Getting Started with the Google Cloud SDK on ThingsPro 2.0 to Publish Modbus Data and Subscribe to Cloud Services

Moxa Technical Support Team
support@moxa.com

Contents

 Released on October 13, 2017

About Moxa

Moxa is a leading provider of edge connectivity, industrial computing, and network infrastructure solutions for enabling connectivity for the Industrial Internet of Things. With over 30 years of industry experience, Moxa has connected more than 50 million devices worldwide and has a distribution and service network that reaches customers in more than 70 countries. Moxa delivers lasting business value by empowering industry with reliable networks and sincere service for industrial communications infrastructures. Information about Moxa's solutions is available at info@moxa.com.

How to Contact Moxa

Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

MOXA
Reliable Networks ▲ Sincere Service

Moxa Tech Note **Getting Started with the Google Cloud SDK on ThingsPro 2.0**

1 Introduction

The purpose of this tech note is to provide you with step-by-step instructions on how to run the Google Cloud SDK on ThingsPro.

The instructions in this document are suitable for developers who are already familiar with the Google Cloud Platform console. To learn more about this technology, please visit: https://cloud.google.com/solutions/iot-overview

ThingsPro Gateway, which is part of the ThingsPro Suite, is a software package that acts as a protocol gateway between the information technology (IT) and operation technology (OT) worlds. On the IT side, ThingsPro Gateway supports a variety of IT protocols such as Microsoft Azure, Wonderware Online Insight, AWS, Generic MQTT, HTTP, Sparkplug, and the M2X cloud server. On the OT side, ThingsPro Gateway supports the Modbus protocol and provides a programming environment for users to develop their own proprietary OT protocols.

2 Application Scenario

Moxa's UC-8112-LX, an embedded computer with the Debian Linux distribution preinstalled, is used as the gateway computer. The ThingsPro Gateway tool is then installed on this UC-8112-LX gateway computer. We will leverage ThingsPro's Modbus engine to read data from a Modbus device and transmit this data to the Google Cloud Platform so that this data can be used by IT applications.

Figure 1 shows an overview of the Google Cloud Platform architecture. Moxa's ThingsPro Gateway, in its role as the "gateway device", reads Modbus data from "Constrained Devices". The data is then published to the "Ingestion" function block in the Google Cloud Platform, which uses the "Cloud Pub/Sub" function to manipulate data and publish it to the "Pipelines" function block. To learn more about the publisher/subscriber model of the Google Cloud platform, refer to https://cloud.google.com/solutions/iot-overview#pipeline_processing_tasks.

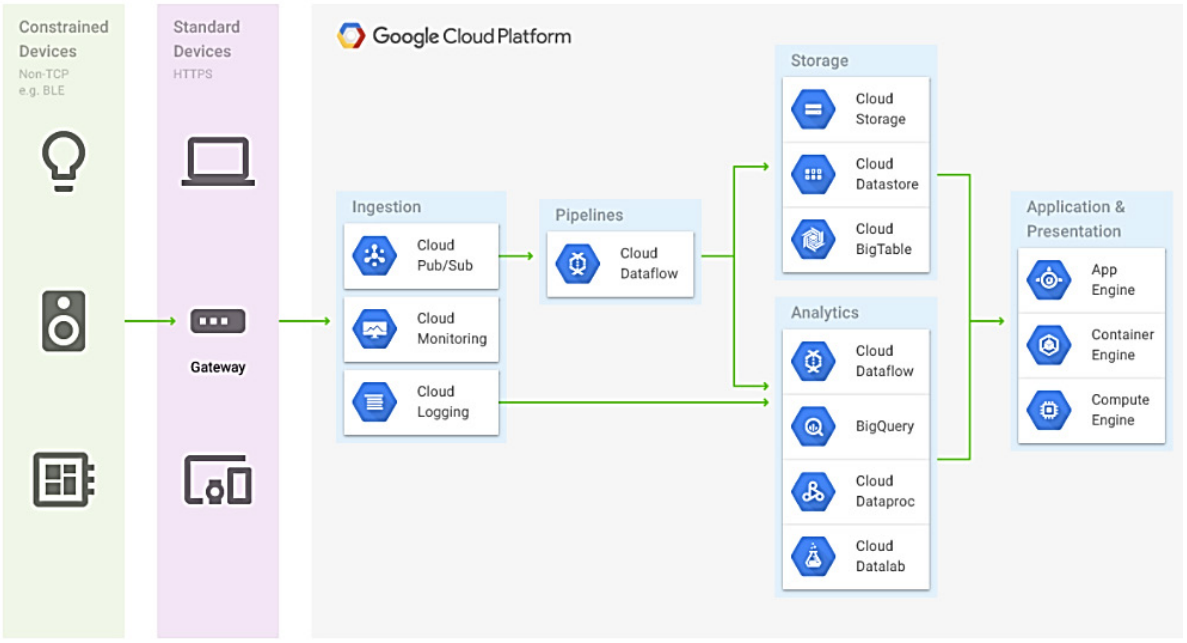


Figure 1: Google Cloud Platform Architecture

3 Prerequisites

- An eligible* Moxa device (e.g., UC-8112-LX) with ThingsPro Gateway software installed
- A Google Cloud Platform user account

*Refer to the ThingsPro datasheet for a list of eligible Moxa devices

4 Solution

To transfer Modbus data from ThingsPro to the Google Cloud platform, do the following:

