# IBM Watson IoT – T1 – MQTT Client

Commissioning task T1

In this commissioning task a normal computer with Windows operating system and a network connection to Internet will be needed. No special hardware or software will be needed.

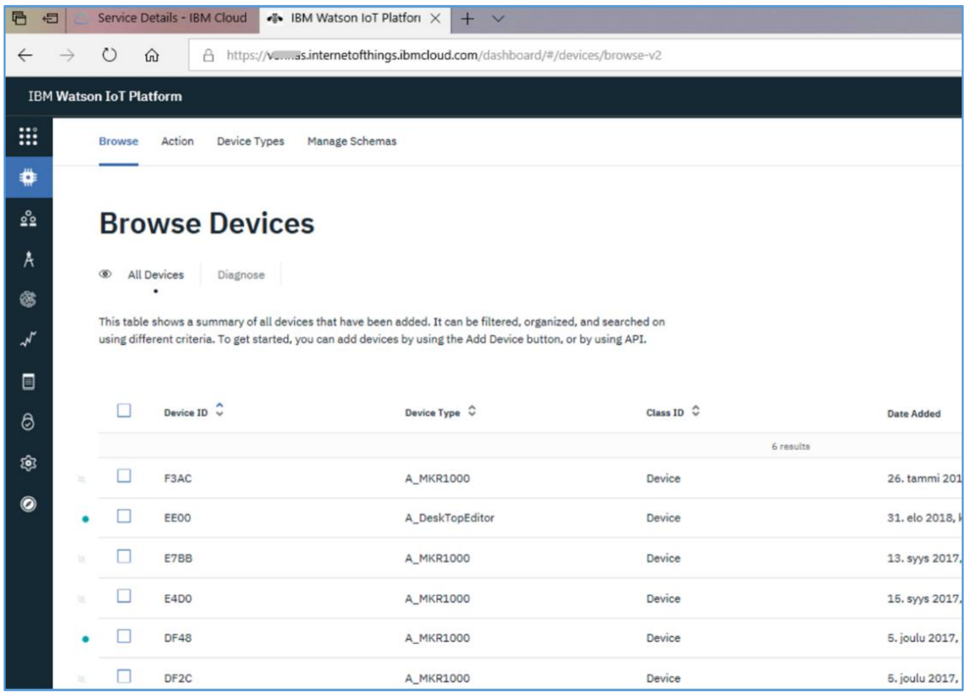
## 1. Environment for completing this exercise – IBM Bluemix Watson IoT

Please study the instruction sheet " IBM Watson - MKR1000 – Getting Started". According to these instructions you will be registering yourself to IBM Bluemix and you will be defining an "organization" and creating two IoT devices.

The instruction ask you to create Arduino MKR1000 –development card as your device. Anyway, this time we will be creating a "device", the role of which we can play with almost any MQTT client. MQTT client apps are available for Windows, Android and iOS.

Please create a "device" or two according to the instructions. The type can be for example "A_DeskTop_Editor". Please remember to save in safe place the just created Organization ID, Device Type, Device ID, Authentication method, Authentication token! IT IS EXTREMELY IMPORTANT TO SAVE ON A TEXT FILE THE AUTHENTICATION TOKEN ! You will not be able to see again the token after closing this page!

Fig 1.1 One desk top application and several real devices visible in IBM Watson IoT device list. Those marked with green dot are online at the moment.


Service Details - IBM Cloud
IBM Watson IoT Platforn
https://[redacted]s.internetofthings.ibmcloud.com/dashboard/#/devices/browse-v2
IBM Watson IoT Platform
Browse
Action
Device Types
Manage Schemas
All Devices
Diagnose

Browse Devices

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.

