Driver Installation and Configuration Guide

Modification date	Revision Record
2020.05.03	The method of driver installation on Windows, Linux and Android 4.4/5.x operating system.
2021.06.06	Added the method of driver installation on Android 6.x/7.x/8.x operating system.
2024.08.07	Added the description of AWS IoT service provided by UCloudLink and testing methods.
2024.09.04	Added the method and testing instructions for using your own AWS IoT service.
Notes: This document is designed for the operating system of Windows, Linux and Android 4.4 and higher	

Contents

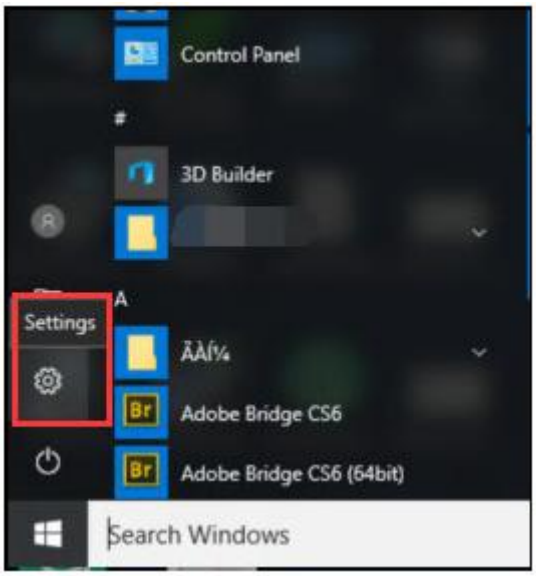
1. Driver Installation on Windows

1.1 Disable Driver Signature Enforcement

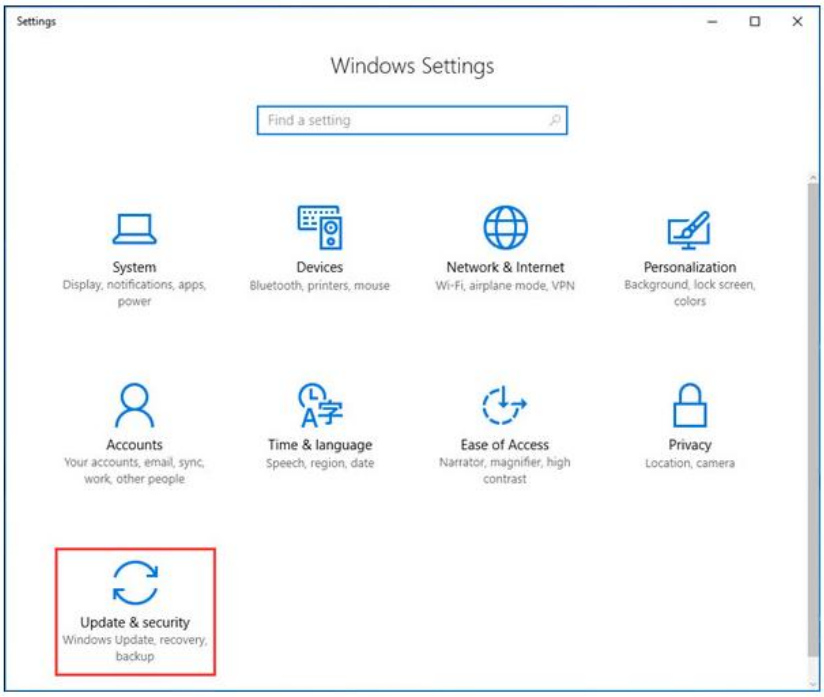
If you want to install a driver not signed by Microsoft on Windows operating system, you'll need to disable Driver Signature Enforcement, or the installation could fail.

1.1.1 Windows 10 Operating System

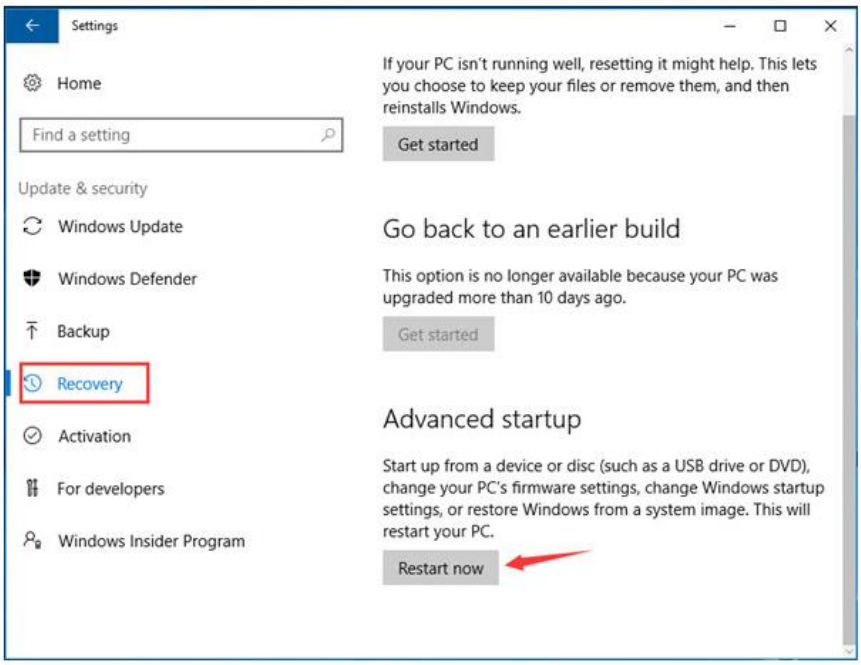
Step 1: Press Win key and click **Settings** icon. You can go into **Settings**.



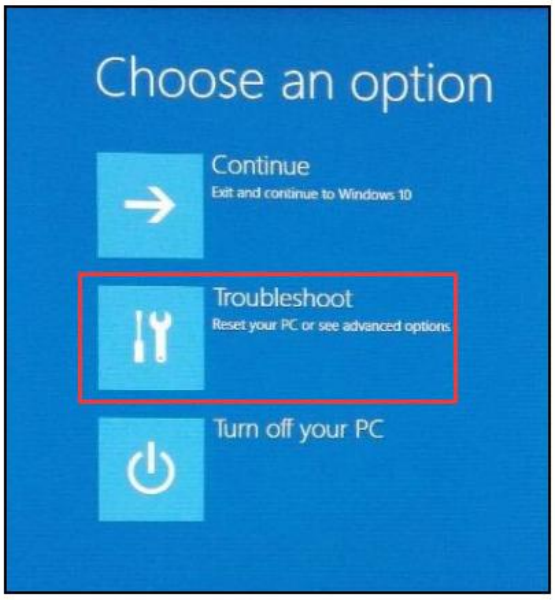
Step 2: In Settings, you can catch the **Update and Security** option in here. And choose it.



