u-connectXpress

IoT cloud connectivity

Application note

Abstract

This Application note provides information on how to configure and setup the connection for the most popular cloud services, such as u-blox Thingstream, Amazon Web Services (AWS), and Azure, using u-connectXpress software.

UBX-19010078 - R06

C1-Public

www.u-blox.com

Document information

Title	**u-connectXpress**	
Subtitle	IoT cloud connectivity	
Document type	Application note	
Document number	UBX-19010078	
Revision and date	R06	15-Jul-2022
Disclosure restriction	C1-Public	

This document applies to the following products:

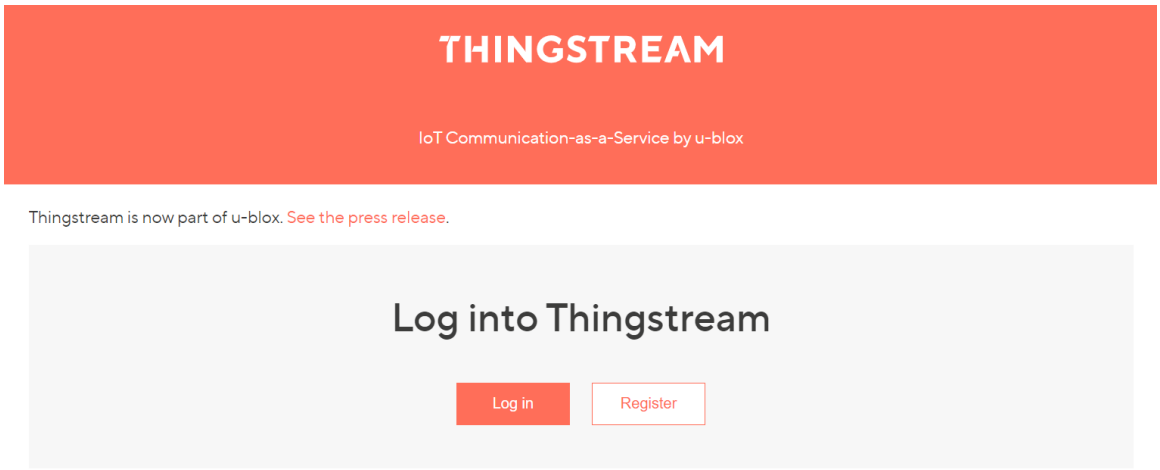
Product name	Software version
NINA-W131	2.1.x onwards
NINA-W132	2.1.x onwards
NINA-W151	All
NINA-W152	All
NINA-W156	3.1.x onwards
ODIN-W260	7.0.x onwards
ODIN-W262	7.0.x onwards
ODIN-W263	7.0.x onwards

Contents

1 Configuring Thingstream

1.1 Cloud server configuration

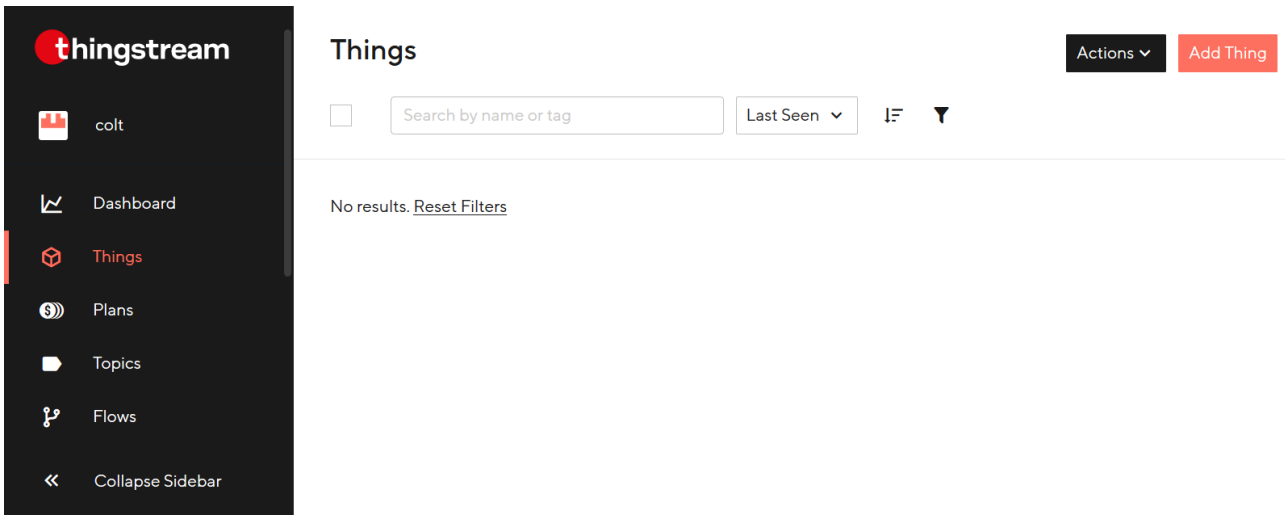
1. Log in to your Thingstream account, or create a new account at: https://thingstream.io/



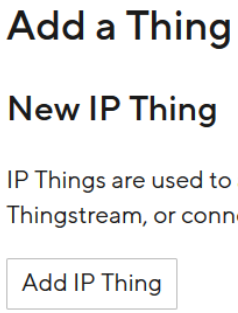
☞ When registering and creating a new (free) account, you need to select a unique domain name, like your company name for example. Follow the registration instructions to activate your account.

2. Navigate to https://portal.thingstream.io/app/things

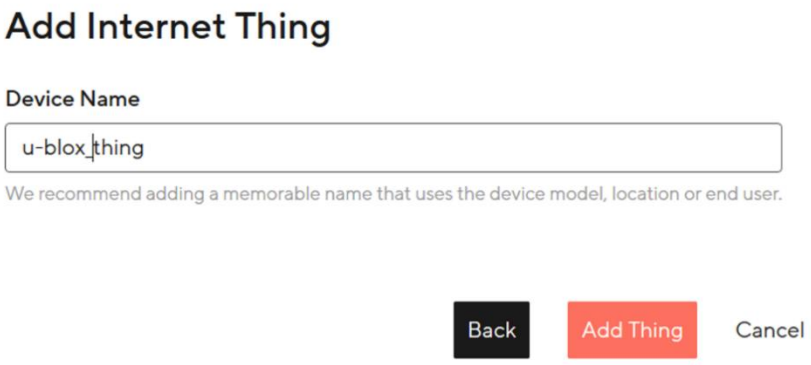
3. Go to **Things** in the navigation menu, and then select **Add Thing**.



