How to Use ThingsPro Gateway to Publish Processed Edge Data to the AWS IoT Cloud

Miles Wang
Application Engineer
miles.wang@moxa.com

Contents

Released on August 31, 2017

1 Background

Moxa's ThingsPro Gateway provides a convenient way to acquire device data through Modbus RTU or Modbus/TCP protocol. In addition, you can create your own data tags and automatically transmit processed data to a remote IoT cloud using the ThingsPro Gateway software.

2 Requirements

- ThingsPro Gateway software installed on an eligible Moxa device (e.g., UC-8112-LX)
- AWS IoT user account

3 Overview

In this tech note, we explain how you can take advantage of the **Custom Equipment Management** function in ThingsPro Gateway to read temperature values from a device via Modbus/TCP protocol and publish the current temperature, the highest temperature, and lowest temperature values to the AWS IoT cloud using the following steps:

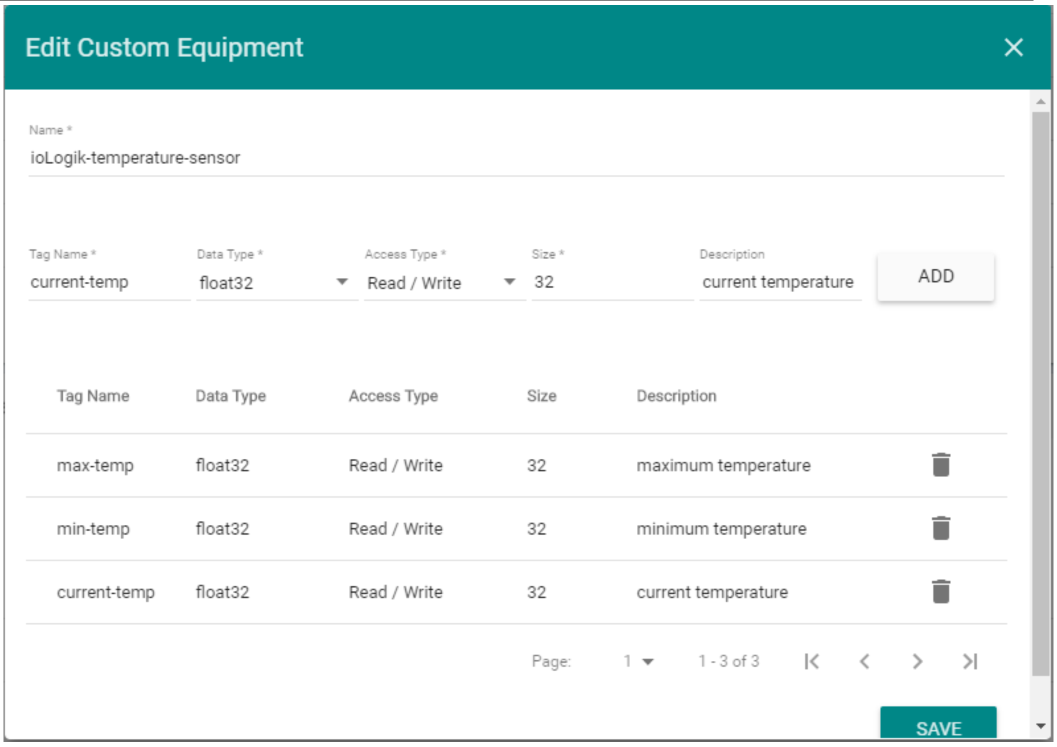
- Creating Custom Tags/Registers
- Setting Up a Modbus TCP Device
- Running a Program to Read and Publish Data from the Device to the AWS IoT Cloud
- Enabling the AWS IoT Client in the ThingsPro Gateway
- Checking the Data on AWS IoT

3.1 Creating Custom Tags/Registers

To create the custom tags required for this project, do the following:

1. Connect to the ThingsPro Gateway homepage using the Chrome browser.
2. In the main menu, click **MODBUS & Logging** and select the **EQUIPMENT TEMPLATE** tab.
3. In the **Custom Equipment Management** section, click ≡+ to open the **Edit Custom Equipment** page
4. Fill in a name of custom equipment ("ioLogik-temperature-sensor" in this example)
5. Add three custom tags—`current-temp`, `max-temp`, and `min-temp`—as shown in Figure 1.