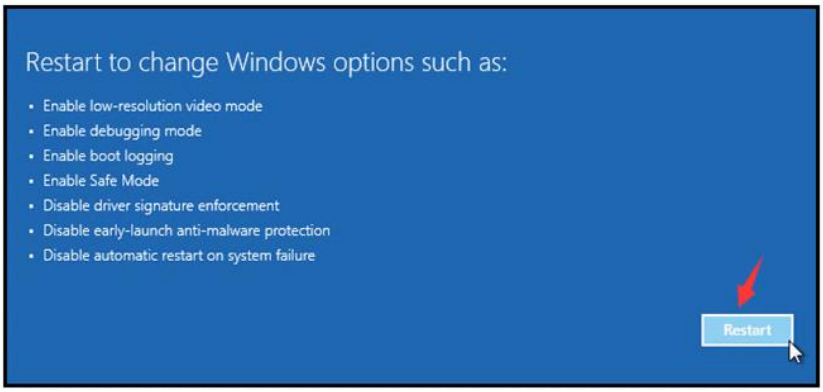
Step 3: Please click the **Recovery** option on the left menu. You can find the details on the right. Please catch **Advanced startup** option and hit **Restart Now**.



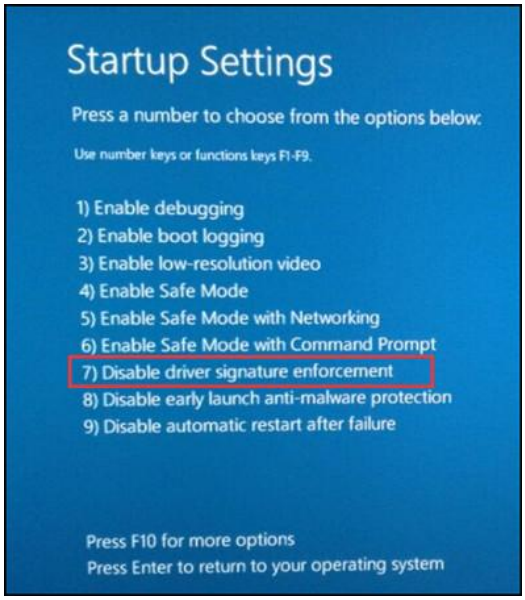
Step 4: Please wait a few seconds, you can enter **Choose an Option** interface. Please select **Troubleshoot** option.



Step 5: In **Troubleshoot**, you can select these options step by step: **Advanced options > Startup Settings > Restart**.

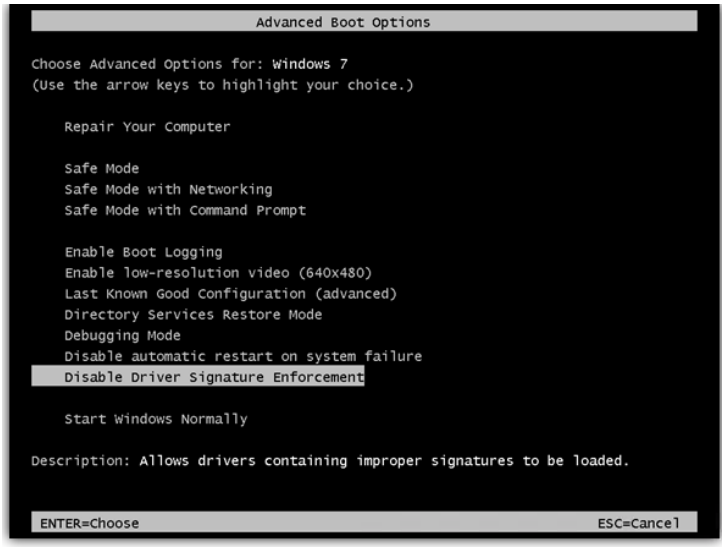


Step 6: You can go into the **Startup Settings** interface. There are 9 features in the list. Please press functions key **F7** or press the **Number Key 7** to choose the **Disable driver signature enforcement**.



1.1.2 Windows 7 64 Operating System

During booting of PC, press "**F8**" key continuously until you get the **Advanced Boot Options** menu, then select "**Disable Driver Signature Enforcement**".



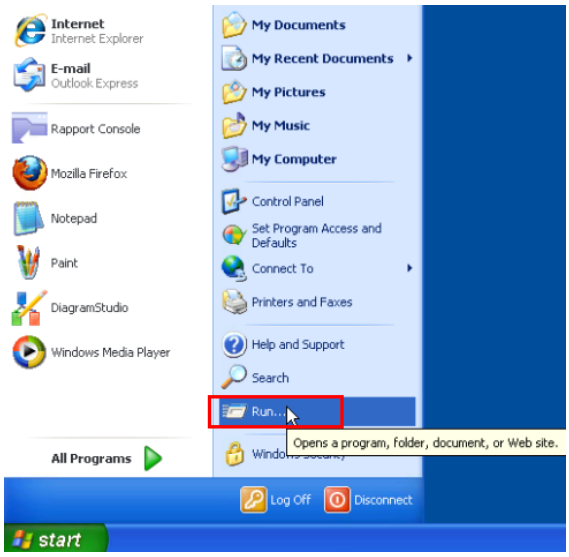
Notes: This solution is only used to disable **Driver Signature Enforcement** just once

1.1.2 Windows 7 32 Operating System

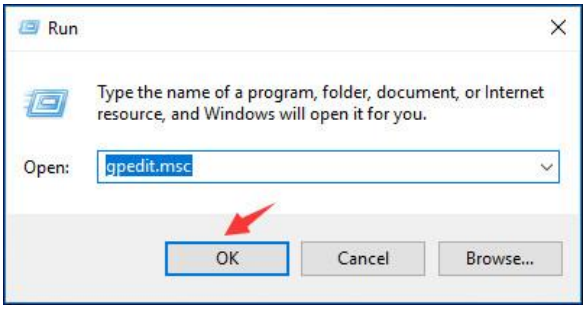
Disabling driver enforcement works differently on Windows 7 32 and Windows 64.

