Intel® IoT Gateways: Publishing Data to an MQTT Broker Using Python

Getting Started Guide

November 2015

Order No.: 333067-001

Revision History

Date	Revision	Description
November 2015	001	Initial public release

Contents

1.0 Introduction

Note: For an online version of this document, see https://software.intel.com/en-us/SetupGateway-MQTT.

Applicable to These Intel® IoT Gateway Products

- Intel® IoT Gateways based on Intel® Atom™ processors.
- Intel® IoT Gateways based on Intel® Core™ processors.
- Intel® IoT Gateways based on Intel® Quark™ processors.

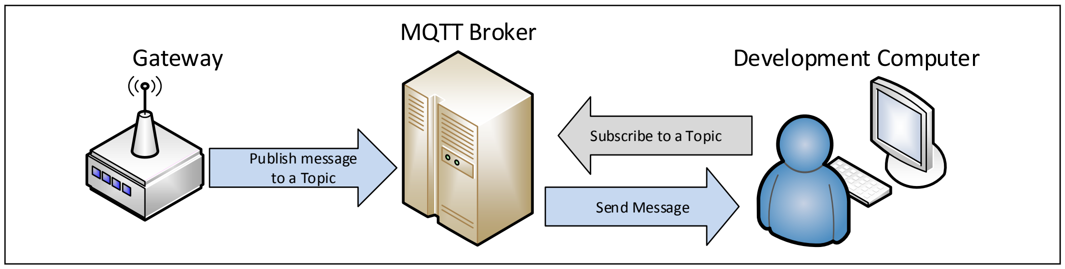
How These Instructions Help You

These instructions will guide you through the steps to publish data from an Intel® IoT Gateway to an MQTT broker (server) using a Python* Script. On a Development Computer you will **subscribe** to an MQTT message test topic on an independent MQTT message broker (server). After subscribing, any messages **published** to the topic on the remote MQTT broker are sent to the Development Computer.

You will do this by:

1. Installing Python MQTT software on a Development Computer.
2. Setting up an MQTT broker.
3. Installing Python MQTT software on your Gateway.
4. Creating and running a Python script to send MQTT messages from the Gateway, through the broker, to the Development Computer.

The figure below shows the connections between the Gateway, the MQTT broker and the Development Computer.



What is MQTT?

MQTT is a messaging protocol that can be used to transmit data from a Gateway to a local server or cloud server. The server relays the data to remote clients that are subscribed to receive it.

From http://en.wikipedia.org/wiki/MQTT:

"MQTT (formerly MQ Telemetry Transport) is a publish-subscribe based light weight messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a small code footprint is required and/or network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message.

There are several MQTT brokers available. They vary in their feature set and some of them implement additional features on top of the standard MQTT functionality. For a comparison of several brokers, go here: https://github.com/mqtt/mqtt.github.io/wiki/server-support"

What is Mosquitto?

Mosquitto is an open source (BSD licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1. This guide gives examples of using Mosquitto to setup an MQTT message broker on a local network. See http://mosquitto.org for details.

Python and MQTT

The Python programming language can be used to generate MQTT messages. You might choose to use Python if it is simple to incorporate the Python code into your primary gateway application, and if you are familiar with Python. Python is sometimes convenient to use because it does not need to be compiled; it is interpreted on the Gateway at run time.

A Python interpreter is installed by default on an Intel® IoT Gateway running Wind River® Linux, therefore the Python scripts will run directly on the Gateway. Python interpreter version 2.7.2 is included with Wind River Linux 5 and Intelligent Device Platform XT 2. Python interpreter versions 2.7.3 and 3.3.3 are included with Wind River® Linux 7 and Intelligent Device Platform XT 3.1.

For more information on Python, see https://www.python.org/.

What is Paho?

From http://www.eclipse.org/paho/clients/python/:

"The Paho Python Client provides a client class with support for both MQTT v3.1 and v3.1.1 on Python 2.7 or 3.x. It also provides some helper functions to make publishing one-off messages to an MQTT server very straightforward."

2.0 Development Computer Setup

In this section, you will set up your Development Computer to subscribe to an MQTT message test topic on an Internet MQTT message broker. After your Development Computer has subscribed to a topic, messages published to the topic are sent to the Development Computer.

To setup your Development Computer you will:

1. Install the Paho Python MQTT software.
2. Test the Paho Python MQTT software by subscribing to an Internet MQTT broker.

Perform the steps in this section on your Development Computer.

Prerequisites

- A Development Computer running Ubuntu Linux version 12.04 or above.
- You have an Internet connection from your Development Computer
- Your Development Computer has a network connection to the MQTT Broker.
  - The MQTT broker could be on an internal network or on the Internet.
  - The network firewall must allow messages to and from the Development Computer on ports 1883, 8883, 8884, and 8885.