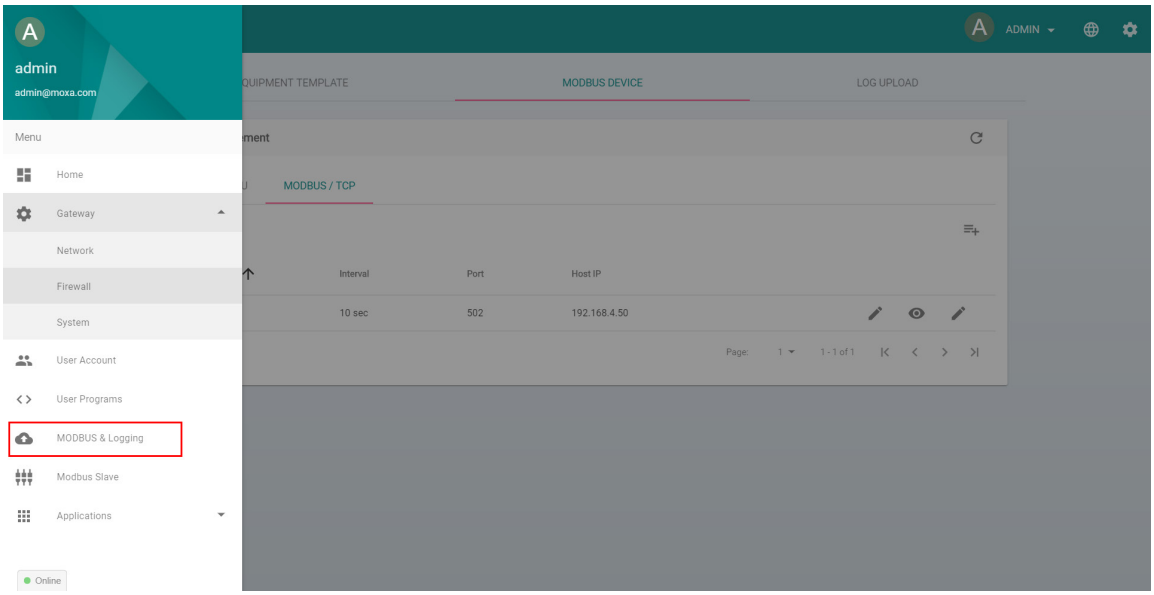
1. Set up ThingsPro Gateway to read Modbus data from "Constrained Devices"
2. Install the Google Cloud SDK on ThingsPro Gateway
3. Publish test data to the Google Cloud Platform Pub/Sub Service
4. Publish ThingsPro Modbus data to the Google Cloud Platform Pub/Sub service

4.1 Set up ThingsPro Gateway to read Modbus data from "Constrained Devices"

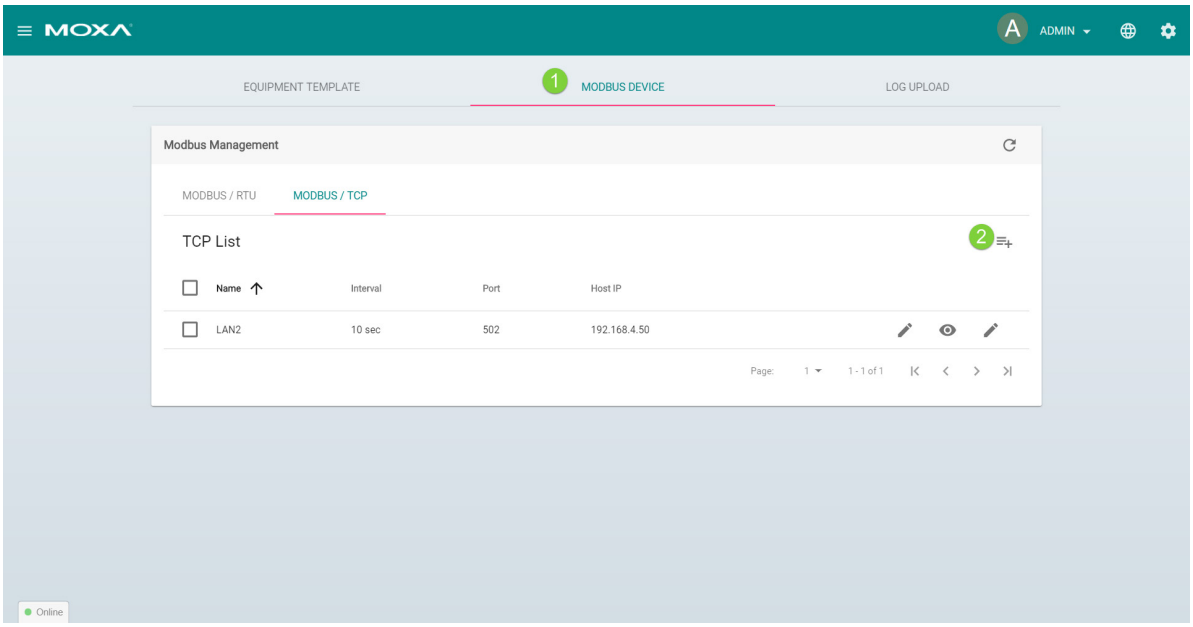
You must first connect your devices to the ThingsPro Gateway. In our example, we use a Moxa ioLogik-E1242 device.