Note: The custom tags created here are used in the example code in Section 3.3.



Name *: ioLogik-temperature-sensor

Tag Name *	Data Type *	Access Type *	Size *	Description
current-temp	float32	Read / Write	32	current temperature

Tag Name	Data Type	Access Type	Size	Description
max-temp	float32	Read / Write	32	maximum temperature
min-temp	float32	Read / Write	32	minimum temperature
current-temp	float32	Read / Write	32	current temperature

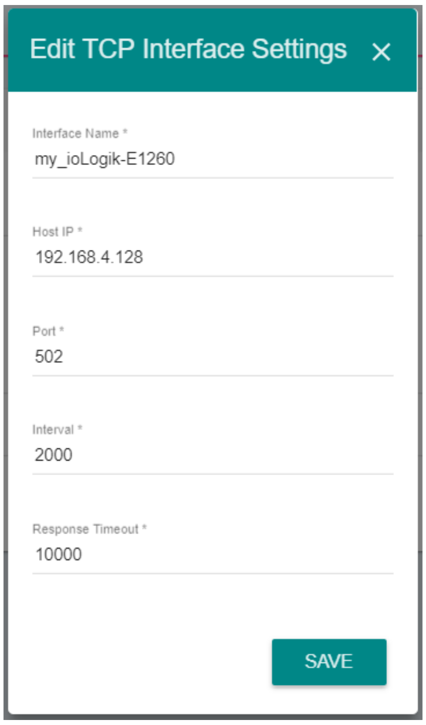
Page: 1 1 - 3 of 3

Figure 1

6. Click **SAVE**

3.2 Setting Up a Modbus TCP Device

1. In the main menu, click **MODBUS & Logging** and select the **MODBUS DEVICE** tab.
2. Click on the **MODBUS/TCP** tab.
3. Click ≡+ to open **Edit TCP Interface Settings** page.
4. Specify the **Interface Name**, **Host IP**, **Port**, **Interval**, and **Response Timeout** values for the TCP interface.
   The **Host IP** is the IP address of Modbus TCP Server. In this example, a temperature sensor is connected to an ioLogik E1260, which allows a Modbus client in the ThingsPro Gateway to pull temperature values via the Modbus TCP protocol at time intervals specified in the **Interval** field (unit: millisecond). See Figure 2 for an example configuration.