The method of disabling driver enforcement on Windows 7 32 is the same as Windows XP. To setup, follow the steps as below:

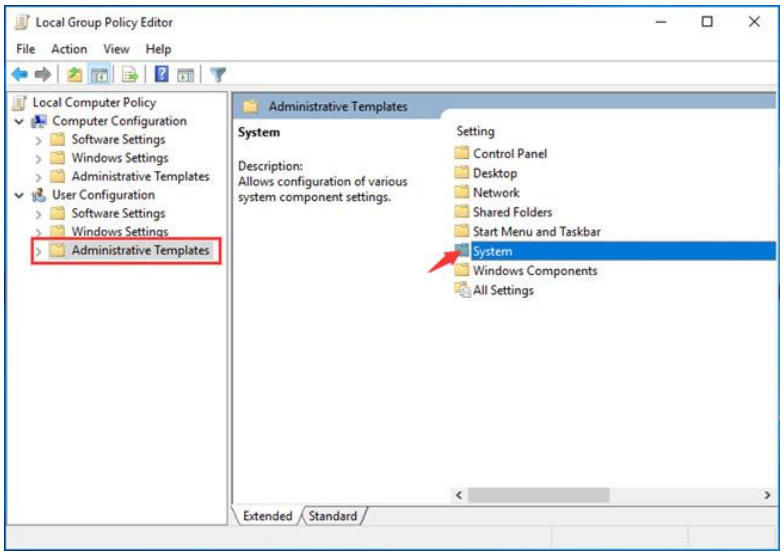
Step 1: Go to **Start** menu and choose **Run**



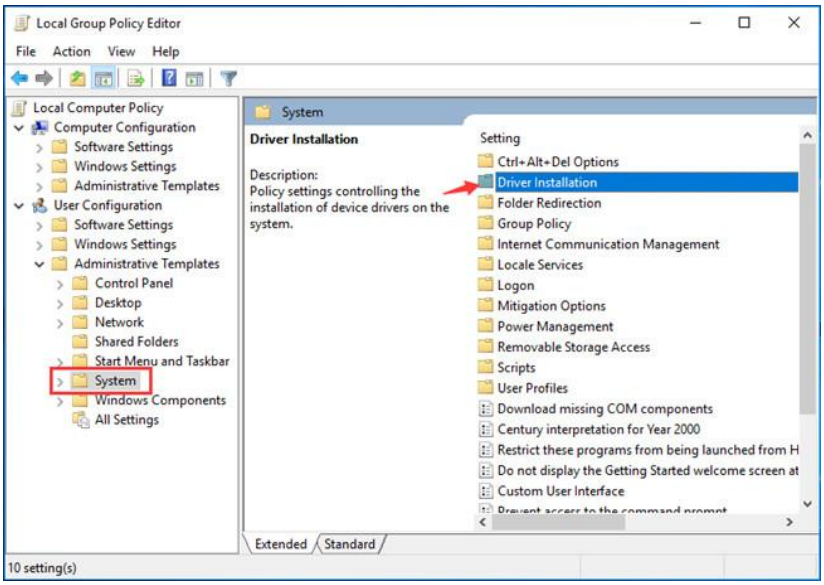
Step 2: Enter ***gpedit-msc*** into the box. You can go into **Local Group Policy Editor**.



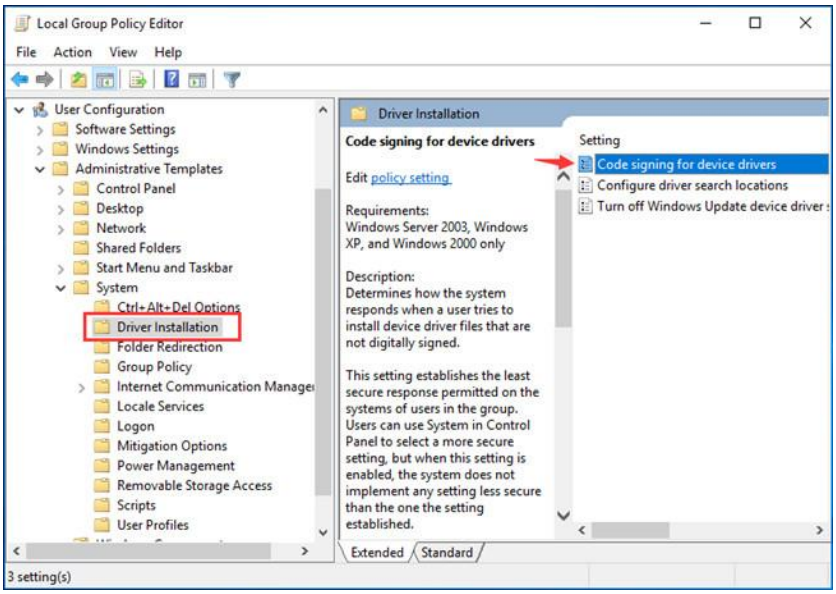
Step 2: In **Local Group Policy Editor**, please select **Administrative Templates** on the left category. You can find the **Setting** list on the right. Please double-click on **System** option.



Step 3: In **System**, there are some programs setting category on the right. Please choose **Driver Installation** and double-click it.

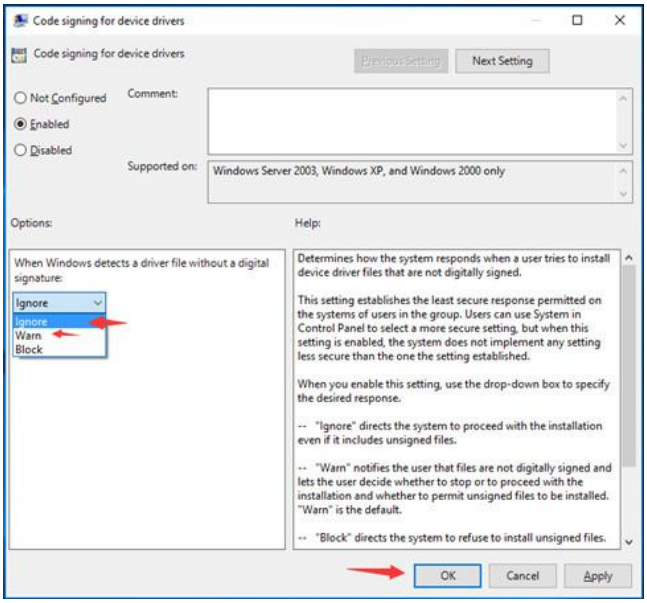


Step 4: Please catch the **Code signing for device drivers** in the **Driver Installation** and double-click it.