To establish a connection between the ThingsPro Gateway and the ioLogik-E1242 device, do the following:

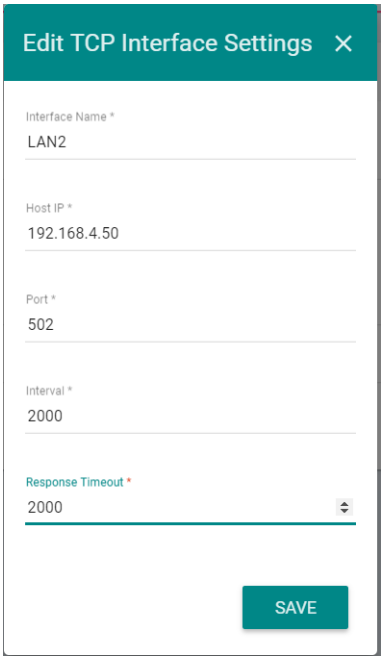
1. Log in to the ThingsPro Gateway web console.
2. Click Main Menu, and select **MODBUS & Logging**.



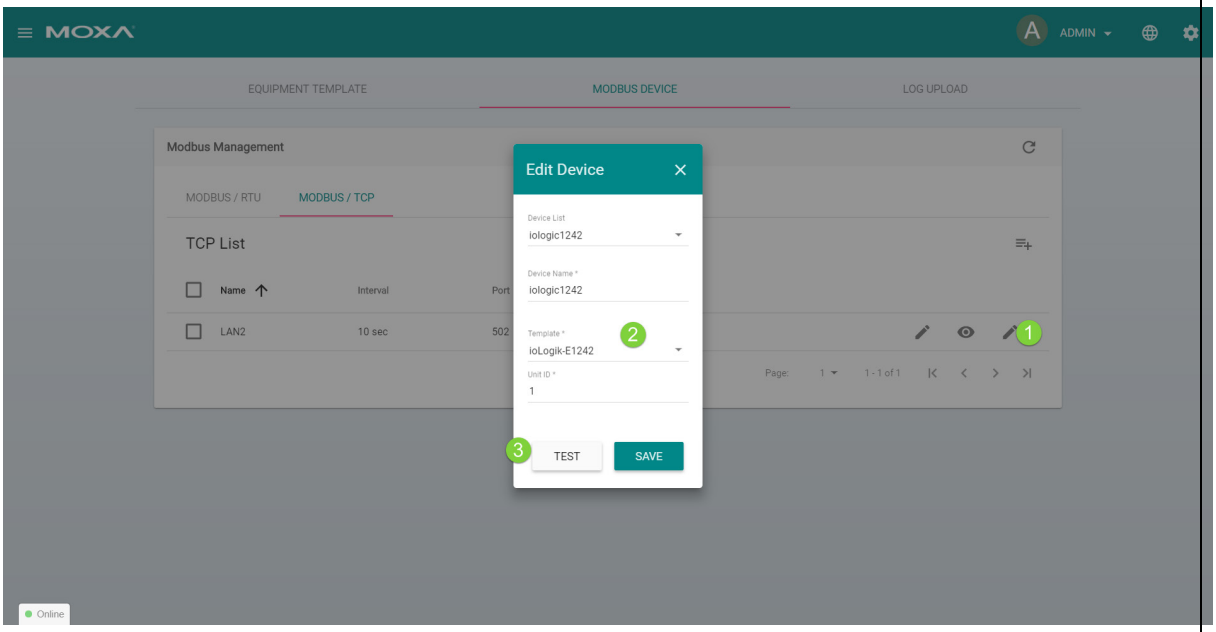
3. Create a new Modbus TCP polling rule for your device.



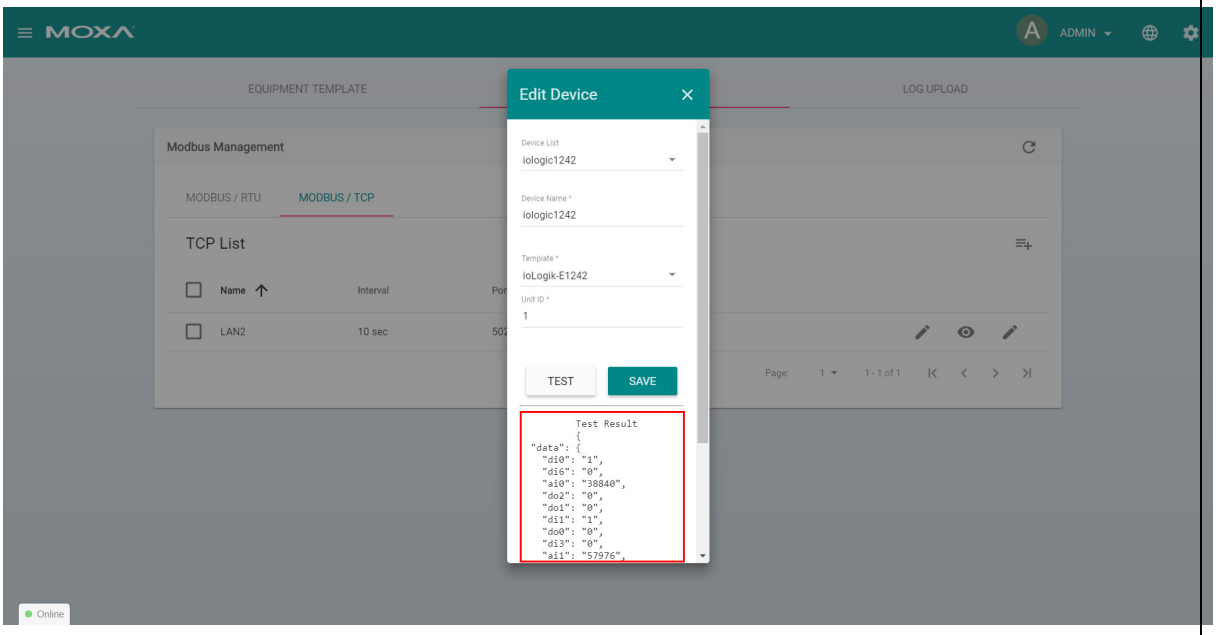
In this example, the ioLogik-E1242 Modbus device is set up with **192.168.4.50** as the IP address. The query interval for the device is set to **2000** milliseconds.



