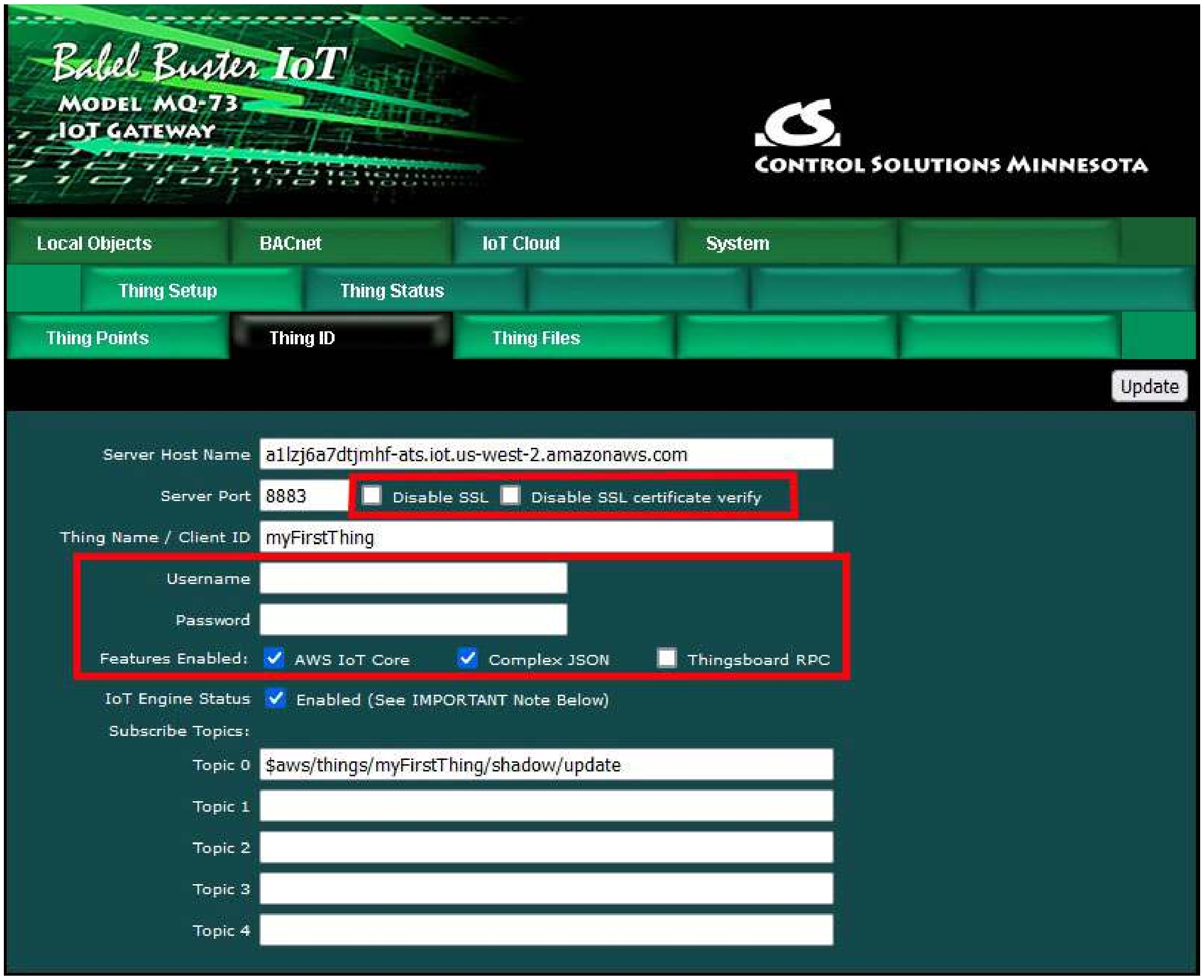
MQTT User Guide Addendum - Mosquitto MQTT

(Updated 7-Nov-2022)

Firmware in MQTT gateways has been updated to provide support for generic non-AWS MQTT brokers such as Mosquitto MQTT. The only visible changes in the web UI are on the Thing ID page. Check boxes have been added in addition to input for username and password. The MQ-73 is illustrated here, but the same change and instructions that follow will apply to any Babel Buster IoT with MQTT.