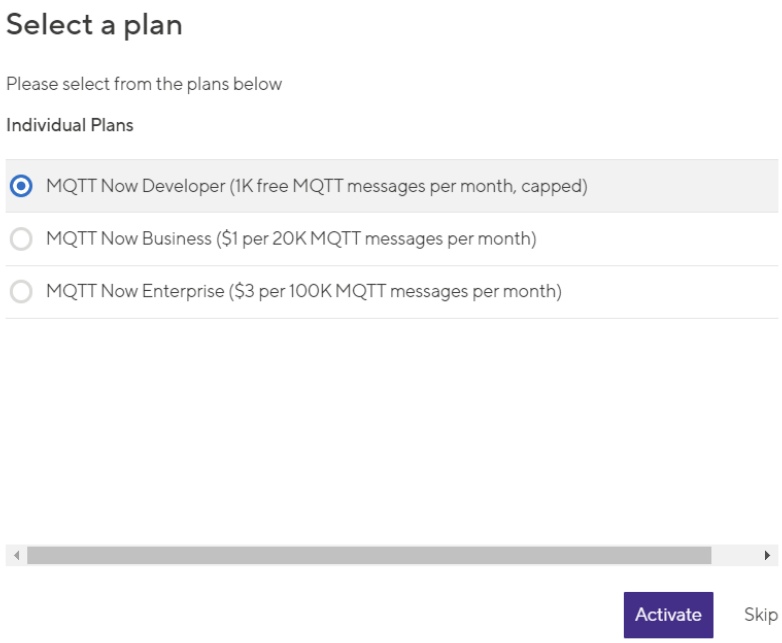
4. Select **Add IP Thing**.



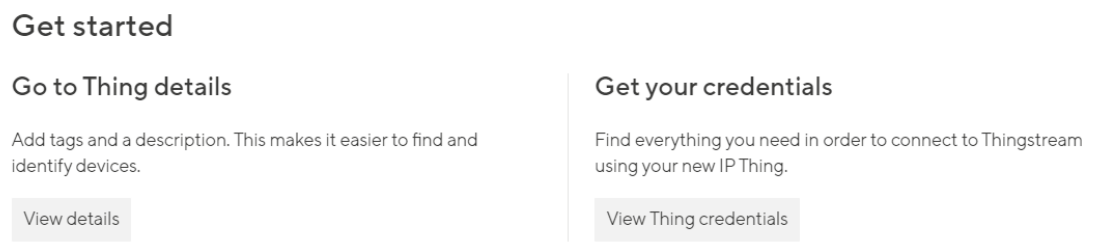
5. Choose an appropriate device name for your internet Thing.



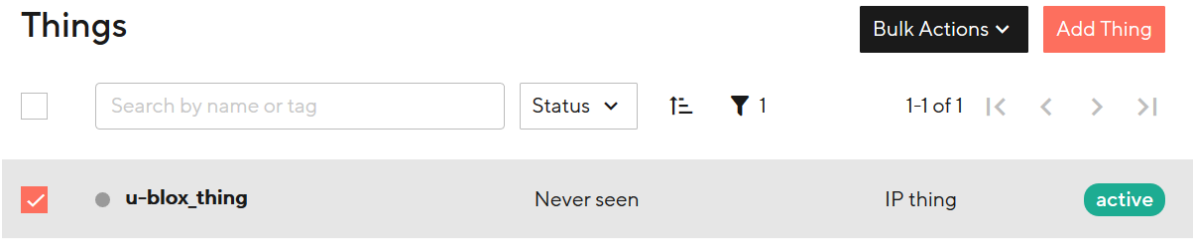
6. Select a plan and then select **Activate**.



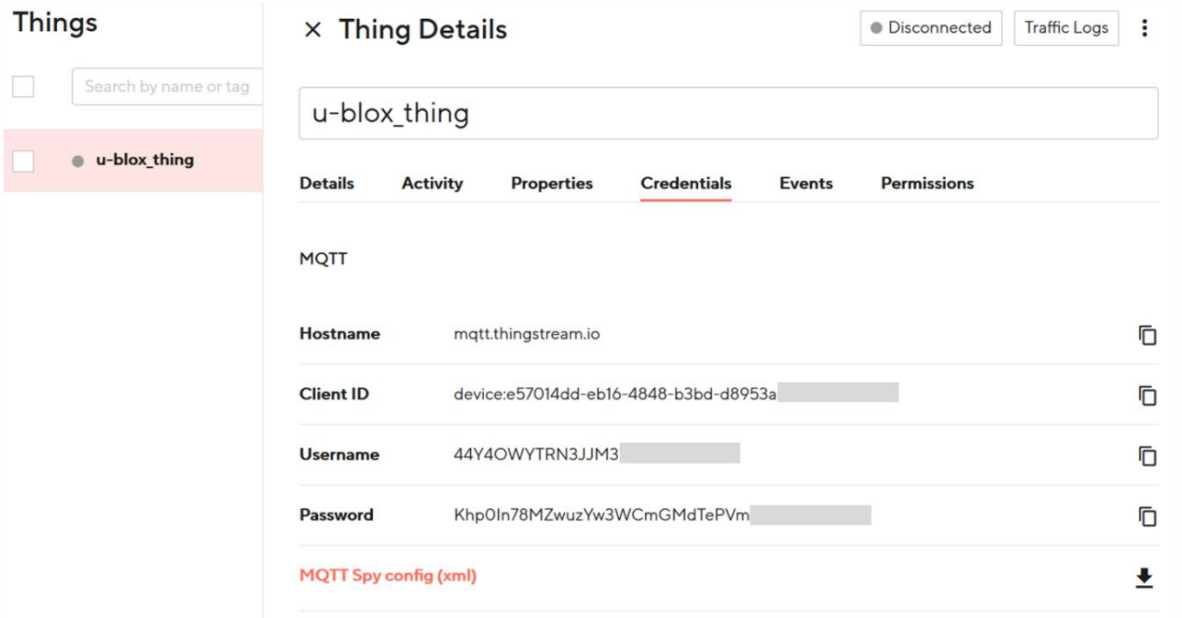
7. From the **Get started** dialog, select **View Thing credentials**.



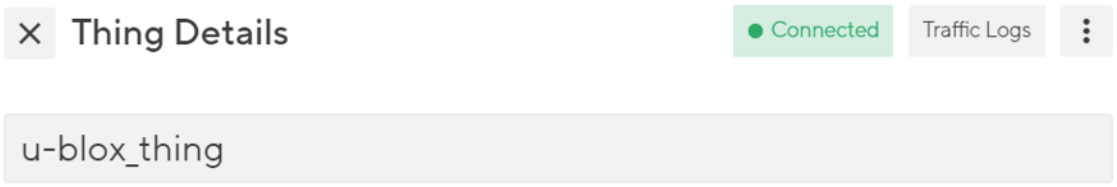
8. Select your device to open the **Thing Details**.



