として DigiCert 社の「DigiCert SHA2 High Assurance Server CA」を使用しています。
SSL/TLS 通信を行う場合は同社の「DigiCert High Assurance EV 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 2019 年 7 月現在のものです。

5.4. お客様が自ら用意するもの

以下のすべてをお客様にて自ら用意する必要があります。

① サーバ（活用システム）
② サーバ（活用システム）上でデータ活用するアプリ
③ ネットワーク回線
④ エッジ上で動作するアプリ
⑤ エッジ（エッジコンピュータ、ゲートウェイ装置、デバイス、センサ、他）
⑥ 本サービスのご契約

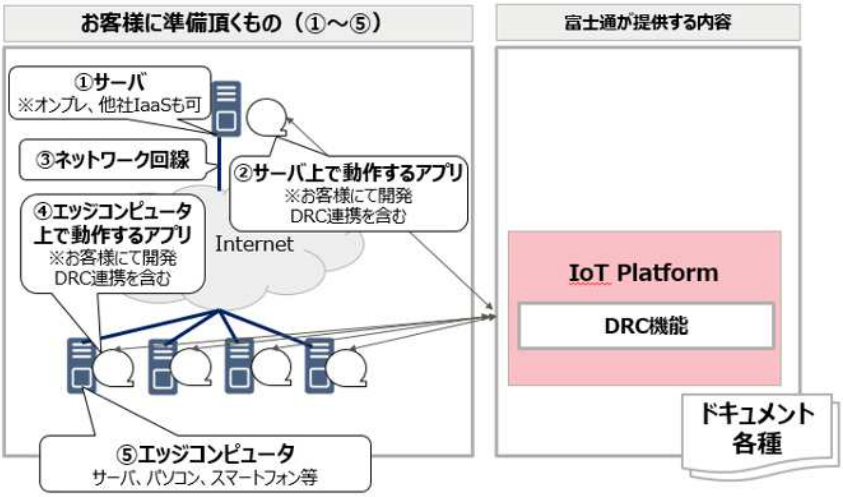


図　お客様に準備頂くもの

尚、本機能は本サービスのリソース_JSON を介して実現します。そのため、エッジ上で動作するアプリ、データを活用するアプリはネットワーク回線を介して本サービスに接続できる必要があります。

第6章　開発の流れ

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。

システム全体の設計を行う
↓
設計に基づきポータル画面より設定を行う
↓
リソース、アクセスコードの作成を行う
↓　・本機能の有効化、エッジ情報の登録、メタデータを納するリソースの作成
アプリ（エッジ／活用システム）の開発を行う
↓　・メタデータ収集アプリの開発（エッジ側）
↓　・検索指示読取・実データ送信を行うする前処理アプリの開発（エッジ側）
↓　・活用システムアプリの開発（活用システム側）
アプリの配備、及び機器の接続を行う
↓
利用を開始する

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「IoT Platform ユーザーガイド」第 5 章をご参照ください。

第7章　開発するアプリ・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。

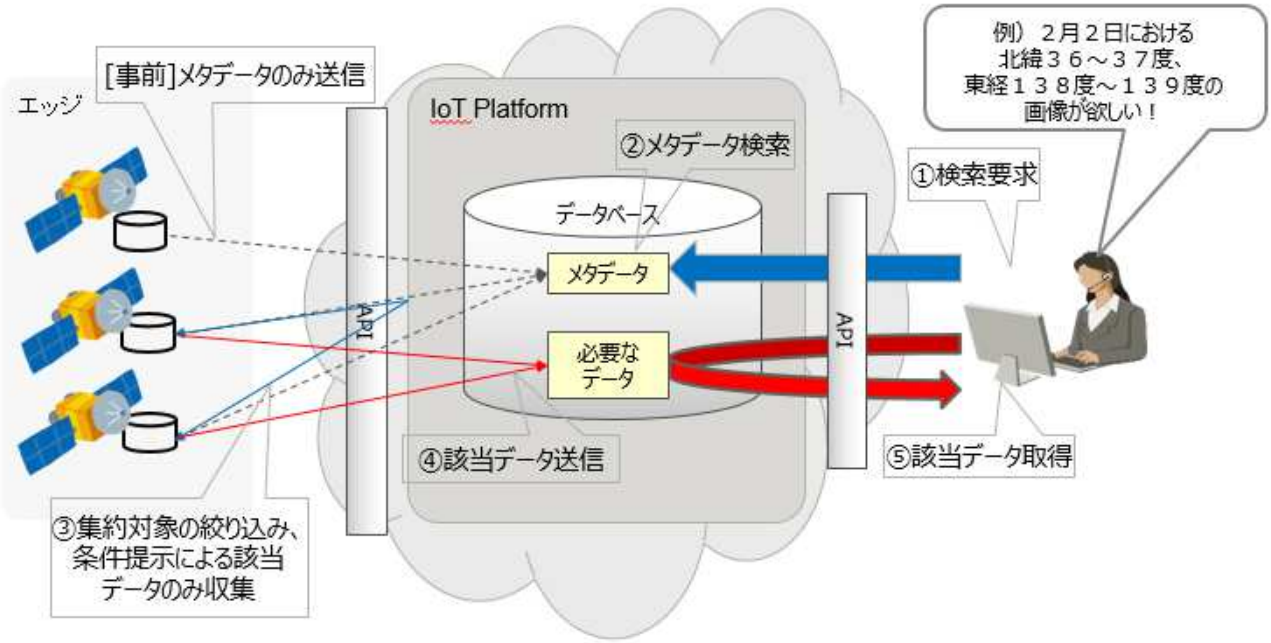
7.1. ユースケースのシナリオについて

アプリ開発の流れを紹介するにあたり、今回活用するユースケースについて紹介します。
今回は宇宙に点在する人工衛星をエッジとして、人工衛星に蓄積された画像を効率よく利活用する例について紹介します。
（尚、本シナリオは仮想のユースケースであり、実際の人工衛星に適用した事例ではありません。ご了承願います。）