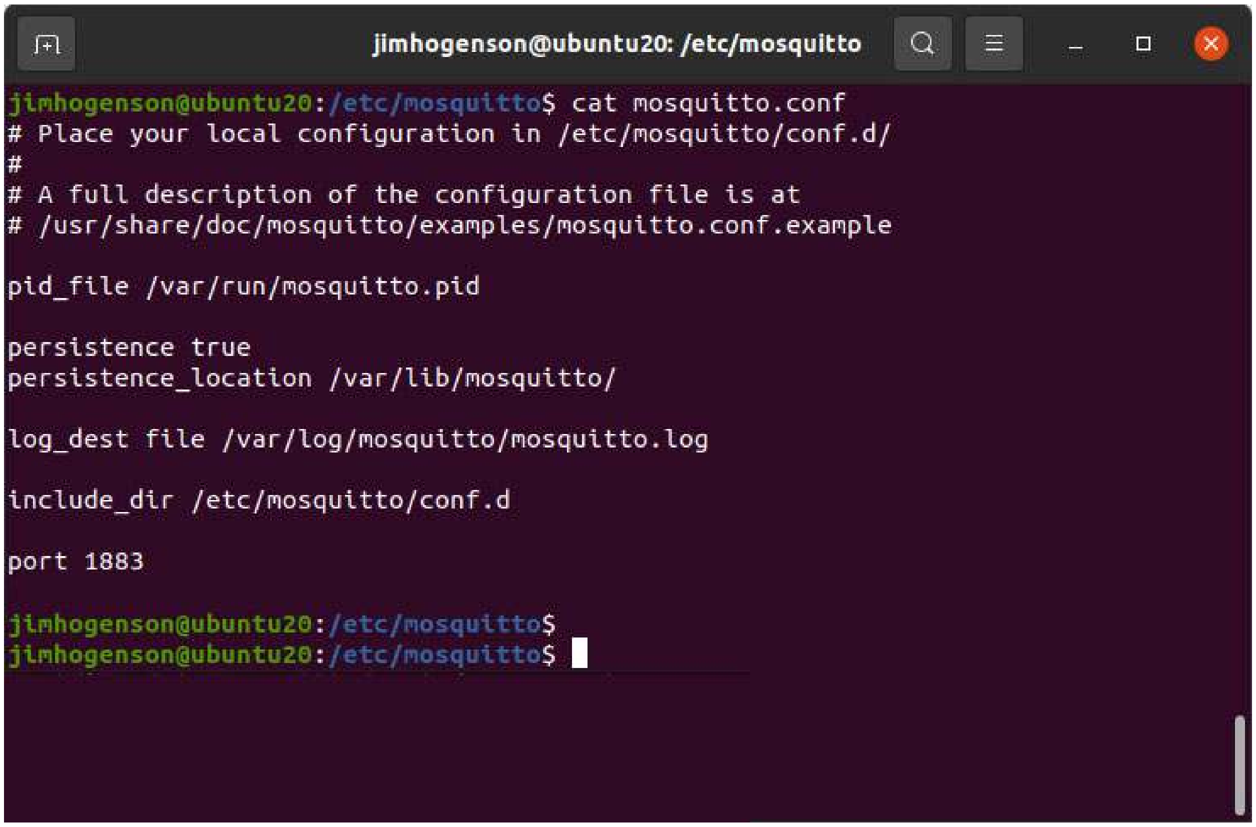


Server Host Name: a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com
Server Port: 8883 ☐ Disable SSL ☐ Disable SSL certificate verify
Thing Name / Client ID: myFirstThing
Username:
Password:
Features Enabled: ✓ AWS IoT Core ✓ Complex JSON ☐ Thingsboard RPC
IoT Engine Status: ✓ Enabled (See IMPORTANT Note Below)
Subscribe Topics:
Topic 0: $aws/things/myFirstThing/shadow/update
Topic 1:
Topic 2:
Topic 3:
Topic 4:

If you will be using AWS IoT for your MQTT broker, simply *check AWS IoT Core and Complex JSON* as highlighted above, leave username and password *blank*, and disregard the rest of this addendum. Nothing about AWS IoT support has changed. Refer to the respective user guide for your gateway.

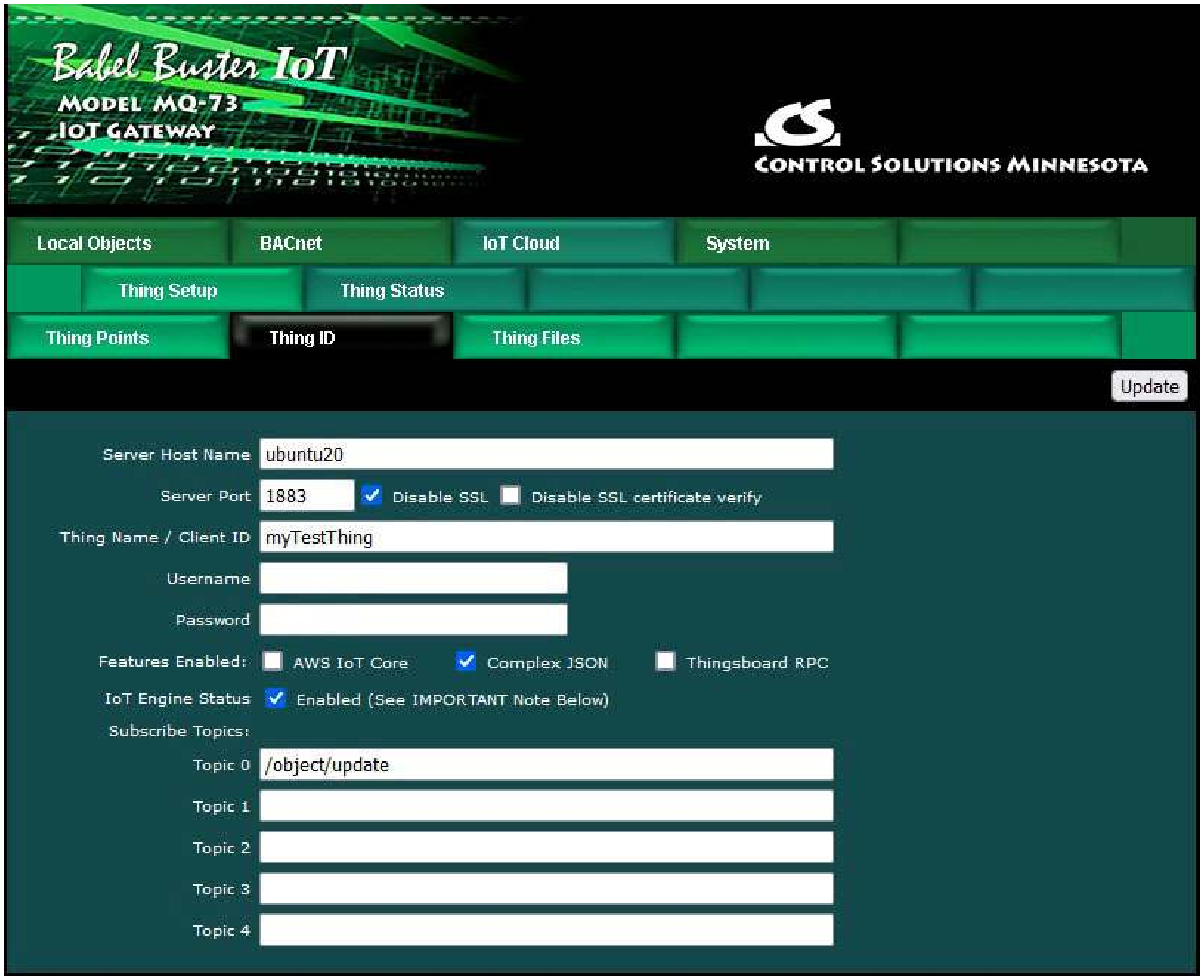
To install Mosquitto MQTT if you have not done so already, follow instructions at http://www.steves-internet-guide.com/mosquitto-broker/.