Install the Paho Python MQTT Software

Perform these steps on your Development Computer.

1. Download the latest Paho MQTT `tar.gz` file from http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt.python.git/. The latest file is at the top of the list under the heading **Download**. As of October 20, 2015, this file is called `org.eclipse.paho.mqtt.python-1.1.tar.gz`.
2. Put the file in a directory of your choice.
3. Extract the file. For example:

```
tar -xf org.eclipse.paho.mqtt.python-1.1.tar.gz
```

4. The contents extract to a directory with a name similar to the `tar.gz` file name. Change to this directory, replacing the directory name if necessary. For example:

```
cd org.eclipse.paho.mqtt.python-1.1
```

5. Run the Python setup and install command:

```
python setup.py install
```

The software is now installed on your Development Computer.

Test the Paho Python MQTT Commands

Test your Development Computer's Paho Python scripts, and its connection to an MQTT broker by subscribing to an active topic.

For example, if you are using an Internet broker, you can use the test.mosquitto.org test site to check proper network connection and broker subscription as follows:

1. Go to the Paho MQTT Python examples directory:

```
cd org.eclipse.paho.mqtt.python/examples
```

2. Modify the `sub.py` script to change the `mqttc.connect` and `mqttc.subscribe` lines:

```
mqttc.connect("test.mosquitto.org", 1883, 60)
mqttc.subscribe("#", 0)
```

   In this script, the following apply:

   - `mqttc.connect`
     - `test.mosquitto.org`: Name of MQTT broker you will connect to.
     - `1883`: Port used for the MQTT messages.
     - `60`: Timeout in seconds.
   - `mqttc.subscribe`
     - `#`: Wildcard character that indicates you are subscribing to all topics on the broker.
     - `0`: Quality of Service. For information about Quality of Service options, see https://www.eclipse.org/paho/files/mqttdoc/Cclient/qos.html.

3. Execute the script:

```
python sub.py
```

   If the network connections and the Python scripts are working correctly, then the `test.mosquitto.org` broker will send multiple lines of data. The content of the data is not relevant. If you do not receive data lines, double-check the network connections and the lines you edited in the script.

4. Press `Ctrl-C` to stop the script.

3.0 MQTT Broker Setup and Subscription

The previous section guided you through installing and testing the Paho Python MQTT software. The next step is to make sure your MQTT broker is set up, and ready to receive and send messages. To complete the examples in this guide, choose one of the following options for the location of your MQTT broker:

- **Existing broker on either a local network or on the Internet (Cloud):** You will use an existing broker on your network or on the Internet (Cloud). This broker is ready to send and receive messages and you have the ability to add a topic to this broker if a suitable topic is not already available.
- **New broker on a local network using Paho Python:** You will establish an MQTT broker on your Development Computer using Paho Python libraries. The Development Computer connects to the Gateway through a local network.
- **New broker on a local network using mosquitto:** You will establish an MQTT broker on your Development Computer using mosquitto. The Development Computer connects to the Gateway through a local network.

Use an Existing Broker on a Local Network

Ideally, you should have an MQTT message broker setup for your application. This broker should be on your desired network, configurable, and ready to send and receive messages for your application.

If you do not have an existing MQTT broker, you have a few options to use a simple MQTT broker for testing. These are covered below. This guide will not cover server or network setup.

If you have an existing MQTT broker, continue to Gateway Setup.

Use an Existing MQTT Broker on the Internet Using Paho Python

Several MQTT brokers are available on the Internet for use as test brokers. This guide uses the one at http://test.mosquitto.org.

Follow the steps in this section to use the Paho Python libraries that you installed earlier to setup a test topic called `mytopic` at http://test.mosquitto.org for MQTT message subscription and publication.

Perform these steps on your Development Computer.

1. Connect the Development Computer to the Internet.
2. Open a new Linux terminal window.
3. Go to the Paho examples directory:

```
cd org.eclipse.paho.mqtt.python-1.1/examples
```

4. Modify the `sub.py` script to change the `mqttc.connect` and `mqttc.subscribe` lines:

```
mqttc.connect("test.mosquito.org",1883, 60)
mqttc.subscribe("mytopic", 0)
```

In this script, the following apply:

- `mqttc.connect`
  - `test.mosquitto.org`: The Internet broker URL.
  - `1883`: Network port used for MQTT messages.
  - `60`: Timeout in seconds.
- `mqttc.subscribe`
  - `mytopic`: Name of your new MQTT topic. The topic is established automatically on the broker if it does not already exist.
  - `0`: Quality of Service.