本シナリオにおいて、人工衛星が複数打ち上げられています。打ち上げられた人工衛星は地上の画像を撮影し人工衛星内に蓄積しています。しかしながら、地上と人工衛星の間の帯域が限られており、常に最新の画像をすべて取得することができません。そこで、本機能を活用し、撮影したメタデータのみを地上に送信し、人工衛星に蓄積された画像を検索できるようにしました。これにより、過去に遡って必要な画像データを効率よく収集することができるようになります。

今回、人工衛星は検索時のキーとなるメタデータとして撮影した日時情報と緯度経度情報(geo_lat,geo_lng)を送信するものとします。また利用者は今回蓄積されたメタデータを基に、2019 年 2

月 2 日における北緯(geo_lat)が 36～37 度、東経(geo_lng)が 138 度～139 度の画像を検索し取得するものとします。取得指示を出すエッジとデータ数は 10 台までで 1 時間経過しても収集できなかった場合は指示を打ち切るものとします。尚、エッジである人工衛星に蓄積できるデータは 7 日分とします。



7.2. 本機能の内部構成

本機能の構成は以下の図のようになっています。図中のアルファベットは本機能のリソース_JSON を表しています。

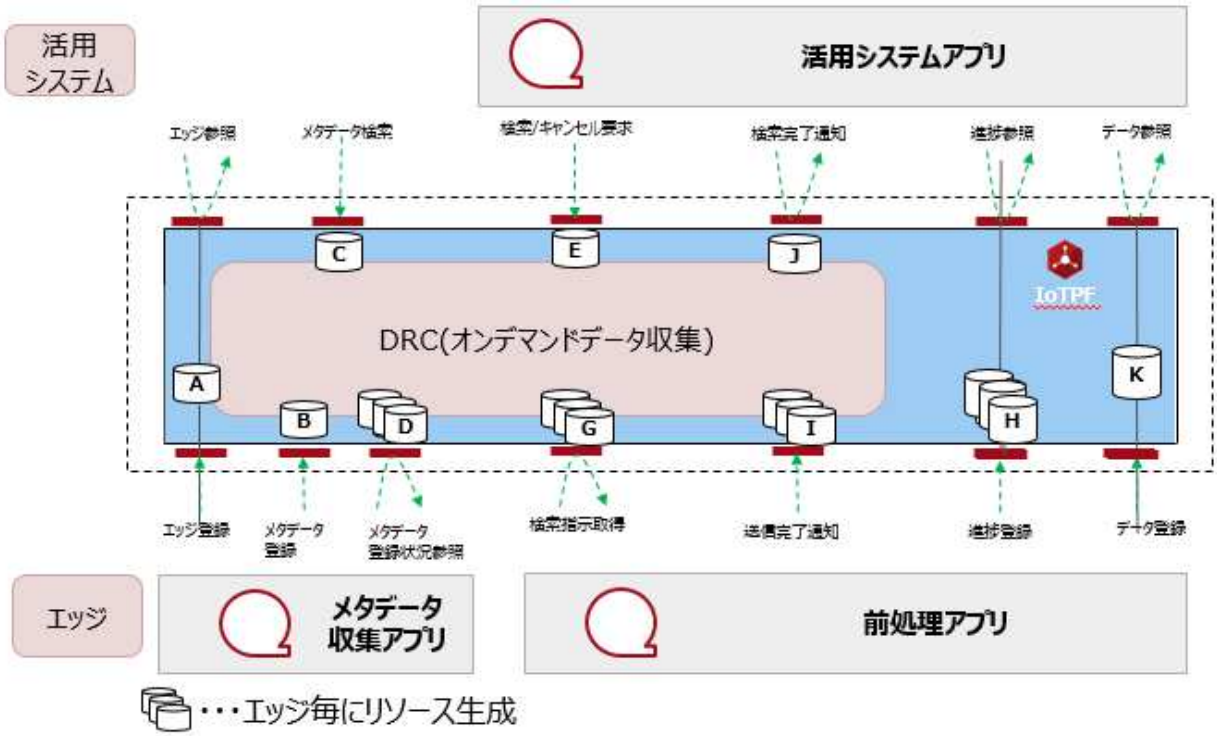


図　内部構成と開発するアプリの一覧

本機能を実現するためには以下 3 つのアプリ開発する必要があります。

１．エッジにおけるメタデータ収集アプリ

センサや動画像からメタデータを生成し本機能に登録する機能が必要です。あわせて、センサや動画像をエッジ側に蓄積する機能が必要です。開発に使用するリソース_JSON は以下のとおりです。

No	リソースパス	用途	詳細参照先
A	<prefix>/meta/gws	エッジ情報の登録、参照	エッジ情報取得 エッジ情報登録
B	<prefix>/meta/publish	メタデータの登録	メタデータ登録
C	<prefix>/meta/keys/<key>/<partition_id>	分解されたメタデータの検索・取得	分解されたメタデータ取得
D	<prefix>/meta/confirm/<gw_id>	メタデータ登録結果の参照	メタデータ登録確認

２．活用システムにおける活用システムアプリ

どのようなデータを検索・抽出したいか指示する機能が必要です。検索指示後は本機能が収集・蓄積したデータを収集することによって、可視化や機械学習などを行う機能が必要です。開発に使用するリソース_JSON は以下のとおりです。

No	リソースパス	用途	詳細参照先
E	<prefix>/request/search	検索要求/キャンセル要求の登録	検索・キャンセル要求
J	<prefix>/response/search	指示完了通知の登録	指示完了通知
H	<prefix>/response/progress/<gw_id>	指示進捗状況の登録、参照	進捗確認 進捗通知
K	<prefix>/response/preprocessed	検索結果(実データ)の登録、参照	データ取得 データ送信