6 results

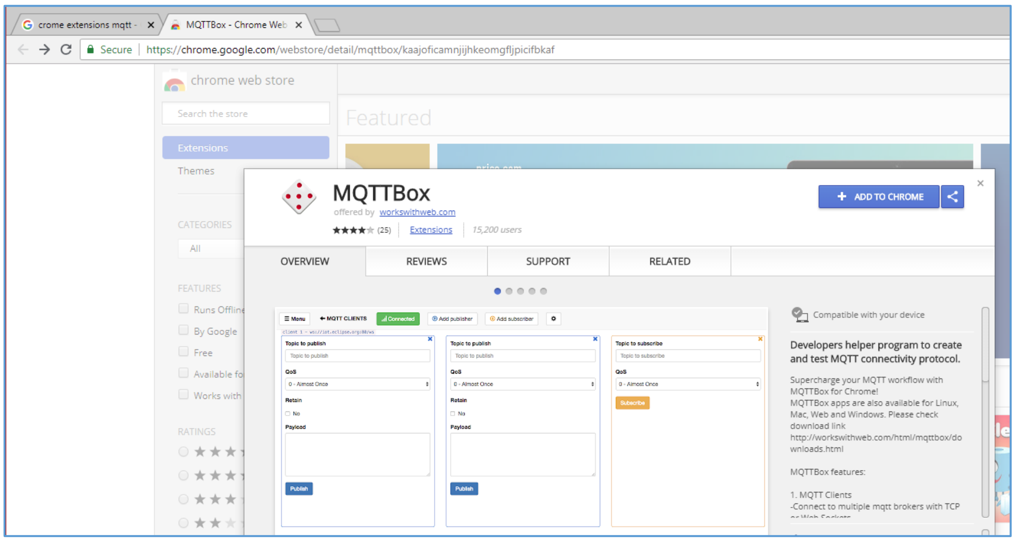
| | Device ID | Device Type | Class ID | Date Added |
|---|---|---|---|---|
| | F3AC | A_MKR1000 | Device | 26. tammi 201 |
| ● | EE00 | A_DeskTopEditor | Device | 31. elo 2018, k |
| | E7BB | A_MKR1000 | Device | 13. syys 2017, |
| | E4D0 | A_MKR1000 | Device | 15. syys 2017, |
| ● | DF48 | A_MKR1000 | Device | 5. joulu 2017, |
| | DF2C | A_MKR1000 | Device | 5. joulu 2017, |

2. Monitoring MQTT broker with browser extension, Windows app, Android App or iOS App.

## 2.1 Selecting an app

Web browsers have extensions for various purposes. Open your browser and search for “browser extension mqtt”. As a result you should get a link to for example MQTTBox Chrome Web Store. Follow the link for installation.

Fig 2.1.1 MQTTBox as browser extension / 2018



Developers helper program to create and test MQTT connectivity protocol.

Supercharge your MQTT workflow with MQTTBox for Chrome!
MQTTBox apps are also available for Linux, Mac, Web and Windows. Please check download link http://workswithweb.com/html/mqttbox/downloads.html

MQTTBox features:

1. MQTT Clients
-Connect to multiple mqtt brokers with TCP

An MQTT APP can be installed on Windows computer, too. In the Microsoft Store search for mqtt in department Apps.

Fig 2.1.2. Search results in Microsoft Store for mqtt / 2018

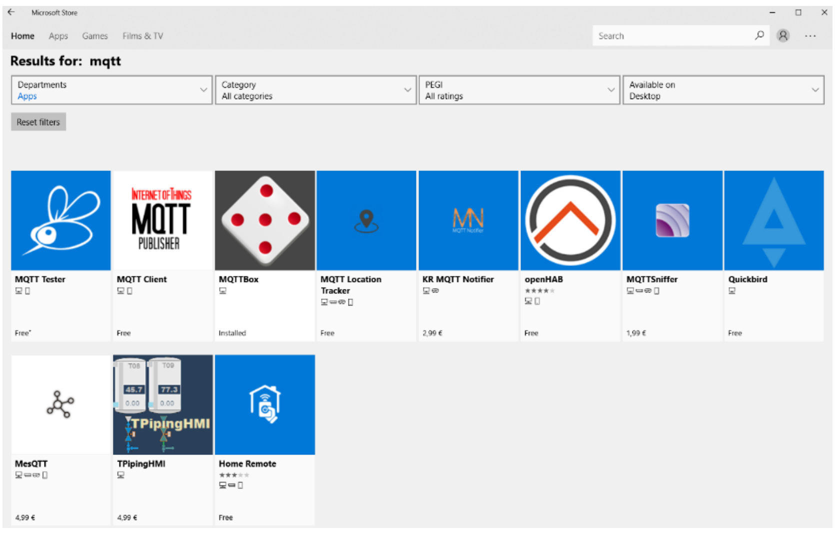
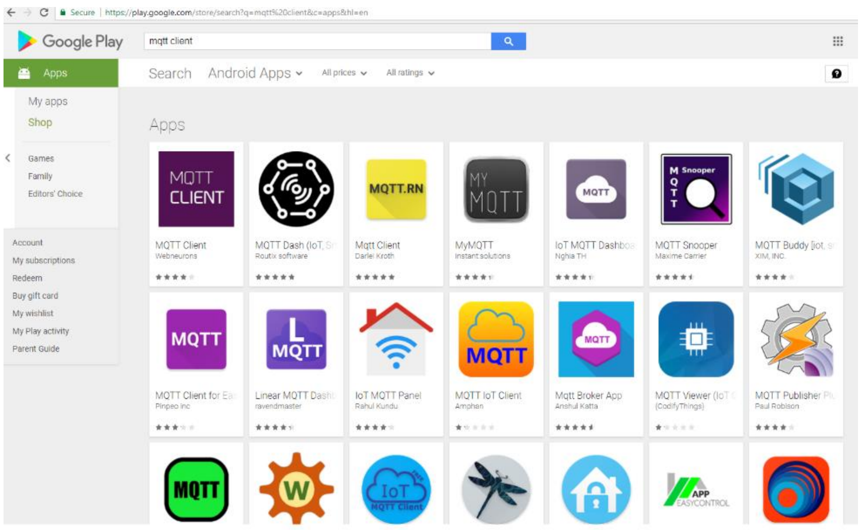


Fig 2.1.3 Android apps for MQTT client in Google Play / 2018



Install the MQTTbox either as browser extension or as a Windows APP. It seems to be easy to use.