Edit TCP Interface Settings

Interface Name *: my_ioLogik-E1260

Host IP *: 192.168.4.128

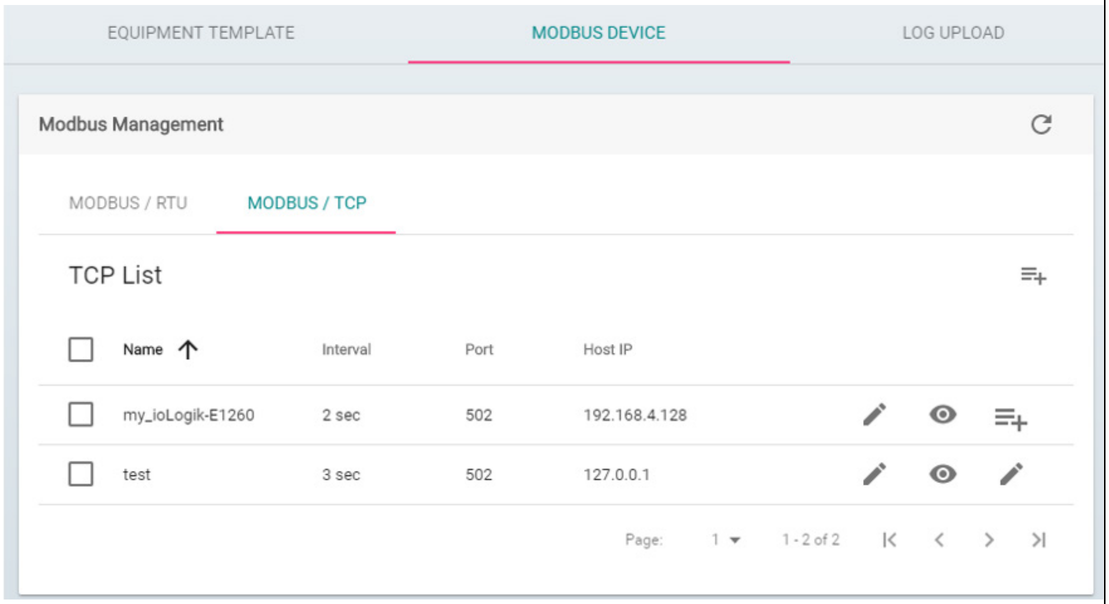
Port *: 502

Interval *: 2000

Response Timeout *: 10000

Figure 2

5. Click **SAVE**
   A new item, **`my_ioLogik-E1260`** is created in TCP List.



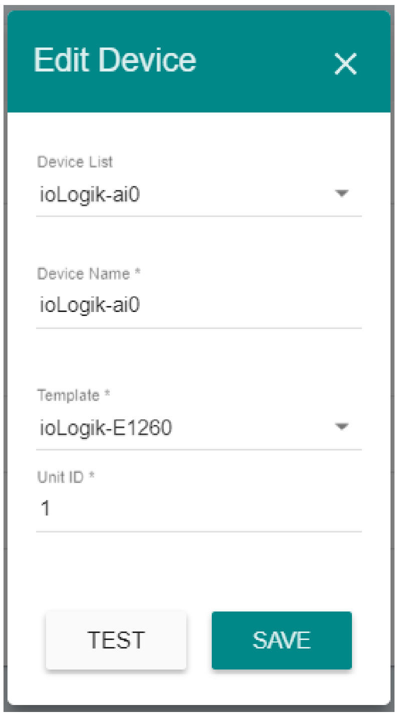
TCP List

Name	Interval	Port	Host IP
my_ioLogik-E1260	2 sec	502	192.168.4.128
test	3 sec	502	127.0.0.1

Figure 3

6. Click ≡+ corresponding to the `my_ioLogik-E1260` device to open the **Edit Device** page.
7. In the **Template** field, select the built-in template `ioLogik-E1260`
8. Specify the **Device Name** (`ioLogik-ai0` in this example).

Note: The device name, `ioLogik-ai0`, specified here has been used to read the Modbus TCP values in the example code in Section 3.3.



Device List: ioLogik-ai0

Device Name *: ioLogik-ai0

Template *: ioLogik-E1260

Unit ID *: 1

Figure 4

3.3 Running a Program to Read and Publish Data from the Device to the AWS IoT Cloud

We have created an example Python program to read Modbus TCP temperature value every five seconds and publish the value of a user-defined tag (`max-temp`) value to ThingsPro Gateway's MQTT server if the current temperature value is higher than the maximum value it has recorded. ThingsPro Gateway will then publish this maximum temperature value to AWS IoT cloud. A similar approach is used to transmit the minimum temperature to the AWS IoT cloud.

To run the example program, do the following:

1. Copy and save the sample code given below in the **/home/moxa/usertag-example.py** file on the UC-8112-LX computer.

Note: At the beginning of the example code, you must call the Moxa proprietary library `libmxidaf_py` to import the modules `Modbus`, `TagV2`, `Tag`, `Time` and `Value` to enable your program to read Modbus data from the ThingsPro Gateway. For more detailed information on the data acquisition API, refer to *ThingsPro Programmer's Guide*.

```
-------------------------------------------------------------------
usertag-example.py
-------------------------------------------------------------------
#!/usr/bin/env python
import time
from libmxidaf_py import Modbus, TagV2, Tag, Time, Value


modbus = Modbus.instance()
tagv2_tmin = TagV2.instance()
tagv2_tmax = TagV2.instance()
tagv2_t = TagV2.instance()


global tmin, tmax
tmin=50# minimum temperature
tmax=0 # maximum temperature


def update_tmin():
 global tmin
 if (tag.value().as_float()<tmin):
     tmin=tag.value().as_float()
     #  update user tag "min-temp"
     at = Time.now()
```

```
    tagv2_tmin.publish(

        "ioLogik-temperature-sensor",

        "min-temp",

        Tag(

            tag.value(),

            at,

            "C")

    )

    print 'tmin: '

    print (tmin)


def update_tmax():

global tmax

if (tag.value().as_float()>tmax):

    tmax=tag.value().as_float()

    #  update user tag "max-temp"

    at = Time.now()


    tagv2_tmax.publish(

        "ioLogik-temperature-sensor",

        "max-temp",

        Tag(

            tag.value(),

            at,

            "C")

    )

    print 'tmax: '

    print (tmax)
```

```
def update_t():
 #  update user tag "current-temp"
 at = Time.now()
 tagv2_t.publish(
     "ioLogik-temperature-sensor",
     "current-temp",
     Tag(
         tag.value(),
         at,
         "C")
 )
 print  (tag.value().as_float())


while (True):
 tag = modbus.read("ioLogik-ai0", "rtdi0")  # A temperature sensor is
connected to ioLogik AI0. The tag name "rtdi0" is pre-defined in template
"ioLogik-E1260" in EQUIPMENT TEMPLATE to read temperature value.
 update_tmin()
 update_tmax()
 update_t()
 time.sleep(5)
```