３．エッジにおける前処理アプリ

エッジより、本機能が指示した要求を読み取る機能、および読み取り内容に従ってデータを抽出・加工し、本機能へデータを送信する機能が必要です。開発に使用するリソース_JSON は以下のとおりです。

No	リソースパス	用途	詳細参照先
G	<prefix>/request/clients/<gw_id>	検索指示/キャンセル指示の参照	検索指示・キャンセル指示取得
I	<prefix>/response/sent_flags/<gw_id>	検索結果の登録	送信完了通知
H	<prefix>/response/progress/<gw_id>	指示進捗状況の登録、参照	進捗確認 進捗通知
K	<prefix>/response/preprocessed	検索結果(実データ)の登録、参照	データ取得 データ送信

本章では具体的な API 発行例を紹介しながら各アプリの役割について説明します。例を実行するにあたって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/

7.3. 費用見積もりについて

本機能では、本サービスの内部処理にて従量サービスで提供している API(データアクセス)を利用します（以降：内部処理）。また、内部処理についてはお客様利用分として計上します。想定以上の利用料金となることを防ぐために、本機能利用時は内部処理を考慮されたうえでアプリケーションを設計してください。なお、内部処理で用いる API 回数は以下のとおりです。

１．メタデータ登録時
本機能 API 回数＝200×メタ数×（メタデータ登録回数＋１分あたりにメタデータ登録するエッジ数）
２．検索要求登録時
本機能 API 回数＝（検索要求時指定の検索対象メタ数×検索条件にヒットする GW 台数）×２
３．各リソースのポーリング
エッジ情報リソース：5 分に 1 回
メタデータリソース：1 分に 1 回
検索要求リソース：1 分に 1 回
送信完了リソース：1 分に 1 回（エッジ数には依存しません）

具体例：
１．メタデータ登録時
メタ数：３
メタデータ登録回数：１回／秒
１分あたりにメタデータ登録するエッジ数：５台
の場合

秒間あたりの本機能 API 回数＝200×1×（3＋5÷60）＝616(API)
※Bulk Insert 機能を用いてデータ登録するため、費用計算時は 200 を乗算します。

２．メタデータ登録時
検索要求時指定の検索対象メタ数：２
検索条件にヒットするエッジ台数：２
の場合

１検索要求あたりの本機能 API 回数＝（2×2）×２=8(API)

※従量料金　10,000API あたり 10 円

特に１．では Bulk Insert 機能を使って登録するため、「メタ数」「メタデータ登録負荷」「メタデータ登録するエッジ台数」の３つについて充分に考慮してアプリケーション設計いただくようお願いします。

7.4. ポータル画面の操作

ポータル画面より、本機能の初期設定を行います。
設定方法については、「サービスポータル操作マニュアル」を参照ください。
以降、以下の prefix リソース、アクセスコードを設定したものとして説明していきます。

prefix リソース	drc
アクセスコード	drcaccesscode

正しく登録できた場合、以下のリソースが作成されます。ポータル画面または API より以下のリソースが作成されていることを確認してください。

drc/meta/gws
drc/meta/publish
drc/meta/request/search
drc/meta/response/search
drc/meta/response/preprocessed

7.5. エッジ情報の登録(A)

はじめに 7.2 節に記載の A「エッジ情報」に動作するエッジ情報を追加します。
ここではエッジの ID として「gw-001」「gw-002」「gw-003」「gw-004」「gw-005」を登録するものとします。
以下のリソースにエッジの情報を記載します。

以降 7.5 章から 7.9 章まで記載している API 実行時の凡例ですが、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。
詳細は「IoT Platform API リファレンス」を参照してください。

リソース種別	リソース_JSON
リソースパス	drc/meta/gws
データ形式	JSON

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 drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/gws?$bulk=single_resource_path

BODY	[
	{
	"_data": {
	"gw_id": "gw-001"
	}
	},
	{
	"_data": {
	"gw_id": "gw-002"
	}
	},
	{
	"_data": {
	"gw_id": "gw-003"
	}
	},
	{
	"_data": {
	"gw_id": "gw-004"
	}
	},
	{
	"_data": {
	"gw_id": "gw-005"
	}
	}
	]

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 drcaccesscode' -d '[
 {
  “gw_id” : “gw-001"
 },
 {
  “gw_id” : “gw-002"
 },
 {
  “gw_id” : “gw-003"
 },
 {
  “gw_id” : “gw-004"
 },
 {
  “gw_id” : “gw-005"
}
]' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/gws?$bulk=single_resource_path '
```

正しく API を発行できた場合、レスポンスとして「200」が返ります。

正しく登録できた場合、以下のリソースが作成されます。ポータル画面または API より以下のリソースが作成されていることを確認してください。

```
drc/meta/confirm/gw-001
 :
drc/meta/confirm/gw-005
drc/response/progress/gw-001
 :
drc/response/progress/gw-005
drc/response/sent_flags/gw-001
 :
drc/response/sent_flags/gw-005
drc/request/clients/gw-001
 :
drc/request/clients/gw-005
```

7.6. メタデータを格納するリソースの作成(C)

次に 7.2 節に記載の C「分解されたメタデータ」のためのリソースを作成する必要があります。
今回の例では位置情報（緯度 geo_lng、経度 geo_lat）をメタデータの key として登録するため以下のリソースを追加します。必要に応じて追加してください。ここでエッジは実データを 7 日分保存できるものとするため、メタデータの保持期間も 7 日とします。保持期間はエッジにおける実データの最大保存期間を指定することを推奨します。

geo_lng

リソース種別	リソース_JSON
リソースパス	drc/meta/keys/geo_lng
データ形式	JSON

geo_lat

リソース種別	リソース_JSON
リソースパス	drc/meta/keys/geo_lat

データ形式	JSON

API を呼び出すのに必要なパラメータは以下の通りです。
geo_lng

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 drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/keys/geo_lng

BODY	{
	"resource" : {
	"retention_period" : 7
	}
	}

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 drcaccesscode' 'http://<zone>.fujitsu.com/v1/<
テナント ID>/drc/meta/keys/geo_lng' -d '{
"resource" : {
  "retention_period" : 7
}
}'
```