The configuration file for Mosquitto is found in /etc/mosquitto/mosquito.conf and the minimum configuration would look like the example below.



```
jimhogenson@ubuntu20:/etc/mosquitto$ cat mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

port 1883

jimhogenson@ubuntu20:/etc/mosquitto$
jimhogenson@ubuntu20:/etc/mosquitto$
```

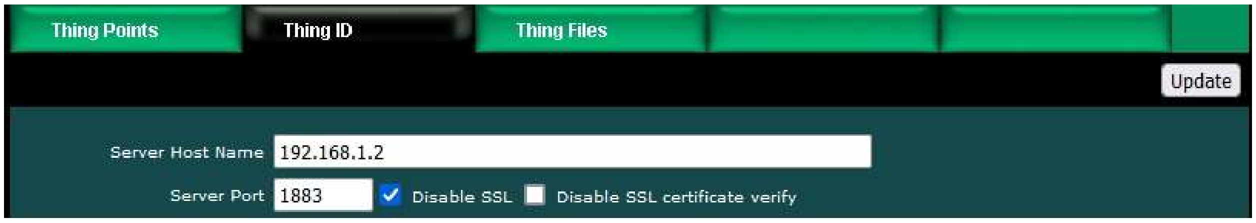
Using a bare minimum configuration with no SSL and no username/password, the Thing ID page would look like the following screen shot.



Server Host Name	ubuntu20
Server Port	1883 [✓ Disable SSL] [ ] Disable SSL certificate verify
Thing Name / Client ID	myTestThing
Username	
Password	
Features Enabled:	[ ] AWS IoT Core [✓ Complex JSON] [ ] Thingsboard RPC
IoT Engine Status	[✓ Enabled (See IMPORTANT Note Below)]
Subscribe Topics:	
Topic 0	/object/update
Topic 1	
Topic 2	
Topic 3	
Topic 4	

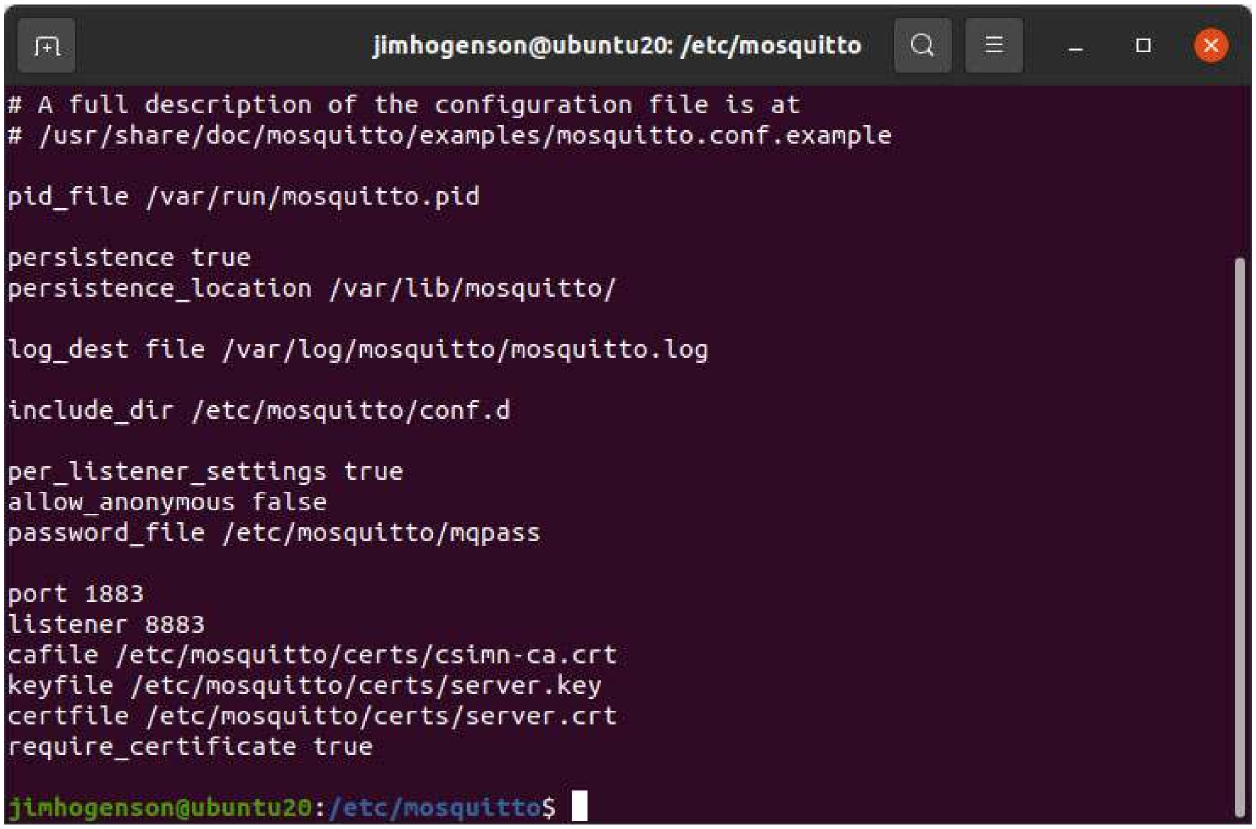
Note that 'ubuntu20' as host name has been added to the local DNS server. DNS lookup of 'ubuntu20' returns the IP address of the local Mosquitto MQTT server. The IP address of the local DNS server has also been entered as primary DNS on the Network page in this gateway.

You can also enter the local server's IP address directly as illustrated below if preferred.



Server Host Name	192.168.1.2
Server Port	1883 [✓ Disable SSL] [ ] Disable SSL certificate verify

Adding both SSL certificates and username/password requirements is illustrated in the mosquitto.conf file pictured below.

