

# DRC-DA プロトタイプについて

## API 一覧

### ディレクトリノード

ディレクトリノードは、IoT-PF による web API を持つ。詳細は [IoT-PF の API リファレンスマニュアル \(https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v3/apireference.pdf\)](https://iot-docs.jp-east-1.paas.cloud.global.fujitsu.com/ja/manual/v3/apireference.pdf) を参照のこと。

IoT-PF の応答には応答内容そのものである `_data` フィールド以外に、`_resource_path` 及び `_date` というフィールドが存在するが、IoT-DA 利用の際には意味を持たない。以下では、`_data` フィールドのみについて記述する。

URI に含まれる空白及びシングルクォートは、URL エンコードが必要になることに注意。

### キー情報取得

取得したいデータが持つメタ情報でもって検索し、該当データを持つインデックスノードの `gw_id` を取得する。

- リクエスト先 URI & メソッド  
GET /keyvalues/<key>/\_past?<filter>=value eq '<value>'&<filter>=gen\_time gt '<unixtime>'
- 応答  
[{"gw\_id":<gw\_id>, "idx\_id":<idx\_id>, "gen\_time":<time>}, {...},...]

### ゲートウェイ情報取得

キー情報取得で得た `gw_id` を指定し、インデックスノードの詳細情報を取得する。

本 API で得られる `gw_ip_and_port` がインデックスノードにアクセスするためのエンドポイントとなる。

- リクエスト先 URI & メソッド (全ゲートウェイ情報)  
GET /gws/\_past
- 応答  
[{"id":<gw\_id>, "address":<gw\_ip\_and\_port>, "status":<status>, "location":<location>},...]  
status 及び location に関しては未実装のため、null が入る。
- リクエスト先 URI & メソッド (個別ゲートウェイ情報)  
GET /gws/\_past?<filter>=id eq '<gw\_id>'
- 応答  
{"id":<gw\_id>, "address":<gw\_ip\_and\_port>, "status":<status>, "location":<location>}

### インデックスノード

インデックスノードは、RabbitMQ による非同期型 (MQTT) インタフェースを持つ。

### インデックス登録

- 送信用 topic  
drc-da/input/index-key/<response\_queue\_id>  
response\_queue\_id には, UUID v4 によって生成された識別子を用いることとする.  
この ID は, 後述の応答受信用 topic に用いる.
- ペイロード  
**sink\_info** フィールドのフォーマットは不定. 今後 IoT-DA が生データ管理まで行う際には規定する予定である.

```
{
  "gen_time":<time>,
  "meta":{ #required
    <key_name>:<value>,...
  },
  "dir_keys":[
    <key_name>,
    <key_name>,...
  ],
  "sink_info":{
    #how to access the sink
    "name":<name>,
    "sink_address":<sink_address>,
    "protocol":<protocol>,
    "content_path":<content_path>
  }
}
```

- 応答受信用 topic  
drc-da/response/<response\_queue\_id>
- 応答内容

<TBD>

## インデックス検索

- 送信用 topic  
drc-da/output/index-key/<response\_queue\_id>
- ペイロード  
クエリの記述方法は, 基本的に MongoDB のクエリ記述方法に従う.

```
{
  "meta":{
    <search_key>:<search_value>,
    <search_key>:{
      "$gte":<value>,
      "$lte":<value>
    }
  },
  "gen_time":{
    "$gte":<unix time>,
    "$lte":<unix time>
  },
  # if want to get the certain index entry
  "idx_id":<idx_id>
}
```

## インデックス通知

- 送信用 topic  
drc-da/output/set-subscription/<response\_queue\_id>
- ペイロード  
インデックス検索と同形。ただし、応答用トピックをサブスクライブし続けることで、登録されたインデックスを即座に受け取る事ができる。

# Getting Started

## 前提

DRC-DA を動作させるためには、コンポーネントとして IoT-PF が必須である。事前に担当者に連絡し、DRC-DA がデフォルトで使用するリソース、及びkeyvalues/ 以下に登録する可能性のあるメタ情報キーのリソースを作成し、これに対するアクセスコード（DRC-DA のコンフィグには read/write 権を持つもの、利用者に対しては keyvalues/\* に対する read 権を持つもの）を発行しておく必要がある。また利用者は、担当者からアクセスコードを受領しておく。