正しく API を発行できた場合、レスポンスとして「201」が返ります。

API を呼び出すのに必要なパラメータは以下の通りです。
geo_lat

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 drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/keys/geo_lat

BODY	{
	"resource" : {
	"retention_period" : 7
	}
	}

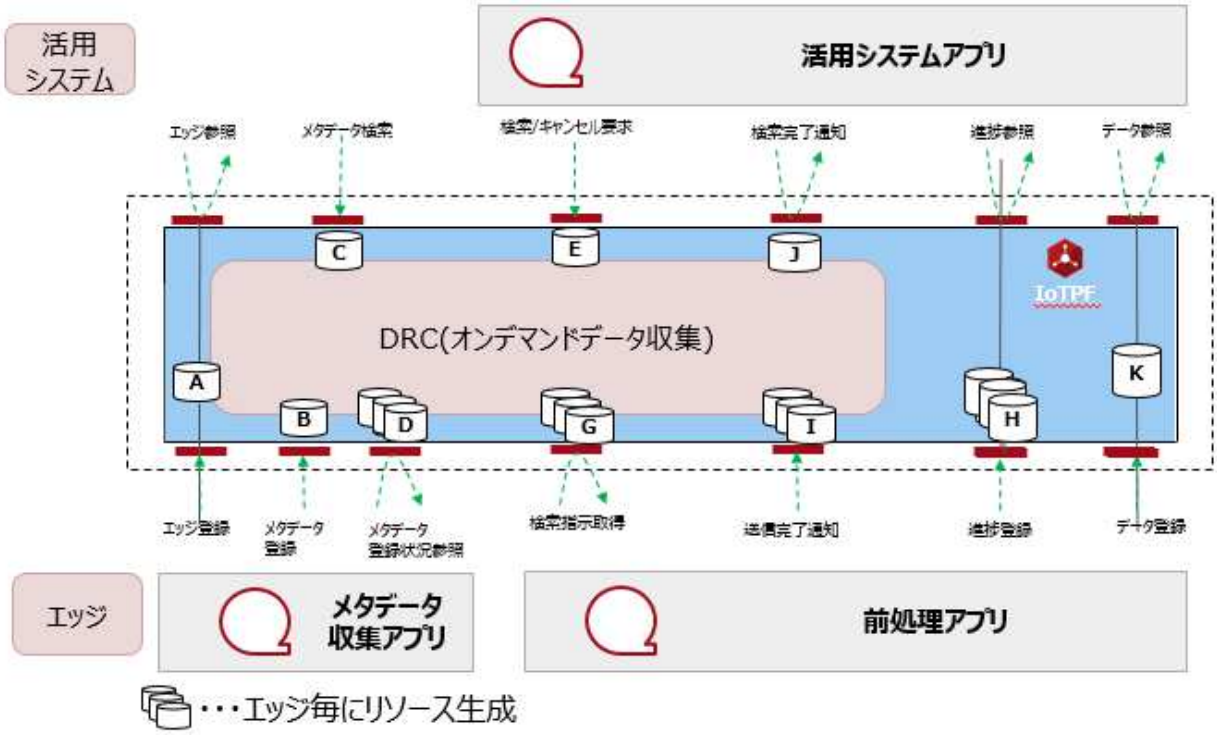
例えば、LinuxOS の端末から curl コマンドを用いて API を呼び出す場合は以下のように実行します。

```
$ curl -i -X POST -H 'Authorization: Bearer drcaccesscode' 'http://<zone>.fujitsu.com/v1/<
テナント ID>/drc/meta/keys/geo_lat' -d '{
"resource" : {
  "retention_period" : 7
}
}'
```

正しく API を発行できた場合、レスポンスとして「201」が返ります。

7.7. エッジにおけるメタデータ収集アプリの開発

本節ではエッジにおけるメタデータ収集アプリにおいて必要な開発手順について記載します。



エッジにおけるメタデータ収集アプリが実施すべき処理の流れについて説明します。

メタデータ登録(B)

↓idx_id を使って一意に登録します。

メタデータ登録状況参照(D)

登録した際の idx_id を使って検索し状況を確認します。

7.7.1. メタデータ登録（B）

エッジにて収集したセンサ・動画像の情報からメタデータを抽出し、本機能へ送信します。また、センサ・動画像の情報をエッジにおいて蓄積します。

ここではエッジである人工衛星が撮影した画像から以下のメタデータを抽出した場合において、メタデータを本機能へ登録する方法を紹介します。

画像ファイル名	S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559.jpg
時刻	2019-02-02T01:28:59.024Z
緯度(geo_lat)	36.12427601168043
経度(geo_lng)	138.77760353347315

以下のリソース_JSON に対してメタデータを登録します。

リソース種別	リソース_JSON
リソースパス	drc/meta/publish
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	PUT
ヘッダフィールド名	Authorization
ヘッダフィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/publish

BODY	{
	"idx_id":"S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559",
	"gw_id":"gw-001",
	"meta": {
	"_date" : "20190202T012859.024Z",
	"geo_lat" : 36.12427601168043,
	"geo_lng" : 138.77760353347315
	}
	}