Step 5: At this time, you can make some settings in **Code signing for device drivers** in order to Windows detects a driver file without a digital signature.

Please hit the **Enabled** button. You can select the **Ignore** or **Warn** in the drop-down menu in **Options** box. Then, please don't forget press **OK**. The configuration is finished.



1.2 Install Driver

Proceed to install your driver.

For Windows 32 operating system, run **32install.exe**
For Windows 64 operating system, run **64install.exe**

During the installation procedure, Windows will inform you that can't verify the publisher of this driver software. At this point, ignore the warning message and choose "**Install this driver software anyway**" to complete the installation.

2. Driver Installation on Linux

Module M2 can be recognized as RNDIS, high-capacity storage device and AT commands port in Host.

2.1 RNDIS Host Configuration

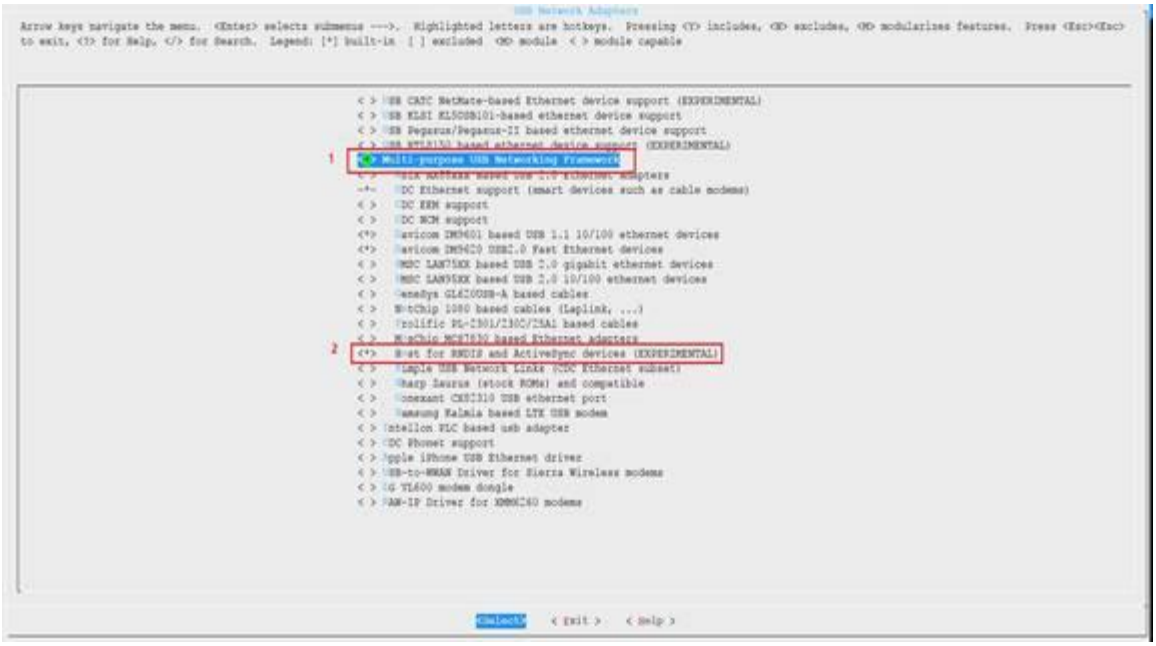
To configure the Host for RNDIS, refer to the following steps:

Go to the root directory of the Linux Kernel source code and execute the command:

 # make menuconfig

Open the Linux Kernel Configuration Menu

Go to Device Drivers > Network device support > USB Network Adapters > Multi-purpose USB Networking Framework



Select **Host for RNDIS and ActiveSync device** and press save key. Then the Kernel is configured successfully.

 # make zImage

Compile its corresponding kernel.

2.2 AT Commands Serial Configuration

To configure support of the USB serial, refer to the following steps:

Go to the root directory of the Linux Kernel source code and execute the command:

 # make menuconfig

Open the Linux Kernel Configuration Menu

2.2.1. Use GSM/CDMA Modems

l Configure module

Go to Device Drivers > USB support > USB Serial Converter support > USB driver for GSM and CDMA modems

Select USB driver for GSM and CDMA modems and press save key, then exit.

l Add VID and PID of M2

Edit the VID and PID of Kernel: **go to drivers/usb/serial/option.c**

Add the code and PID of Kernel

```
80
81 /* ucloudlink vendor id */
82 #define UKL_VENDOR_ID    0x1782
83
```

Add the blacklist of M2

```
514
515 static const struct option_blacklist_info ukl_m2_blacklist = {
516     .sendsetup = BIT(3),
517 };
518
```

Add VID and PID to option_ids

```
1232    { USB_DEVICE_AND_INTERFACE_INFO(UKL_VENDOR_ID, 0x5D22, 0xff, 0x00, 0x00),
1233      .driver_info = (kernel_ulong_t)&ukl_m2_blacklist },
1234
```

AT commands of M2 have been added completely so far.

Compile kernel and burn Host image, then connect M2 to the Host, then you can find the **/dev/ttyUSB0** under **/dev/**

Notes: It is required to compile the Kernel for this procedure as below, and you use # modprobe usbserial vendor=0x1782 product=0x5D22 in the Linux operating system such as ubuntu. Then /dev/ttyUSB* appears.

2.2.2. Use USB Serial Driver

To edit Linux Kernel source code, go to **drivers/usb/serial/generic.c**

```
33
34 //static __u16 vendor  = 0x05f9;
35 //static __u16 product = 0xffff;
36 static __u16 vendor  = 0x1782;
37 static __u16 product = 0x5D22;
38
39
```

If you would like to compile USB Serial into Host Linux Kernel, compile Host Kernel and burn Host image, then connect M2 to the device, you can find the **/dev/ttyUSB0** under **/dev/**

If you would like to compile USB Serial into module, execute the command as below:

 # insmod usbserial vendor=0x1782 product=0x5d22

2.3 RNDIS Network Configuration on Linux

After you have RNIDS Host module compiled or inserted into Linux Kernel, when you start up the system and insert the module, you will see similar prompts shown as below:

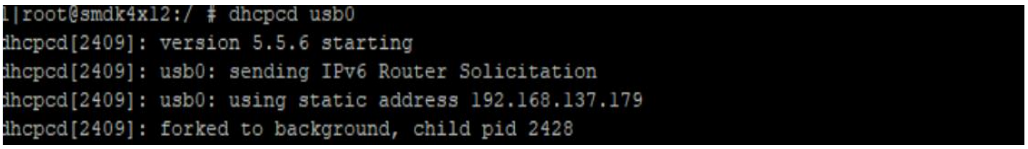
New rndis-host device of usb0 is detected.