The MQTTbox can be installed as well directly from an .exe file which is available at http://workswithweb.com/mqttbox.html

A number of apps are available for Android and iOS as well.

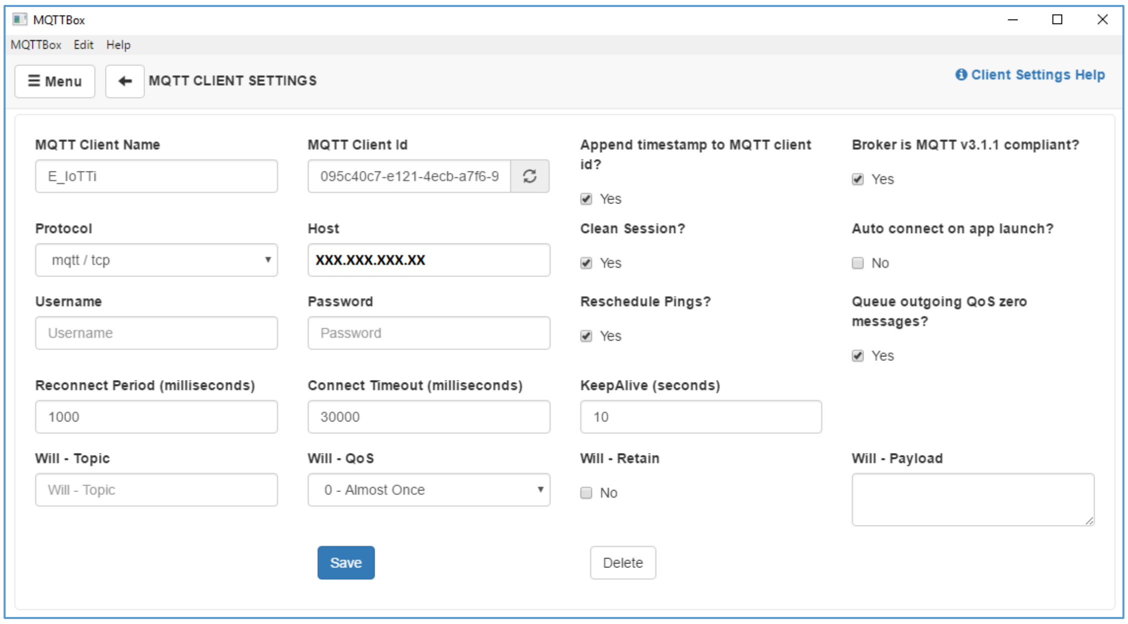
In opening the APP or extension you will be given a possibility to add a client. Please check out that in the app you selected you will be able to define in minimum:

- MQTT Client name
- MQTT Client Id
- Protocol
- Host
- Username
- Password
- Topic
- Quality of Service
- Payload Type
- Payload

Possibility on setting another port than standard 1883 would be a bonus!

## 2.2 Connecting to mosquitto mqtt broker

Fig. 2.2.1 MQTTBox settings for a new client.



| | | | |
|---|---|---|---|
| MQTT Client Name: E_IoTTi | MQTT Client Id: 095c40c7-e121-4ecb-a7f6-9 | Append timestamp to MQTT client id?: ✓ Yes | Broker is MQTT v3.1.1 compliant?: ✓ Yes |
| Protocol: mqtt / tcp | Host: XXX.XXX.XXX.XX | Clean Session?: ✓ Yes | Auto connect on app launch?: No |
| Username: Username | Password: Password | Reschedule Pings?: ✓ Yes | Queue outgoing QoS zero messages?: ✓ Yes |
| Reconnect Period (milliseconds): 1000 | Connect Timeout (milliseconds): 30000 | KeepAlive (seconds): 10 | |
| Will - Topic: Will - Topic | Will - QoS: 0 - Almost Once | Will - Retain: No | Will - Payload: |

For the settings fill in the text boxes as seen above.

In the **MQTT Client Name** you can create your own client name.

The **MQTT Client Id** is generated automatically. Selecting the **append time stamp to MQTT client id** helps avoiding to contact the server again with exactly the same client id.

In **Host** please type the IP address of the mqtt broker you are connecting to. HAMK offers for education and research a broker at iot.xxxxxxxx.hamk.fi ( your instructor may give you the address). The mosquito project offers a mqtt broker for test purposes, more information: https://mosquitto.org/ and https://test.mosquitto.org/ . Please use the mosquito broker if no other instructions were given.