## 準備

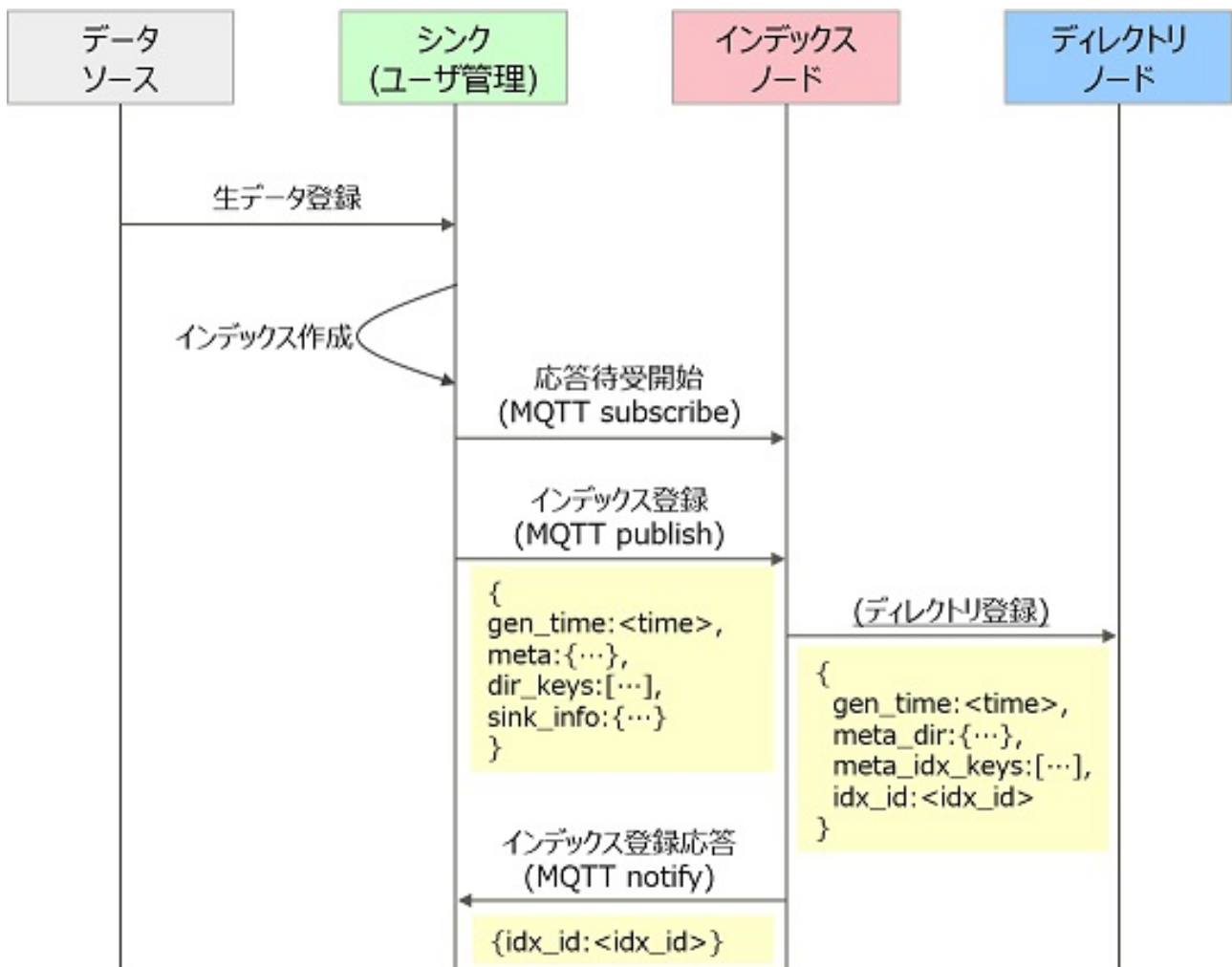
scripts/iotda\_client\_mqtt.rb を手元にダウンロードする。

加えて、ruby (v2.1 以降) 実行環境を構築し、scripts/iotda\_client\_mqtt.rb に必要は gem をインストールする。

```
gem install mqtt
gem install json
```

直接 MQTT クライアントを使うことも可能。その場合には、scripts/iotda\_client\_mqtt.rb が参考になる。

## インデックス情報の登録



登録者は、あらかじめ DRC-DA インデックスノードのエンドポイントを知っている想定とする。

1. 必要ライブラリ, 及び `iotda_client_mqtt.rb` のインポート

```
require '../scripts/iotda_client_mqtt'
require 'optparse'
require 'json'
```

