```
{
        "timestamp":＜日時＞,
        "source": ＜データソース＞,
        "data":[{"temperature":気温,"pressure":気圧, "humidity":湿度}]
}
```

timestamp：データ全体の日時。YYYYMMDDhhmmss 形式の文字列。省略時には現在の日時がサーバ側で設定される。

source：データ送信元の ID を任意の文字列で指定する。tag(source)に入る。通常は装置識別子。

data：データの配列。キーが field で値が value に入る。順序に意味は持たせない。基本的には任意のフィールド名を付けられる。

リスト 1

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import sys
import json
import datetime
import influxdb_client
from influxdb_client.client.write_api import SYNCHRONOUS

DEBUG = False # True

# InfluxDB にアクセスするライブラリ

class InfluxDB:
    # コンストラクタ
    def __init__(self, url, org, token):
        print("Constructor of InfluxDB")
        self.url = url
        self.organization = org
        self.api_token = token
        # InfluxDB へ接続するクライアント作成
        self.client = influxdb_client.InfluxDBClient(
            url=url,
            token=token,
            org=org
        )
        # Bucket API 呼び出し準備
        self.bucket_api = self.client.buckets_api()
        # 検索 API 呼び出し準備
        self.query_api = self.client.query_api()
        # 書き込み API 呼び出し準備
        self.write_api = self.client.write_api(write_options=SYNCHRONOUS)

    # デストラクタ
    def __del__(self):
        print("Destructor of InfluxDB")


    # Bucket 作成
    def create_bucket(self, bucket_name):
        print("create_bucket")
        # bucket 存在確認
        bkt = self.bucket_api.find_bucket_by_name(bucket_name=bucket_name)
        if DEBUG : print(bkt)

        # 存在しなければ bucket 作成
```

```
48              if bkt == None:
49                  print("bucket " + bucket_name + " is Not exist.")
50                  self.bucket_api.create_bucket(bucket_name=bucket_name, org=self.organization)
51
52
53          # Bucket 削除
54          def delete_bucket(self, bucket_name):
55              print("delete_bucket")
56              # bucket 存在確認
57              bkt = self.bucket_api.find_bucket_by_name(bucket_name=bucket_name)
58              if DEBUG : print(bkt)
59
60              #  存在すれば bucket 削除
61              if bkt != None:
62                  print("bucket " + bucket_name + " exist.")
63                  self.bucket_api.delete_bucket(bkt)
64
65          def json2point(self, measurement, data_json):
66              data =json.loads(data_json)
67              if DEBUG : print(data)
68
69              # tag
70              p = influxdb_client.Point(measurement).tag("source", data['source'])
71
72              # timestamp
73              if 'timestamp' in data:
74                  if data['timestamp']:
75                      p.time(datetime.datetime.strptime(data['timestamp'], "%Y%m%d%H%M%S"))
76
77              # field
78              r = []
79              it = None
80              for d in data['data']:
81                  if DEBUG : print(d)
82                  for k,v in d.items():
83                      if DEBUG : print(k,v)
84                      p.field(k, v)
85
86              return p
87
88
89          # データ保存
90          """
91          データは以下の JSON 形式でもらう。timestamp は UTC で指定。
92          {
93                  "timestamp":,
94                  "source":,
95                  "data":[]
96          }
97          """
98          def put_data(self, bucket_name, measurement, data):
99              print("put_data:", measurement, data)
100             r = self.json2point(measurement, data)
101             self.write_api.write(bucket=bucket_name, org=self.organization, record=r)
102
103
104         # データ取得
105         def get_data(self, bucket_name, measurement, start=None, stop=None, source=None):
106             print("get_data:",bucket_name, measurement, start, stop)
```

