How to secure an MQTT connection from IBM Integration Bus to the IBM Watson IoT Platform on Bluemix

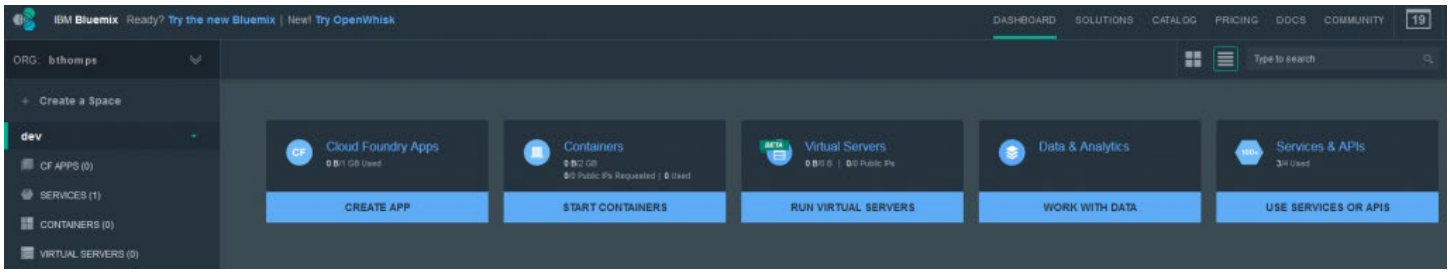
BenThompsonIBM
Published on May 27, 2016 / Updated on August 7, 2018

MQ Telemetry Transport, or MQTT for short, is a lightweight messaging protocol which is particularly useful in situations where you wish to minimize network bandwidth and device resource requirements (such as battery life on a mobile phone for example). MQTT uses a publish-subscribe style of messaging so that the information provider (publisher) can be decoupled from the consumer of the information (subscriber). One of the best known usages and widespread implementations of MQTT as a protocol is as the transport underpinning Facebook messenger

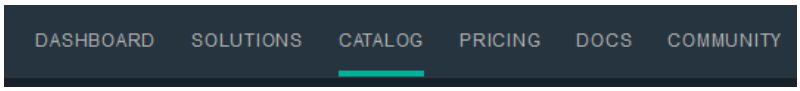
IBM Integration Bus provides an MQTTPublish message flow processing node and an MQTTSubscribe message flow processing node as one of the new features in IIBv10. These nodes communicate with an MQTT server (external to IIB) using MQTT over TCPIP. With the latest fix pack, v10.0.0.5, we have extended this support to also allow SSL to be used to encrypt the MQTT protocol. This blog post describes how you can use these IIB capabilities to securely exchange information with an MQTT server. There are lots of MQTT servers available, some of which are licensed through a vendor such as IBM, who provide an MQTT server implementation as part of the MQ product. There are also open source options available such as Mosquitto for example, which is EPL/EDL licensed.

This post will demonstrate the use of IBM's Watson IoT platform as an MQTT server, which you can try out for free through Bluemix. The instructions which follow describe how you can quickly get this service up and running from scratch.