4. Apply the Modbus TCP polling engine rule that you just created to the predefined ioLogik-E1242 Modbus template in ThingsPro Gateway.



5. Click **TEST** to verify that ThingsPro Gateway is polling data from the Modbus device.



If the test result shows no errors, the Modbus device is set up correctly.

If the test result shows errors, it could be because of one of the following:

- Modbus device is set up with an incorrect IP address
- Router firewall is blocking the Modbus protocol
- Modbus device allows only one TCP session

4.2 Install the Google Cloud SDK on ThingsPro Gateway Device

The Google Cloud SDK is available for download from the Google website at:
https://cloud.google.com/pubsub/docs/quickstart-client-libraries

You will need a Google account to use the Pub/Sub service. Once you are set up with an account, follow the instructions given in the "Before you begin" section shown in Figure 2.

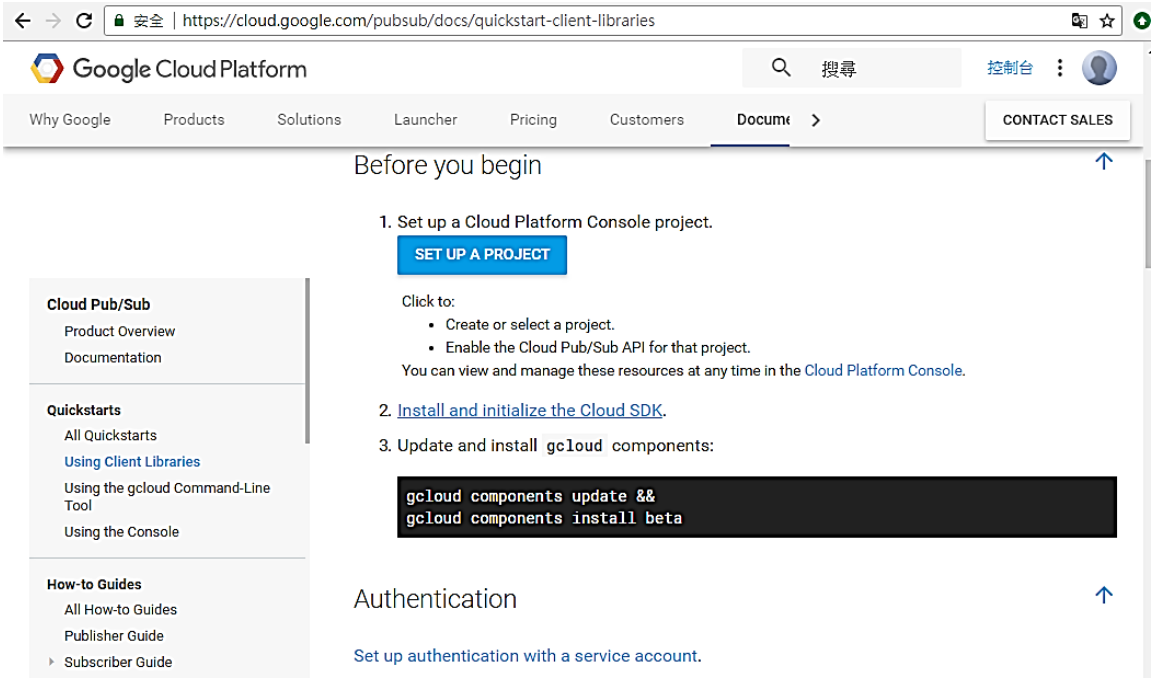


Figure 2: Setting Up the Google Cloud SDK

Step 1: Set up a project

In this example, we create a new project called "My Project". The project ID "my-project-1504251921418" was assigned automatically when the project was created.