That indicates module is set up successfully. For different platforms, we will use RNDIS to access Internet. The ipv4 of module is 192.168.137.0/24, and default gateway and DNS are 192.168.137.129.

2.3.1. Android 4.4

- Automatically assign IP:

  #dhcpcd usb0



After execution, you can use the command of netcfg to check the information as below:

```
root@smdk4x12:/ #
root@smdk4x12:/ # netcfg
lo       UP                                   127.0.0.1/8   0x00000049 00:00:00:00:00:00
rmnet0   DOWN                                   0.0.0.0/0   0x00001002 4e:c0:15:04:86:ea
sit0     DOWN                                   0.0.0.0/0   0x00000080 00:00:00:00:00:00
ip6tnl0  DOWN                                   0.0.0.0/0   0x00000080 00:00:00:00:00:00
usb0     UP                           192.168.137.179/24   0x00001043 da:bd:3e:25:36:12
eth0     UP                                     0.0.0.0/0   0x00001003 00:00:ff:ff:00:00
root@smdk4x12:/ #
```

- Configure DNS service(Optional)

If you have configured DNS service, you can skip this step.

Use the ndc service came from Android 4.4 to set up DNS server address, execute the command as below:

Set up the default gateway

#ndc resolver setdefaultif usb0

```
root@smdk4x12:/ # ndc resolver setdefaultif usb0
200 0 Resolver command succeeded
```

Set up DNS

#ndc resolver setifdns usb0 "" 192.168.137.129

```
root@smdk4x12:/ # ndc resolver setifdns usb0 "" 192.168.137.129
200 0 Resolver command succeeded
```

2.3.2. Android 5.x, 6.x

- Automatically assign IP

  #dhcpcd usb0

  These are assigned ip, mask, gateway, dns

```
shell@tulip-p1:/ #
shell@tulip-p1:/ # getprop dhcp.usb0.ipaddress
192.168.137.122
shell@tulip-p1:/ # getprop dhcp.usb0.mask
255.255.255.0
shell@tulip-p1:/ # getprop dhcp.usb0.gateway
192.168.137.129
shell@tulip-p1:/ # getprop dhcp.usb0.dns1
192.168.137.129
```

- Create network number

For Android 5.0 and higher, add netid. Firstly create netid, for example, if the network number is 100

 # ndc network create 100

The operation is successful if it returns 200, 0 or success.

- Add network card of usb0 into the network that Id is 100

  # ndc network interface add 100 usb0

  The operation is successful if it returns 200, 0 or success.

- Set up router and gateway

  #ndc network route add 100 usb0 0.0.0.0/0 192.168.137.129

  The operation is successful if it returns 200, 0 or success.

- Set netid 100 as default network connection

  # ndc network default set 100

  The operation is successful if it returns 200, 0 or success.

- Configure dns

  # ndc resolver setnetdns 100 "" 192.168.137.129

  The operation is successful if it returns 200, 0 or Resolver command succeeded.

Then the network configuration is complete, and you can access Internet using Ping.

2.3.3. Android 7.x, 8.x

- Assign IP

The ipv4 address of M2 module is 192.168.137.0/24（the address is 192.168.137.x, and mask is 255.255.255.0）. Default gateway and DNS are 192.168.137.129.

The IP is assigned manually to usb0 and usb0 up

#ndc interface setcfgusb0 192.168.137.139 24 up

- Create network number

For Android 5.0 and higher, add netid. Firstly create netid, for example, if the network number is 100.

#ndcnetworkcreate100

The operation is successful if it returns 200, 0 or success.

- Add gateway of usb0 to network that the ID is 100

  #ndc network interface add 100 usb0

  The operation is successful if it returns 200, 0 or success.

- Set up router and gateway

  #ndc network route add 100 usb00.0.0.0/0 192.168.137.129

  The operation is successful if it returns 200, 0 or success.

- Set netid 100 as default network connection

  #ndc network default set 100

  The operation is successful if it returns 200, 0 or success.

- Set up DNS

  The operation is successful if it returns 200, 0 or Resolver command succeeded.

The network configuration is complete, and you can assess Internet using Ping.

Notes: Source code of network configuration in Android system is /system/netd/server/CommandListener.cpp, refer to source code for details.

3 Hardware description

3.1 Datasheet

https://www.ucloudlink.com/file/uCloudlink_GLMM20A01_Mini_PCIe_Hardware_Design_V2.3.pdf

3.2 Standard kit contents

1. A GLAMM20A02 module

4 MQTT Support

4.1 AT Commands for MQTT

The AT commands for MQTT communication as follows.

AT#MQTTCONN
Connect to an MQTT server. This command can set parameters such as the server's IP address and port number.
The complete format :
AT#MQTTCONN=client_id,username,password,broker_url,broker_port,keepalive_time,clean_session

AT#MQTTPUB
Publish an MQTT message. This command can set parameters such as the message topic, quality of service level, and message content.
Command format：
AT#MQTTPUB=pub_topic,pub_qos,pub_retained,pub_msg
Please note that pub_msg is a hexadecimal string. For example, if you want to send 'hello', the contents of pub_msg should be the hexadecimal string of the ASCII code for each letter in 'hello', whitch is '68656C6F'.

AT#QMTSUB
Subscribe to an MQTT message. This command can set parameters such as the subscription topic and quality of service level.
Command format：
AT#MQTTSUB=sub_topic,sub_qos

AT#MQTTUNSUB
Unsubscribe from a topic on an MQTT server.
Command format：
AT#MQTTUNSUB=unsub_topic

AT#MQTTDISC
Disconnect from an MQTT server. This command can disconnect the current MQTT connection.
Command format：
AT#MQTTDISC

AT#MQTTSTAT?
Query the connection status of an MQTT client
Command format：
AT#MQTTSTAT?

4.2 Connect to AWS IoT service provided by UCloudlink

When the device is activated and successfully connected to the network, it will automatically apply for a certificate with the device IMEI as the things name from AWS IoT. Then, AT commands can be used to connect to AWS IoT, subscribe, publish messages, and perform other related operations.

