

Publish Data to the IBM Cloud

Date	4 November 2022
Team Id	PNT2022TMID17100
Project Name	Project - Signs with smart connectivity for Better road safety

Signs with smart connectivity for Better road safety

Python code to access subscriber:

```
import paho.mqtt.client as paho
import time
import random
def on_publish(client, userdata, mid):
    print("Publish the data")

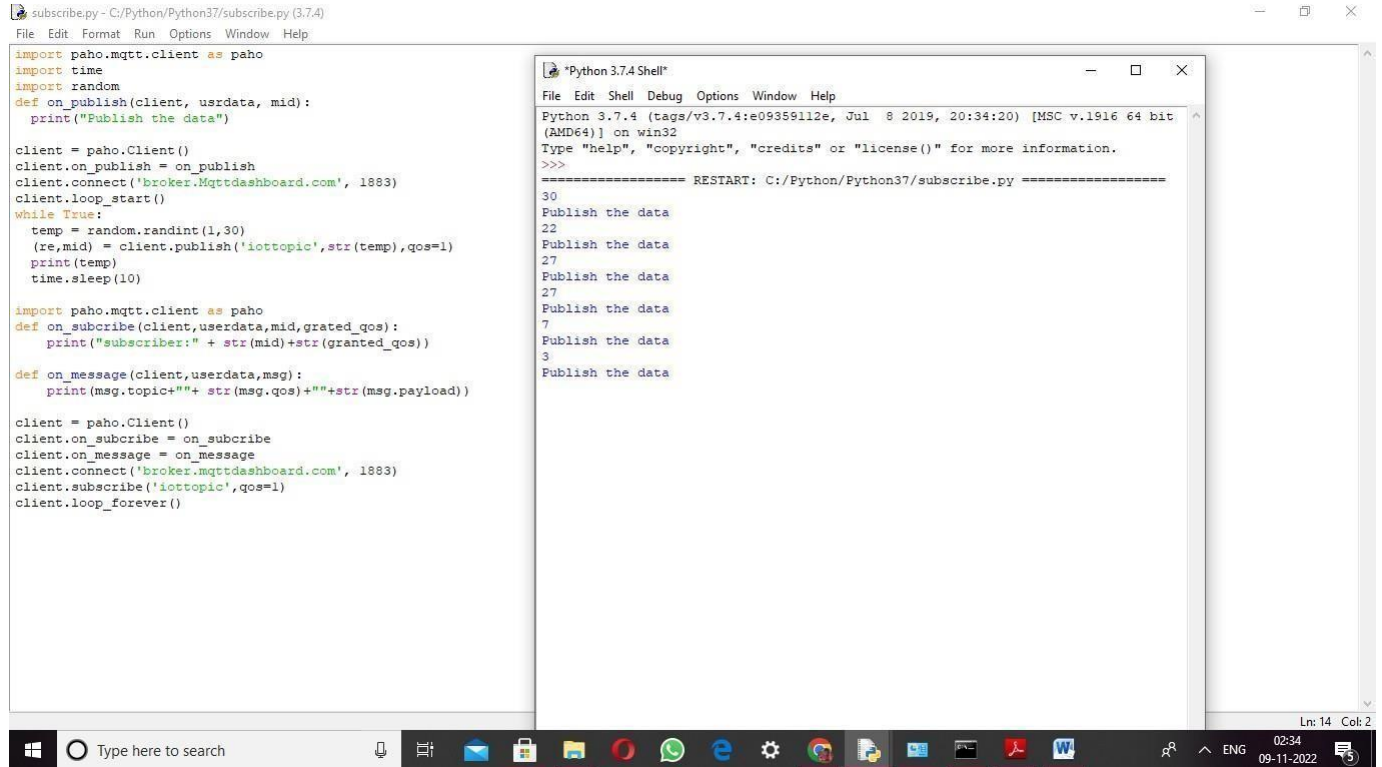
client = paho.Client()
client.on_publish = on_publish
client.connect('broker.Mqttdashboard.com', 1883)
client.loop_start()
while True:
    temp = random.randint(1,30)
    (re,mid) = client.publish('iottopic',str(temp),qos=1)
    print(temp)
    time.sleep(10)

import paho.mqtt.client as paho
def on_subscribe(client,userdata,mid,grated_qos):
    print("subscriber:" + str(mid)+str(granted_qos))

def on_message(client,userdata,msg):
    print(msg.topic+""+ str(msg.qos)+""+str(msg.payload))

client = paho.Client()
client.on_subscribe = on_subscribe
client.on_message = on_message
```

```
client.connect('broker.mqttdashboard.com', 1883)
client.subscribe('iottopic', qos=1)
client.loop_forever()
```