In MQTTbox there is no option for port. The MQTT standard port 1883 will be used.

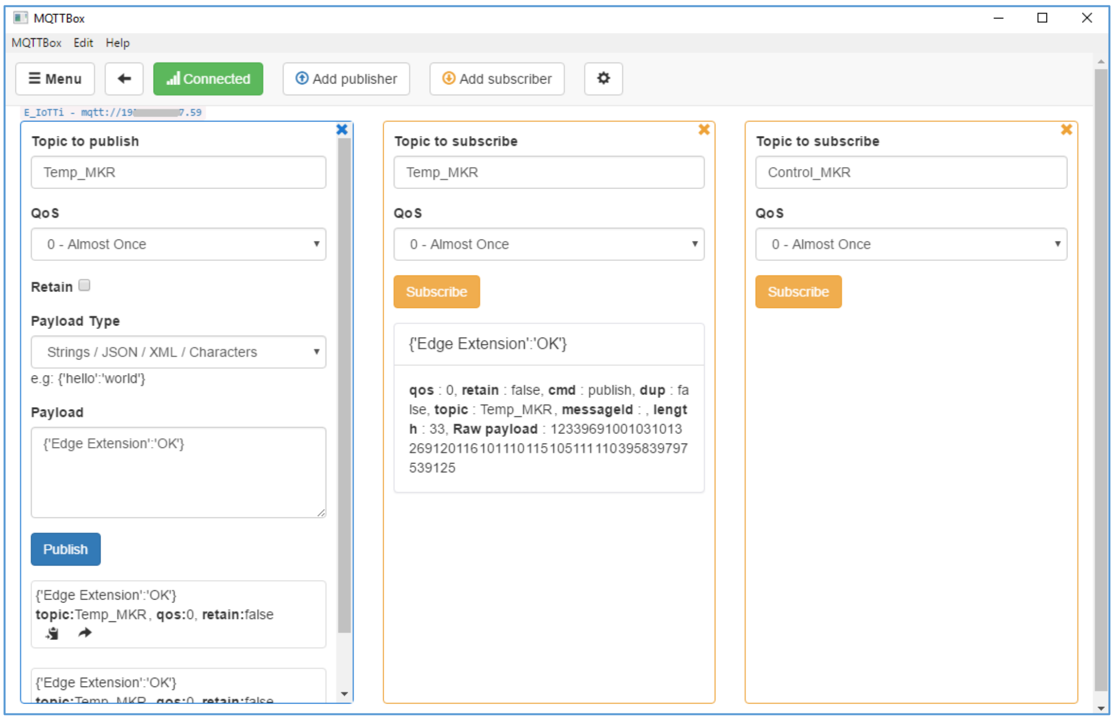
Select the Protocol mqtt/tcp.

User name and password can be left empty when there is no user authentication set in this MQTT broker.

For detailed information on the other settings please have a look at web page https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment

Once the connection has been established you can test it. To succeed with connecting you need to be in a network where communication to port 1883 is open in the firewall. In HAMK premises that is HAMKvisitor WLAN-network. Typically that port is open in home network or a network shared as WLAN from your own mobile phone.

Fig 2.2.2 Broker tested with Windows APP MQTTBox. In browser extension MQTTBox the user interface is exactly the same.



E_IoTTi - mqtt://19[redacted]7.59

Topic to publish: Temp_MKR
QoS: 0 - Almost Once
Retain
Payload Type: Strings / JSON / XML / Characters
e.g: {'hello':'world'}
Payload: {'Edge Extension':'OK'}

{'Edge Extension':'OK'}
**topic:**Temp_MKR, **qos:**0, **retain:**false

{'Edge Extension':'OK'}
**topic:**Temp_MKR, **qos:**0, **retain:**false

Topic to subscribe: Temp_MKR
QoS: 0 - Almost Once

{'Edge Extension':'OK'}

**qos** : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : Temp_MKR, **messageId** : , **length** : 33, **Raw payload** : 12339691001031013269120116101110115105111110395839797539125