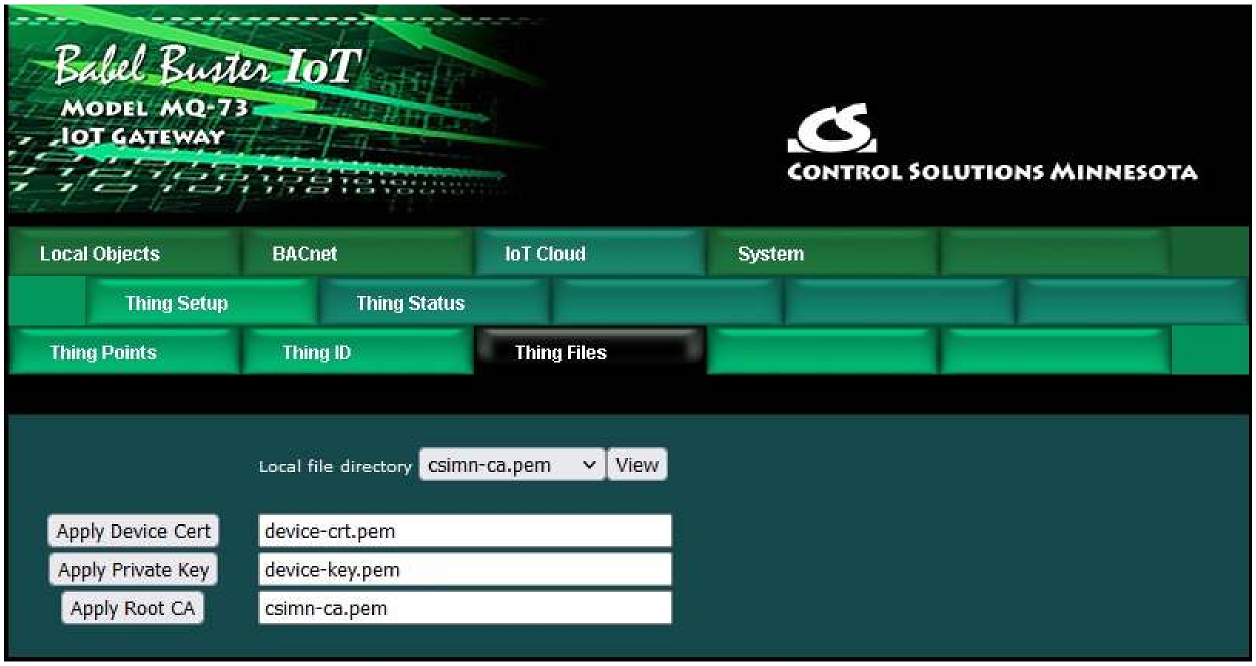


```
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

per_listener_settings true
allow_anonymous false
password_file /etc/mosquitto/mqpass

port 1883
listener 8883
cafile /etc/mosquitto/certs/csimn-ca.crt
keyfile /etc/mosquitto/certs/server.key
certfile /etc/mosquitto/certs/server.crt
require_certificate true

jimhogenson@ubuntu20:/etc/mosquitto$
```

Follow instructions for mosquitto_passwd (under Documentation at mosquitto.org) for creating the password file and adding usernames to it.

To create your own SSL certificates for both the Mosquitto server and the client (Babel Buster IoT), follow instructions at http://mosquitto.org/man/mosquitto-tls-7.html and see also https://asciinema.org/a/201826.

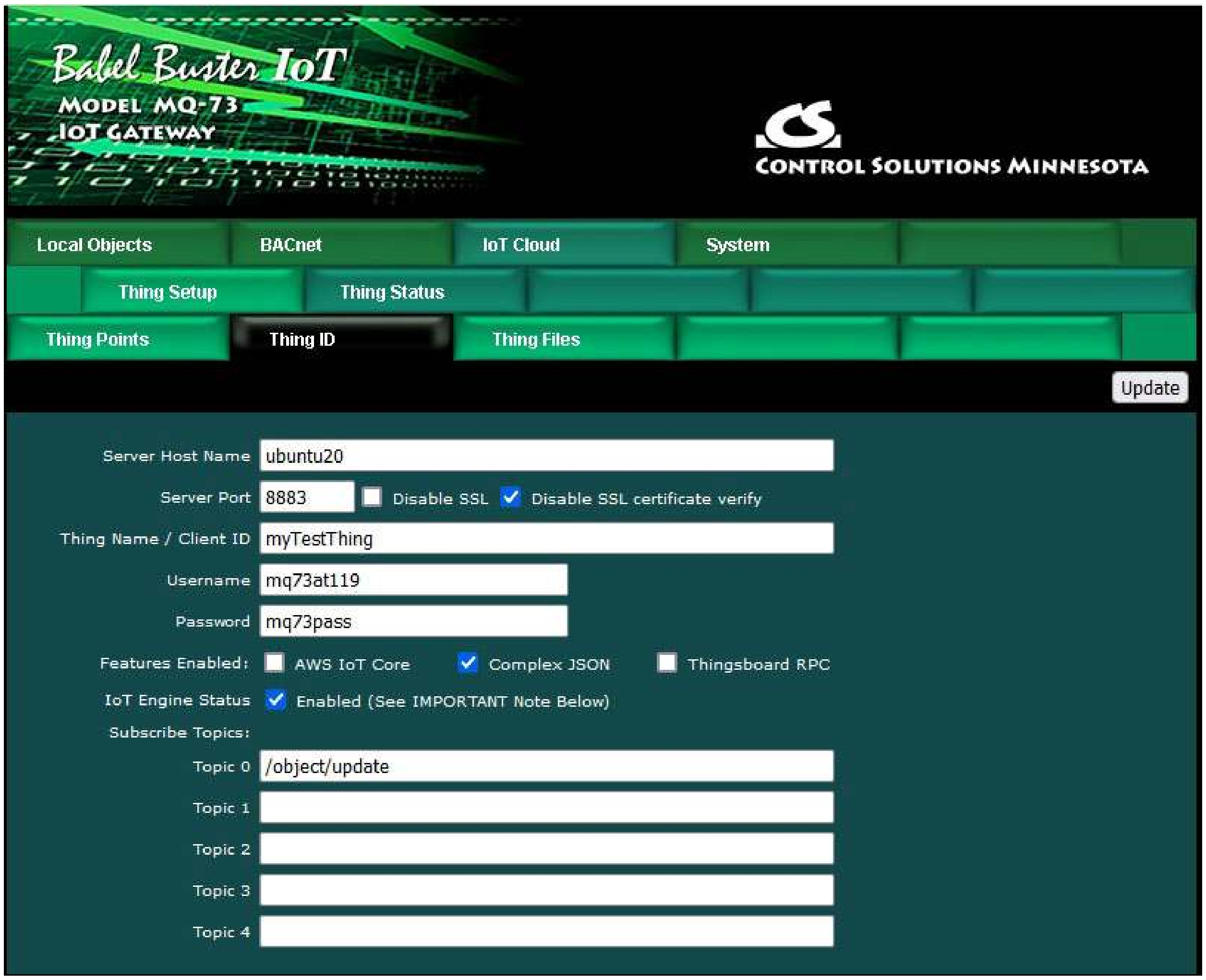
Certificates for use with Mosquitto are uploaded and installed in the same manner as for AWS. An example of the Thing Files page is illustrated below.