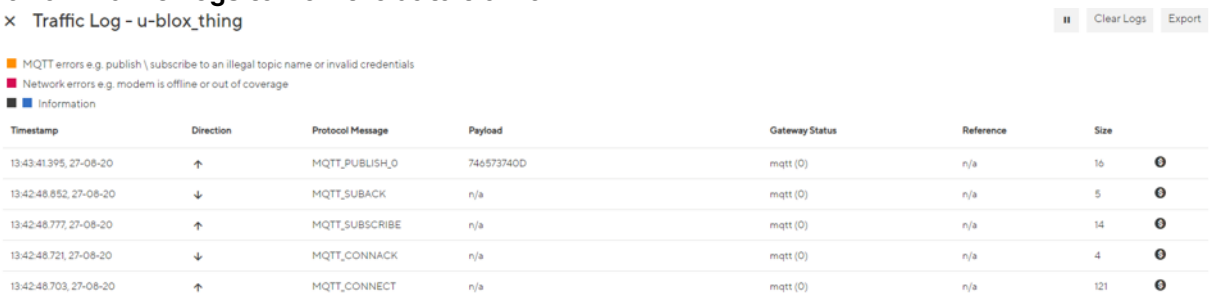
9. Select **Credentials** and make note of your **Thing Details**. You need this information to connect the module to the cloud later.



10. Configure and connect your device (see section **Error! Reference source not found.**). The **Connected** button turns green when the device is connected.



11. Click on **Traffic Logs** to view the data traffic.



1.2 Installing certificates to module

Install the server certificates to the module using AT-commands or s-center, as described in sections 1.2.1 and 1.2.2 respectively.

☞ Note that Thingstream certificates are located on Amazon servers. Amazon CA root certificate can be downloaded from: https://www.amazontrust.com/repository/AmazonRootCA1.pem

1.2.1 Installing certificates on the module using AT commands

The sequence below configures Wi-Fi and installs the certificates to the module.

1. Upload the local Amazon certificate file to the physical module as a trusted root (CA) certificate using the command `AT+USECMNG=0,0`.

☞ For further information about `AT+USECMNG`, see the u-connect AT commands manual [1] and u-connectXpress user guide [3].

2. Set up a network connection to establish the connection using the following the AT commands:

```
AT+UWSC=<configuration_id>,<param_tag>,<param_val1>[,<param_val2>,...,<param_valn>]
AT+UWSCA=<config_id>,<action>
```

☞ The commands given in the example above connect Wi-Fi. For more information about the Wi-Fi connection setup, see the u-connectXpress user guide [3].

3. Connect to the Thingstream cloud server using the `AT+UDPC` command and MQTT URL, as shown in the given connection parameters and subsequent command example below.

   Client Id=device:57014dd-eb16-4848-b3bd-d8953a123456789
   Username = 44Y4OWYTRN3JJM3123456789
   Password = KgpOln78MZwuzYw3WCmGMdTePVm123456789
   Encryption = 3 – Only allow TLS 1.2

Trusted Root (CA) = AmazonRootCA1.pem
Publish topic = testpub
Subscribe topic = testsub

```
at+udcp=mqtt://mqtt.thingstream.io:8883/?
encr=3&
ca=AmazonRootCA1.pem
&client=device:57014dd-eb16-4848-b3bd-d8953a123456789
&user=44Y4OWYTRN3JJM3123456789
&passwd=KgpOln78MZwuzYw3WCmGMdTePVm123456789
&pt=testpub
&st=testsub
```

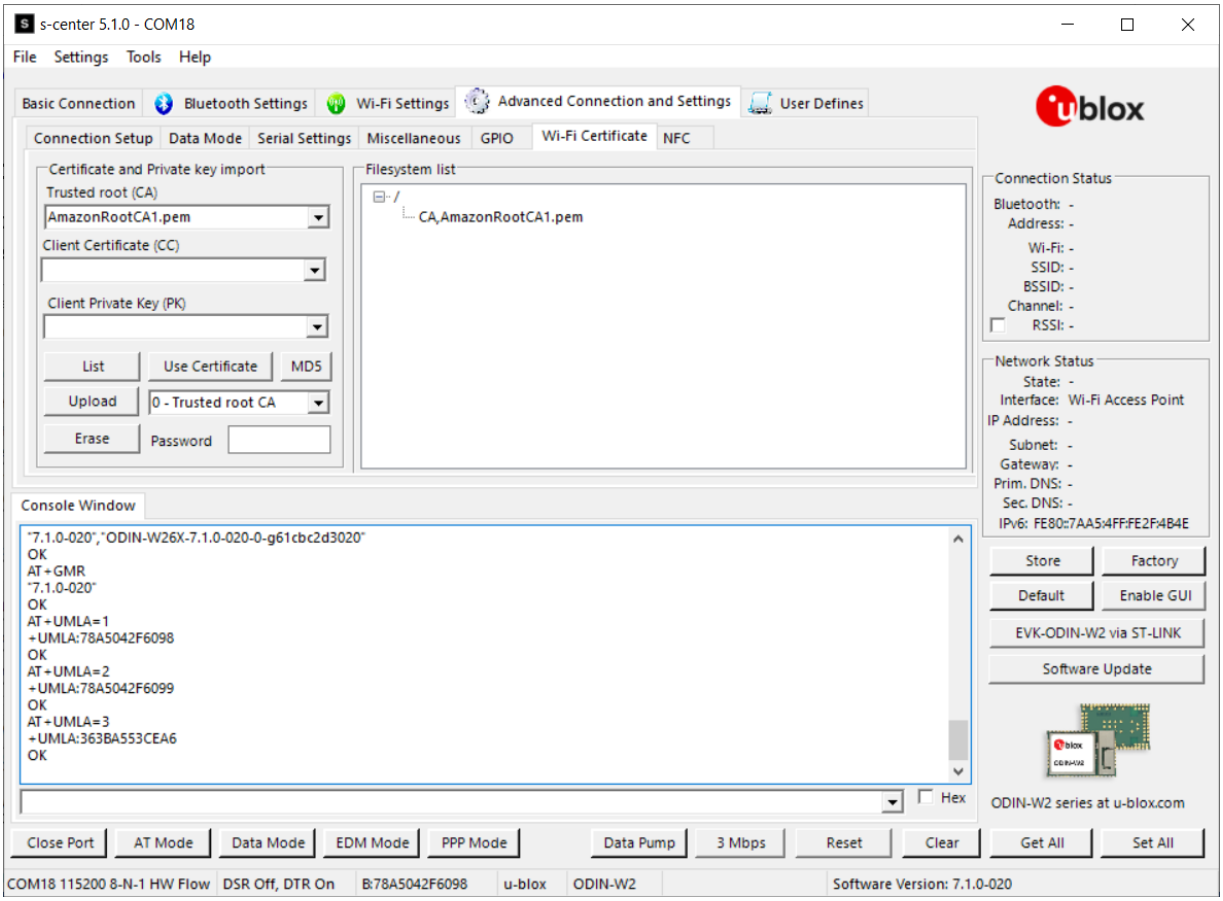
where:

- encr=3 = makes the connection using TLS 1.2. By default, connections use the less secure TLS versions 1.0 or 1.1.
- ca = the internal name given to the Thingstream Trusted Root (CA) server certificate when uploaded to the module.
- client = the Client ID set to the thing in Thingstream
- user = the token/Username generated while creating the device in the Thingstream creation of the thing.
- passwd = the token/Password generated while creating the device in the Thingstream creation of the thing.
- pt = Publish topic
- st = Subscribe topic

1.2.2 Installing certificates on the module using s-center

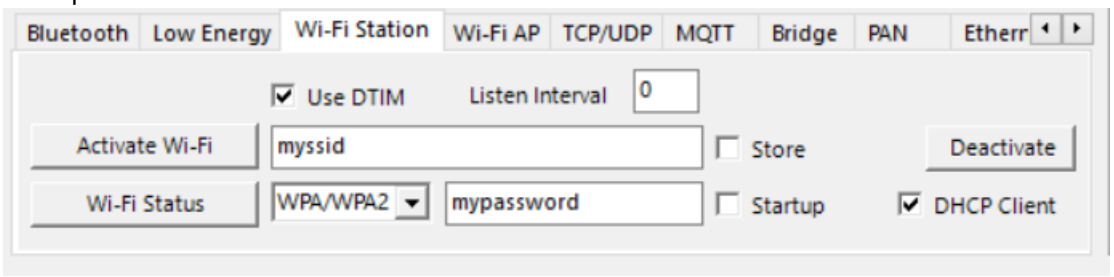
The sequence below configures Wi-Fi and installs the certificates to the module.

1. Using s-center, upload the local Amazon certificate file to the module as a trusted root (CA) certificate.



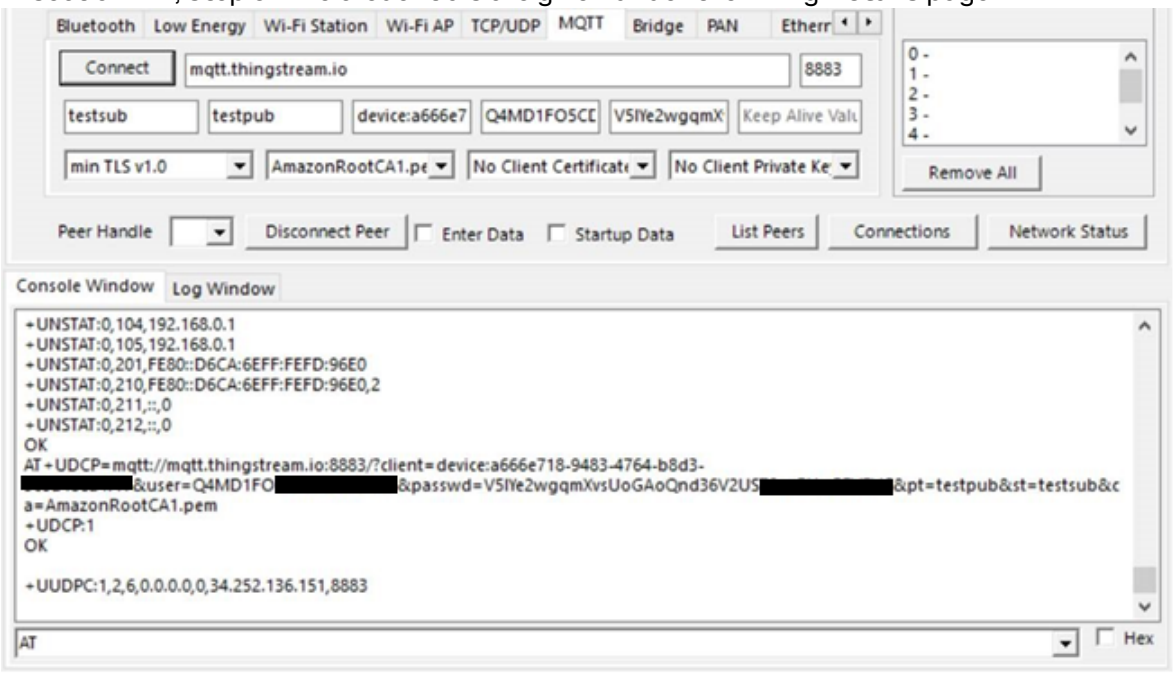
For further information about `AT+USECMNG`, see the u-connect AT commands manual [1] and u-connectXpress user guide [3].

2. Set up a network connection.



The example above connects Wi-Fi. For more information about the Wi-Fi connection setup, see the u-connectXpress user guide [3].

3. **Connect** to the u-blox Thingstream with MQTT using the Thing credentials previously noted in section 1.1, step 9. The credentials are given under the Thing **Details** page.



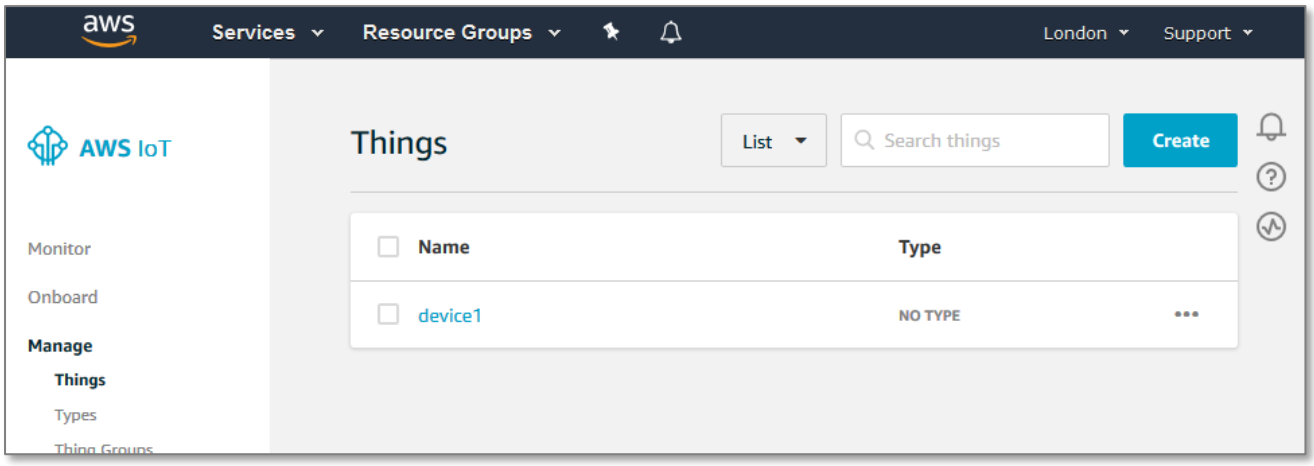
☞ For more information about MQTT, see the u-connectXpress MQTT application note [2].