5. Run the `sub.py` script:

```
python sub.py
```

Subscribing to a new topic called `mytopic` on the test.mosquitto.org broker automatically establishes that topic. Now, when the Gateway (or any other computer) publishes an MQTT message to `mytopic` on the test.mosquitto.org broker, the message will be sent to the Linux terminal where the `sub.py` script is running.

Continue to Gateway Setup.

Set up an MQTT Broker on a Local Network Using Paho Python

Use the steps in this section if your goal is to use your Development Computer as a local network test broker using the Paho Python libraries installed earlier.

Perform these steps on your Development Computer.

1. Connect the Development Computer to the same network as your Gateway.
2. Open a new Linux terminal window.
3. Go to the Paho examples directory:

```
cd org.eclipse.paho.mqtt.python-1.1/examples
```

4. Modify the `sub.py` script to change the `mqttc.connect` and `mqttc.subscribe` lines:

```
mqttc.connect("localhost",1883, 60)
mqttc.subscribe("mytopic", 0)
```

In this script, the following apply:

- `mqttc.connect`
  - `localhost`: An alias to the development computer's local loopback IP address (typically 127.0.0.1).
  - `1883`: Network port used for MQTT messages.

    — `60`: Timeout in seconds.
- `mqttc.subscribe`
    — `mytopic`: Name of the MQTT topic. The topic is established automatically on the broker if it does not already exist.
    — `0`: Quality of Service.

5. Run the `sub.py` script:

```
python sub.py
```

Subscribing to a new topic called `mytopic` on the local host automatically starts the local broker and establishes that topic for other subscribers. Now, when the Gateway publishes an MQTT message to `mytopic` at the Development Computer's IP address, the message will be sent to the Linux terminal where the `sub.py` script is running.

Continue to Gateway Setup.

Set up an MQTT Broker on a Local Network Using Mosquitto

Use the steps in this section if your goal is to use your Development Computer as a local network test broker using the open source application called mosquitto.

Note: For details about mosquitto use and syntax, see http://mosquitto.org, or use `<mosquitto_command> --help`, where `<mosquitto_command>` is the command that you need help with.

To start a broker process on the Development Computer:

1. Open a new Linux terminal window.

   The broker will run in this window. You can still receive MQTT messages on this Development Computer. The MQTT messages will be received in a different terminal window.
2. Install mosquitto if you do not already have it installed.

```
sudo apt-get install mosquitto
```

3. Create a topic and subscribe to it with the `mosquitto_sub` command.

```
mosquitto_sub -d -h localhost -t mytopic
```

   In this script, the following apply:
   - `-d`: Enable debug messages.
   - `-h localhost`: An alias to the development computer's local loopback IP address (typically 127.0.0.1).
   - `-t mytopic`: Name of the MQTT topic. The topic is established automatically on the broker if it does not already exist.
4. Check that the Development Computer is connected to the same network as the Gateway.

Subscribing to a new topic called `mytopic` on the local host automatically starts the local broker and establishes that topic for other subscribers. Now, when the Gateway publishes an MQTT message to `mytopic` at the Development Computer's IP address, the message will be sent to the Linux terminal where the `mosquitto_sub` command is running.

Continue to Gateway Setup.

4.0 Gateway Setup

To setup your Gateway, you will:

1. Install the Paho Python MQTT software on the Gateway.
2. Connect your Gateway to a test MQTT broker to confirm your connections and the functionality of a test script.

Prerequisites

- This example assumes the Gateway is running Wind River® Linux 7 with Wind River® Intelligent Device Platform XT 3.1. The procedures in this chapter may be different if the Gateway is running a different operating system.
- Your Gateway operating system must be created with the inclusion of `--with-template=feature/mqtt` in the configure statement. See https://software.intel.com/en-us/SetupGateway-hardware.
- Your Gateway is on the same network as your MQTT broker if you are using an internal network.
- Your Gateway is connected to the Internet if you are using an MQTT broker on the Internet. Your Internet firewall must allow traffic on ports 1883, 8883, 8884, and 8885.

Install the Paho Python MQTT Software

Perform these steps directly on your Gateway.

1. Download the latest Paho MQTT `tar.gz` file from http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt.python.git/. The latest file is at the top of the list under the heading **Download**. As of October 20, 2015, this file is called `org.eclipse.paho.mqtt.python-1.1.tar.gz`.
2. Put the file in a directory of your choice.
3. Extract the file. For example:

```
tar -xf org.eclipse.paho.mqtt.python-1.1.tar.gz
```

4. The contents extract to a directory with a name similar to the `tar.gz` file name. Change to this directory, replacing the directory name if necessary. For example:

```
cd org.eclipse.paho.mqtt.python-1.1
```

5. Run the Python setup and install command:

```
python setup.py install
```

The software is now installed on your Gateway.