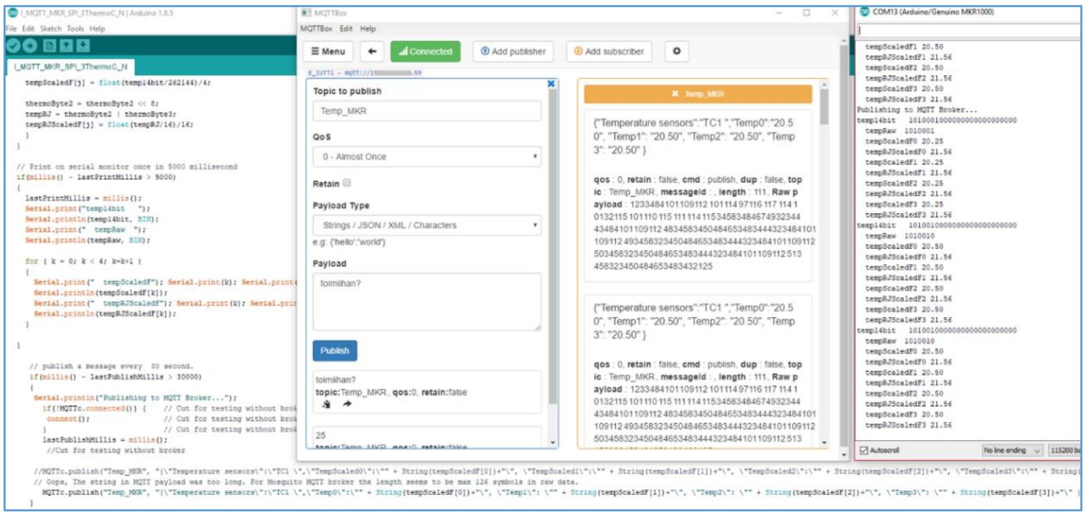
Topic to subscribe: Control_MKR
QoS: 0 - Almost Once

Broker is tested with sending a payload in JSON format into a new topic. A subscriber is created to listen to that topic. It seems to work all right as seen on the picture above.

You may as well test the operation with IoT hardware. This is not necessary if the hardware is not available. An example with Arduino microcontroller board is given.

Arduino MKR1000 is powered up next. It starts publishing the temperature data. In MQTTBox a new subscriber is created  to listen for topic Temp_MKR. The MKR1000 is publishing in JSON format under this topic. Seems to work all right.

Fig 2.2.3. Publishing several temperature sensor values in JSON format under topic “Temp_MKR” to MQTT broker and subscribing to the topic “Temp_MKR” in MQTT client.


I_MQTT_MKR_SPI_3ThermoC_N | Arduino 1.8.5
File Edit Sketch Tools Help
I_MQTT_MKR_SPI_3ThermoC_N
MQTTBox
MQTTBox Edit Help
Menu
Connected
Add publisher
Add subscriber
Publish
Temp_MKR
COM13 (Arduino/Genuino MKR1000)
Autoscroll
No line ending
115200 b

```
      tempScaledF[j] = float(temp14bit/262144)/4;

      thermoByte2 = thermoByte2 << 8;
      tempRJ = thermoByte2 | thermoByte3;
      tempRJScaledF[j] = float(tempRJ/16)/16;
      }
    }

  // Print on serial monitor once in 5000 millisecond
  if(millis() - lastPrintMillis > 5000)
  {
    lastPrintMillis = millis();
    Serial.print("temp14bit   ");
    Serial.println(temp14bit, BIN);
    Serial.print("  tempRaw  ");
    Serial.println(tempRaw, BIN);

    for ( k = 0; k < 4; k=k+1 )
    {
      Serial.print("  tempScaledF"); Serial.print(k); Serial.print(
      Serial.println(tempScaledF[k]);
      Serial.print("  tempRJScaledF"); Serial.print(k); Serial.prin
      Serial.println(tempRJScaledF[k]);
    }

  }

   // publish a message every  30 second.
   if(millis() - lastPublishMillis > 30000)
   {
    Serial.println("Publishing to MQTT Broker...");
      if(!MQTTc.connected()) {    // Cut for testing without brok
       connect();                 // Cut for testing without brok
      }                           // Cut for testing without brok
      lastPublishMillis = millis();
       //Cut for testing without broker

    //MQTTc.publish("Temp_MKR", "{\"Temperature sensors\":\"TC1 \",\"TempScaled0\":\"" + String(tempScaledF[0])+"\", \"TempScaled1\":\"" + String(tempScaledF[1])+"\", \"TempScaled2\":\"" + String(tempScaledF[2])+"\", \"TempScaled3\":\"" + String(
    // Oops, The string in MQTT payload was too long. For Mosquito MQTT broker the length seems to be max 126 symbols in raw data.
      MQTTc.publish("Temp_MKR", "{\"Temperature sensors\":\"TC1 \",\"Temp0\":\"" + String(tempScaledF[0])+"\", \"Temp1\": \"" + String(tempScaledF[1])+"\", \"Temp2\": \"" + String(tempScaledF[2])+"\", \"Temp3\": \"" + String(tempScaledF[3])+"\" }
   }
```

E_IoTTi - mqtt://1[redacted].59

Topic to publish: Temp_MKR
QoS: 0 - Almost Once
Retain
Payload Type: Strings / JSON / XML / Characters
e.g: {'hello':'world'}
Payload: toimiihan?