2. Log in to the UC-8112-LX via the serial console or SSH.
3. Create an executable Python script using the command:

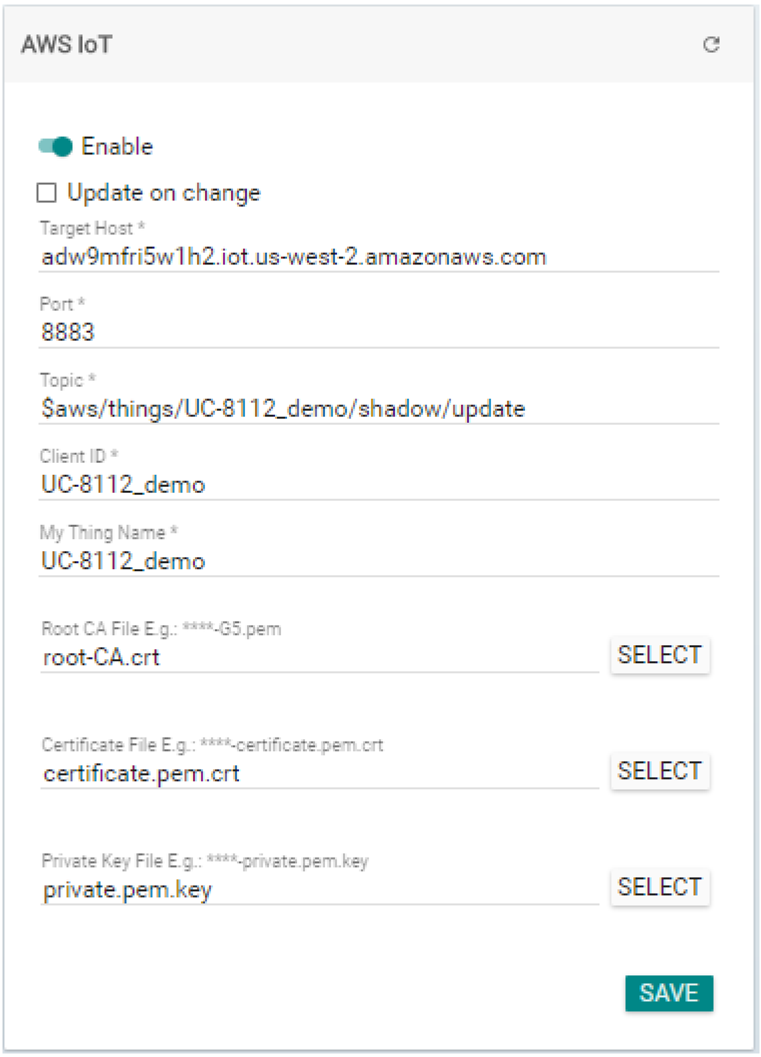
   ```
   moxa@Moxa:~$ chmod +x usertag-example.py
   ```

4. Run the Python program using the following command:

   ```
   moxa@Moxa:~$ ./usertag-example.py
   ```

3.4 Enabling the AWS IoT Client in the ThingsPro Gateway

Follow the instructions in the ThingsPro Software Suite User's Manual to configure the AWS IoT client on the ThingsPro Gateway. The **AWS IoT** Client configuration window can be accessed from the main menu and is listed under the **Applications** section. Make sure you enable the AWS IoT on ThingsPro Gateway as per the example shown in as shown in Figure 5.