The screenshot shows a Windows desktop environment. On the left, a text editor window titled 'subscribe.py - C:/Python/Python37/subscribe.py (3.7.4)' contains a Python script. The script imports paho.mqtt.client as paho, time, and random. It defines an on_publish function that prints 'Publish the data'. It then creates a paho.Client, sets on_publish, connects to 'broker.mqttdashboard.com' on port 1883, starts the loop, and enters a while True loop that publishes random data to 'iottopic' every 10 seconds. Below this, it defines an on_subscribe function that prints subscriber information, sets on_subscribe, connects to the same broker, subscribes to 'iottopic' with qos=1, and enters a loop_forever loop. On the right, a 'Python 3.7.4 Shell' window shows the output of the script, displaying 'Publish the data' multiple times. The Windows taskbar at the bottom shows the search bar, task view, and various application icons. The system tray on the right shows the date and time as 02:34 on 09-11-2022.

```
import paho.mqtt.client as paho
import time
import random
def on_publish(client, userdata, mid):
    print("Publish the data")

client = paho.Client()
client.on_publish = on_publish
client.connect('broker.mqttdashboard.com', 1883)
client.loop_start()
while True:
    temp = random.randint(1,30)
    (re,mid) = client.publish('iottopic',str(temp),qos=1)
    print(temp)
    time.sleep(10)

import paho.mqtt.client as paho
def on_subscribe(client,userdata,mid,grated_qos):
    print("subscriber:" + str(mid)+str(granted_qos))

def on_message(client,userdata,msg):
    print(msg.topic+" "+ str(msg.qos)+" "+str(msg.payload))

client = paho.Client()
client.on_subscribe = on_subscribe
client.on_message = on_message
client.connect('broker.mqttdashboard.com', 1883)
client.subscribe('iottopic',qos=1)
client.loop_forever()
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python/Python37/subscribe.py =====
30
Publish the data
22
Publish the data
27
Publish the data
27
Publish the data
7
Publish the data
3
Publish the data
```

PROGRAM:

#IBM Watson IOT Platform

```
#pip install wiotp-sdk
```

```
import wiotp.sdk.device
```

```
import time
```

```
import random
```

```
myConfig = {
```

```
"identity": {
```

```
"orgId": "gsqz5f",
```

```
"typeId": "NANDY",
```

```
"deviceId": "12345" },
```

```
"auth": { "token": "9876543210" }
```

```
}
```

```
def myCommandCallback(cmd):
```

```

print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
m=cmd.data['command']
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()
while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()

```

The screenshot shows a Python script running in a shell window. The script is titled "publish.py" and is located at "C:/Python/Python37/publish.py (3.7.4)". The script imports the necessary modules and defines a configuration object for the IBM IoT Platform. It then enters a loop where it generates random temperature and humidity values, creates a data object, and publishes it to the platform. The output of the script is shown in the shell window, displaying a series of "Published data Successfully" messages with the generated temperature and humidity values.

```

import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "gaqz56",
        "typeId": "RABBY",
        "deviceId": "12345" },
    "auth": { "token": "9876543210" }
}
def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()
while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()