```
107         # 検索
108         """
109         start,stop は UTC で指定。
110         """
111         range = ''
112         if start != None:
113             range = '|> range(start: ' + str(int(datetime.datetime.strptime(start, "%Y%m%d%H%M%S").replace(tzinfo=datetime.timezone.utc).timestamp()))
114         else:
115             range = '|> range(start: 0'
116 
117         if stop != None:
118             range = range + ', stop: ' + str(int(datetime.datetime.strptime(stop, "%Y%m%d%H%M%S").replace(tzinfo=datetime.timezone.utc).timestamp())) + ')'
119         else:
120             range = range + ', stop: now())'
121 
122         query = ' from(bucket:"' + bucket_name + '") ' + range
123 
124         if measurement:
125             query = query + '|> filter(fn:(r) => r._measurement == "' + measurement + '")'
126 
127         if source:
128             query = query + '|> filter(fn:(r) => r.source == "' + source + '")'
129 
130 
131         if DEBUG : print(query)
132 
133         result = self.query_api.query(org=self.organization, query=query)
134 
135         if DEBUG:
136             for table in result:
137                 #print(table)
138                 for record in table.records:
139                     print(record.get_measurement())
140                     print(record.get_time())
141                     print(record.get_field())
142                     print(record.get_value())
143                     for v in record.values:
144                         print(v)
145                         print(record[v])
146 
147         """
148         結果は、以下の形式で返す
149         {
150                     "start":"",
151                     "stop":"",
152                     "bucket":
153                     data:[{"timestamp":"xxx","measurement":"mmm", "source":"xxx", フィールド:値},...]
154            }
155 
156         """
157         res = {"start":start,"stop":stop, "bucket":bucket_name}
158         dp = []
159         for table in result:
160             for record in table.records:
161                 if DEBUG: print("RECORD: ",record)
162                 d = {"timestamp":record.get_time().strftime('%Y%m%d%H%M%S'), "measurement":record.get_measurement(), "source":record['source'],
record.get_field():record.get_value()}
163                 if DEBUG : print(d)
```

```
165                     dp.append(d)
166
167             res['data'] = dp
168             if DEBUG : print(res)
169
170             return res
```

リスト２

```
 1  #!/usr/bin/env python
 2  # -*- coding: utf-8 -*-
 3
 4
 5  import sys
 6  import json
 7
 8  from fastapi import APIRouter
 9  from pydantic import BaseModel
10  from typing import Union, List, Optional
11
12  from LIB import db
13  from LIB import influxdb
14
15  DEBUG = False # True
16
17  router = APIRouter(
18      prefix="/data",
19      tags=["data"],
20  )
21
22  # InfluxDB へのアクセス情報取得
23  s = db.DB()
24  idb_rec = s.select(db.DsInfluxDB, "InfluxDB")
25  if DEBUG : print("idb_rec=",idb_rec)
26  if idb_rec != None:
27      url = "http://" + idb_rec.address + ":" + str(idb_rec.port)
28      organization = idb_rec.organization
29      token = idb_rec.token
30  else:
31      url = "http://localhost:8086"
32      organization = "MyDataServer"
33      token = "jG99-jU4YqMZixUN3IGoBCdQ7iAbekCXPkaM2SEvHXt0lXEQKMDkJppaiXYn66bnsOuaEDZ323k l p3EYIHw-zg=="
34
35  if DEBUG:
36      print("url = ", url)
37      print("organization = ", organization)
38      print("token = ", token)
39
40  idb = influxdb.InfluxDB(url, organization, token)
41
42
43  """
44  InfluxDB
45  """
46  class DList(BaseModel):
47      timestamp: Optional[str]
48      source: str
```

```
49          data: List[dict]
50
51      @router.get("/")
52      async def get_data(channel_id: str, measurement: Union[str, None] = None, start: Union[str, None] = None, stop: Union[str, None] = None):
53          print("get_data", channel_id, measurement, start, stop)
54          """
55          データを取得する
56          """
57          res = idb.get_data(channel_id, measurement, start, stop, None)
58          if DEBUG : print(res)
59
60          return {"Status": "OK", "data":res}
61
62
63
64      @router.post("/{channel_id}/{measurement}")
65      async def put_data(channel_id: str, measurement: str, data : DList):
66          print("put_data:", channel_id, measurement, data)
67          """
68          データを登録する
69          """
70          idb.put_data(channel_id, measurement, data.json())
71          return {"Status": "OK"}
72
73
```

リスト３