toimiihan?
**topic:**Temp_MKR, **qos:**0, **retain:**false

25
**topic:**Temp_MKR, **qos:**0, **retain:**false

{"Temperature sensors":"TC1 ","Temp0":"20.50", "Temp1": "20.50", "Temp2": "20.50", "Temp3": "20.50" }

**qos** : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : Temp_MKR, **messageId** : , **length** : 111, **Raw payload** : 1233484101109112101114971161171141013211510111011511111411534583484674932344434841011091124834583450484653483444323484101109112493458323450484653483444323484101109112503458323450484653483444323484101109112513458323450484653483432125

{"Temperature sensors":"TC1 ","Temp0":"20.50", "Temp1": "20.50", "Temp2": "20.50", "Temp3": "20.50" }

**qos** : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : Temp_MKR, **messageId** : , **length** : 111, **Raw payload** : 1233484101109112101114971161171141013211510111011511111411534583484674932344434841011091124834583450484653483444323484101109112493458323450484653483444323484101109112503458323450484653483444323484101109112513

```
  tempScaledF1 20.50
  tempRJScaledF1 21.56
  tempScaledF2 20.50
  tempRJScaledF2 21.56
  tempScaledF3 20.50
  tempRJScaledF3 21.56
Publishing to MQTT Broker...
temp14bit   1010001000000000000000000
  tempRaw  1010001
  tempScaledF0 20.25
  tempRJScaledF0 21.56
  tempScaledF1 20.25
  tempRJScaledF1 21.56
  tempScaledF2 20.25
  tempRJScaledF2 21.56
  tempScaledF3 20.25
  tempRJScaledF3 21.56
temp14bit   1010010000000000000000000
  tempRaw  1010010
  tempScaledF0 20.50
  tempRJScaledF0 21.56
  tempScaledF1 20.50
  tempRJScaledF1 21.56
  tempScaledF2 20.50
  tempRJScaledF2 21.56
  tempScaledF3 20.50
  tempRJScaledF3 21.56
temp14bit   1010010000000000000000000
  tempRaw  1010010
  tempScaledF0 20.50
  tempRJScaledF0 21.56
  tempScaledF1 20.50
  tempRJScaledF1 21.56
  tempScaledF2 20.50
  tempRJScaledF2 21.56
  tempScaledF3 20.50
  tempRJScaledF3 21.56
```

The maximum length of the MQTT payload seems to be 126 symbols in the raw data. Othervise the message seems not to be available for subscription. In your string it is rather difficult to count the exact number of characters. Although in MQTT the payload maximum size should be 256 mb!

JSON format is not necessary. Plain text or integer or float values can be used as well. The message payload will be sent as character string.

**TASK A.**

Please study from MQTT documentation what is said about MQTT Topic and topic hierarchy.

Please define for yourself a number of topics with some kind of hierarchy. Just write down your topics in a text sheet.

Publish messages to MQTT broker using the lowest level of your topic hierarchy.

Subscribe on MQTT broker on your topics. Try with different hierarchies. Try what are the wildcards # and + .

**In the report,** write some lines about what you learned about using the + and # ! Attach in your report a screen shot image on successful testing.

3. Connecting to IBM Cloud Watson IoT platform

In your MQTT client try to establish the client with parameters corresponding your "device" you defined in IBM Watson IoT.

Instruction are given here for the MQTTBox application. But the same instructions can be used for configuring any other mqtt client as well.

Fill in the authentication information you got when creating your organization ID and device:

| | |
|---|---|
| Server: | <OrganizationID>.messaging.internetofthings.ibmcloud.com |
| Port: | 1883 |
| ClientID: | d:<OrganizationID>:<deviceType>:<DeviceID> |
| Username: | use-token-auth |
| Password: | <AuthenticationToken> |
| Clean Session: | OFF or ON |
| SSL: | OFF ( and in Watson IoT please remember to select Security Settings, TSL Optional ) |

The message topic needs to be in certain format:

iot-2/evt/<topic>/fmt/json

The <topic> can be replaced with any text but please avoid special symbols like scandinavain ö, ä and space.