AWS IoT

Enable: on
Update on change: off

Target Host *
adw9mfri5w1h2.iot.us-west-2.amazonaws.com

Port *
8883

Topic *
$aws/things/UC-8112_demo/shadow/update

Client ID *
UC-8112_demo

My Thing Name *
UC-8112_demo

Root CA File E.g.: ****-G5.pem
root-CA.crt

Certificate File E.g.: ****-certificate.pem.crt
certificate.pem.crt

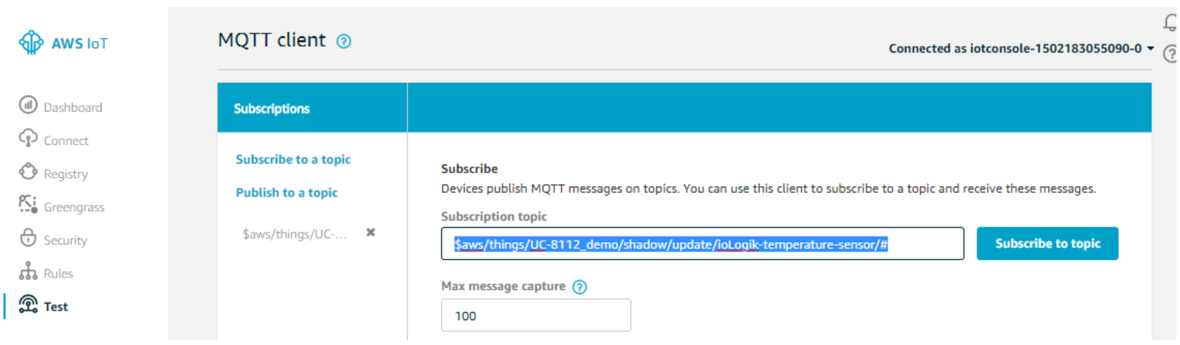
Private Key File E.g.: ****-private.pem.key
private.pem.key

Figure 5

3.5 Checking the Data on AWS IoT

You can subscribe the topic of your interest on the AWS IoT Console. In order to subscribe to the data acquired in the example illustrated in this document, type the topic, **$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#** in the **Subscription topic** field on the **MQTT client** page and click **Subscribe to topic**, as shown in Figure 6.

Note: The strings marked in red in the subscription topic will vary based on the AWS IoT *Thing Shadow* for your device.



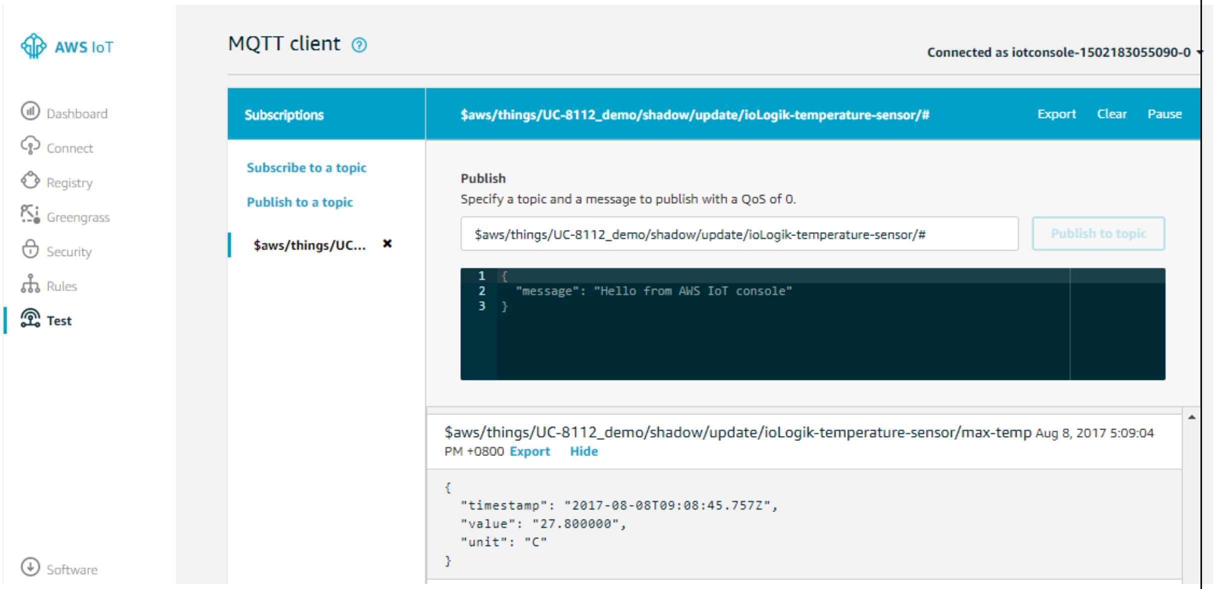
Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic: $aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#