```

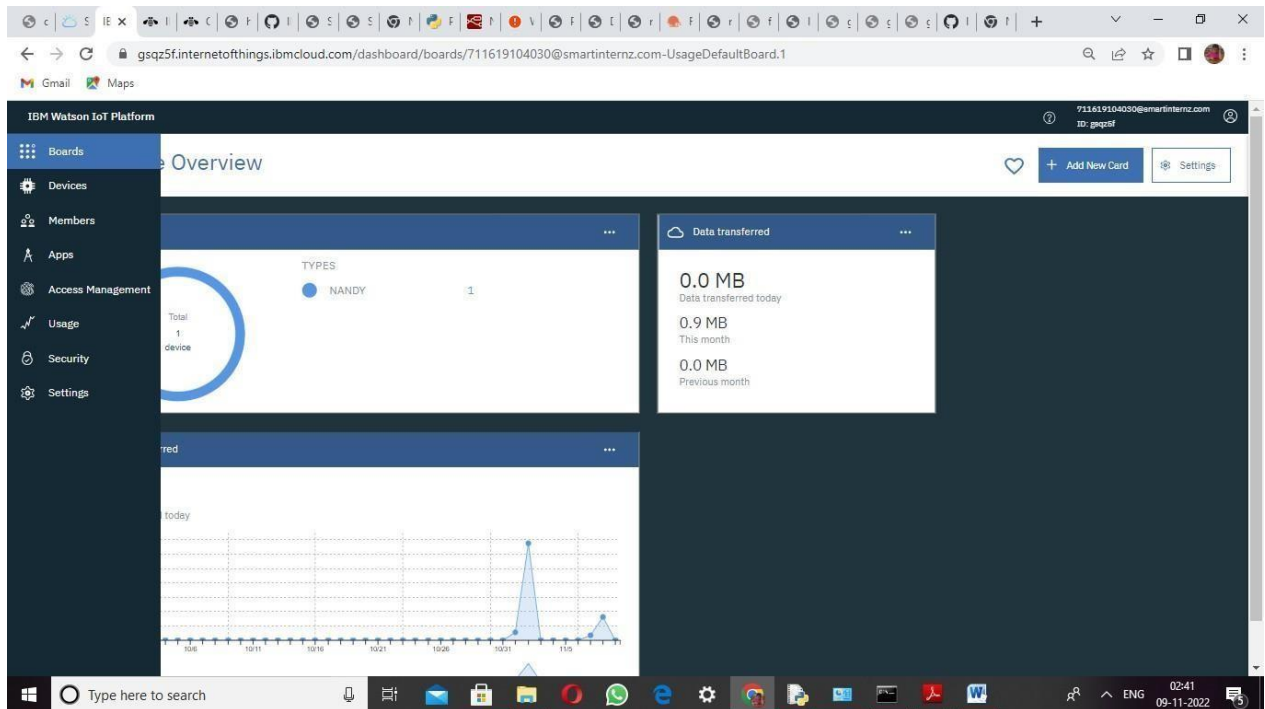
Python 3.7.4 Shell

```

Published data Successfully: %s ('temperature': 50, 'humidity': 45)
Published data Successfully: %s ('temperature': 8, 'humidity': 28)
Published data Successfully: %s ('temperature': 46, 'humidity': 1)
Published data Successfully: %s ('temperature': 78, 'humidity': 49)
Published data Successfully: %s ('temperature': 81, 'humidity': 41)
Published data Successfully: %s ('temperature': 73, 'humidity': 80)
Published data Successfully: %s ('temperature': 76, 'humidity': 34)
Published data Successfully: %s ('temperature': 2, 'humidity': 81)
Published data Successfully: %s ('temperature': 33, 'humidity': 32)
Published data Successfully: %s ('temperature': 18, 'humidity': 76)
Published data Successfully: %s ('temperature': 68, 'humidity': 79)
Published data Successfully: %s ('temperature': 85, 'humidity': 39)
Published data Successfully: %s ('temperature': -10, 'humidity': 96)
Published data Successfully: %s ('temperature': 112, 'humidity': 23)
Published data Successfully: %s ('temperature': 63, 'humidity': 52)
Published data Successfully: %s ('temperature': -18, 'humidity': 36)
Published data Successfully: %s ('temperature': 103, 'humidity': 70)
Published data Successfully: %s ('temperature': 38, 'humidity': 12)
Published data Successfully: %s ('temperature': 6, 'humidity': 51)
Published data Successfully: %s ('temperature': 0, 'humidity': 71)
Published data Successfully: %s ('temperature': 76, 'humidity': 24)
Published data Successfully: %s ('temperature': -16, 'humidity': 73)
Published data Successfully: %s ('temperature': -8, 'humidity': 35)
Published data Successfully: %s ('temperature': 60, 'humidity': 49)
Published data Successfully: %s ('temperature': 79, 'humidity': 97)
Published data Successfully: %s ('temperature': 93, 'humidity': 72)
Published data Successfully: %s ('temperature': -13, 'humidity': 72)
Published data Successfully: %s ('temperature': 67, 'humidity': 90)
Published data Successfully: %s ('temperature': 108, 'humidity': 83)
Published data Successfully: %s ('temperature': 71, 'humidity': 32)
Published data Successfully: %s ('temperature': 47, 'humidity': 75)
Published data Successfully: %s ('temperature': 65, 'humidity': 16)
Published data Successfully: %s ('temperature': 10, 'humidity': 83)
Published data Successfully: %s ('temperature': 24, 'humidity': 76)
Published data Successfully: %s ('temperature': 109, 'humidity': 31)
Published data Successfully: %s ('temperature': 15, 'humidity': 24)
Published data Successfully: %s ('temperature': 0, 'humidity': 34)
Published data Successfully: %s ('temperature': 44, 'humidity': 87)
Published data Successfully: %s ('temperature': 99, 'humidity': 94)