2. クライアントの初期化

```
params = ARGV.getopts(' ', 'host:', 'port:')
client = IoTDAClient.new(host: params["host"], port: params["port"])
```

3. 応答用キュー識別子の生成, 及び応答用キューのサブスクライブ

```
queue_name = SecureRandom.uuid.to_s
client.subscribe_queue(queue_name: queue_name)
sleep(5)
```

`sleep(5)` を行うのは、応答用キューが準備出来ていない状態で、DRC-DA からの応答メッセージが送信されてしまうことを避けるため。

4. インデックス情報の準備

```

device_id = "dev-01"
gen_time = Time.now.to_i
dir_keys = ["location", "device_id"]
meta = {:location => "loc-A", :device_id => device_id, :temp => 25.5, :humid => 60}
sink_info = {:address => "192.168.0.1:8080", :protocol => "http"}

```

上記 API の規定に従って、インデックス情報を準備する。

#### 5. インデックス情報の送信, 及び応答待ち

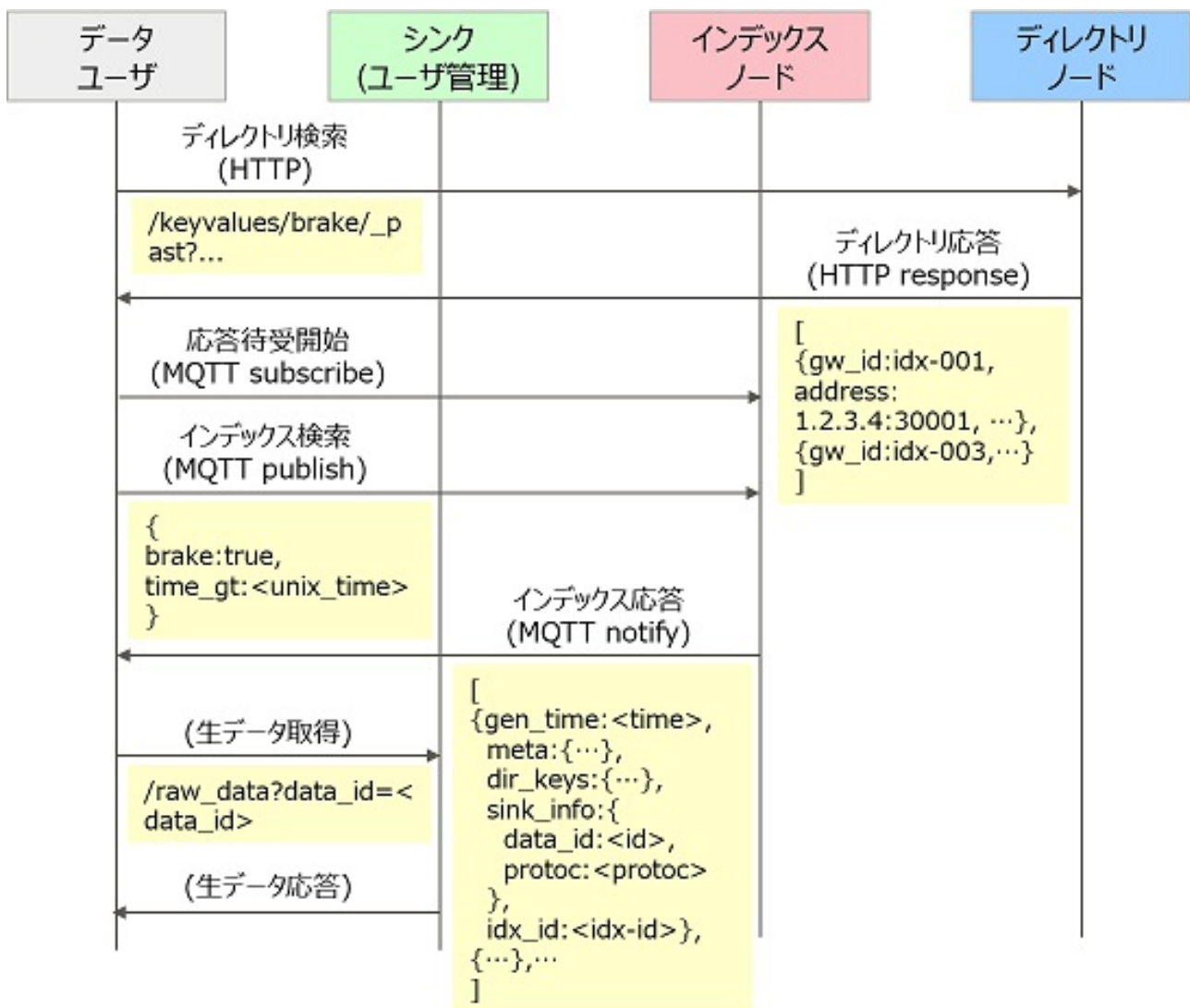
```

client.publish_idx_v2(sink_info: sink_info, dir_keys: dir_keys, meta: meta, gen_time: gen_time,
queue_name: queue_name)
sleep(5)