Max message capture: 100

Figure 6

Once you subscribe to the topic, you will be able to view the max-temp, min-temp, and current-temp data on the AWS IoT console as shown in Figure 7, Figure 8, and Figure 9.



$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#

Specify a topic and a message to publish with a QoS of 0.

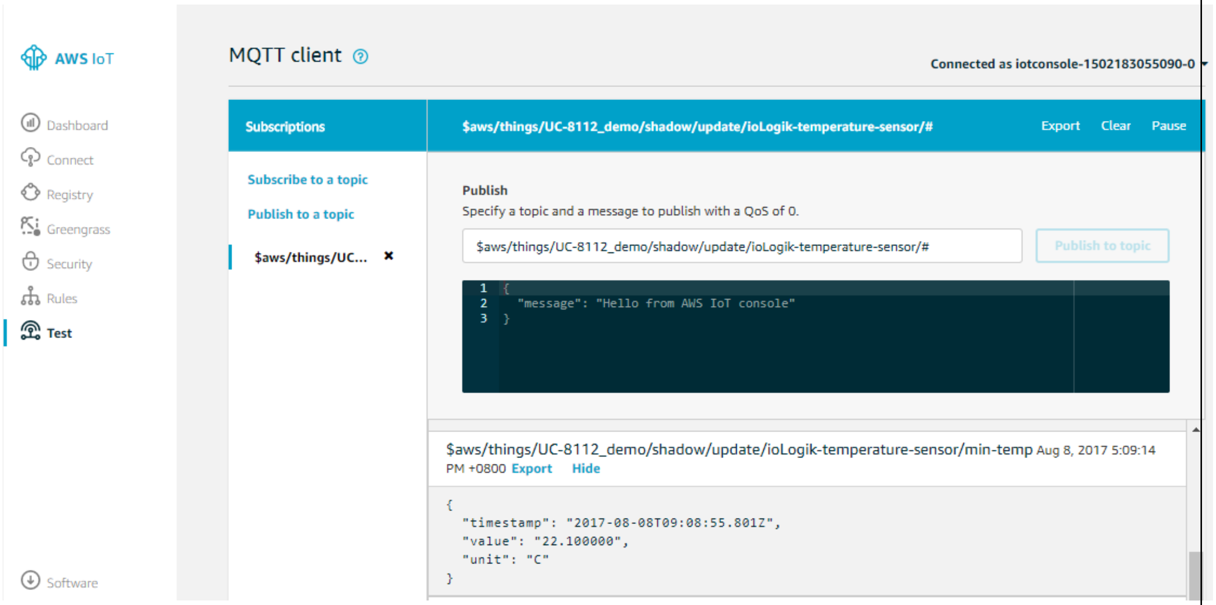
$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#

```
{
  "message": "Hello from AWS IoT console"
}
```

$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/max-temp Aug 8, 2017 5:09:04 PM +0800

```
{
  "timestamp": "2017-08-08T09:08:45.757Z",
  "value": "27.800000",
  "unit": "C"
}
```

Figure 7: User tag data (max-temp)