```

Publish the data to the ibm cloud:



The screenshot shows the IBM Watson IoT Platform dashboard. The left sidebar contains navigation links: Boards, Devices, Members, Apps, Access Management, Usage, Security, and Settings. The main area is titled 'Devices' and shows a list of devices. The table below shows the details of a device with ID 12345.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By	Device Class
12345	Connected	NANDY	Device	Nov 3, 2022 12:12 AM		711619104030@smarinternz.com	

Identity | Device Information | Recent Events | State | Logs

Device ID: 12345
Device Type: NANDY
Date Added: Nov 3, 2022 12:12 AM
Added By: 711619104030@smarinternz.com
Connection Status: Connected
Connection Time: Nov 9, 2022 2:40 AM
Client Address: 106.222.125.219 SecureToken

IBM Watson IoT Platform

711619104030@emartinternz.com
ID: gsqz5f

Browse Action Device Types Interfaces

Add Device

12345 Connected NANDY Device Nov 3, 2022 12:12 AM 711619104030@smartinternz.com

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
status	{"temperature":27,"humidity":72}	json	a few seconds ago
status	{"temperature":3,"humidity":56}	json	a few seconds ago
status	{"temperature":21,"humidity":54}	json	a few seconds ago
status	{"temperature":3,"humidity":28}	json	a few seconds ago
status	{"temperature":0,"humidity":85}	json	a few seconds ago

Items per page 50 | 1-1 of 1 item

1 of 1 page < 1 >

Type here to search

ENG

02:31 09-11-2022

gsqz5f.internetofthings.ibmcloud.com/dashboard/devices/browse

GmailMaps

IBM Watson IoT Platform

711619104030@smarterintmz.com
ID: gsqz5f

BrowseActionDevice TypesInterfaces

Add Device

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By	Device Class
12345	Connected	NANDY	Device	Nov 3, 2022 12:12 AM		711619104030@smarterintmz.com	

IdentityDevice InformationRecent EventsStateLogs

Showing Raw Data | No Interfaces Available

Property	Value	Type	Event	Last Received
temperature	-10	Number	status	a few seconds ago
humidity	10	Number	status	a few seconds ago

Items per page 50 | 1-1 of 1 item

1 of 1 page

Type here to search

02:45
09-11-2022