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 drcaccesscode' -d '{
 "idx_id":"S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559",
 "gw_id":"gw-001",
 "meta": {
   "_date" : "20190202T012859.024Z",
   "geo_lng" : 36.12427601168043,
   "geo_lat" : 138.77760353347315
 }
}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/publish'
```

正しく API を発行できた場合、レスポンスとして「200」が返ります。

7.7.2. メタデータ登録確認（D）

登録されたメタデータは検索しやすいよう分解され整理されます。次に、この登録・分解が成功したか否かを確認します。以下のリソース_JSON に結果が格納されるため、以下のリソース_JSON に対して idx_id が「S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559」に該当する実データを検索・参照します。

リソース種別	リソース_JSON
リソースパス	drc/meta/confirm/gw-001
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	GET
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/confirm/gw-001/_past?$filter=idx_id eq S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559
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。

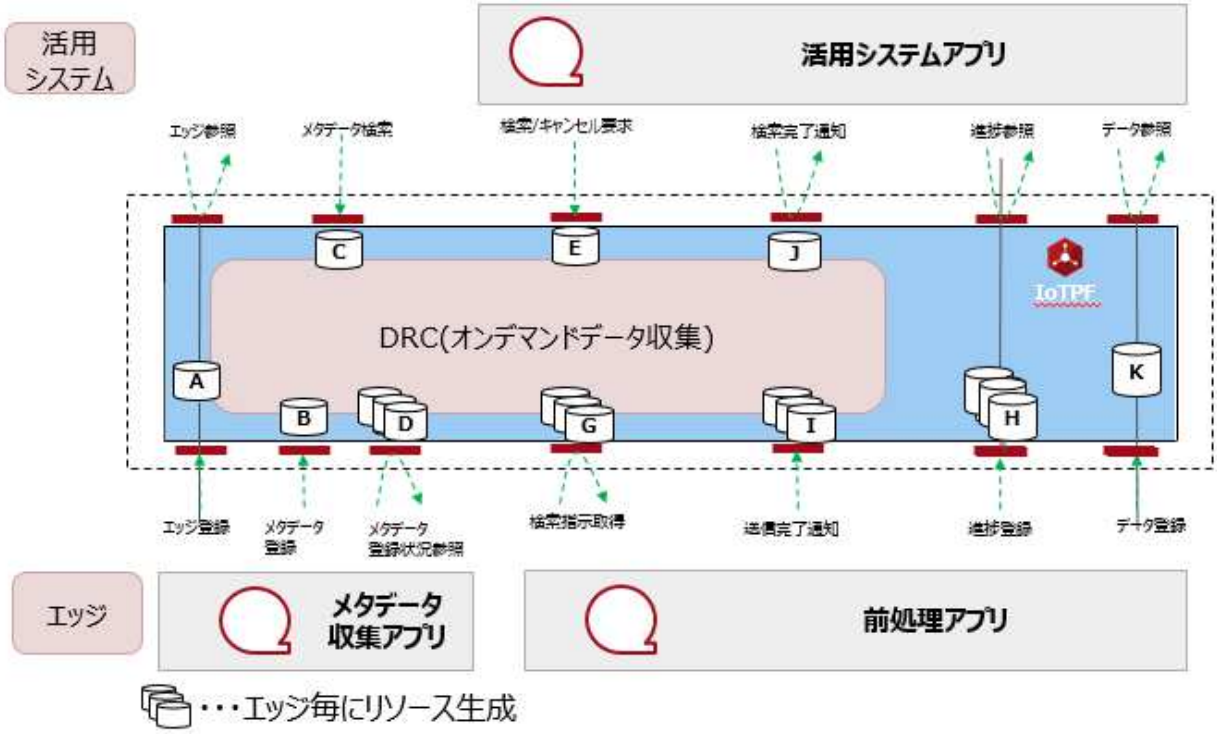
```
$ curl -i -X GET -H 'Authorization: Bearer drcaccesscode'
'http://<zone>.fujitsu.com/v1/< テ ナ ン ト  ID>/drc/meta/confirm/gw-001/_past?$filter=idx_id
eq "S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559"'
```

メタデータの分解に成功した場合、status に succeeded が記載されます。以下の例の場合 geo_lng、geo_lat 両方共成功していることが確認できます。

```
{
 "idx_id" : "S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559",
 "status": {
    "geo_lat": "succeeded",
    "geo_lng": "succeeded",
    "_date" : "20190202T013000.000Z"
}
```

7.8. 活用システムアプリの開発

本節では活用システムにおいて必要な開発手順について記載します。



活用システムにおける活用システムアプリが実施すべき処理の流れについて説明します。

検索要求(E)
　　↓　システム全体で一意な req_id を作成し、これを用いて検索します。
検索結果取得(J)　任意
　　↓　検索要求が正しく本システムに要求されたかを確認します。
進捗状況の取得(H)　任意
　　↓　エッジの検索処理状況を知りたい場合に取得します。
検索要求のキャンセル(E)　任意
　　↓　検索要求をキャンセルしたい場合に実行します。
検索結果取得（J）
　　↓　検索要求時に使用した req_id を用い検索結果を確認します。
実データの取得
　　検索結果が正常に終わった場合、実データを取得します。

7.8.1. 検索要求（E）

収集したメタデータを使って、エッジに対して検索要求を行います。Graceful 機能を使って意図せずエッジに高負荷な指示を出さないよう条件を絞り込むことができます。

ここでは例として、2019 年 2 月 2 日における北緯(geo_lng)が 36～37 度、東経(geo_lat)が 138 度～139 度の画像を取得、ただし取得指示を出すエッジと実データ数は 10 台までで 1 時間経過しても収集できなかった場合は指示を打ち切りするケースについて紹介します。

以下のリソース_JSON に結果が格納されるため、以下のリソース_JSON に対して検索要求データを登録します。

リソース種別	リソース_JSON
リソースパス	drc/request/search
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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。
geo_lng の$filter 条件は「"_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 36 and value lt 37"」、geo_lat の$filter 条件は「"_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 138 and value lt 139"」であるものとします。

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 drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/search

BODY	{
	"req_id" : "request001",
	"type" : "search",
	"preproc" : "returnJPG",
	"keys" : [
	"geo_lng",
	"geo_lat"
	],
	"filters" : [
	"_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 36 and value lt 37",
	"_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 138 and value lt 139"
	],
	"graceful": {
	"searchreq_all_timeout" : 3600,
	"number_limit" : 10,
	"gw_limit" : 10
	},
	"verbose" : true
	}

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 drcaccesscode' -d '{
 "req_id" : "request001",
 "type" : "search",
 "preproc" : "returnJPG",
 "keys" : [
  "geo_lng",
  "geo_lat"
 ],
 "filters" : [
 "_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 36
 and value lt 37",
```

```
 "_date gt 20190202T000000.000Z and _date lt 20190203T000000.000Z and value gt 138
 and value lt 139"
  ],
  "graceful": {
    "searchreq_all_timeout" : 3600,
    "number_limit" : 10,
    "gw_limit" : 10
  },
 "verbose" : true
}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/search'
```