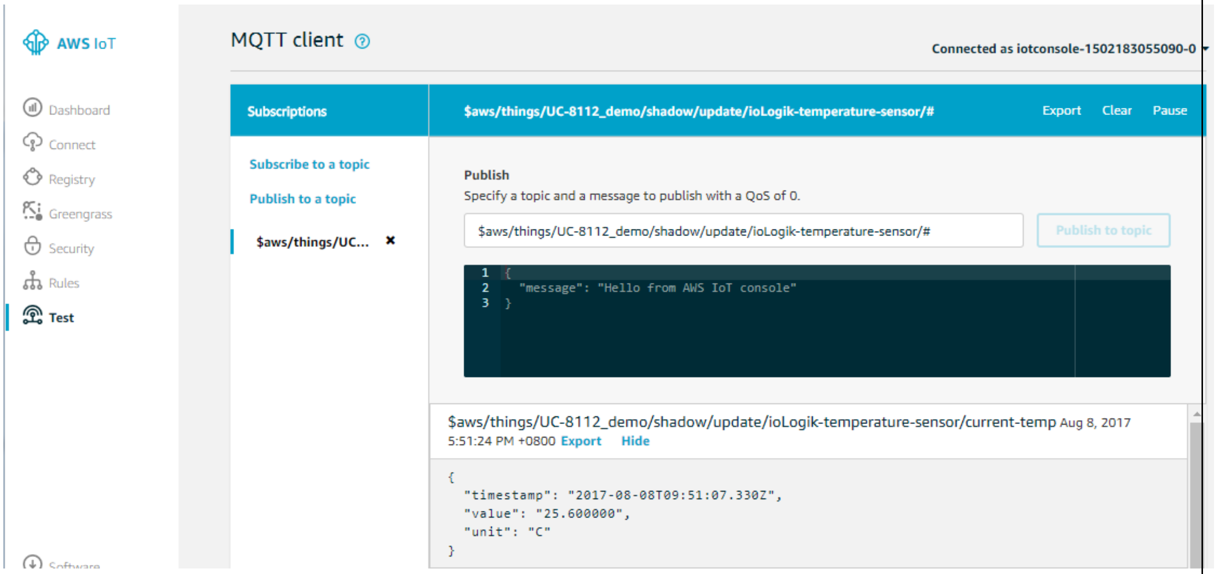

AWS IoT
Dashboard
Connect
Registry
Greengrass
Security
Rules
Test
Software
MQTT client
Connected as iotconsole-1502183055090-0
Subscriptions
Subscribe to a topic
Publish to a topic
$aws/things/UC...
$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#
Export
Clear
Pause
Publish
Specify a topic and a message to publish with a QoS of 0.
$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#
Publish to topic

```
{
  "message": "Hello from AWS IoT console"
}
```

$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/min-temp Aug 8, 2017 5:09:14 PM +0800 Export Hide

```
{
  "timestamp": "2017-08-08T09:08:55.801Z",
  "value": "22.100000",
  "unit": "C"
}
```

Figure 8: User tag data (min-temp)


AWS IoT
Dashboard
Connect
Registry
Greengrass
Security
Rules
Test
Software
MQTT client
Connected as iotconsole-1502183055090-0
Subscriptions
Subscribe to a topic
Publish to a topic
$aws/things/UC...
$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#
Export
Clear
Pause
Publish
Specify a topic and a message to publish with a QoS of 0.
$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/#
Publish to topic

```
{
  "message": "Hello from AWS IoT console"
}
```

$aws/things/UC-8112_demo/shadow/update/ioLogik-temperature-sensor/current-temp Aug 8, 2017 5:51:24 PM +0800 Export Hide

```
{
  "timestamp": "2017-08-08T09:51:07.330Z",
  "value": "25.600000",
  "unit": "C"
}
```

Figure 9: User tag data (current-temp)

3.5.1 Deploying Your Custom Program on ThingsPro Gateways

Once you have completed developing your own program development, follow the instructions given in the previous sections of this document to create the required custom tags. You can then run your program permanently using the **User Programs** function provided in the ThingsPro Gateway. Refer to chapter "Managing User Programs" in the *ThingsPro Software Suite User's Manual* for details.

If you want to deploy this program to a group of ThingsPro Gateways located at different field sites, you can export the settings from the ThingsPro Gateway which has the right configuration by using **ThingsPro Device Enablement Utility** and use the Restore function provided by the **ThingsPro Server** to push the configuration to the ThingsPro Gateways in the group. Refer to the chapter "Restoring the Configuration Setting for a Device Group" in the *ThingsPro Software Suite User's Manual* for details.

4 Additional Reading

- ThingsPro Programmer's Guide
  https://www.moxa.com/support/DownloadFile.aspx?type=support&id=14768
- Latest version of the *ThingsPro Software Suite User's Manual* available at:
  https://www.moxa.com/support/sarch_result.aspx?prod_id=5151&type_id=7&type=doc