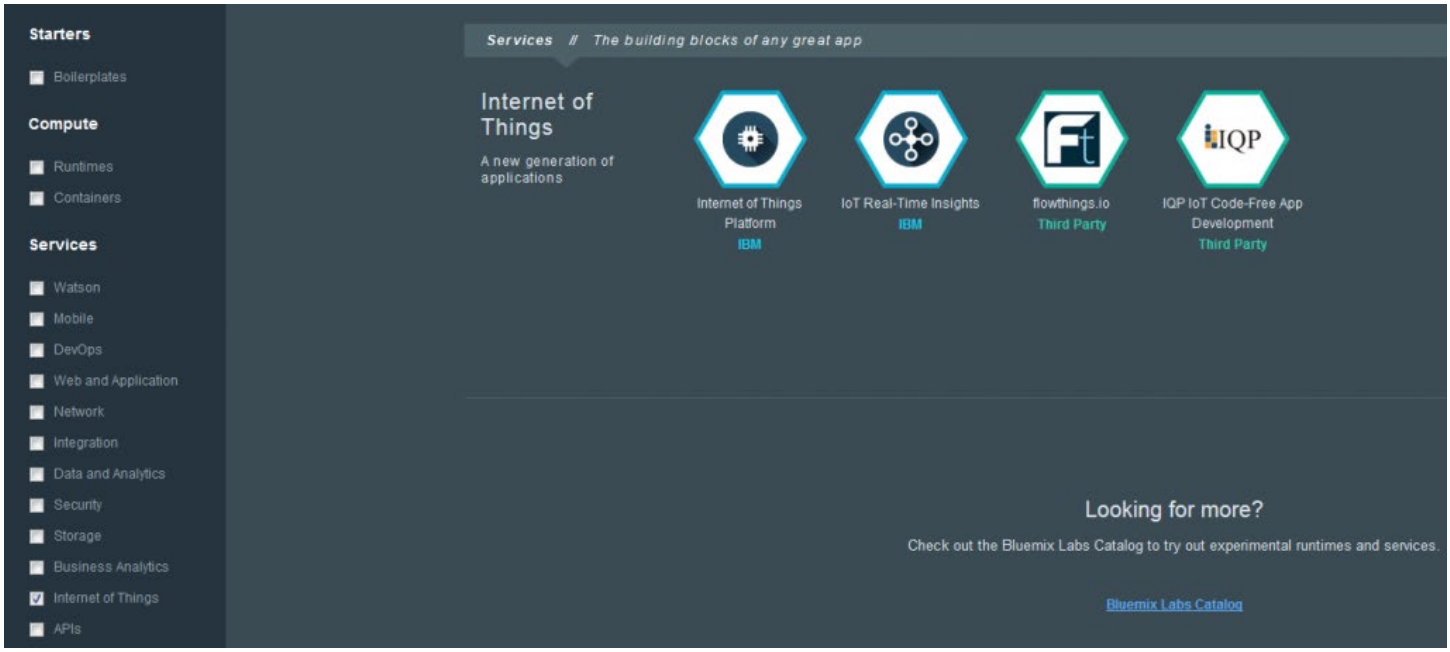
If you have an id already then start by logging on to Bluemix . If not, you can easily register for an id



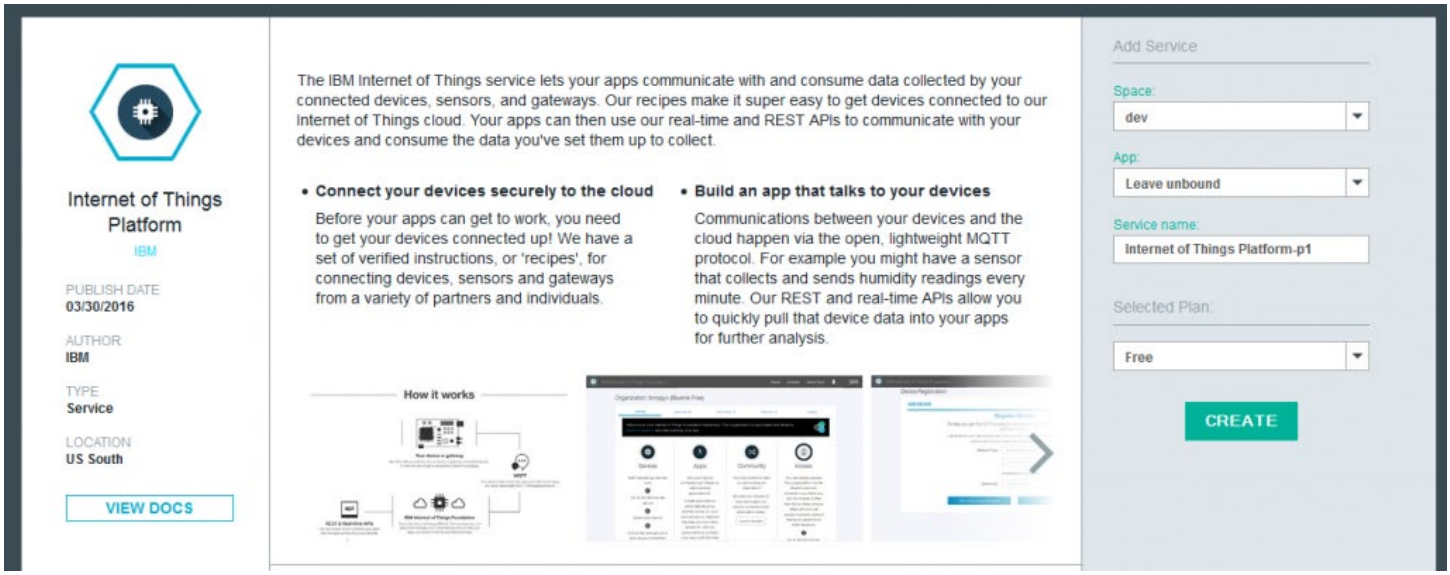
Using the link at the top of the Browser screen, navigate to the Bluemix Catalog:



Apply a filter to the Catalog view using the options on the left of the screen; Under `Services`, select the `Internet of Things` checkbox. The filtered view of available Services should look like this:

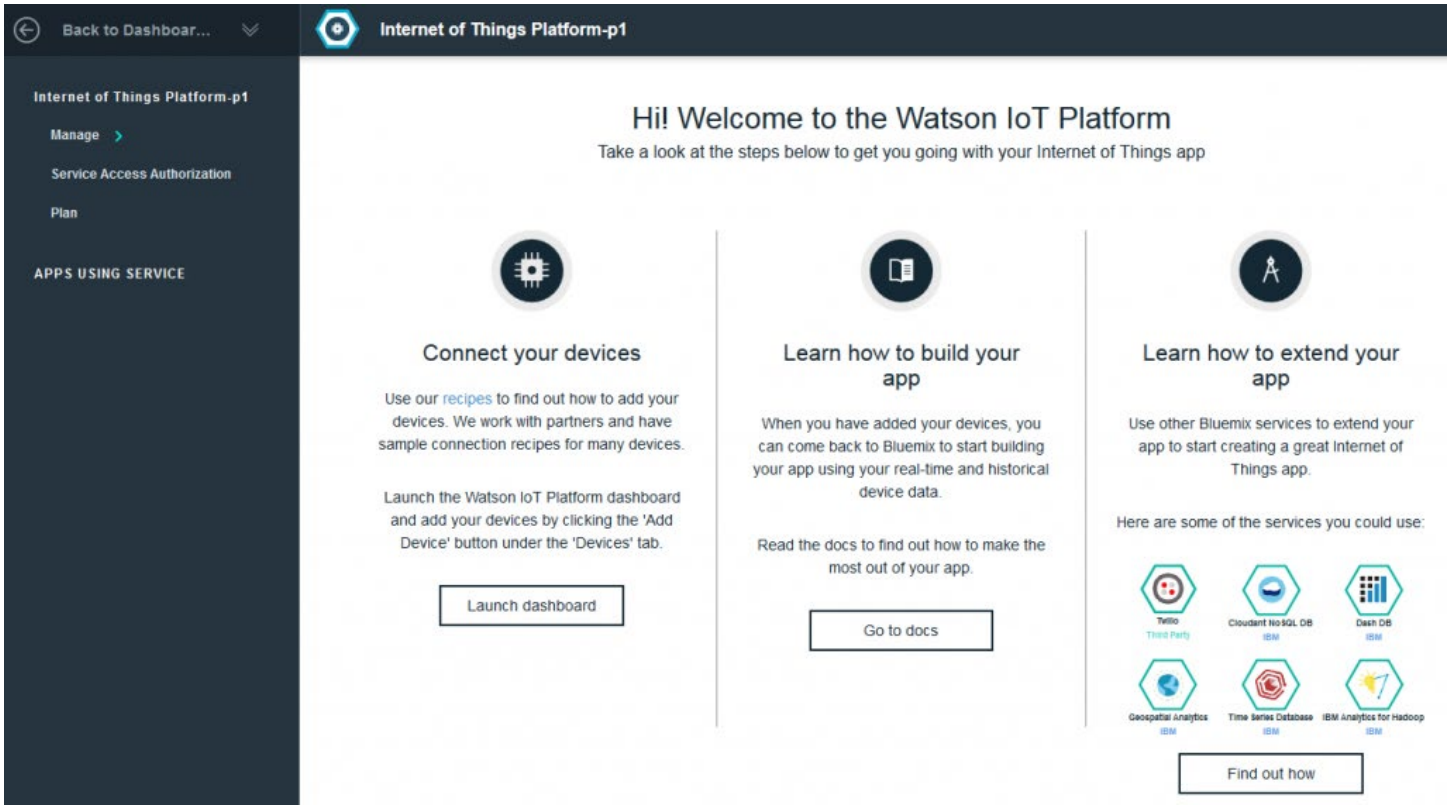


4. Click the `Internet of Things Platform` tile:



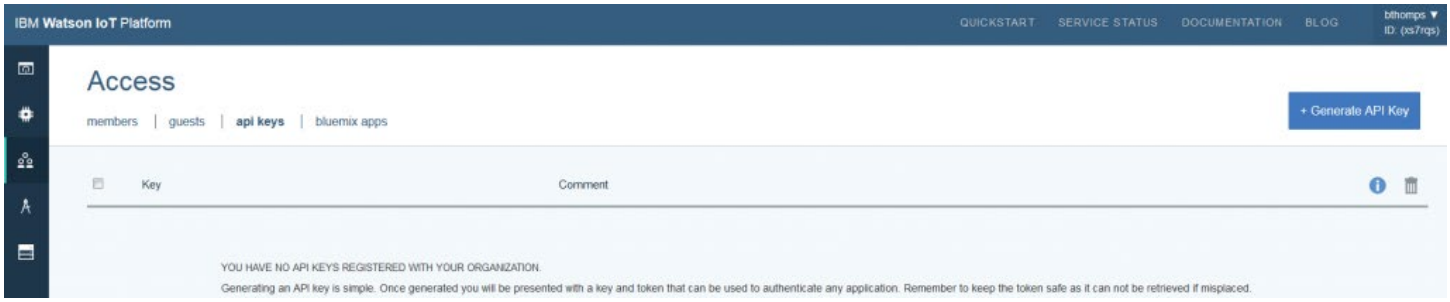