Test the Connection to the MQTT Broker (Internet Broker Example)

Test your Gateway's Paho Python scripts, and its connection to an MQTT broker by subscribing to an active topic.

For example, if you are using an Internet broker, you can use the test.mosquitto.org test site to check proper network connection and broker subscription as follows:

1. Go to the Paho MQTT Python examples directory:

```
cd org.eclipse.paho.mqtt.python/examples
```

2. Modify the `sub.py` script to change the `mqttc.connect` and `mqttc.subscribe` lines:

```
mqttc.connect("test.mosquitto.org", 1883, 60)
mqttc.subscribe("#", 0)
```

   In this script, the following apply:

   - `mqttc.connect`
     - `test.mosquitto.org`: Name of MQTT broker you will connect to.
     - `1883`: Port used for the MQTT messages.
     - `60`: Timeout in seconds.
   - `mqttc.subscribe`
     - `#`: Wildcard character that indicates you are subscribing to all topics on the broker.
     - `0`: Quality of Service. For information about Quality of Service options, see https://www.eclipse.org/paho/files/mqttdoc/Cclient/qos.html.

3. Execute the script:

```
python sub.py
```

   If the network connections and the Python scripts are working correctly, then the `test.mosquitto.org` broker will send multiple lines of data. The content of the data is not relevant. If you do not receive data lines, double-check the network connections and the lines you edited in the script.

4. Press `Ctrl-C` to stop the script.

5.0 Sending and Receiving MQTT Messages

With your Development Computer and Gateway configured for MQTT and your MQTT broker set up, you are ready to create and run a Python script to publish MQTT messages.

These steps use Paho Python examples to create an initial script that you can later modify to meet your needs. Create the Python script either directly on your Gateway or on your Development Computer. If you create the script on your Development Computer, copy it to your Gateway to run it.

Security Considerations

Be aware of the following when creating and running your Python scripts:

- If you use the Python interpreter, IMA security is not an issue. IMA security allows the interpreter to run new scripts.
- If you do not use the Python interpreter and you have IMA security enabled, then you must sign the script before you can run it.
- If you have McAfee Embedded Control enabled on your Gateway, you must add your script to your Whitelist.

Note: This guide assumes that neither IMA nor Whitelisting is enabled on your Gateway.

Create and Run a Python MQTT Publish Script

1. Go to the Paho `examples` directory:

```
cd org.eclipse.paho.mqtt.python-1.1/examples
```

2. Copy and rename the `pub-single.py` script:

```
cp pub-single.py my-pub.py
```

3. Edit `my-pub.py` to change the last line so it points to the MQTT broker and the topic that you set up in Development Computer Setup:

```
publish.single("mytopic","Hello", hostname="<broker_location>")
```

   In this script, the following apply:

   - "`mytopic`": Topic name on the broker.
   - "`Hello`": Message content.
   - "`<broker_location>`": MQTT broker location. If you are using your Development Computer as your MQTT broker, then you should use the local network IP address of your Development Computer. For example `192.168.1.5`. If you are using an Internet MQTT broker, then you can use the IP address or the server name of the broker. For example: `test.mosquitto.org`.

4. If you edited the `my-pub.py` script on your Development Computer, copy it to your Gateway.
5. Run the `my-pub.py` Python script on the Gateway to publish the message from the Gateway to the `mytopic` topic on your MQTT broker:

```
python my-pub.py
```

Receive MQTT Messages on Your Development Computer

When your Gateway publishes a message to a topic on an MQTT broker, the broker sends that message to all computers that are subscribed to the topic on that MQTT broker.

You just issued the command `python my-pub.py` on your Gateway. This resulted in a "Hello" message being sent to your MQTT broker and from there to your Development Computer. This should now be displayed on your Development Computer as:

```
mytopic 0 Hello
```

In this message, the following apply:

- `mytopic`: Topic name.
- `0`: Quality of Service.
- `Hello`: The message.

6.0 Conclusion

Congratulations! Using the Python programming language, you sent an MQTT message (`Hello`) from the Gateway, through the topic (`mytopic`) on the MQTT broker (`localhost`), and then to a Development Computer that subscribed to that MQTT topic. (`mytopic`)

Continue to Development Suggestions for ideas to extend the exercises in this guide.

7.0 Development Suggestions

Consider the following ideas to enhance the use of MQTT messages in your environment:

- Get data from your Gateway sensors and publish sensor data to an MQTT broker.
- Push messages into a local database or to a Cloud database.
- Increase the complexity of the messages you send.
- Experiment with Paho Python functions to send multiple messages with a single command.
- Add security features to the MQTT message communications through encryption, whitelisting, or other means.