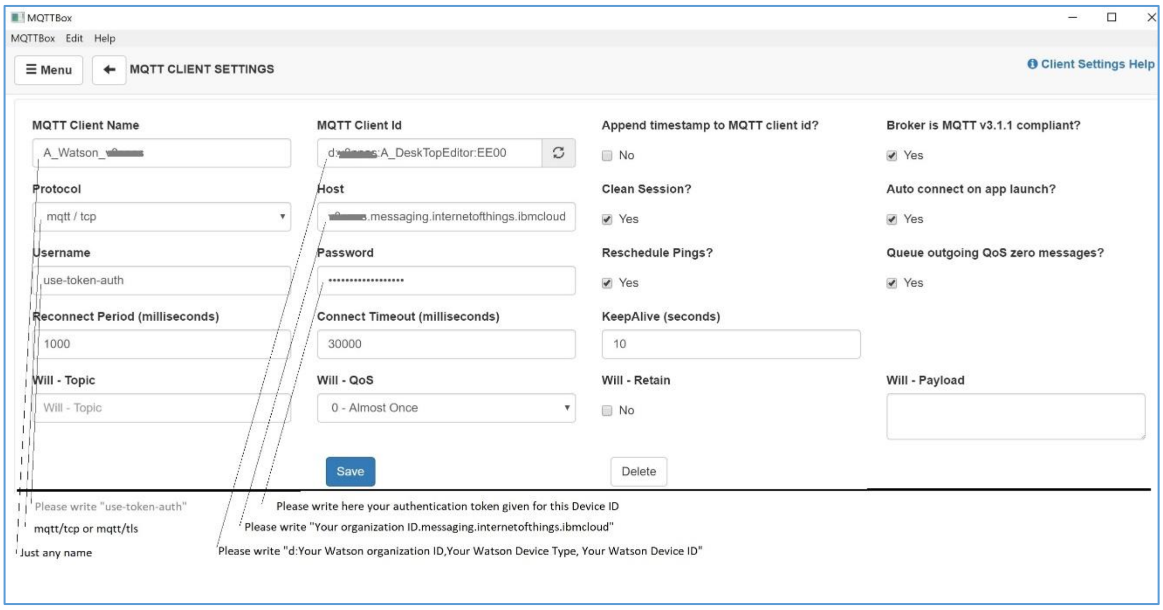
In the above in the topic it is defined that the message payload needs to be in JSON format. For example:

{ "name":"Pekka", "age":41, "city":"Oulu" }

## MQTTBox application used for connecting the Watson IoT

An Application for MQTT messaging can be used for connecting to IBM Watson IoT platform. The client settings and topic settings just need to be created just in the correct format.

Fig 3.2.1 Settings on the MQTT Client window on MQTTBox application. In addition to notes on the picture please note that “No” is selected for the Append time stamp to MQTT client ID!



| | | | |
|---|---|---|---|
| MQTT Client Name | MQTT Client Id | Append timestamp to MQTT client id? | Broker is MQTT v3.1.1 compliant? |
| A_Watson_[redacted] | d:[redacted]:A_DeskTopEditor:EE00 | No | ✓ Yes |
| Protocol | Host | Clean Session? | Auto connect on app launch? |
| mqtt / tcp | [redacted].messaging.internetofthings.ibmcloud | ✓ Yes | ✓ Yes |
| Username | Password | Reschedule Pings? | Queue outgoing QoS zero messages? |
| use-token-auth | •••••••••••••••••• | ✓ Yes | ✓ Yes |
| Reconnect Period (milliseconds) | Connect Timeout (milliseconds) | KeepAlive (seconds) | |
| 1000 | 30000 | 10 | |
| Will - Topic | Will - QoS | Will - Retain | Will - Payload |
| Will - Topic | 0 - Almost Once | No | |

Please write "use-token-auth"