Local file directory csimn-ca.pem

Apply Device Cert: device-crt.pem
Apply Private Key: device-key.pem
Apply Root CA: csimn-ca.pem

The Thing ID page when SSL and username/password are configured in Mosquitto would appear as in the screen shot below.



Server Host Name: ubuntu20

Server Port: 8883 ☐ Disable SSL ☑ Disable SSL certificate verify

Thing Name / Client ID: myTestThing

Username: mq73at119

Password: mq73pass

Features Enabled: ☐ AWS IoT Core ☑ Complex JSON ☐ Thingsboard RPC

IoT Engine Status: ☑ Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0: /object/update

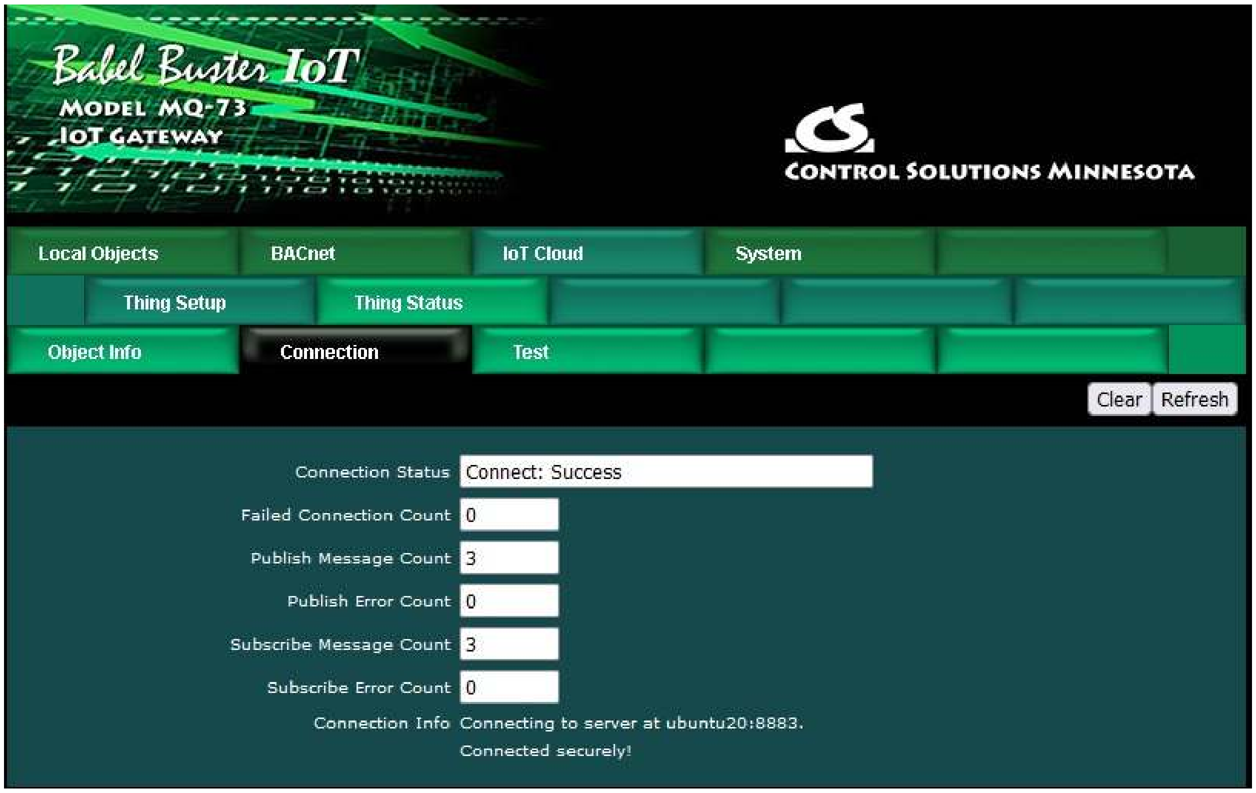
Topic 1:

Topic 2:

Topic 3:

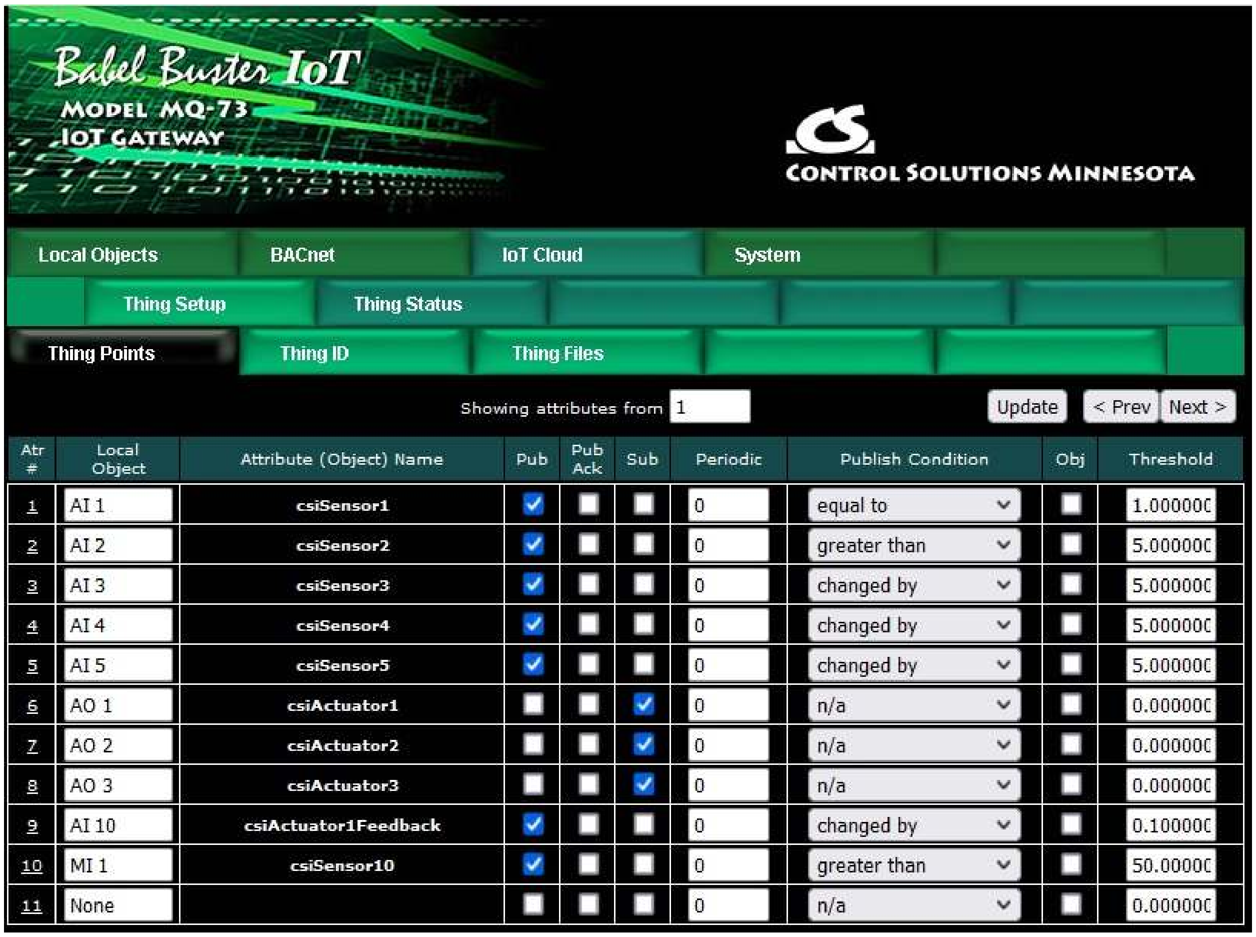
Topic 4:

Upon successful connection, you should see the "success" indication as pictured below.



Connection Status	Connect: Success
Failed Connection Count	0
Publish Message Count	3
Publish Error Count	0
Subscribe Message Count	3
Subscribe Error Count	0
Connection Info	Connecting to server at ubuntu20:8883. Connected securely!

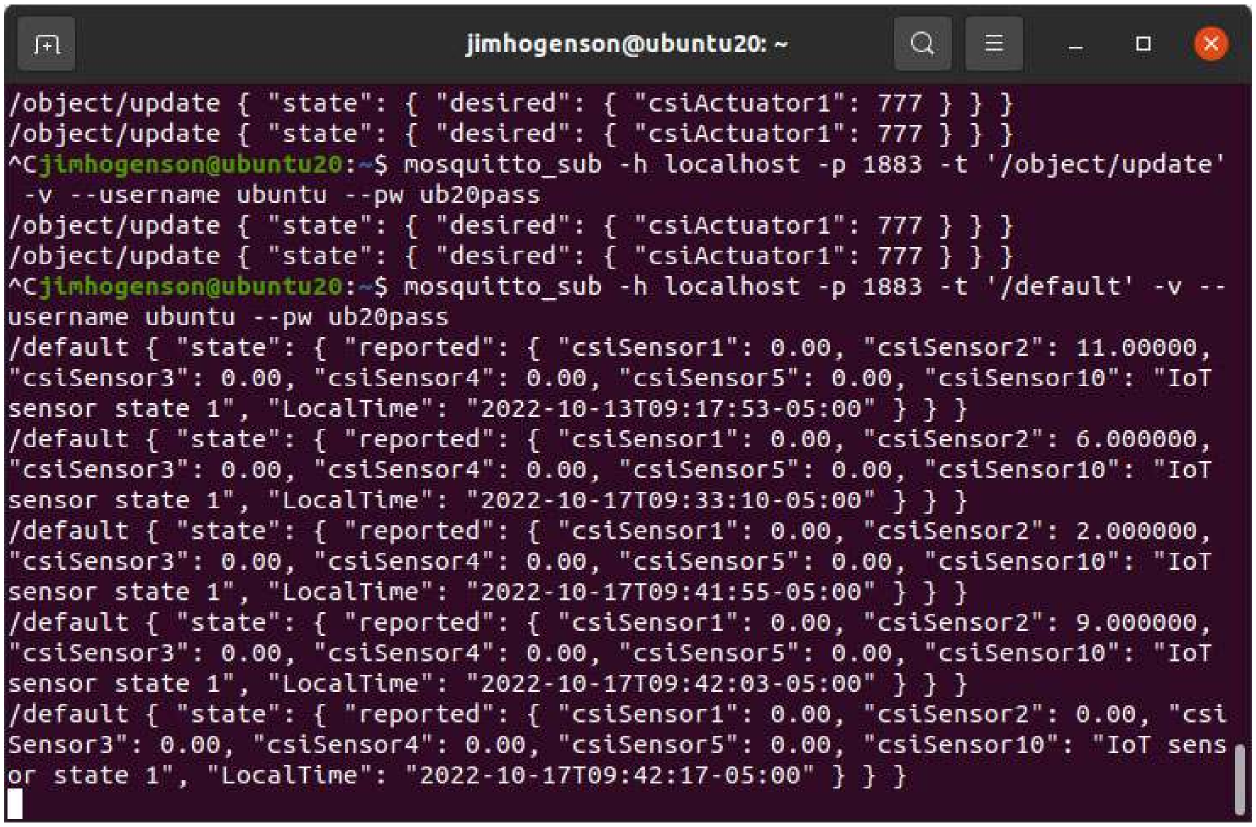
The same set of Thing Points, along with the same publish and subscribe rules, as used for AWS will work the same with Mosquitto MQTT or any other MQTT broker.