Click the `Create` button to create an instance of the service. The Free Plan will be more than sufficient for our demonstration purposes as it allows you to register up to 10 application bindings, 20 devices and exchange 100MB of data.

5. You should now be faced with the Watson IoT welcome screen:



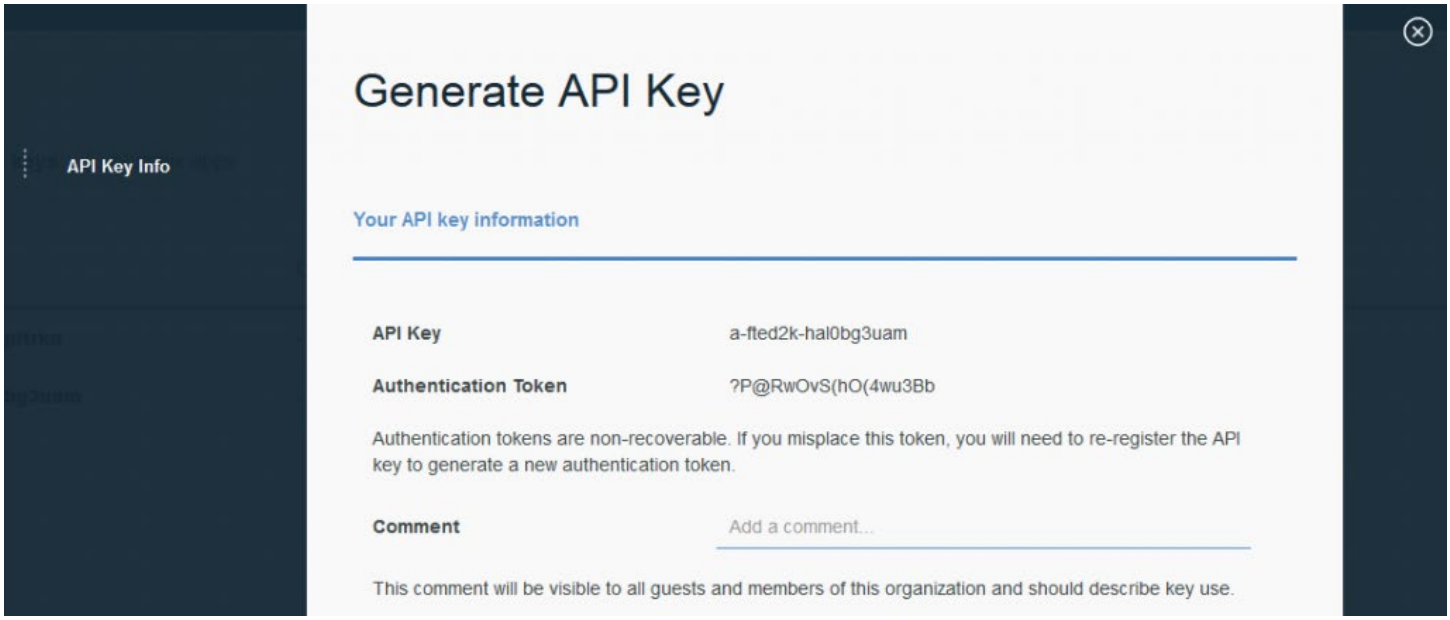
Click the `Launch dashboard` button.