mqtt/tcp or mqtt/tls

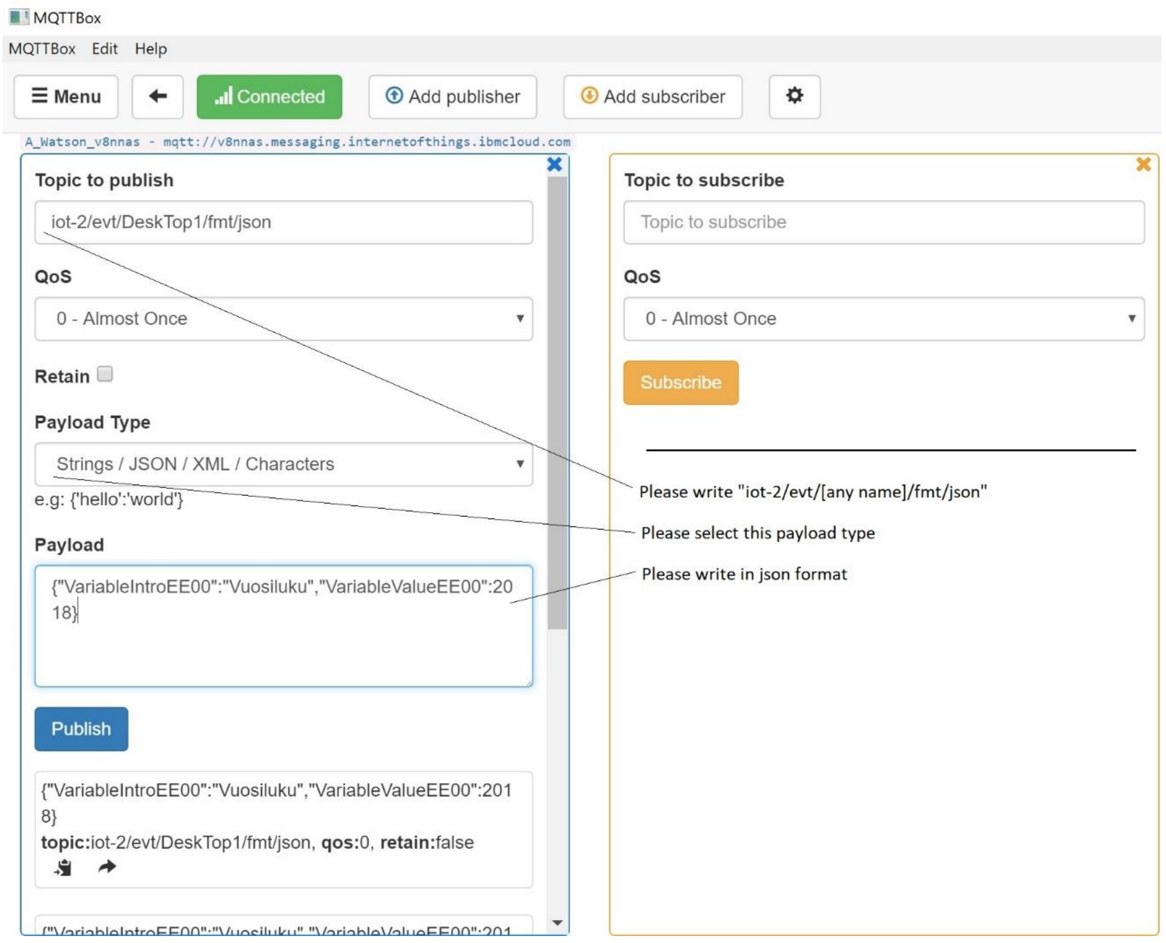
Just any name

Please write here your authentication token given for this Device ID

Please write "Your organization ID.messaging.internetofthings.ibmcloud"

Please write "d:Your Watson organization ID,Your Watson Device Type, Your Watson Device ID"

You can remove the setting Auto connect on app launch. This way you will be able to look at the topic-payload-window without at the same time connecting to the broker.


MQTTBox
MQTTBox Edit Help
Menu
Connected
Add publisher
Add subscriber
A_Watson_v8nnas - mqtt://v8nnas.messaging.internetofthings.ibmcloud.com
Topic to subscribe
QoS
0 - Almost Once
Subscribe
Please write "iot-2/evt/[any name]/fmt/json"
Please select this payload type
Please write in json format

Topic to publish: iot-2/evt/DeskTop1/fmt/json

QoS: 0 - Almost Once

Retain

Payload Type: Strings / JSON / XML / Characters

e.g: {'hello':'world'}

Payload:

```
{"VariableIntroEE00":"Vuosiluku","VariableValueEE00":2018}
```

Publish

```
{"VariableIntroEE00":"Vuosiluku","VariableValueEE00":2018}
topic:iot-2/evt/DeskTop1/fmt/json, qos:0, retain:false
```

Fig 3.2.2 Definition of topic for publishing in Topic window on MQTTBox application

**Please note than in the IBM Cloud Watson ( and in another commercial IoT Platfrom ) it is not possible to subscribe an event. The events are “private” messages from the device to the IoT Platform. With your device credentials you could subcribe a command if your IoT Platform would publish a command for your device.**

**TASK B**

Try out publishing data into the IBM Watson IoT. Can you publish a topic and a JSON formatted value? Can you publish text strings inside the JSON formatted payload? Can you publish number values inside the JSON formatted payload?

In the MQTTBox or almost any other mqtt client you can try to publish the following payload as an event send by a sensor device:

{"d":{"tempCount":156,"temp4position":"Lab1","temp4value":23,"temp5position":"Cellar2","temp5value":18}}

It is a JSON formatted payload.

Try to publish this or something similar to the IBM Watson!

If you work with microcontroller applications you can try to publish the same content from a microcontroller. In C++ for Mbed OS operating system microcontroller this would look like:

```
sprint(buffer,
“{\"d\":{\"tempCount\":%d,\"temp4position\":\"Lab1\",\"temp4value\":23,\"temp5position\":\
"Cellar2\",\"temp5value\":%d}}”,temp4ADC,temp5ADC);
```

This line would create a string type variable buffer. This string would be sent as the payload.

In the IBM Cloud IoT Platform open your Device. Look for the Recent event, State and Logs.

You will get a JSON object. Open the object. You will see the Property – value pairs you wrote in the payload.

In the IBM Cloud Watson IoT the JSON format {“d”:{“hour”:7}} would open as an object correctly. But {“d”:{“hour”:07}} would print in Watson IoT completely something else….try it! What did you get? …. You have got some ASCII symbols. Every character in the payload text is given with the number from the ascii symbol table.

**In the report,** write some lines about what you learned!