Showing attributes from 1

Atr #	Local Object	Attribute (Object) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Obj	Threshold
1	AI 1	csiSensor1	✓			0	equal to		1.00000C
2	AI 2	csiSensor2	✓			0	greater than		5.00000C
3	AI 3	csiSensor3	✓			0	changed by		5.00000C
4	AI 4	csiSensor4	✓			0	changed by		5.00000C
5	AI 5	csiSensor5	✓			0	changed by		5.00000C
6	AO 1	csiActuator1			✓	0	n/a		0.00000C
7	AO 2	csiActuator2			✓	0	n/a		0.00000C
8	AO 3	csiActuator3			✓	0	n/a		0.00000C
9	AI 10	csiActuator1Feedback	✓			0	changed by		0.10000C
10	MI 1	csiSensor10	✓			0	greater than		50.0000C
11	None					0	n/a		0.00000C

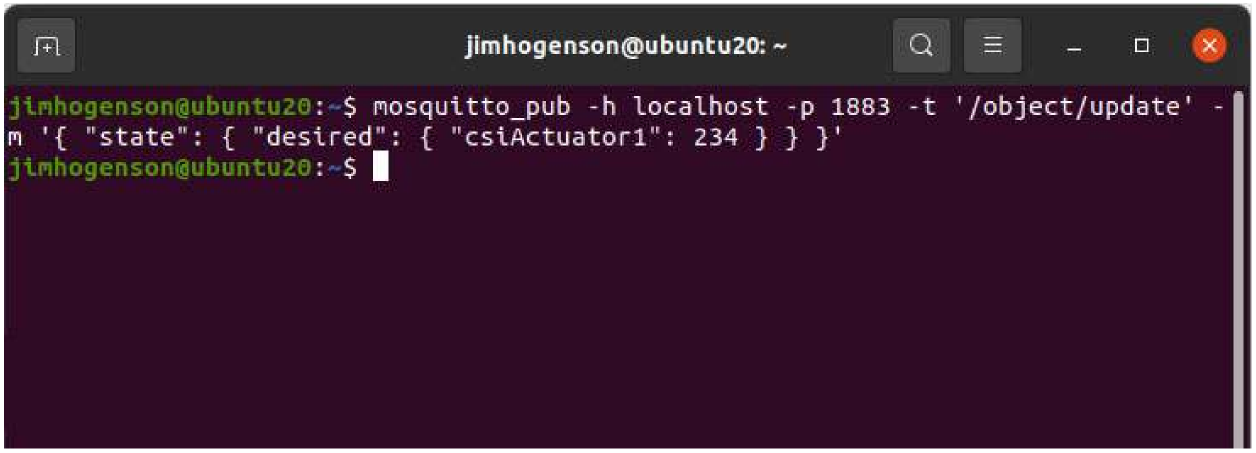
The following screen shot shows using the mosquitto_sub utility to subsrcribe to the default topic for testing the Babel Buster IoT publish to that topic. The mosquitto_sub is among the utilities installed when you install Mosquitto on your Linux server. Refer to mosquitto.org Documentation for further instructions on using mosquitto_sub.



jimhogenson@ubuntu20: ~

```
/object/update { "state": { "desired": { "csiActuator1": 777 } } }
/object/update { "state": { "desired": { "csiActuator1": 777 } } }
^Cjimhogenson@ubuntu20:~$ mosquitto_sub -h localhost -p 1883 -t '/object/update'
 -v --username ubuntu --pw ub20pass
/object/update { "state": { "desired": { "csiActuator1": 777 } } }
/object/update { "state": { "desired": { "csiActuator1": 777 } } }
^Cjimhogenson@ubuntu20:~$ mosquitto_sub -h localhost -p 1883 -t '/default' -v --
username ubuntu --pw ub20pass
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 11.00000,
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT
sensor state 1", "LocalTime": "2022-10-13T09:17:53-05:00" } } }
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 6.000000,
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT
sensor state 1", "LocalTime": "2022-10-17T09:33:10-05:00" } } }
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 2.000000,
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT
sensor state 1", "LocalTime": "2022-10-17T09:41:55-05:00" } } }
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 9.000000,
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT
sensor state 1", "LocalTime": "2022-10-17T09:42:03-05:00" } } }
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 0.00, "csi
Sensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT sens
or state 1", "LocalTime": "2022-10-17T09:42:17-05:00" } } }
```

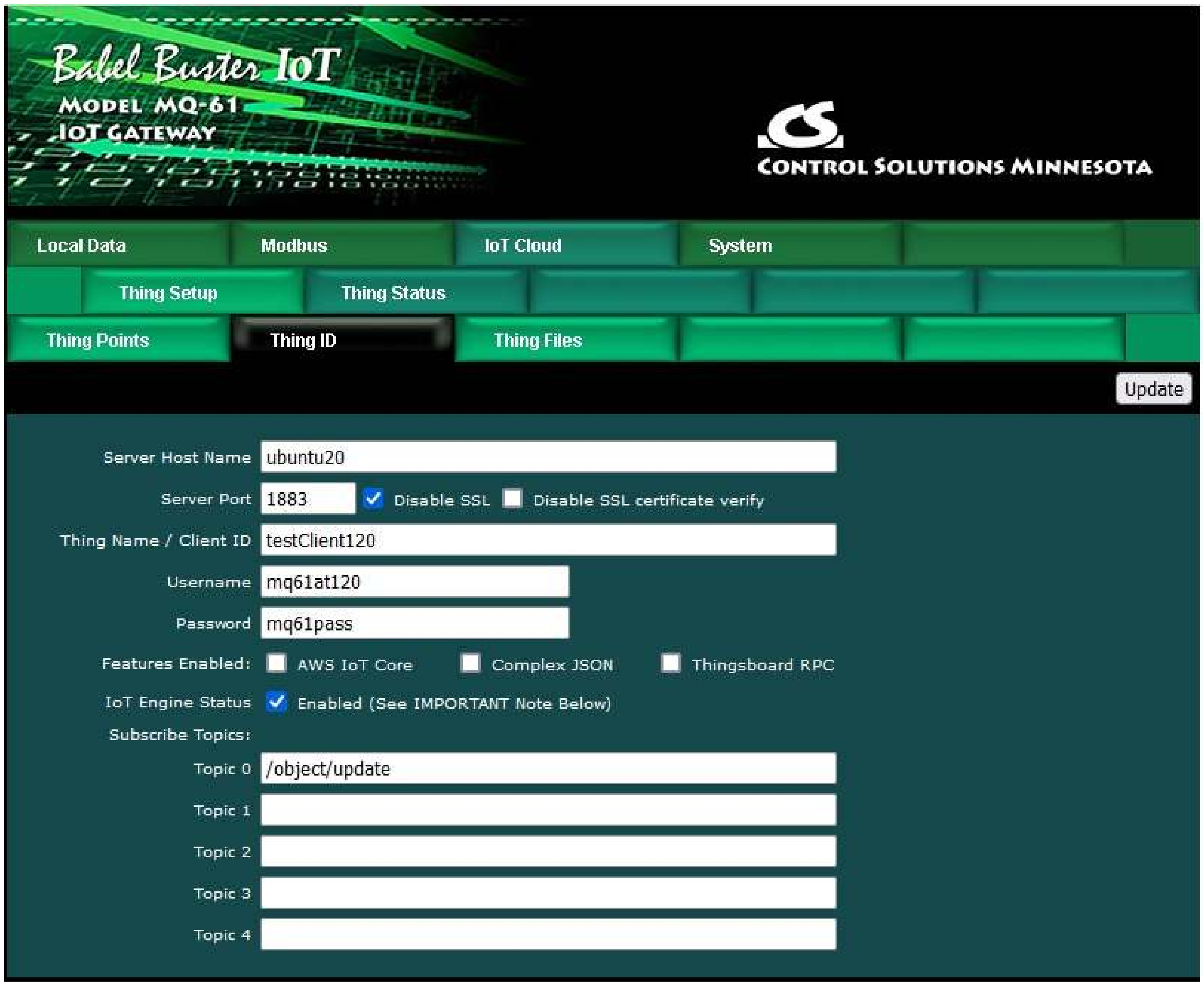
The following screen shot shows an example of publishing from the test client to Babel Buster IoT using the mosquitto_pub utility. This example was created prior to adding username/password to this instance of the broker.