Here is an example :

AT#MQTTCONN=353682680008768,,,a3r68s0rrta0ju-ats.iot.us-west-2.amazonaws.com,8883,30,1

OK

AT#MQTTSUB=EJSImJ6JIIS5I/353682680008768/usr/get,1

OK

AT#MQTTPUB=EJSImJ6JIIS5I/353682680008768/usr/update,1,0,68656C6F

OK

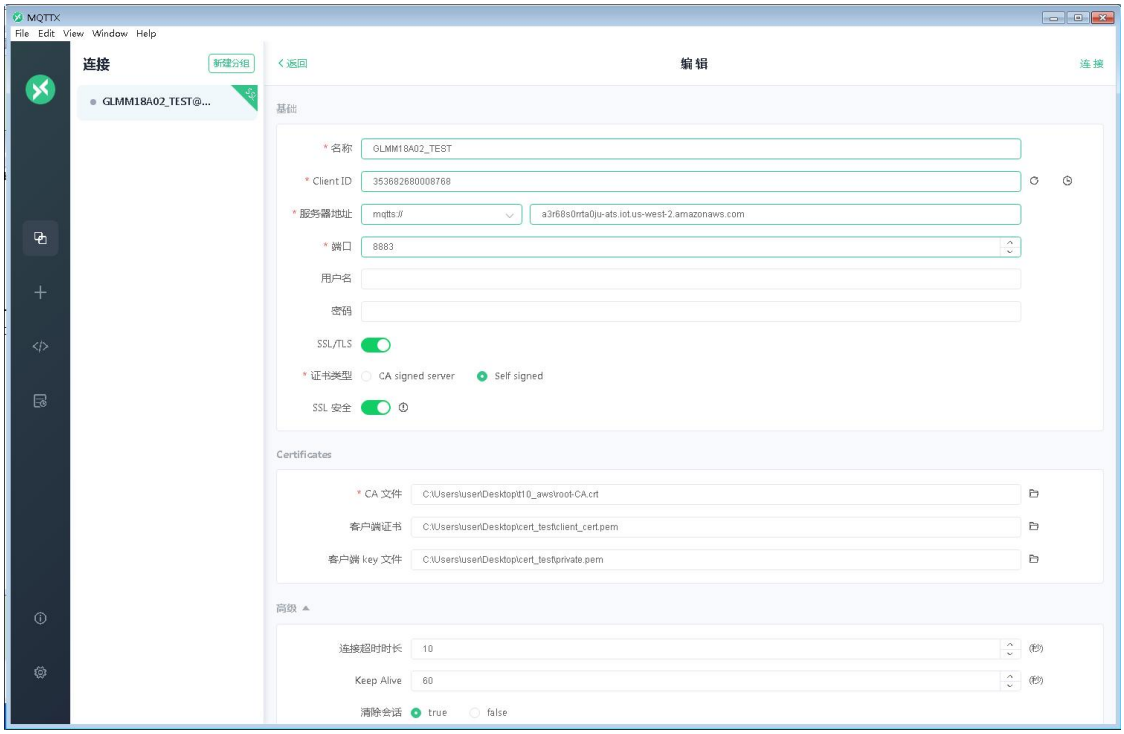
AT#MQTTRECV

61777320696F7420636F6E6E6563746564
OK

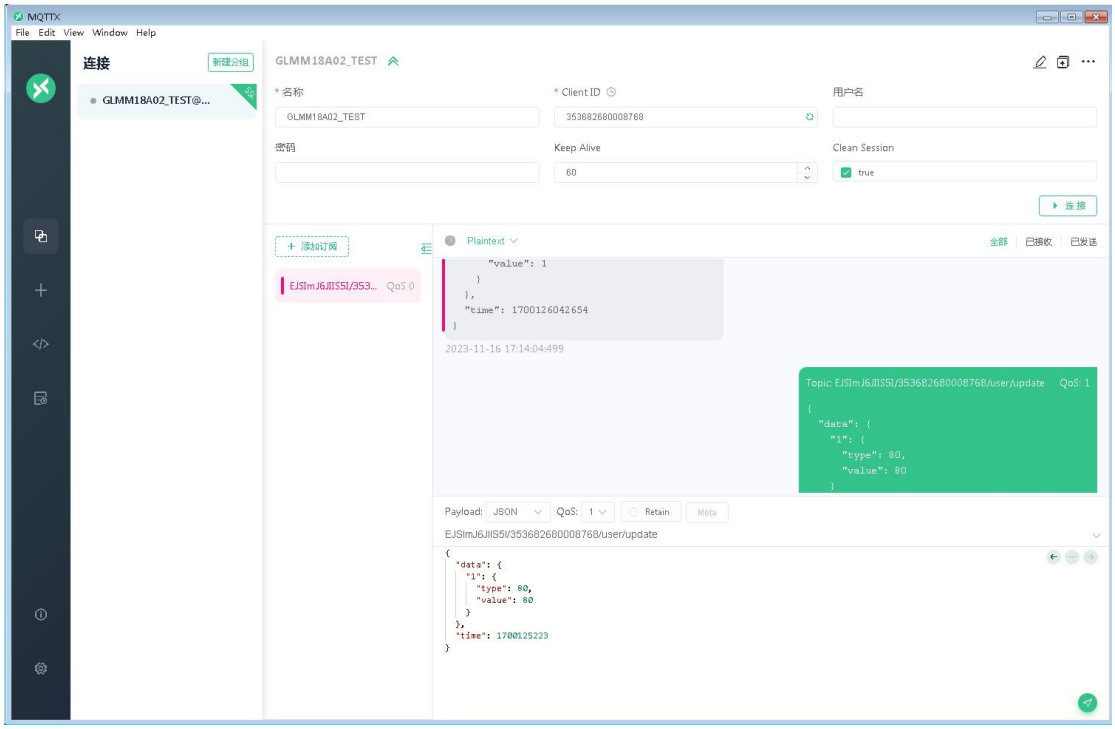
Note: the IMEI in topic string need to be replaced with the actually test device IMEI.

4.2.1 Test AWS IOT service provided by UCloudlink

Certification and private key can be provided only for test device. The configuration of MQTTX tool is as follows.