2 Configuring AWS IoT Core

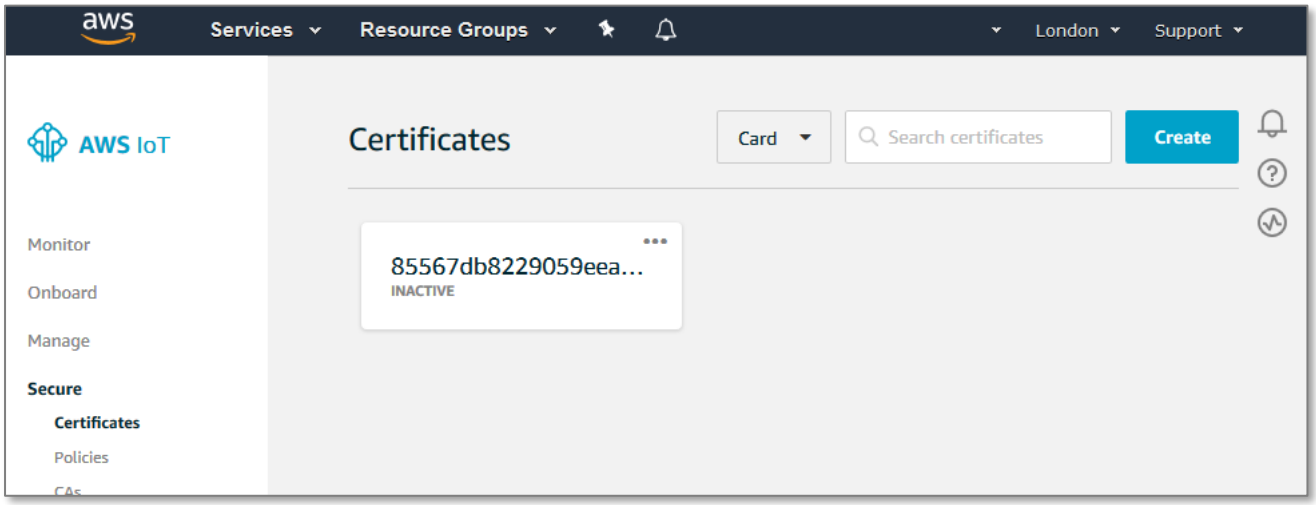
NINA-W1 and ODIN-W2 has passed the AWS Device Qualification Program.

2.1 Cloud server configuration

1. Create a Thing (IoT Core \ Manage \ Things). This is a representation and record of the physical device to be connected to the cloud. Give the Thing a name, for example “device1”. Create Thing without certificate (the certificate will be created later).



2. Create certificate using AWS IoT’s CA. Download the certificate, public key, private key, and the server certificate for AWS IoT.

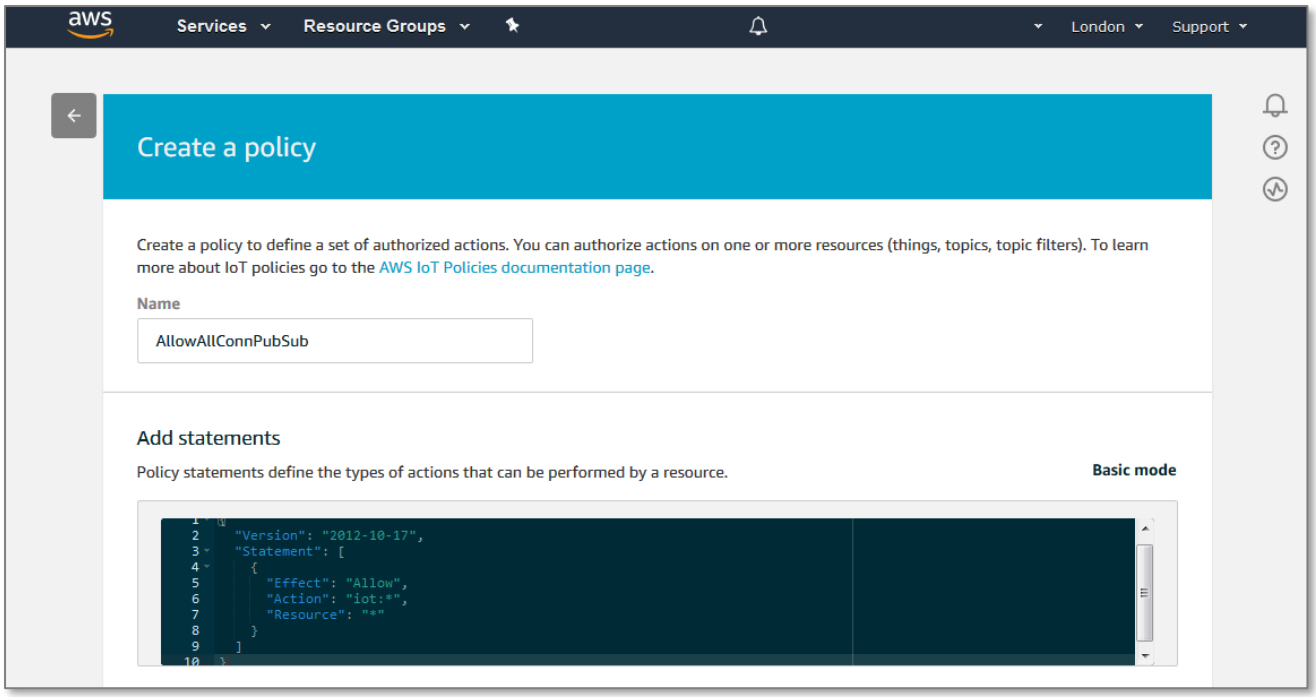


3. Create a policy, which allows the certificate holder to publish to all topics and subscribe to all topic filters. Enable Advanced mode and copy and paste the following example policy statement:

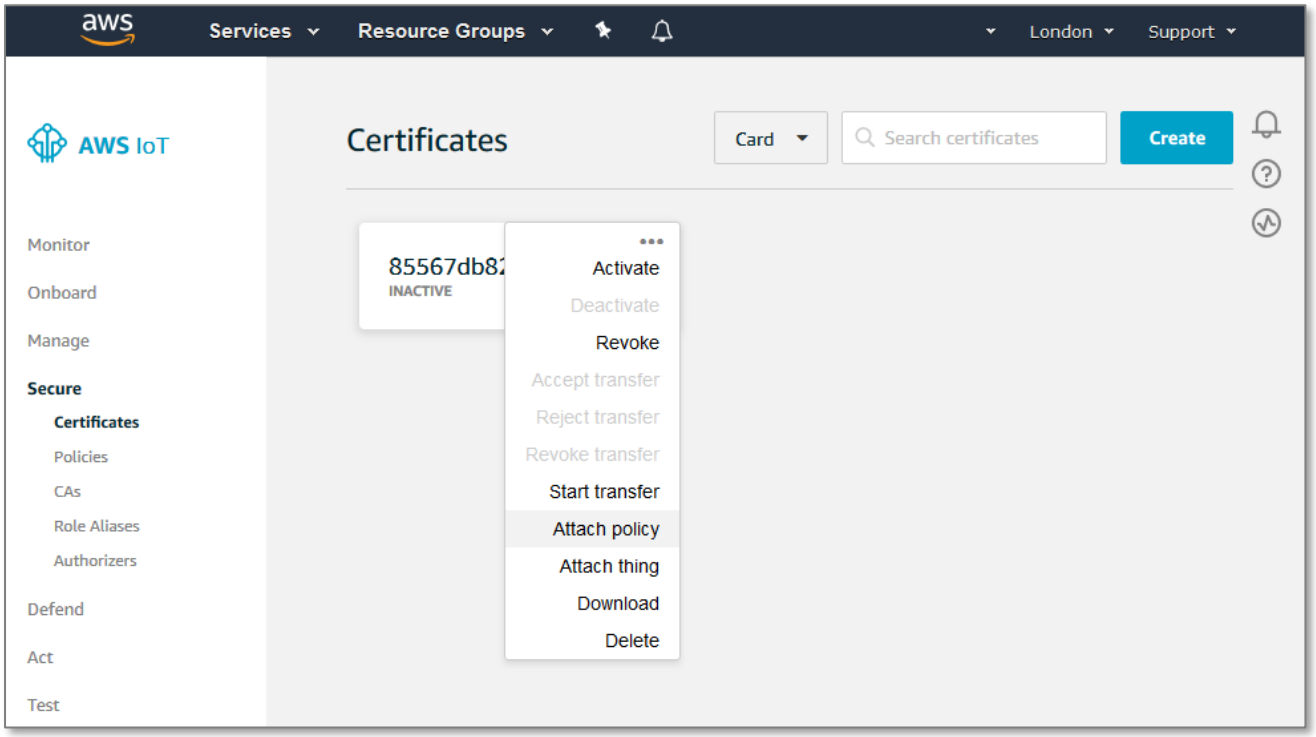
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

⚠ The example policy above must ONLY be used for the example in this application note. Do NOT use the above policy example it in production code. Instead, use a policy which explicitly specifies the allowed actions per resources, such as:

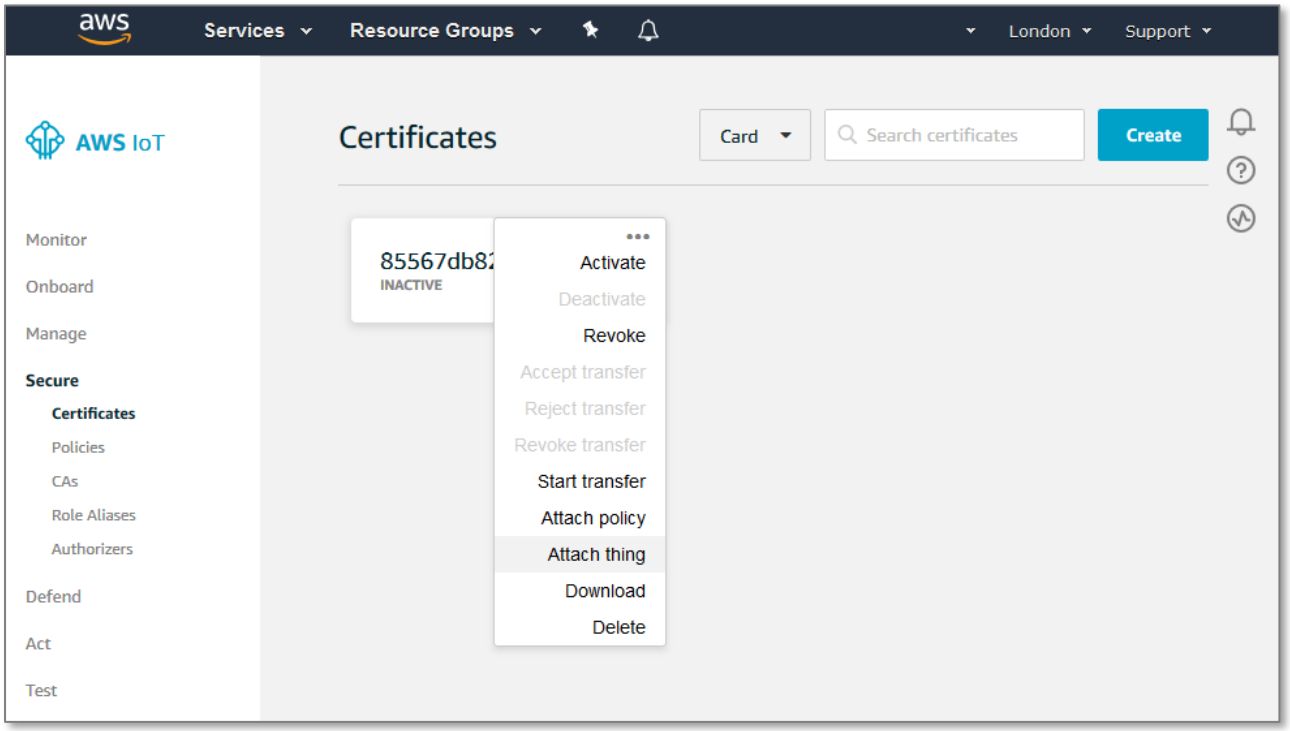
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-west-2:123456789012:
                   client/${iot:Connection.Thing.ThingName}"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:
        topicfilter/$aws/things/${iot:Connection.Thing.ThingName}/shadow/*",
        "arn:aws:iot:us-west-2:123456789012:
         topicfilter/${iot:Connection.Thing.ThingName}/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:
         topic/$aws/things/${iot:Connection.Thing.ThingName}/shadow/*",
        "arn:aws:iot:us-west-2:123456789012:
         topic/${iot:Connection.Thing.ThingName}/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:
         topic/$aws/things/${iot:Connection.Thing.ThingName}/shadow/*",
        "arn:aws:iot:us-west-2:123456789012:
         topic/${iot:Connection.Thing.ThingName}/*"
      ]
    }
  ]
}
```



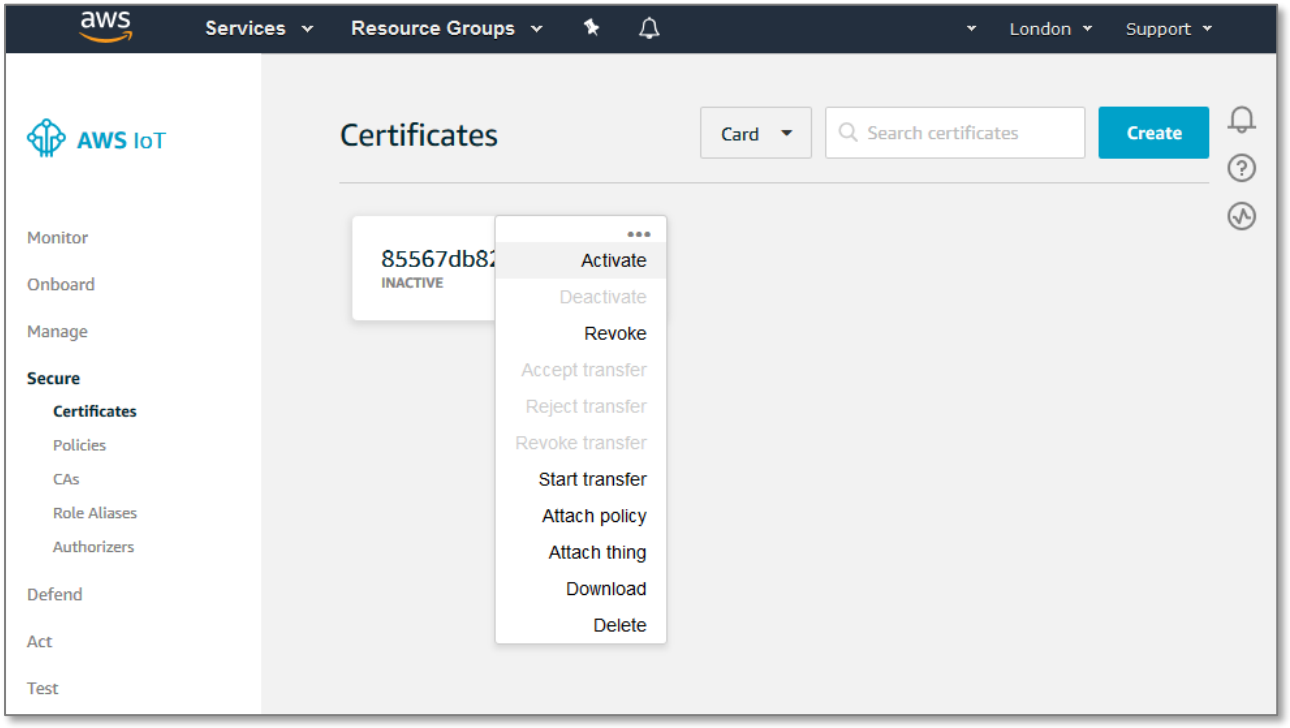
4. Attach the previously created policy to the previously created certificate.



5. Attach the previously created thing to the previously created certificate.



6. Activate the certificate.



2.2 Installing certificates to module

1. Upload the AWS IoT server certificate; the certificate can be downloaded from the following URL:

   https://www.amazontrust.com/repository/AmazonRootCA1.pem

   `AT+USECMNG=0,0,<internal_name>,<data_size>`

   See the u-connect AT commands manual [1] for additional information.

2. The Thing's certificate, public and private keys correspond to what is described elsewhere as the client or device certificate, public and private keys. Upload the client key and client certificate generated in step 2 of the cloud server configuration described in section 2.1.

   `(certificate) AT+USECMNG=0,1,<internal_name>,<data_size>`

   `(private key) AT+USECMNG=0,2,<internal_name>,<data_size>`

3. Set up a network connection. Example (Wi-Fi):

   `AT+UWSC=configuration_id>,<param_tag>,<param_val1>[,<param_val2>,...,<param_valn>]`

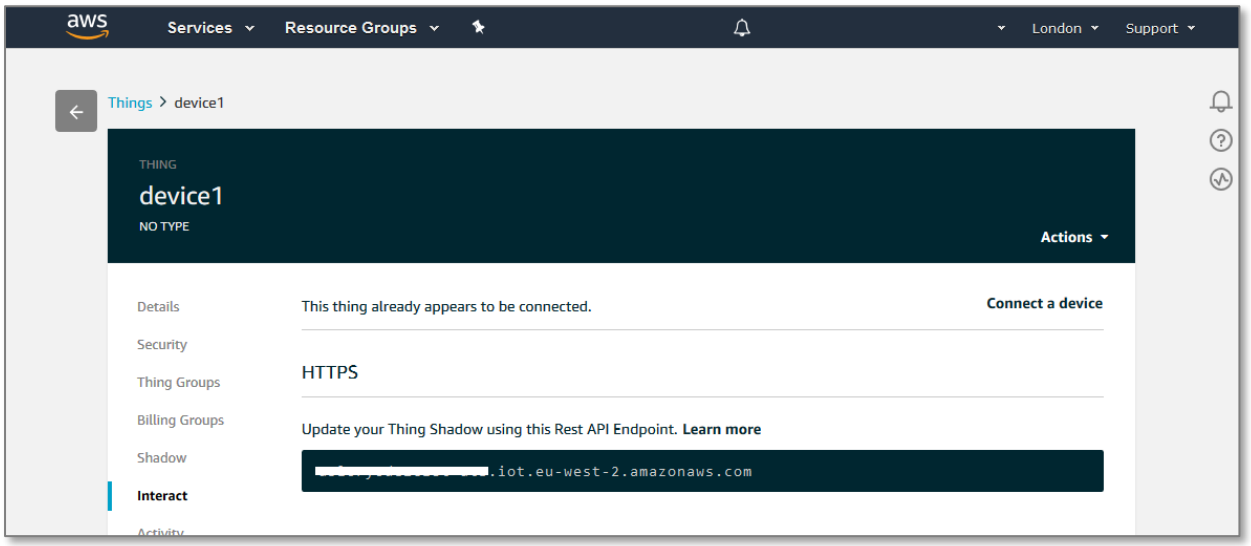
   `AT+UWSCA=config_id>,<action>`

4. Connect to AWS IoT using MQTT. For more information see the u-connectXpress MQTT application note [2].

   `at+udcp=mqtt://endpoint:8883/?ca=server_cert&cert=device_cert&privKey=device_key&pt=publish_topic&st=subscribe_topic`

   where:

   endpoint = your unique address to connect the "Thing" to AWS IoT, which can be found under Manage > Things > thing_name > Interact > HTTPS. The address is the same as the Rest API endpoint:



   server_cert = the internal name given to the AWS IoT server certificate when uploaded to the module.

   device_cert = the internal name given to the client certificate when uploaded to the module

   device_key = the internal name given to the client private key when uploaded to the module

   publish_topic = any topic that is valid in MQTT and module

   subscribe_topic = any topic that is valid in MQTT and module

Example