正しく API を発行できた場合、レスポンスとして「200」が返ります。

Memo

preproc はエッジにおいて行いたい処理の名前になります。
もしも、エッジにおいて、動画のエンコード方法や圧縮率など検索内容によって異なる前処理を用意したい場合、preproc を異なる名前にすることによって、エッジの前処理を変更することができます。
その場合、エッジは preproc に応じた前処理を予め用意しておく必要があります。

7.8.2. 進捗状況取得（H）

エッジから活用システムに対して検索処理の途中状況通知がある場合に進捗状況を取得することがでできます。なお、進捗状況データの取得は必須ではありませんが、検索処理を円滑に利用したい場合等にご利用ください。

リソース種別	リソース_JSON
リソースパス	drc/response/progress/$all
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	GET
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/progress/$all/_past?$filter=req_id eq 'request001'
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。

```
$ curl -i -X GET -H 'Authorization: Bearer drcaccesscode'  'http://<zone>.fujitsu.com/v1/<
テナント ID>/drc/response/progress/$all/_past?$filter=req_id eq 'request001'
```

進捗状況データがあった場合、以下のような JSON 取得できます。

```
{
  "req_id": " request001",
  "%complete": 50,
  "gw_id": "gw-001",
  "date": "20190305T123456.000Z",
  "num_tasks": 10,
  "preproc": "returnJPG"
}
```

7.8.3. 検索要求のキャンセル(E)

検索要求が timeout する前に検索要求を中断したい場合は検索要求をキャンセル指定で発行することで中断できます。この場合、該当の検索要求処理は終了しますが、キャンセル時点の検索結果を取得できます。

検索要求をキャンセルしたい場合、以下のリソース_JSON に対して検索キャンセルデータを登録します。

リソース種別	リソース_JSON
リソースパス	drc/request/search
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	PUT
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/search

BODY	{
	"req_id" : "request001",
	"type" : "cancel"
	}

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 drcaccesscode' -d '{
 "req_id" : "request001",
 "type" : "cancel"
}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/search'
```

正しく API を発行できた場合、レスポンスとして「200」が返ります。

7.8.4. 検索結果取得（J）

活用システムアプリにおいて、検索要求（req_id:request001）を行った結果を取得するケースについて紹介します。エッジは完了するまで５分に１回を目安に定期実行してください。

リソース種別	リソース_JSON
リソースパス	drc/response/search
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	GET
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/search/_past?$filter=req_id eq 'request001'
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。

```
$ curl -i -X GET -H 'Authorization: Bearer drcaccesscode' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/search/_past?$filter=req_id eq 'request001'
```

検索に成功した場合、data_status_summary が全件 succeeded となります。
7.9.4 節にて送信した実データの格納先より実データを取得してください。

```
{
  "req_id" : "request001",
  "preproc" : "returnJPG",
  "keys" : [
    "geo_lng",
    "geo_lat"
   ],
   "filters" : [
   "_date gt '20190202T000000.000Z' and _date lt '20190203T000000.000Z and value gt
  36 and value lt 37",
 "_date gt '20190202T000000.000Z'and _date lt '20190203T000000.000Z and value gt 138
 and value lt 139"
  ],
  "graceful": {
    "searchreq_all_timeout" : 3600,
    "number_limit" : 10,
    "gw_limit" : 10
  },
  "verbose" : "false",
  "detail" : {
    "inprogress" : "false",
    "gws_status_summary" : {
```