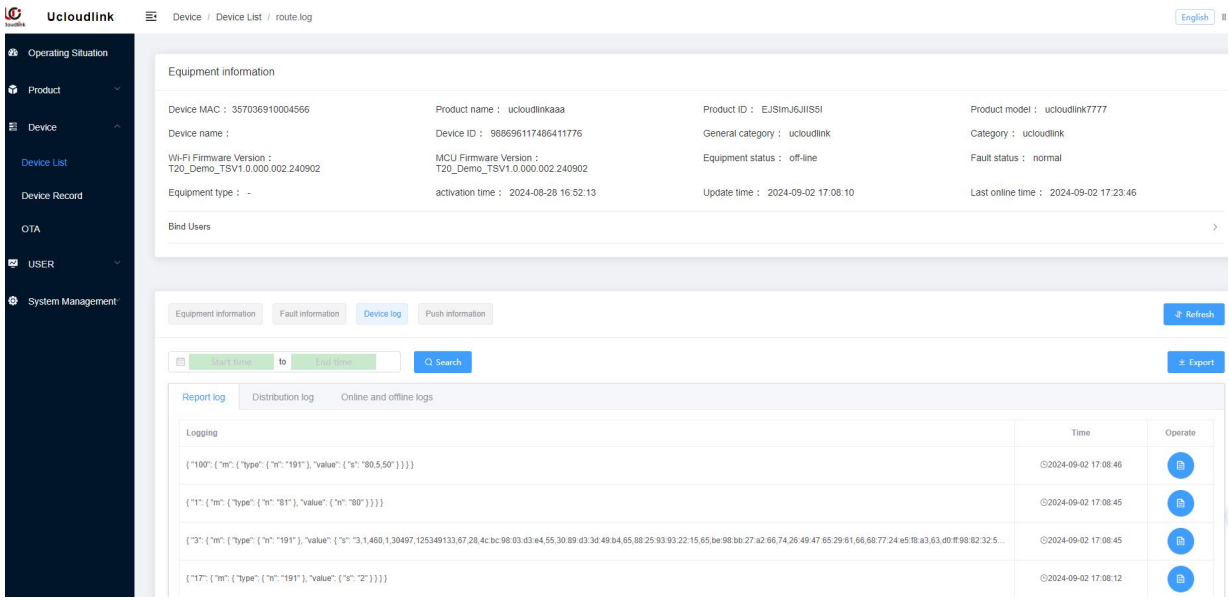


After the configuration is complete, click the connect button to connect to aws iot server. Subscribe the topic on which the test device publish messages (EJSImJ6JIIS5I/353682680008768/usr/update) in the MQTT tool to receive messages sent by the test device. Sending messages to the topic subscribed by the test device(EJSImJ6JIIS5I/353682680008768/usr/get) with MQTT tool, the test device will receive messages.



4.2.2 Using the device management platform provided by UCloudlink

After the device is activated, it will automatically request an AWS certificate and connect to the AWS IoT service provided by UCloudlink. You can manage the device on the UCloudlink device management platform, including viewing messages, sending messages, and performing various control operations on the device. We will create a specific account for you to manage all of your devices.



4.3 Connecting to your own AWS IoT service

4.3.1 Setup your AWS account and permissions

If you do not have an existing AWS account and user, refer to the online AWS documentation at:
https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html

Then , follow the steps below to get started.

Sign up for an AWS account:
https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#aws-registration

Create an administrative user:
https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#create-an-admin

Open the AWS IoT console:
https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#iot-console-signin

4.3.2 Create resources in AWS IoT

Create IoT resources, an IoT access policy, and a Thing object. When creating a Thing object, select "Auto-generate certificate" in the certificate options, and in the "Attach a policy to the certificate" step, select the policy generated in the previous step.
Download and save the generated certificate, and import the certificate into the device in the following steps.

Create an AWS IoT Policy：

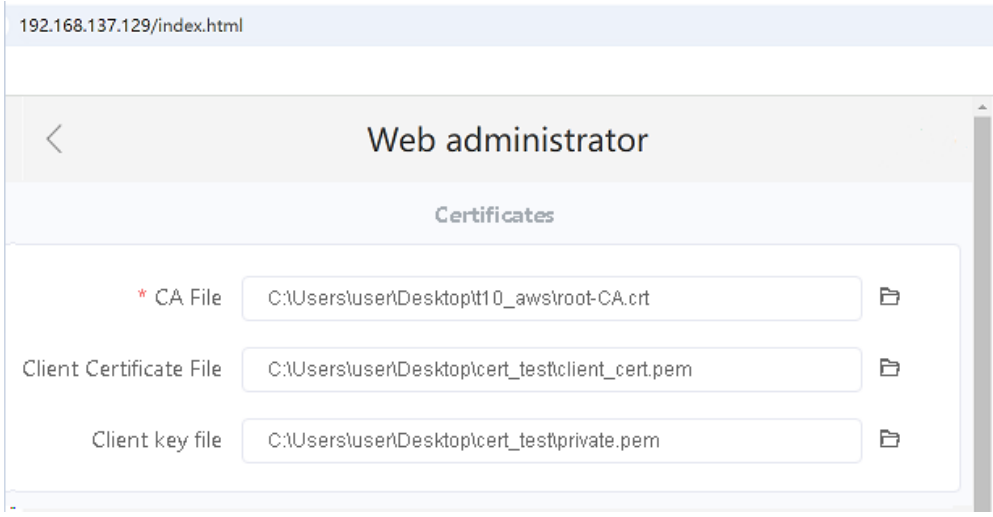
https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-resources.html#create-iot-policy

Create a thing object：

https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-resources.html#create-aws-thing

4.3.3 Provision the device with credentials

Place the test device on the small board , and then insert the small board into the USB port of the test computer.Use a web browser to access 192.168.137.129 and login to the web service page of the device. Then select the "AWS IoT" option and import the certificate, private key, and CA certificate on the "Certificates" page.



Buy link for the mini PCIe to USB board:

https://www.amazon.com/-/zh/dp/B01EMIOBD2/ref=sr_1_5?crid=2NVUXFNJS9GSJ&dib=eyJ2IjoiMSJ9.d6-9fODQ7WaOBIAh7QOFeMfM7QNBBIrJKEJs7EilOI3FIaK9SaacBx1QZR-XQdRWCtuQyOf5-ZVIIMCt4AEvWND7OtQYNOqTwKUSdKHiuJPYqia2BIAuxUN3nEuReyfE_paWVnYuAhcjEri4WO7riDqaBSo_MPOwJSarLwRA8zdMHOSgaShEfzh82RJO61BsEZPjUIOeOq-wW5WyFk3xC8sdXnPPtrlsCZLrKOCz0uY.L7F8IAJi5KVw_7q1bAthJudNAVkQESJ9AXycnYnJy9g&dib_tag=se&keywords=mini%2Bpcie%2Bto%2Busb%2B3.0&qid=1725344218&sprefix=minipcie%2Caps%2C320&sr=8-5&th=1

4.3.4 Connecting the device to AWS IoT Core

Open the PuTTY tool on the testing computer to send AT commands. Execute the following AT command to connect to AWS IoT Core.