jimhogenson@ubuntu20: ~

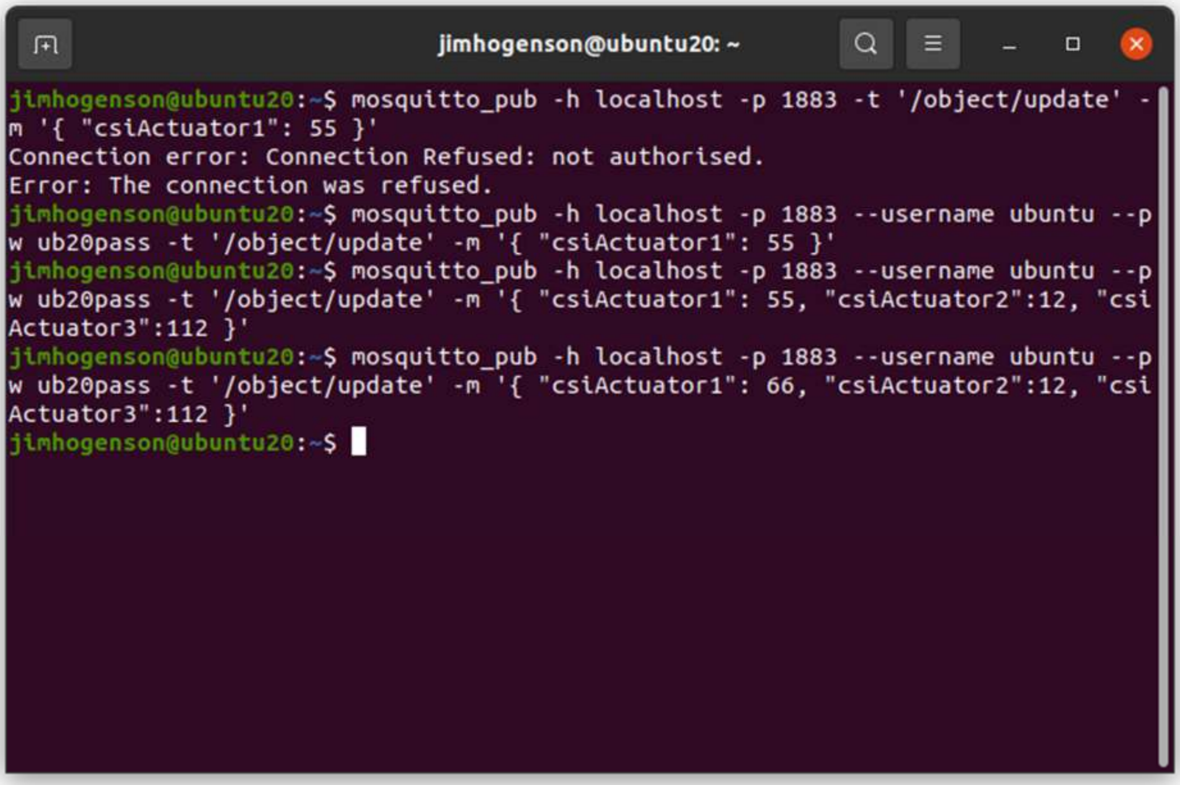
```
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 -t '/object/update' -
m '{ "state": { "desired": { "csiActuator1": 234 } } }'
jimhogenson@ubuntu20:~$
```

The JSON expected by AWS IoT Core is a complex object structure. You have the option of keeping this complex structure, or using "simple" JSON. Some applications may require just simple JSON. To switch to simple JSON, just un-select Complex JSON on the Thing ID page as illustrated below.



Server Host Name	ubuntu20
Server Port	1883 [✓] Disable SSL [ ] Disable SSL certificate verify
Thing Name / Client ID	testClient120
Username	mq61at120
Password	mq61pass
Features Enabled:	[ ] AWS IoT Core [ ] Complex JSON [ ] Thingsboard RPC
IoT Engine Status	[✓] Enabled (See IMPORTANT Note Below)
Subscribe Topics:	
Topic 0	/object/update
Topic 1	
Topic 2	
Topic 3	
Topic 4	

The screen shot below shows publishing "simple" JSON to the broker which in turn will forward this message to the MQ device. Compare this to the mosquitto_pub example above.



```
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 -t '/object/update' -
m '{ "csiActuator1": 55 }'
Connection error: Connection Refused: not authorised.
Error: The connection was refused.
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p
w ub20pass -t '/object/update' -m '{ "csiActuator1": 55 }'
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p
w ub20pass -t '/object/update' -m '{ "csiActuator1": 55, "csiActuator2":12, "csi
Actuator3":112 }'
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p
w ub20pass -t '/object/update' -m '{ "csiActuator1": 66, "csiActuator2":12, "csi
Actuator3":112 }'
jimhogenson@ubuntu20:~$ 
```