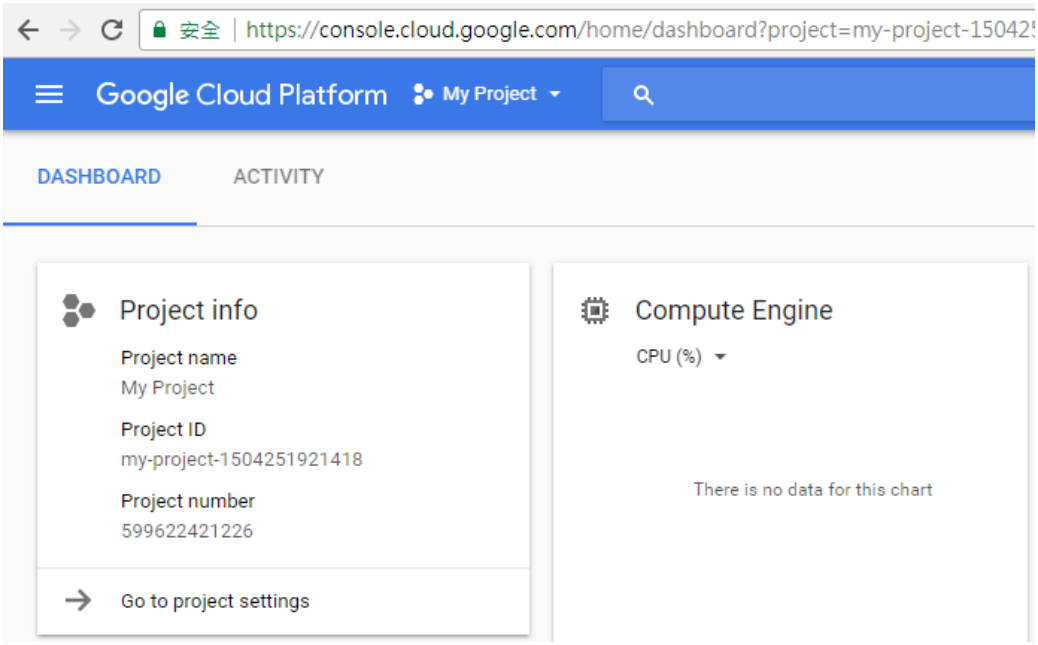


Figure 3: Setting Up a Project

STEP 2: Install and initialize the Google Cloud SDK

1. Connect the console cable to the ThingsPro Gateway Device and set the baudrate to **115200/8N1**.
2. Log in with the username **moxa** and password **moxa**.
3. Switch to the **root** user using the command:
   # sudo su
   The default password is "moxa".
4. Install the Google Cloud SDK.
   Make sure that your ThingsPro Gateway can access the Internet and run the following commands:

```
# apt-get update && apt-get install lsb-release
# export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
# echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" |
sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
# curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -
# sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Wait for about 10 minutes for the installation process to complete. When the installation is complete, run the following command:

```
# pip install virtualenv
```

5. Once you set up the Google Cloud SDK, run the following commands on the ThingsPro Gateway console to download the Python pub/sub example code.

```
# apt-get install git -y
# cd /home/moxa
# mkdir Google
# cd Google
# git clone \
https://github.com/GoogleCloudPlatform/python-docs-samples.git
# cd python-docs-samples/pubsub/cloud-client
```

For details, visit:
https://github.com/GoogleCloudPlatfor/python-docs-samples/tree/master/pubsub/cloud-client

4.3 Publish test data to the Google Cloud Platform Pub/Sub Service and subscribe to data from the service

Follow the instructions given below to publish data to the Google Cloud Platform Pub/Sub service and to subscribe to data from the service.

1. Create a new topic in the Google Cloud Platform Pub/Sub service.
   To be able to publish/subscribe data to/from the Google Pub/Sub service, you must first create a new topic in the Google Cloud Platform Pub/Sub service web console, as shown in Figure 4.

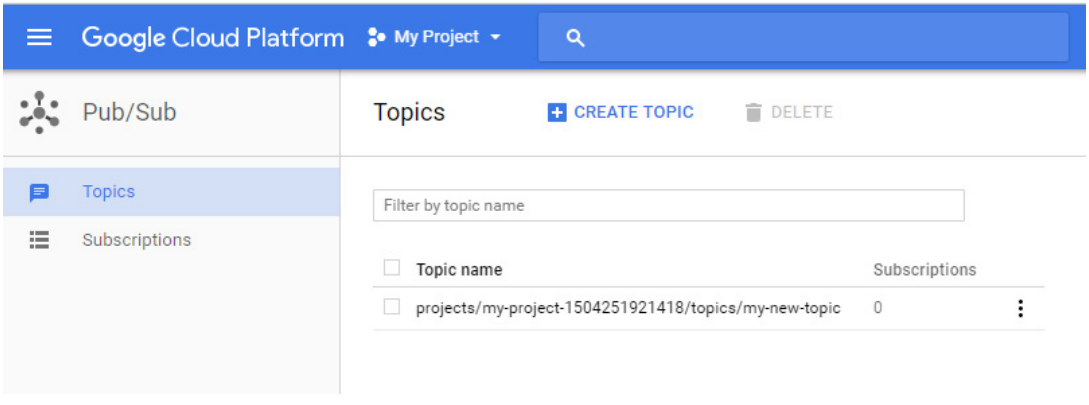


Figure 4 Creating a Topic

Once you have successfully created a new topic (`my-new-topic`, in this example), you will be able to publish and subscribe messages to and from that topic.

2. Set up an authentication process for the cloud client.
   In this example, we use ThingsPro's default credentials and Google Cloud SDK to set up the authentication process. For details, visit: https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/pubsub/cloud-client

   a. Run the following command:
      # gcloud auth application-default login

      A secure link to the **Sign in** page is displayed as follows:

```
root@Moxa:/home/moxa# gcloud auth application-default login
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?redirect_uri=urn%3Aietf%3Awg%3Aoau
th%3A2.0%3Aoob&prompt=select_account&response_type=code&client_id=764086051850-6
qr4p6gpi6hn506pt8ejuq83di341hur.apps.googleusercontent.com&scope=https%3A%2F%2Fw
ww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Faut
h%2Fcloud-platform&access_type=offline
```

   b. Access the link above to view the authorization code.

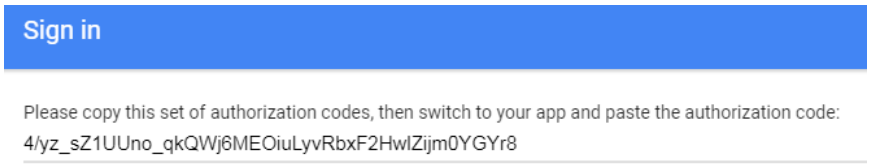


Figure 5 Viewing the Authorization Code

c. Copy the authorization code and paste it in the console.
After you have entered the verification code in the console, the credentials will be generated and stored in the ThingsPro Gateway for Pub/Sub samples as shown in Figure 6.

```
root@Moxa:/home/moxa# gcloud auth application-default login
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?redirect_uri=urn%3Aietf%3Awg%3Aoau
th%3A2.0%3Aoob&prompt=select_account&response_type=code&client_id=764086051850-6
qr4p6gpi6hn506pt8ejuq83di341hur.apps.googleusercontent.com&scope=https%3A%2F%2Fw
ww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Faut
h%2Fcloud-platform&access_type=offline


Enter verification code: 4/yz_sZ1UUno_qkQWj6MEOiuLyvRbxF2Hwl2ijm0YGYr8

Credentials saved to file: [/root/.config/gcloud/application_default_credentials
.json]

These credentials will be used by any library that requests
Application Default Credentials.
root@Moxa:/home/moxa#
```

Figure 6 Pasting the Authorization Code in the Google Cloud Platform Console

3. Install the Python packages required for the virtual environment using the following commands:

```
# cd /home/moxa/Google/python-docs-samples/pubsub/cloud-client
# virtualenv env
# source env/bin/activate
 (env) # pip install -r requirements.txt
```

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# pip i
nstall -r requirements.txt
Collecting google-cloud-pubsub==0.28.2 (from -r requirements.txt (line 1))
  Downloading google_cloud_pubsub-0.28.2-py2.py3-none-any.whl (80kB)
    100% |████████████████████████████████| 81kB 651kB/s
Collecting googleapis-common-protos[grpc]<2.0dev,>=1.5.2 (from google-cloud-pubs
ub==0.28.2->-r requirements.txt (line 1))
  Downloading googleapis-common-protos-1.5.2.tar.gz
Collecting google-gax<0.16dev,>=0.15.13 (from google-cloud-pubsub==0.28.2->-r re
quirements.txt (line 1))
  Downloading google-gax-0.15.14.tar.gz (109kB)
    100% |████████████████████████████████| 112kB 381kB/s
Collecting psutil<6.0dev,>=5.2.2 (from google-cloud-pubsub==0.28.2->-r requireme
nts.txt (line 1))
  Downloading psutil-5.2.2.tar.gz (348kB)
    100% |████████████████████████████████| 358kB 332kB/s
Collecting google-cloud-core<0.28dev,>=0.27.0 (from google-cloud-pubsub==0.28.2-
>-r requirements.txt (line 1))
  Downloading google_cloud_core-0.27.1-py2.py3-none-any.whl (50kB)
    100% |████████████████████████████████| 51kB 857kB/s
Collecting grpcio<2.0dev,>=1.0.2 (from google-cloud-pubsub==0.28.2->-r requireme
nts.txt (line 1))
```

4. Subscribe to the topic that you created (**my-new-topic**) using the command:

```
(env) # python ./subscriber.py my-project-1504251921418 create \
        my-new-topic my-sub
```

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# pytho
n ./subscriber.py my-project-1504251921418  create my-new-topic my-sub
Subscription created: name: "projects/my-project-1504251921418/subscriptions/my-
sub"
topic: "projects/my-project-1504251921418/topics/my-new-topic"
push_config {
}
ack_deadline_seconds: 10