```
(mydata_server) tsuchiya@macbook ~ % pip install influxdb-client
Collecting influxdb-client
  Downloading influxdb_client-1.29.1-py3-none-any.whl (698 kB)
     |████████████████████████████████| 698 kB 4.1 MB/s
Collecting pytz>=2019.1
  Downloading pytz-2022.1-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB 4.2 MB/s
Collecting rx>=3.0.1
  Downloading Rx-3.2.0-py3-none-any.whl (199 kB)
     |████████████████████████████████| 199 kB 3.5 MB/s
Collecting urllib3>=1.26.0
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 2.8 MB/s
Collecting python-dateutil>=2.5.3
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 6.1 MB/s
Requirement already satisfied: certifi>=14.05.14 in ./opt/anaconda3/envs/mydata_server/lib/python3.9/site-packages (from influxdb-client) (2021.10.8)
Requirement already satisfied: setuptools>=21.0.0 in ./opt/anaconda3/envs/mydata_server/lib/python3.9/site-packages (from influxdb-client) (58.0.4)
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, urllib3, rx, pytz, python-dateutil, influxdb-client
Successfully installed influxdb-client-1.29.1 python-dateutil-2.8.2 pytz-2022.1 rx-3.2.0 six-1.16.0 urllib3-1.26.9
(mydata_server) tsuchiya@macbook ~ %
Last login: Wed Jun 22 09:21:01 on ttys005
. /Users/tsuchiya/opt/anaconda3/bin/activate && conda activate /Users/tsuchiya/opt/anaconda3/envs/mydata_server;
(base) tsuchiya@macbook ~ % . /Users/tsuchiya/opt/anaconda3/bin/activate && conda activate /Users/tsuchiya/opt/anaconda3/envs/mydata_server;
(mydata_server) tsuchiya@macbook ~ % pip install influxdb-client
Collecting influxdb-client
```

```
  Downloading influxdb_client-1.29.1-py3-none-any.whl (698 kB)
     |████████████████████████████████| 698 kB 4.1 MB/s
Collecting pytz>=2019.1
  Downloading pytz-2022.1-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB 4.2 MB/s
Collecting rx>=3.0.1
  Downloading Rx-3.2.0-py3-none-any.whl (199 kB)
     |████████████████████████████████| 199 kB 3.5 MB/s
Collecting urllib3>=1.26.0
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 2.8 MB/s
Collecting python-dateutil>=2.5.3
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 6.1 MB/s
Requirement already satisfied: certifi>=14.05.14 in ./opt/anaconda3/envs/mydata_server/lib/python3.9/site-packages (from influxdb-client) (2021.10.8)
Requirement already satisfied: setuptools>=21.0.0 in ./opt/anaconda3/envs/mydata_server/lib/python3.9/site-packages (from influxdb-client) (58.0.4)
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, urllib3, rx, pytz, python-dateutil, influxdb-client
Successfully installed influxdb-client-1.29.1 python-dateutil-2.8.2 pytz-2022.1 rx-3.2.0 six-1.16.0 urllib3-1.26.9
(mydata_server) tsuchiya@macbook ~ %
```

リスト４

```
$ cd $HOME/CQ/mydata_server
$ export PYTHONPATH="`pwd`/LIB:`pwd`/APP:$PYTHONPATH"
$ ./mydata_server.py
```

リスト５

```
INFO:Started server process [13386]
INFO:Waiting for application startup.
INFO:Application startup complete.
INFO:Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

リスト６

```
{
  "timestamp": "20220910113015",
  "source": "test",
  "data": [
    {"temperature":30.0,"pressure":1000.1, "humidity":40.2}
  ]
}
```

リスト７

```
{
  "timestamp": "20220910113115",
  "source": "test",
  "data": [
    {"temperature":31.5,"pressure":1000.1, "humidity":39.2}
  ]
}
```

リスト８