```
      "processing" : 0,
      "succeeded" : 1,
      "failed" : 0,
      "canceled" : 0
    },
    "data_status_summary" : {
      "processing" : 0,
      "succeeded" : 1,
      "failed" : 0,
      "canceled" : 0
    },
    "gws_status" : {
      "gw-001" : {
        "status" : "succeeded"
    }
   }
 }
}
```

Memo

・verbose を true 指定して検索要求を実行した場合、req_id による検索時に複数件の検索結果データが hit します。最新の 1 件を取得したい場合は top=1 指定を追加して API を呼び出してください。
・検索要求が異常終了した場合は gws_status.error_message にエラー要因が格納されますので。要因に応じた対処実施をお願いします。要因については DRC(オンデマンドデータ収集)インターフェースリファレンスを参照願います。

7.8.5. 実データの取得

最後に実データを取得します。
実データの送信先として今回は以下のリソースを使いますが、別のリソースや外部のファイルサーバなど自由に使っていただいて構いません。

リソース種別	リソース_Binary
リソースパス	_bin/jpg
データ形式	バイナリ

アクセスコードは以下を使用します。

アクセスコード	JPGaccesscode
リソースパス	_bin/jpg
アクセス権限	「U」「R」に権限を付与

HTTP メソッド	GET
ヘッダフィールド名 1	Authorization
ヘッダフィールド値 1	Bearer JPGaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/jpg/_present
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。

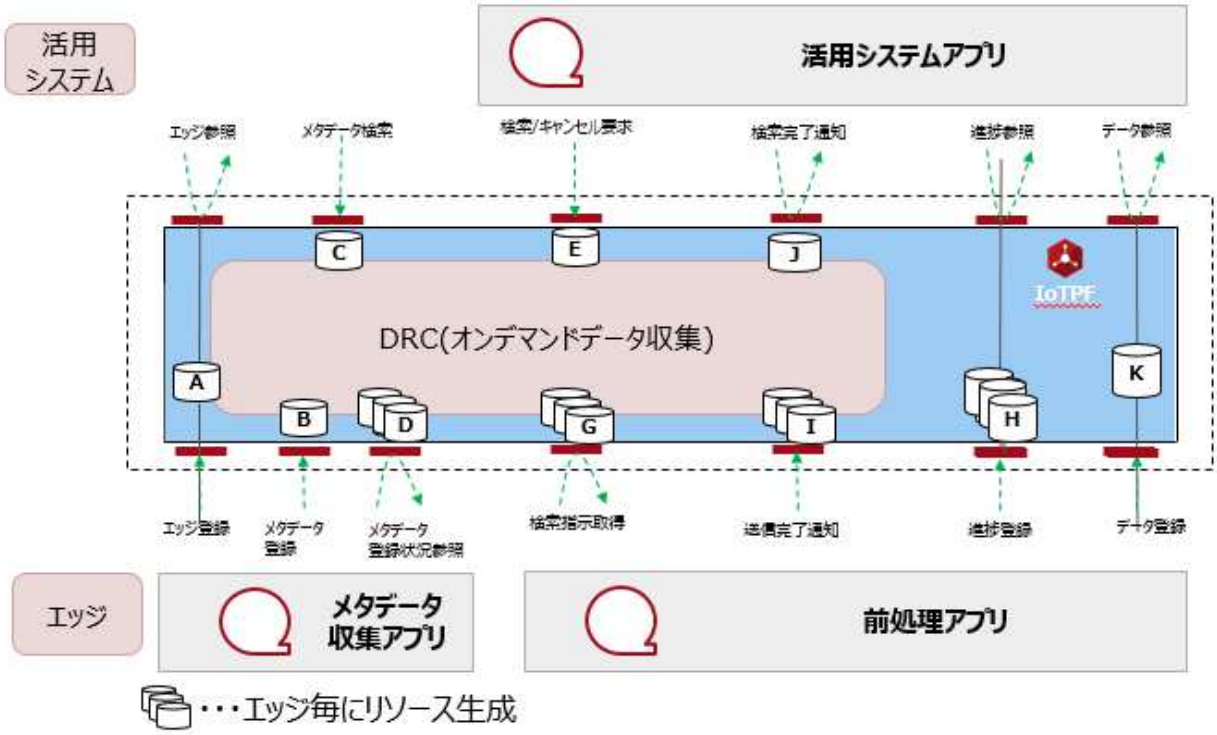
URI の$filter=以降において、非予約文字以外の文字はパーセントエンコードにより置き換える必要があります。「 」（半角スペース）は「%20」、「+」は「%2b」、「'」（シングルクォーテーション）は「%27」、「=」は「%3d」に置き換えてください。

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

正常完了できた場合、レスポンスとして「200」が返り画像データを取得することができます。

7.9. エッジにおける前処理アプリの開発

エッジが本機能の指示を読み取り、前処理を行いその結果を本機能へ送信します。



エッジにおける前処理アプリが実施すべき処理の流れについて説明します。

検索指示取得(G)
　　↓　検索指示を定期的に取得し、前処理をすべきか判断します。
前処理の実施
　　↓　idx_ids、preproc、type に基づき前処理を実施します。
進捗状況の登録(H)　任意
　　↓　活用システムに対して検索処理状況を通知したい場合に登録します。
実データの送信
　　↓　前処理した実データ（ここでは画像）を登録します。
送信完了通知（I）
　　検索指示取得時に記載された req_id を使って実データ送信が終わったことを通知します。

7.9.1. 検索指示取得（G）

エッジが本機能の指示を読み取る API の例を紹介します。
エッジは 5 分に 1 回を目安に定期実行を行い、指示が更新された場合に 7.9.4 節の実データの送信を行ってください。なお、検索指示の刈り取り漏れを防ぐために**前処理アプリは前回(最終)の検索指示を取得した時刻を覚えておくこと**を推奨します。

リソース種別	リソース_JSON
リソースパス	drc/request/clients/gw-001
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	GET
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/clients/gw-001/_past?$filter=_date ge <前回取得した指示の_date の値>
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。

```
$ curl -i -X GET -H 'Authorization: Bearer drcaccesscode' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/request/clients/gw-001/_past?$filter=_date ge 20190203T013002.123Z '
```

検索指示があった場合、以下のような JSON 取得できます。idx_ids はメタデータ登録時に付与した文字列です。（今回のユースケースの場合はファイル名になっています。）対応する実データを 7.9.4 節にて送信します。なお、検索指示が 2 件以上登録されている可能性がありますので、対応する実データの送信についても複数件対処する必要があります。

```
{
  “req_id” : “request001”,
  “type” : “search”,
  "preproc" : "returnJPG",
  “idx_ids” : [
    "S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559"
  ]
}
```

なお、同じ req_id でも再取得時に type が cancel に更新される場合があります。その場合は前処理アプリにて**該当 req_id の検索処理をキャンセルする作りこみ**が必要となります。

7.9.2. 前処理の実施

前処理はお客様で開発されるアプリケーションにて実施ください。前処理として最低限必要な処理は以下のとおりとなります。

1．type が cancel だった場合は、該当の req_id で要求されている、idx_ids および preproc に対応した実データ取得処理をキャンセルする。
2．type が search だった場合は、該当の req_id で要求されている、idx_ids および preproc に対応した実データ取得処理を実行する。

7.9.3. 進捗状況登録（H）

エッジが活用システムに対して検索処理の途中状況を通知したい場合に進捗状況を登録することで通知することができます。なお、進捗状況データの登録は必須ではありませんが、検索処理状況をリアルタイムに通知したい場合等にご利用ください。

リソース種別	リソース_JSON
リソースパス	drc/response/progress/gw-001
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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	PUT
ヘッダ フィールド名	Authorization
ヘッダ フィールド値	Bearer drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/progress/gw-001

BODY	{
	"req_id": " request001",
	"%complete": 50,
	"gw_id": "gw-001",
	"date": "20190305T123456.000Z",
	"num_tasks": 10,
	"preproc": "returnJPG"
	}

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。なお、お客様のご利用状況に応じて追加で他情報を通知したい場合は JSON の key を任意に追加して登録しても問題ありません。