(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client#
```

You are now ready to receive messages through your subscription ("my-sub") to the topic "my-new-topic".

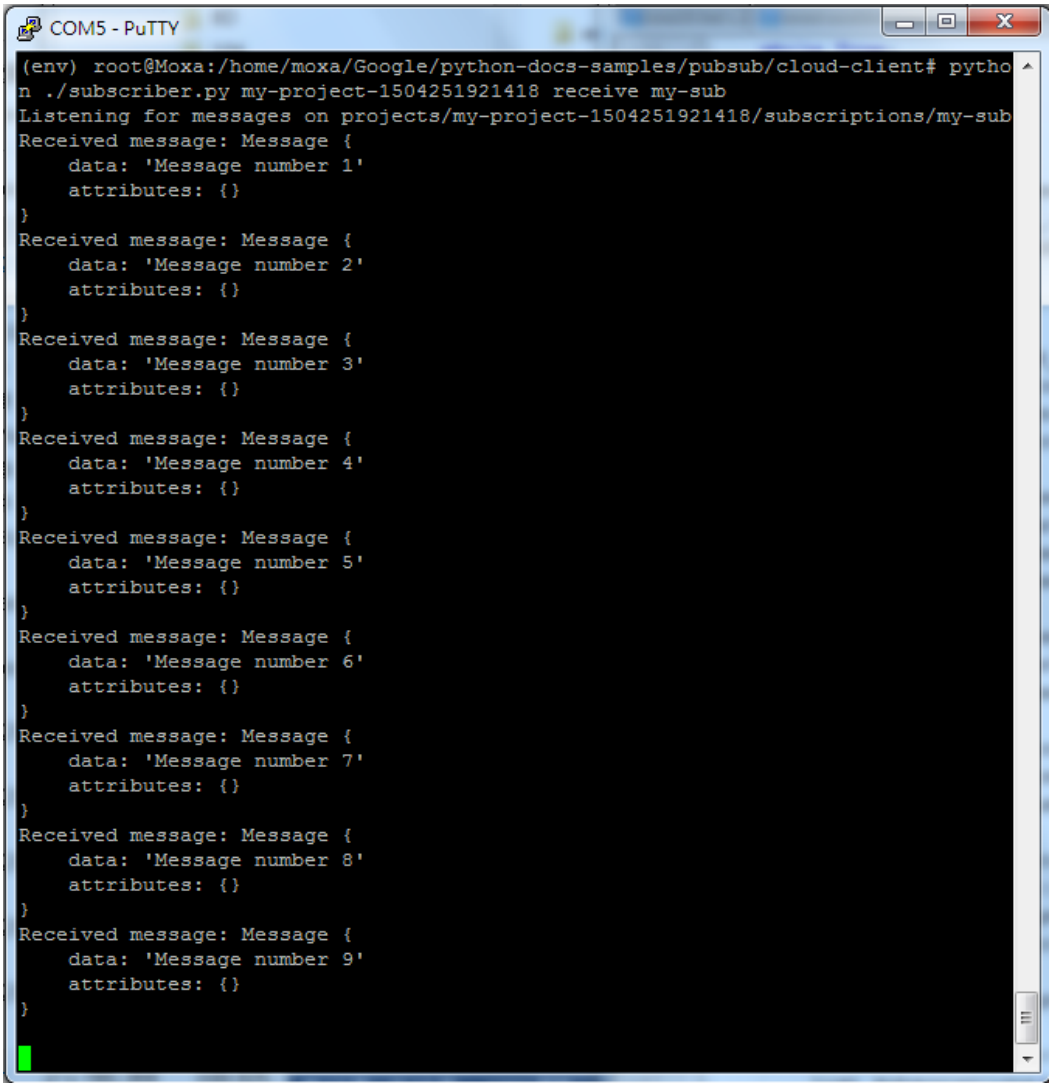
5. To publish messages to the "my-new-topic" topic, use the command:

```
(env) # python ./publisher.py my-project-1504251921418 publish \
        my-new-topic
```

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# pytho
n ./publisher.py my-project-1504251921418 publish my-new-topic
Published messages.
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client#
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client#
```

6. To read the messages that you just published to the topic, use the command:

```
(env) #  python ./subscriber.py my-project-1504251921418 receive my-sub
```



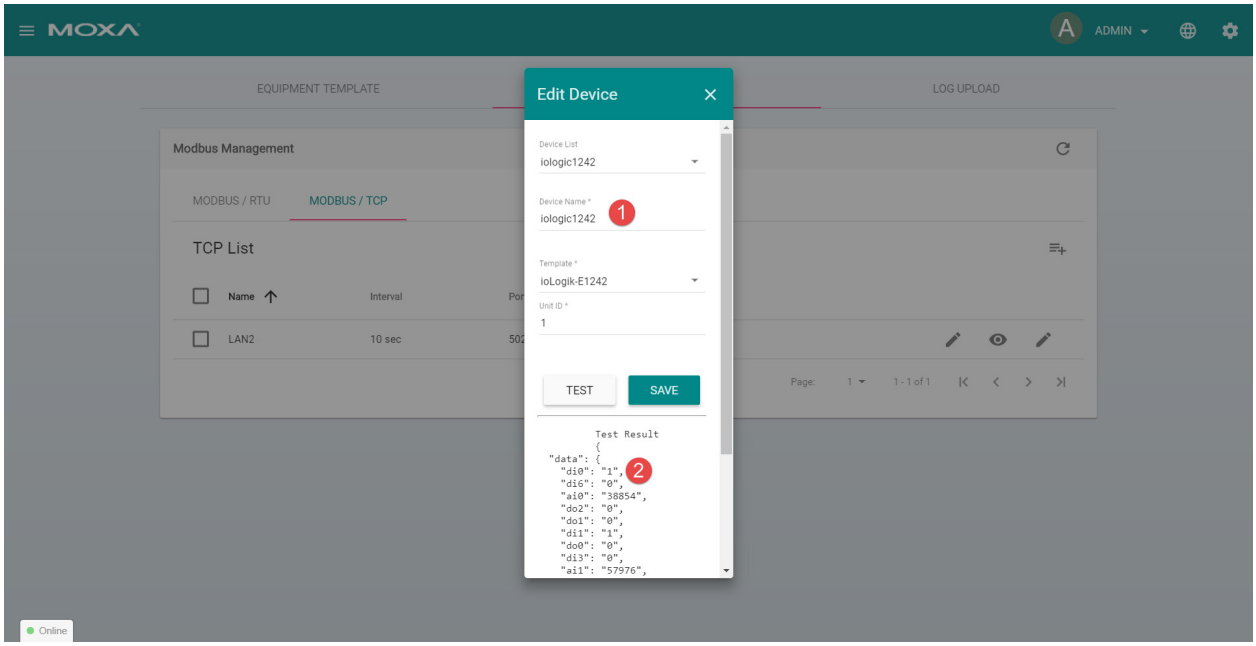
4.4 Publish ThingsPro Modbus data to the Google Cloud Platform Pub/Sub service and subscribe to data from the service