6. Next you'll set up secure access to the service.



Navigate to the Access panel (shown above) using the menu on the left of the IoT dashboard, and then click the blue `Generate API Key` button on the right of the screen.

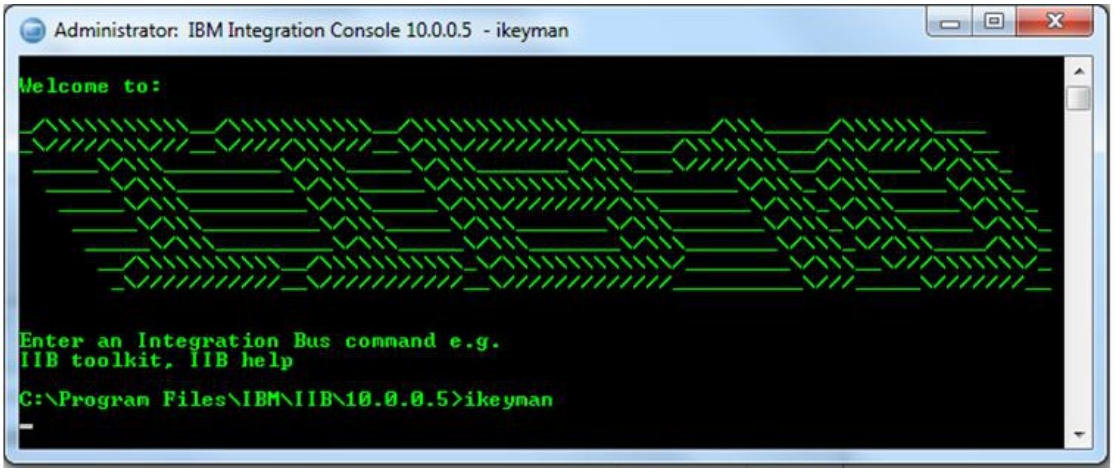
> An API key will be generated for you with an associated Authentication Token. When this is shown, you might want to copy and save this somewhere because once you click finish you will not be shown the value again!! (If this happens, just generate a new API Key and a new associated Authentication Token)



When you registered with the IoT platform, you were given an organization ID, which is a unique 6 character identifier for your account. Applications then require an API Key to connect into that organization. When you make an MQTT connection into the IoT platform these values are used when configuring IBM Integration Bus. The table below should help you understand how this information is structured and may help translate the terminology you might encounter for these properties when using the IoT and IIB documentation:

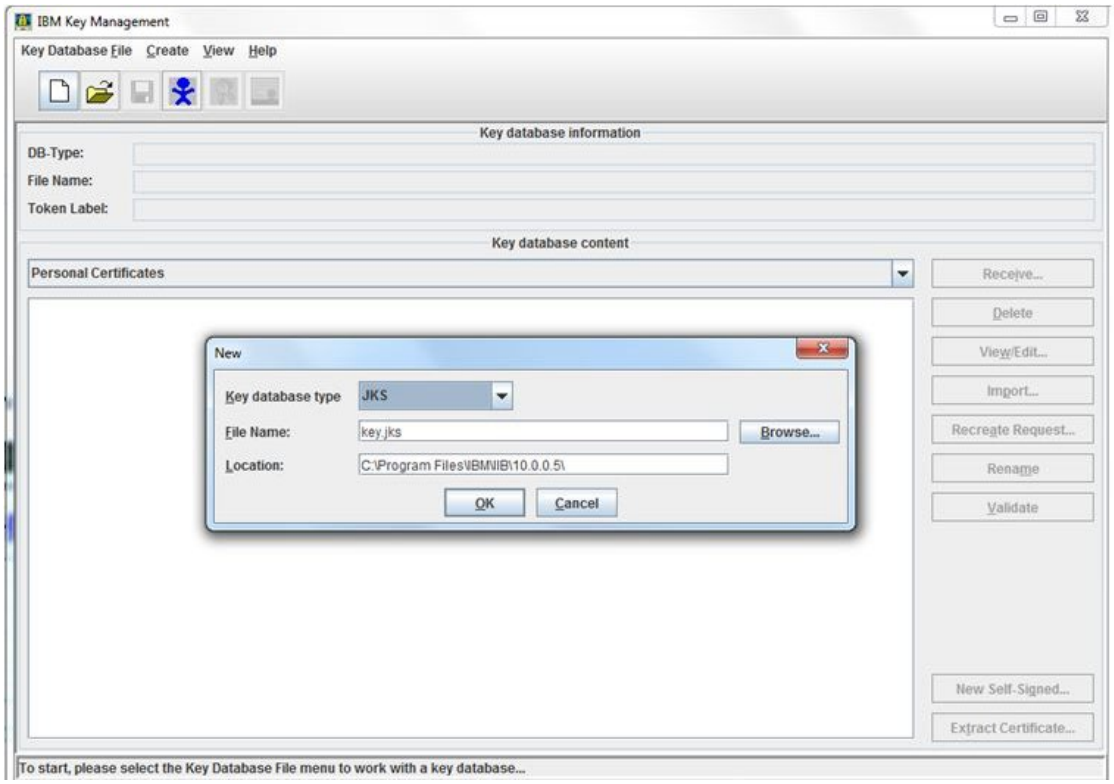
MQTT client ID	a:orgid:app_id	`a:fted2k:IIBPub`
MQTT username	IoT API key	`a-fted2k-hal0bg3uam`
MQTT password	IoT Authentication token	`?P@RwOvS(hO(4wu3Bb`

We will now focus our attention on setting up a Trust Store which will be used by the IIB Integration Server where the message flow which contains the MQTT nodes will be deployed. Launch an IIB Command Console session, ensuring you "Run as administrator" (this will avoid any later issues regarding write permissions when you save the trust store). The command console session will place you in the root directory of the IIB installation, `C:\Program Files\IBM\IIB\10.0.0.5`, from where you can run the IBM Key Management user interface by executing the command `ikeyman`:



This command will launch the Key Management UI, shown below.

Click the New button, and create a new store of JKS type. The rest of these instructions will assume you choose a File Name of `key.jks` and a Location of `C:\Program Files\IBM\IIB\10.0.0.5`



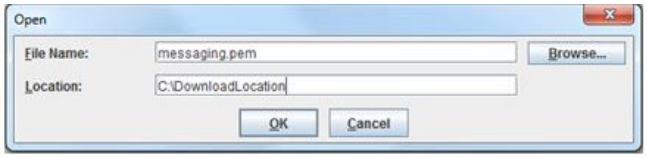
You will then be prompted to supply a password which will be used by the IIB Integration Server to access the Trust Store. The rest of these instructions assume you use a value of `changeit`.

Using the drop-down in the centre of the UI, switch from the `Personal Certificates` view of the key database to the `Signer Certificates` view:

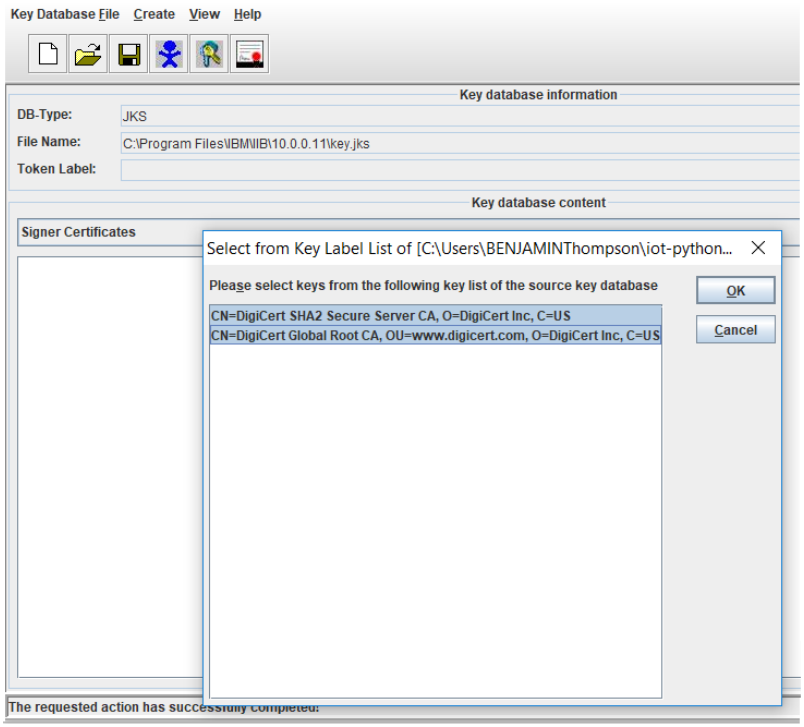
The IoT service (in SSL encryption terms, fulfilling the role of "server") provides a public certificate in .pem format. This is provided as a single file named messaging.pem which contains the entire certificate chain for all IoT services (*.messaging.internetofthings.ibmcloud.com). Download this certificate chain file from here: https://github.com/ibm-watson-iot/iot-python/blob/master/src/ibmiotf/messaging.pem

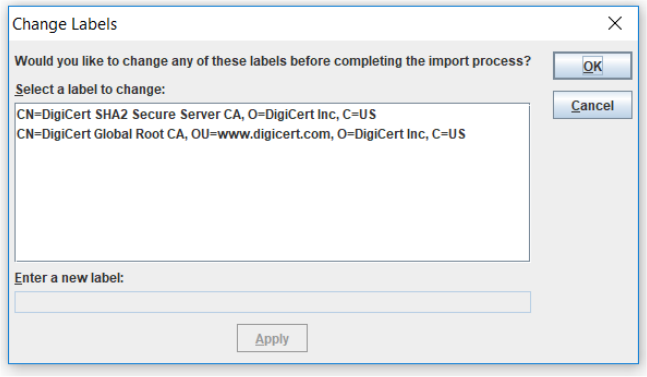
Next click the Add button in the IBM Key Management UI:

In the `Open` dialog, navigate to the `messaging.pem` file which you just downloaded:



When prompted, select and add all of the keys provided in the file.





Now that we have finished preparing the Trust Store, close the IBM Key Management UI and return to the IIB Command Console session that it was launched from, because next we will set up the IIB runtime to use the Trust Store. In the Command Console, if you do not already have an Integration Node defined (you can use the mqsilist command to check if you're not sure), create and start a node with an integration server and display some of its properties using the following commands:

```
mqsicreatebroker TESTNODE_10005

mqsistart TESTNODE_10005

mqsicreateexecutiongroup TESTNODE_10005 -e default

mqsireportproperties TESTNODE_10005 -e default -o ComIbmJVMManager -r
```

Note that at this stage, by default, there are no settings for the properties truststoreType, truststoreFile and truststorePass. For further background information,

this page in the Knowledge Center may be helpful:

Execute this next set of commands to configure the Trust Store settings:

```
mqsichangeproperties TESTNODE_10005 -e default -o ComIbmJVMManager -n truststoreType -v JKS

mqsichangeproperties TESTNODE_10005 -e default -o ComIbmJVMManager -n truststoreFile -v "C:\Program Files\IBM\IIB\10.0.0.5\key.jks"

mqsichangeproperties TESTNODE_10005 -e default -o ComIbmJVMManager -n truststorePass -v default::truststorePass
```

Finally, configure IIB with the password for the Trust Store, and the MQTT user name and password which will be used when forming the SSL connection to IoT. Remember to restart the integration node once you have done this:

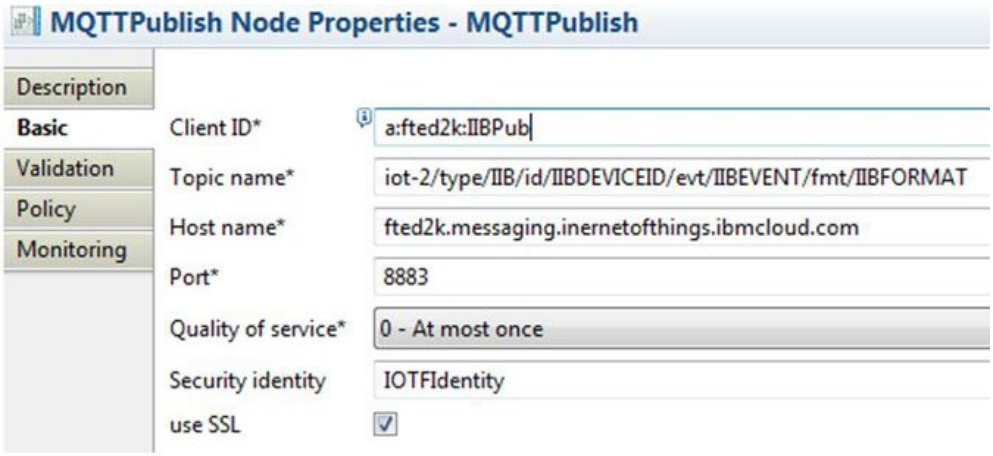
```
mqsisetdbparms TESTNODE_10005 -n default::truststorePass -u na -p changeit

mqsisetdbparms TESTNODE_10005 -n mqtt::IOTFIdentity -u "a-fted2k-hal0bg3uam" -p ?P@RwOvS(hO(4wu3Bb

mqsistop TESTNODE_10005
```

```
mqsistart TESTNODE_10005
```

You have now completed the configuration of the MQTT SSL aspects of the configuration, so all that remains to be done is to create message flows which utilise the MQTT Publish or MQTT Subscribe node. The screen shot below shows an MQTT Publish node which has been configured with settings to match the example values which we have used throughout this post:



As shown above, you can select the `use SSL` checkbox directly on an MQTTPublish or MQTTSubscribe message flow node. It is also possible to dynamically override the combination of the Host name, Port and Use SSL properties by specifying a value named “ConnectionURL” in the LocalEnvironment tree which is passed into the MQTTPublish node. You could do this using a snippet of Compute node ESQL for example.

This example shows an SSL override:

```
SET OutputLocalEnvironment.Destination.MQTT.Output.URL =
'ssl://fted2k.messaging.internetofthings.ibmcloud.com:8883';
```

This example shows a non-SSL override:

```
SET OutputLocalEnvironment.Destination.MQTT.Output.URL =
'tcp://fted2k.messaging.internetofthings.ibmcloud.com:1883';
```

It is also possible to override the MQTT connection used by applying an override on a BAR file:

```
mqsiapplybaroverride -b MQTT_SSLTesting.bar -k MQTTParentApplication -m
MyFlowName#MQTTSubscribe.connectionUrl=ssl://fted2k.messaging.internetofthings.ibmcloud.com:8883
```

TAGS IBM CLOUD, MQTT

BenThompsonIBM

---

2 comments on"How to secure an MQTT connection from IBM Integration Bus to the IBM Watson IoT Platform on Bluemix"

Davey Zheng June 27, 2018

1. Can someone details how to add the certificates into the key.jks from the .pem file. Do we need select/copy/paste to create separated certs from the provided .pem file or just add the .pem file into the keystore?

2. The pem file contains 2 certificates only, however you mentioned the certificates, where is the third one?

Reply (Edit)

BenThompsonIBM August 07, 2018

Hi Davey,

It looks like messaging.pem has subsequently been updated (in October 2017, well after the initial publication of this article) and as you say it now only includes 2 certificates. You can add the whole .pem file into the keystore by selecting the 2 certificates in the ikeyman dialog. We've editted the article to add an extra screenshot showing this step.You can download the messaging.pem file from the github repo indicated in the article (git clone https://github.com/ibm-watson-iot/iot-python.git).

Cheers,
Ben

Reply (Edit)