```
$ curl -i -X PUT -H 'Authorization: Bearer drcaccesscode'  'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/progress/gw-001 ' -d '{
  "req_id": " request001",
  "%complete": 50,
  "gw_id": "gw-001",
  "date": "20190305T123456.000Z",
  "num_tasks": 10,
  "preproc": "returnJPG"
} '
```

正しく API を発行できた場合、レスポンスとして「200」が返ります。

7.9.4. 実データの送信

エッジが本機能へ実データを送信する API の例を紹介します。
実データの送信先として今回は以下のリソースを使いますが、別のリソースや外部のファイルサーバなど自由に使っていただいて構いません。

リソース種別	リソース_Binary
リソースパス	_bin/jpg
データ形式	バイナリ

アクセスコードは以下を使用します。

アクセスコード	JPGaccesscode
リソースパス	_bin/jpg
アクセス権限	「U」「R」に権限を付与

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 JPGaccesscode
ヘッダフィールド名２	x-iotpf-meta-data1
ヘッダフィールド値２	S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/jpg
BODY	登録する JPEG ファイル (S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559.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。

```
$ curl -i -X PUT -H 'Authorization:Bearer JPGaccesscode' -H 'x-iotpf-meta-data1: S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559' --data-binary @./S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559.jpg 'http://<zone>.fujitsu.com/v1/<テナント ID>/_bin/jpg'
```

正常完了できた場合、レスポンスとして「200」が返ります。

7.9.5. 送信完了通知（I）

エッジが 7.9.4 節の実データ送信が終了したことを通知する API の例を紹介します。
以下のリソース_JSON に対して送信完了通知データを登録します。

リソース種別	リソース_JSON
リソースパス	drc/response/sent_flags/gw-001
データ形式	JSON

アクセスコードは以下を使用します。

アクセスコード	drcaccesscode
リソースパス	drc
アクセス権限	「CDL」「P」「G」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。req_id については 7.9.1 節にて読み取ったものをそのまま使用してください。

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 drcaccesscode
URI	http://<zone>.fujitsu.com/v1/<テナント ID>/drc/response/sent_flags/gw-001

BODY	{
	"req_id" : "request001",
	"gw_id" : "gw-001",
	"preproc" : "returnJPG",
	"status": "succeeded"
	}

正常完了できた場合、レスポンスとして「200」が返ります。

第8章　メタデータのパーティション分割（partition_id）について

本ドキュメントでは、すべてのメタデータを key 毎に集約し、そのメタデータを元に検索を行う例を説明しました。もしもサービスを利用するにあたって、エッジの規模が極端に大きくなる場合や拠点によって分割してメタデータを管理したい場合、partition_id によってパーティションをわけることにより、メタデータを分割して管理することができます。

本ドキュメントの例ではメタデータは以下のようにメタデータを作成していました。

リソースパス	概要
drc/meta/keys/geo_lng	経度情報のメタデータ
drc/meta/keys/geo_lat	緯度情報のメタデータ

またメタデータを登録する際は、以下のようなコマンドによって送信されていました。

```
$ curl -i -X PUT -H 'Authorization: Bearer drcaccesscode' -d '{
 "idx_id":"S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559",
 "gw_id":"gw-001",
 "meta": {
   "_date" : "2019-02-02T01:28:59.024Z",
   "geo_lng" : 36.12427601168043,
   "geo_lat" : 138.77760353347315
 }
}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/publish'
```

もしも、メタデータが増大し、東日本(jp-west)と西日本(jp-east)に分割したい場合事前に以下のようなメタデータを作成します。この時すべてのメタデータに対してすべての partition_id（今回は jp-west jp-east）が記載されたリソースを作成してください。

リソースパス	概要
drc/meta/keys/geo_lng	経度情報のメタデータ
drc/meta/keys/geo_lat	緯度情報のメタデータ
drc/meta/keys/geo_lng/jp-west	経度情報のメタデータ（東日本用）
drc/meta/keys/geo_lat/jp-west	緯度情報のメタデータ（東日本用）
drc/meta/keys/geo_lng/jp-east	経度情報のメタデータ（西日本用）
drc/meta/keys/geo_lat/jp-east	緯度情報のメタデータ（西日本用）

メタデータを登録する際は以下のように partition_id を追加して登録してください。

```
$ curl -i -X PUT -H 'Authorization: Bearer drcaccesscode' -d '{
 "idx_id":"S2B_MSIL2A_20190202T012859_N0211_R074_T54SUE_20190202T034559",
 "gw_id":"gw-001",
 "meta": {
   "_date" : "2019-02-02T01:28:59.024Z",
   "geo_lng" : 36.12427601168043,
   "geo_lat" : 138.77760353347315
 },
 "partition_id" : "jp-west"
}' 'http://<zone>.fujitsu.com/v1/<テナント ID>/drc/meta/publish'
```

この時、drc/meta/keys/geo_lng/jp-west、drc/meta/keys/geo_lat/jp-west にメタデータは分割・整理され格納されます。

第9章　サンプルアプリ

本機能のサンプルプログラムを以下において公開しています。使い方については下記 URL をご参照ください。

https://github.com/fjiotpf/drcfs-sample

本プログラムは 7 章で紹介した人工衛星のエッジに搭載することを想定したサンプルアプリです。本アプリは Node-RED 上にて動作し、事前に以下の環境が必要になります。