In this section, we use the ThingsPro example code (included in the UC-8112-LX platform) to read Modbus data queried by the ThingsPro Modbus engine that we set up in the previous section.

1. Copy the ThingsPro Modbus example code (**`subscribe.py`**) to the current directory and rename it as **`thingspro.py`**.

```
jimmy@jimmy-desktop: ~
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# cp /usr/share/mxdaf/libmxidaf_py/sample/subscribe.py thingspro.py
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client#
```

2. Change the Modbus query example code based on your ThingsPro Modbus TCP web-console setup.
   Use the Device Name "iologic1242" (1) and tag name "di0" (2) values in the Modbus TCP web-console setup to edit the **`thingspro.py`** program.



```
jimmy@jimmy-desktop: ~
import time
from libmxidaf_py import TagV2

tagv2 = TagV2.instance()


def on_tag_callback(equipment_name, tag_name, tag):
    print "{}:{}:{}:{}:{}".format(
        equipment_name,
        tag_name,
        tag.at(),
        tag.value(),
        tag.unit())

tagv2.subscribe_callback(on_tag_callback)

tagv2.subscribe(
    "iologic1242",
    "di0")

while(True):
    time.sleep(0.1)
~
~
~
```

3. Run the **thingspro.py** program.
   The ThingsPro Modbus engine will poll data periodically using the specified interval.

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# python thingspro.py
iologic1242:di0:2017-09-18T10:03:44.008Z:1:
iologic1242:di0:2017-09-18T10:03:46.003Z:1:
```

4. Merge the **publisher.py** example code with the **thingspro.py** program as shown below:

```
import argparse 1
import time
from libmxidaf_py import TagV2
from google.cloud import pubsub_v1 2

tagv2 = TagV2.instance()

def publish_messages(project, topic_name, data): 3
        """Publishes multiple messages to a Pub/Sub topic."""
        publisher = pubsub_v1.PublisherClient()
        topic_path = publisher.topic_path(project, topic_name)

    # for n in range(1, 10):
    #     data = u'Message number {}'.format(n)
        # Data must be a bytestring
        data = data.encode('utf-8')
        publisher.publish(topic_path, data=data)

        print('Published messages.')

def on_tag_callback (equipment_name, tag_name, tag):
        print "{}:{}:{}:{}:{}".format(
                equipment_name,
                tag_name,
                tag.at(),
                tag.value(),
                tag.unit())

    publish_messages("my-project-1504251921418", "my-new-topic", "tag: {}, timestamp: {}, value: {}".format(tag_name, tag.at(), tag.value())) 4

tagv2.subscribe_callback(on_tag_callback)

tagv2.subscribe(
        "iologic1242",
        "di0")

while(True):
        time.sleep(0.1)
```

5. Run the **`thingspro.py`** program.
   Each time the Modbus engine reads data, it publishes it to the Google Pub/Sub service (see 3 above).
   The following image shows the data payloads being sent.

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client# python thingspro.py
iologic1242:di0:2017-09-19T02:46:06.007Z:1:
Published messages.
iologic1242:di0:2017-09-19T02:46:08.000Z:1:
Published messages.
iologic1242:di0:2017-09-19T02:46:10.007Z:1:
Published messages.
```

6. To read data from the Google Pub/Sub service, run the **`subscriber.py`** program.

```
(env) root@Moxa:/home/moxa/Google/python-docs-samples/pubsub/cloud-client#  python ./subscriber.py my-project-1504251921418 receive my-sub
Listening for messages on projects/my-project-1504251921418/subscriptions/my-sub
Received message: Message {
    data: 'tag: di0, timestamp: 2017-09-19T02:46:10.007Z, val...'
    attributes: {}
}
Received message: Message {
    data: 'tag: di0, timestamp: 2017-09-19T02:46:08.000Z, val...'
    attributes: {}
}
Received message: Message {
    data: 'tag: di0, timestamp: 2017-09-19T02:46:06.007Z, val...'
    attributes: {}
}
```

The subscriber.py program is used to confirm that your data is being published to the Google Pub/Sub service. Once your data is available to the Pub/Sub service, it can be used by Google Cloud Data Analytics services through the Data Flow service.

5 Additional Reading

https://console.cloud.google.com

https://cloud.google.com/solutions/iot-overview#pipeline_processing_tasks

https://cloud.google.com/pubsub/docs/quickstart-client-libraries

https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/pubsub/cloud-client

http://www.moxa.com/product/uc-8100.htm

https://www.moxa.com/Event/industrial-computers/thingspro-data-acquisition-device-management-platform/resources.htm