```

なお, 上記手順に従ったサンプルスクリプトが tests/pub\_idx\_v2.rb にある。

## ディレクトリ情報の取得



IoT-PF のエンドポイント及びアクセスコードは知っている想定とする。

#### 1. 検索用キーバリューを準備する

- ここでは仮に {"location":"loc-A"} を検索したいとする

- 加えて、生成日時が 2017/07/04 15:13:31 +0900 以降であるとする。UNIX タイムに変換すると、1499148815 になる

## 2. 検索用キーバリューを持つインデックスノード一覧を取得する

- リクエスト: `curl -X GET 'http://<IoT-PF エンドポイント>/keyvalues/location?filter%20%27value%27%20eq%20%27loc-A%27%20gen_time%27%20gt%201499148815'`
- レスポンス: `[{"gw_id":"idx-001", "idx_id":"<uuid>", "gen_time":1499148900}]`

## 3. レスポンスに含まれるインデックスノードの詳細情報を取得する

- リクエスト: `curl -X GET -H 'Authorization: Bearer <token>' 'http://<IoT-PF エンドポイント>/gws?filter%20%27id%27%20eq%20%27idx-001'`
- レスポンス: `{"id":"idx-001", "address":"10.10.10.xx:1883", "status":null, "location":null}`

3 で取得したインデックスノード詳細情報に含まれる `address` が、下記インデックス情報の取得で用いるエンドポイントとなる。

# インデックス情報の取得

ディレクトリ情報の取得で、取得したインデックス情報が登録されている DRC-DA インデックスノードのエンドポイントが取得出来たとする。取得したエンドポイントに対し、以下を実行する。

## 1. 必要ライブラリ、及び `iotda_client_mqtt.rb` のインポート

```
require '../scripts/iotda_client_mqtt'
require 'optparse'
require 'json'
```

## 2. クライアントの初期化

```
params = ARGV.getopts(' ', 'host:', 'port:')
client = IoTDAClient.new(host: params["host"], port: params["port"])
```

## 3. 応答用キュー識別子の生成、及び応答用キューのサブスクライブ

```
queue_name = SecureRandom.uuid.to_s
client.subscribe_queue(queue_name: queue_name, search_keyvalue: search_keyvalue)
sleep(5)
```

