FLOW:

1) Go to tutorials.MQTT-IDAS-CYGNUS-MYSQL
2) *$ docker-compose -p fiware up -d*
3) Go to POSTMAN & check health of 3 services: IoT agent, orion context broker, Cygnus
   To check health,
4) Then check if service is registered, i.e go to GET REGISTERED SERVICES
   If not registered, i.e count:0, then do-> Register a Service **(STEP1)**
5) Check if device is registered, i.e get registered devices. If not regd, then do->Provision an IoT device**(STEP 3)**
6) Run battery_report.py
7) Check if "value" : xyz received in **GEt published data in context broker(STEP 6)**
   If not received, try creating new "provision an IoTdevice" then update topic in python script then follow steps 6& 7.
8) Now stop battery_report.py & go to CYGNUS->Subscribe to context changes
   a. POST & http://localhost:1026/v2/subscriptions/
   b. Add 3 headers
   c. In body, write

```
{
 "description": "Notify Cygnus of all context changes",
 "subject": {
        "entities": [
        {
        "idPattern": ".*"
        }
        ]
 },
 "notification": {
        "http": {
        "url": "http://cygnus:5050/notify"
        }
 },
 "throttling": 1
}
```

   d. Click **SEND.**

for success, Check STATUS- 201 Created

9) Then run py file
10) Then go to **"Check subscriptions is working"**,
    a. GET & http://localhost:1026/v2/subscriptions/
    b. Add 2 headers
    c. Click SEND.
       If status 200 OK then successful.
11) Now, go to tutorials.MQTT-IDAS-CYGNUS-MYSQL/

12) Type $ docker exec -it db-mysql mysql -h mysql-db -P 3306 -u root -p123

```
mysql>
mysql> show databases;
| information_schema |
| mysql              |
| openiot            |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.18 sec)

mysql> show schemas;
| information_schema |
| mysql              |
| openiot            |
| performance_schema |
| sys                |
+--------------------+

mysql> show tables from openiot;
show tables from openiot;
+----------------------------------+
| Tables_in_openiot                |
+----------------------------------+
| urn_ngsi-ld_TempHumd_002_MyDevice |
| urn_ngsi-ld_TempHumd_003_MyDevice |
| urn_ngsi_ld_TempHumd_004_MyDevice |
+----------------------------------+

mysql> SHOW DATABASES;
| information_schema |
| mysql              |
| openiot            |
| performance_schema |
| sys                |

mysql> use openiot;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed

mysql> SELECT * FROM urn_ngsi_ld_TempHumd_004_MyDevice;
```

```
mysql> SELECT recvTime, attrName,attrType,attrValue from
urn_ngsi_ld_TempHumd_004_MyDevice;
+----------------------+-----------------+----------+----------------------+
| recvTime             | attrName        | attrType | attrValue            |
+----------------------+-----------------+----------+----------------------+
| 2020-10-17 19:08:38.144 | temperature:room | Integer  |                      |
| 2020-10-17 19:08:38.144 | TimeInstant      | DateTime | 2020-10-17T19:08:37.00Z |
| 2020-10-17 19:09:10.72  | b                | string   | 909                  |
| 2020-10-17 19:09:10.72  | TimeInstant      | DateTime | 2020-10-17T19:09:09.00Z |
| 2020-10-17 19:09:10.72  | temperature:room | Integer  |                      |
| 2020-10-17 19:09:20.72  | b           | string   | 909                  |
| 2020-10-17 19:09:20.72  | TimeInstant      | DateTime | 2020-10-17T19:09:19.00Z |
| 2020-10-17 19:09:20.72  | temperature:room | Integer  |                      |

| 2020-10-17 19:10:30.216 | TimeInstant      | DateTime | 2020-10-17T19:10:30.00Z |
| 2020-10-17 19:10:30.216 | temperature:room | Integer  |                      |
+----------------------+-----------------+----------+----------------------+
29 rows in set (0.00 sec)
```

Checking Health of IoT agent and Context broker services

POSTMAN:

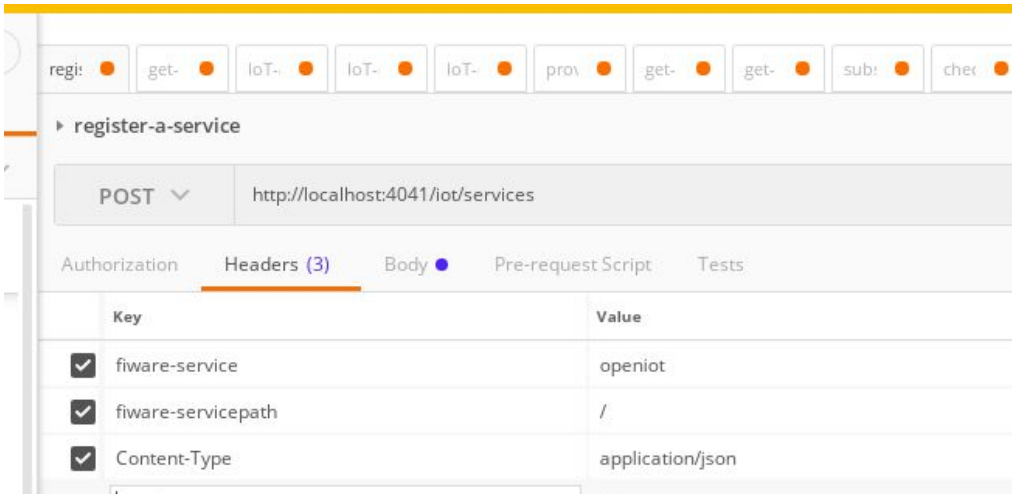
NEW-> REquest-> enter request name as <iot agent> health check->select folder. Click SAVE.
=> In Collections, go to IoT agent health check,
1-select GET
2-http://localhost:4041/iot/about
port number check from 'docker ps' for IoT agent and click SEND.
3-in body, nothing
4- click on SEND.
If STATUS is 200 OK means successful.

=>Similarly For Orion context broker health check

Connecting IoT Devices

STEP 1) Registering a service

a. NEW-> REquest-> enter request name as <register a service>->select folder. Click SAVE.
b. Select **POST** & url = http://localhost:4041/iot/services (POST needs body)
c. Add 3 headers



d. In body, add

```
{
 "services": [
   {
        "apikey":      "5jggokgpepnvsb2uv4s40d59ov",
        "cbroker":     "http://orion:1026",
        "entity_type": "MyDevice",
        "resource":    ""
   }
 ]
}
```

e. **<u>Same APIkey should be in python file.</u>**
Click **SEND.**

for success, Check STATUS- 201 Created

[Provisioning a Service Group for MQTT
For MQTT communication, provisioning supplies the authentication key so the IoT Agent will know which topic it must subscribe to.]

STEP 2) Get registered services

a. New-> REquest-> enter request name as <get registered service>->select folder. Click SAVE.

b. Select **GET** & url = http://localhost:4041/iot/services (GET needs no body)
c. Add 2 headers

d. In body, add nothing
e. Click **SEND.**

for success, Check STATUS- 200 OK

STEP 3) Provision IoT device

a. New-> REquest-> enter request name as <register a iot device>->select folder. Click SAVE.
b. Select **POST** & url = http://localhost:4041/iot/devices (POST needs body)
c. Add 3 headers

d. In body, add

```
{
 "devices": [
  {
        "device_id":   "TempHumd005",
        "entity_name": "urn:ngsi_ld:TempHumd:005",
        "entity_type": "MyDevice",
        "protocol":"JSON",
        "transport":"MQTT",
        "timezone":     "Asia/India",
        "attributes": [
        { "object_id": "t", "name": "temperature:room", "type": "Integer" }
        ] }
 ]
}
```

e. Click **SEND.**

for success, Check STATUS- 201 Created

STEP 4) Get registered devices

a. New-> REquest-> enter request name as <get registered devices>->select folder. Click SAVE.
b. Select **GET** & url = http://localhost:4041/iot/devices (GET needs no body)
c. Add 2 headers

d. In body, add nothing
e. Click **SEND.**

for success, Check STATUS- 200 OK

In body, you'll get

{

"count": 5,
"devices":
}

STEP 5) Publish message to a particular topic "t|30"

a. In Battery_.py python script, change these:
   topic = "TempHumd005/attrs" check latest from get regd devices
   api_key = "/5jggokgpepnvsb2uv4s40d59ov" same as from ‘registering a service STEP1’
b. In **client.publish(topic_battery,jsonFormat_battery),** data must be in ultralight form
c. Now run python script.

STEP 6) GEt published data in context broker

a. New-> REquest-> enter request name as <get published data>->select folder. Click SAVE.
b. Select **GET** & url = http://localhost:1026/v2/entities/urn:ngsi_ld:TempHumd:005/
   Check ‘urn:ngsi_ld:TempHumd:005/’ from provision a device entity-name