Here is an example :

AT#MQTTCONN=353682680008768,,,a3r68s0rrta0ju-ats.iot.us-west-2.amazonaws.com,8883,30,1

OK

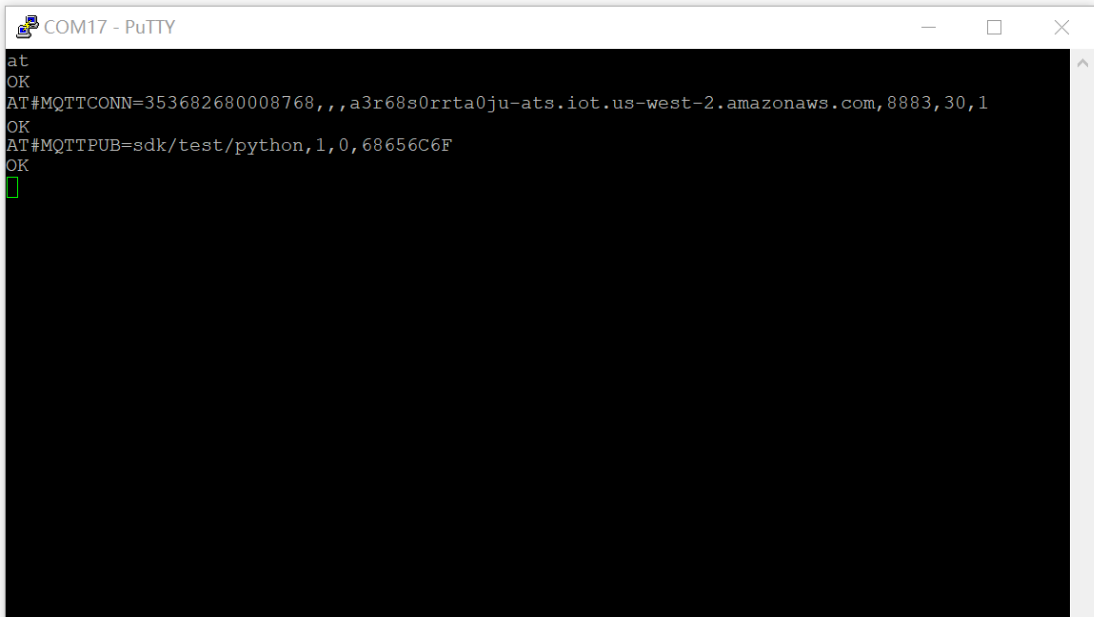
Execute the following AT command to publish messages to AWS IoT Core.

AT#MQTTPUB=sdk/test/python,1,0,68656C6F

OK

Note: the IMEI in topic string need to be replaced with the actually test device IMEI.

the endpoint string need to be replaced with your actually endpoint.



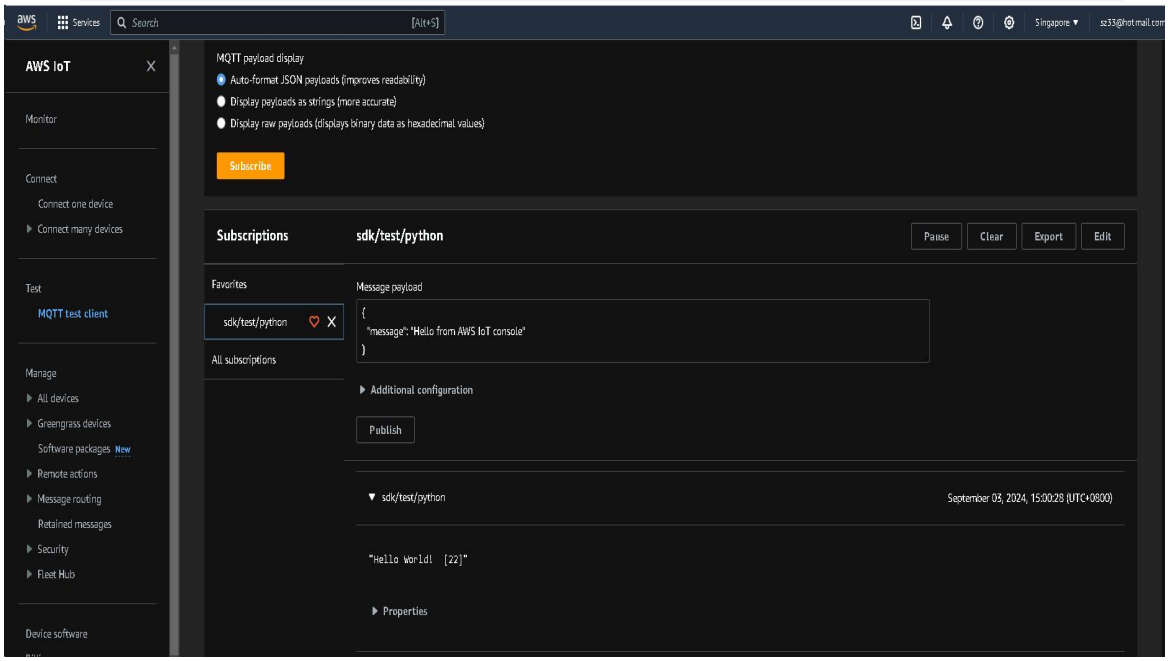
Download link for puTTY:

https://www.putty.org/

4.3.5 Verify messages in AWS IoT Core

Viewing device messages or sending messages to devices can be done using the MQTT tool in AWS IoT.The following link provides detailed instructions on how to perform the operation:

https://docs.aws.amazon.com/iot/latest/developerguide/view-mqtt-messages.html



5 Troubleshooting

1. device cannot start normally after being inserted into the USB port of the computer.

Possible insufficient power supply for USB, try another USB port or switching to another computer.

2. Unable to open the web service page of the device.

Check if the USB RNDIS network card driver is installed correctly.If the driver is not installed , refer to the driver installation section in this document for installation instructions and try again.

3. The device did not respond when sending AT commands using the puTTY tool.

Check if puTTY has selected the correct serial port and open the device manager to view the AT command port of the device and select the corresponding port.

4. Issues related to AWS IoT，refer to the AWS online documentation on Troubleshooting AWS IoT

https://docs.aws.amazon.com/iot/latest/developerguide/iot_troubleshooting.html