## 4. 検索用キーバリューの準備

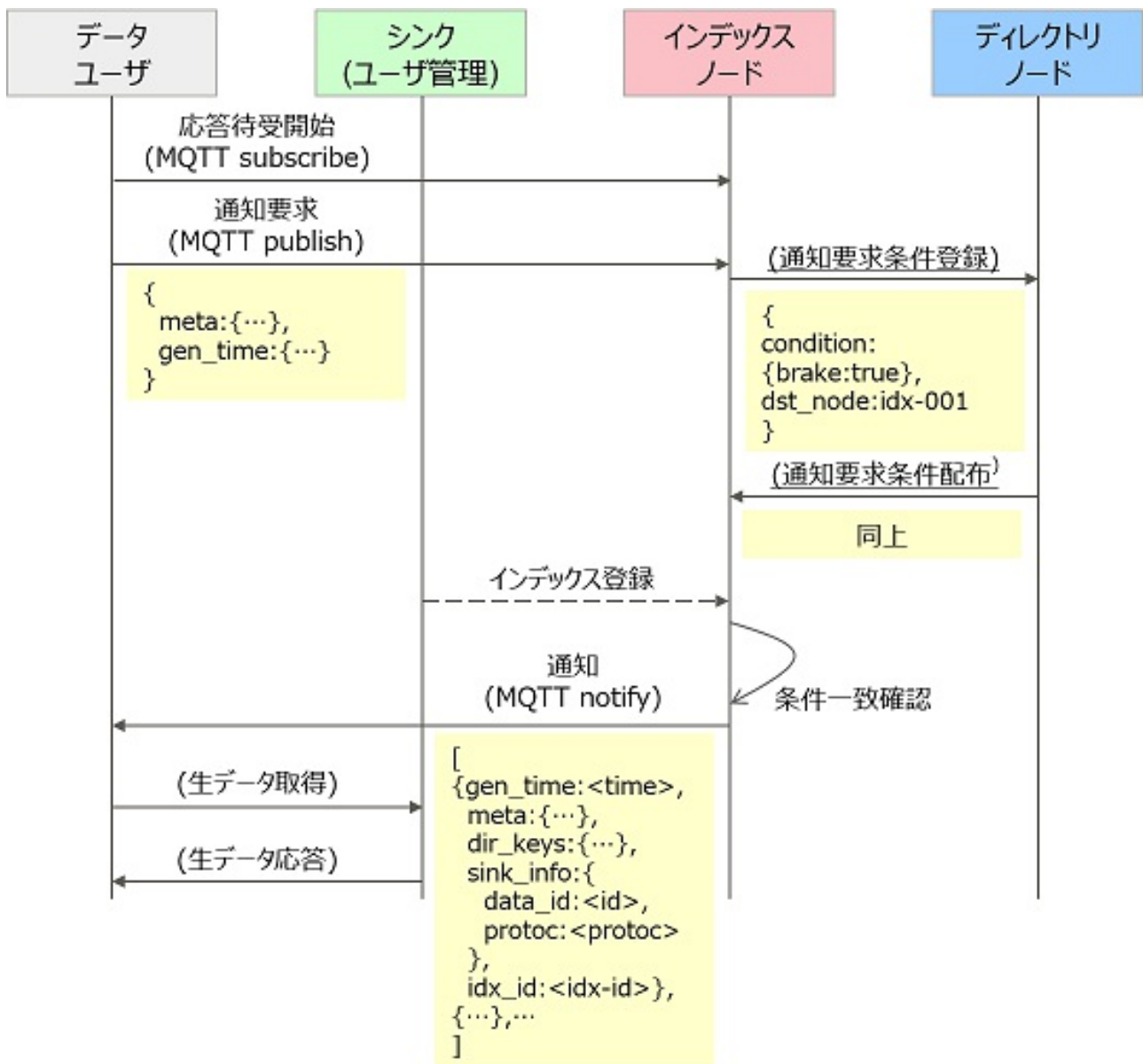
```
search_keyvalue = {"location" => "loc-A", "temp" => 25.5}
```

## 5. インデックス情報取得要求を送信、及び応答待ち

```
client.search_idx_v2(search_keyvalue: search_keyvalue, queue_name: queue_name)
sleep(5)
```

なお、上記手順に従ったサンプルスクリプトが `tests/get_idx_v2.rb` にある。

# インデックス情報の取得（通知待ち）



1 から 4 までの手順は、上記 "インデックス情報の取得" と同様。その後、以下を行う。

5. 通知待ちスレッドを実行、及び応答待ち

```

response = nil
t = Thread.new() do
  response = client.subscribe(node_id: "node-sub", subscribe_keys: search_keyvalue, timeout:
30)
end
t.join
  
```

"インデックス情報の登録" 手順がされることで、上記スレッドが response としてインデックス情報を受け取る。

上記コードでは response を受け取るとスレッドが終了するが、続けてサブスクライブすることも出来る。

## DRC-DA セットアップ手順

### コンフィグ

コンフィグは config/config.properties において記述する.

```
#common settings
LOG_LEVEL={TRACE|DEBUG|INFO|ERROR}

OWN_ID= #attach an unique-id to each node

AS_DIRECTORY=true
AS_INDEX=false

IOTPF_HTTP_HOST=
IOTPF_MQTT_HOST=
IOTPF_MQTT_PORT=
IOTPF_USER=
IOTPF_PASSWORD=
IOTPF_RESOURCE_ROOT=
IOTPF_BIN_RESOURCE_ROOT=
IOTPF_TOKEN=

# For index node settings
RABBITMQ_HOST=
MONGODB_HOST=
MEMCACHED_HOST=

DATA_CLEAR_PERIOD=

# For directory node settings (nothing to be configured now)
```

## Copyright について

COPYRIGHT Fujitsu Limited 2017 and FUJITSU LABORATORIES LTD. 2017