```
at+udcp=mqtt://abcdefghijkl-ats.iot.eu-west-
2.amazonaws.com:8883/?ca=AmazonRootCA1.pem&cert=device.crt&privKey=device.key&pt=te
st/pt&st=test/st
```

5. Switch to Data mode and send data to/from the AWS IoT server; monitor the transferred data as mentioned in Appendix B.2.

3 Configuring Azure IoT Hub

☞ Due to limitations in the maximum length of an AT command for NINA-W13 and NINA-W15 in u-connectXpress software prior to version 3.0, keep all names and identifiers as short as possible. Typically:

- The name of the IoT Hub
- The Device Id of the devices
- The names of the certificates (CA and CC) as well as the client private key (PK).

3.1 Create a new IoT Hub

Unless one already exists, create a new IoT Hub.

1. Navigate to portal.azure.com and select **More Services**.
2. Search for IoT Hub, and then select **Create** in the pop-up window.
3. Create new or select an existing Resource group.
4. Choose your IoT hub name, such as “iot-uconnectxpress-dev”. Microsoft suggest the naming convention “iot-<App Name>-<Environment>”. The given name is publicly visible, so private metadata should not be included. Keep the IoT hub name as short as possible.
5. Step through the configuration wizard. Tags are optional private metadata which you may want or need to manage your hub – especially as your company and your Azure cloud grows. If you are unsure, just add the tag “MyTag”, and value “MyValue”.
6. Once created, go to your IoT Hub (which Microsoft refers to in more general terms, as a resource) and check that it is available under “All resources” from the home screen.

3.2 Configure a module with a X.509 CA Signed certificate

3.2.1 Obtain the CA certificate

Create a CA certificate to use when signing the certificates for each module. The procedures for generating a CA certificate using OpenSSL are described in u-connectXpress Wi-Fi security application note [4].

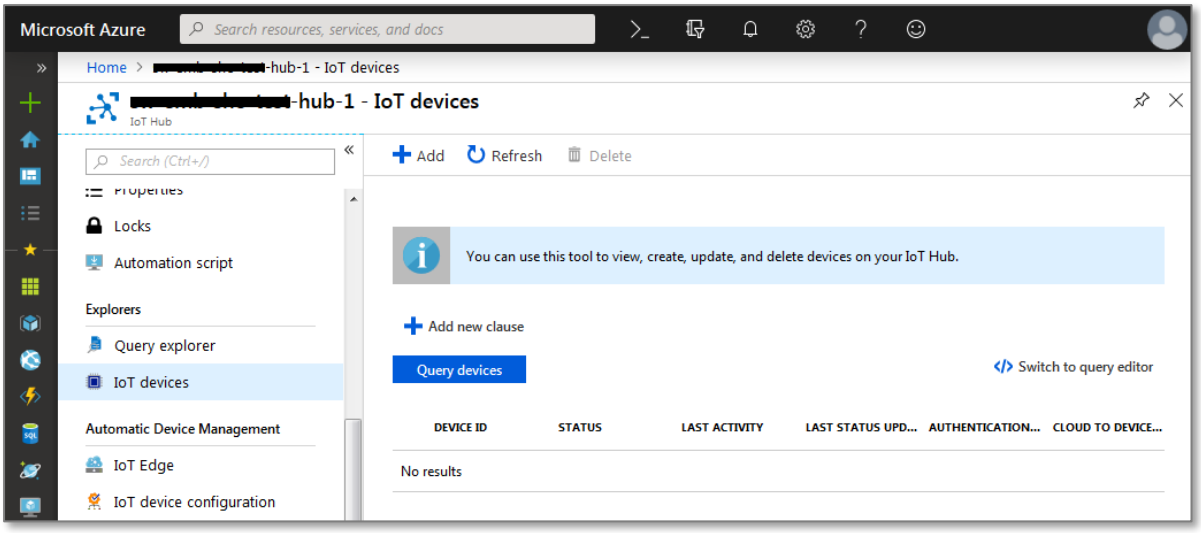
3.2.2 Configure cloud server

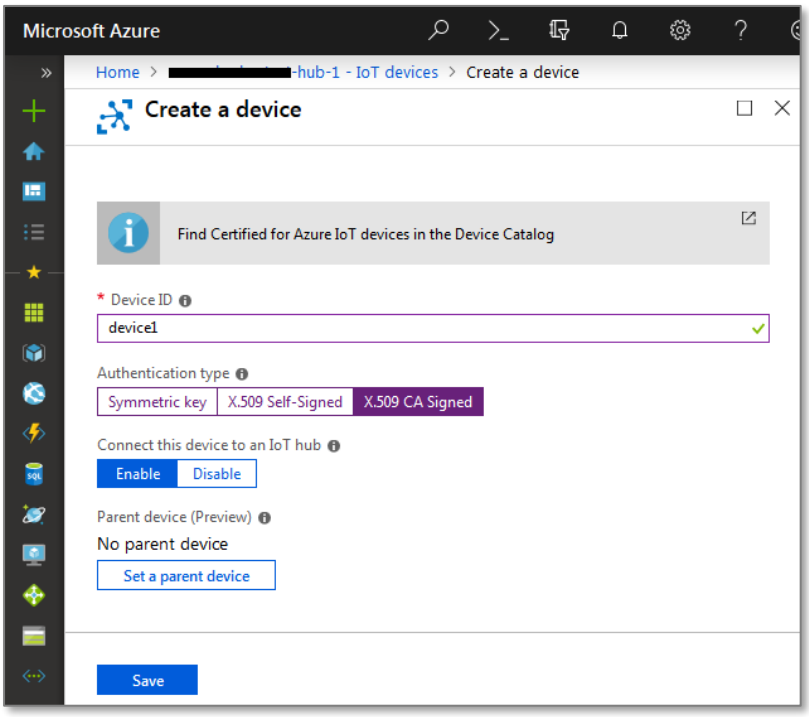
In the IoT Hub resource, you are required to verify that you have the necessary permissions to generate and sign certificates. To configure the cloud server, you must:

- Create the first device
- Sign a client side-certificate for the device
- Upload the CA used to sign the client-side certificate
- Verify the certificate used to sign the client-side certificate

3.2.2.1 Create the first device

1. Go to the IoT Devices tool in the IoT hub Explorer section. Select **New** or **Add** to register a new device.
2. Enter a unique Device ID. For example, enter the device's serial number "device1", or the response to the `AT+UMLA=2` command from your module. If the NINA-W13/W15 module is configured with u-connectXpress software version 3.0 or earlier, keep the device ID as short as possible.
3. Set the Authentication type to "X.509 CA Signed".





4. Select Save to add your new device to the displayed list of IoT devices.

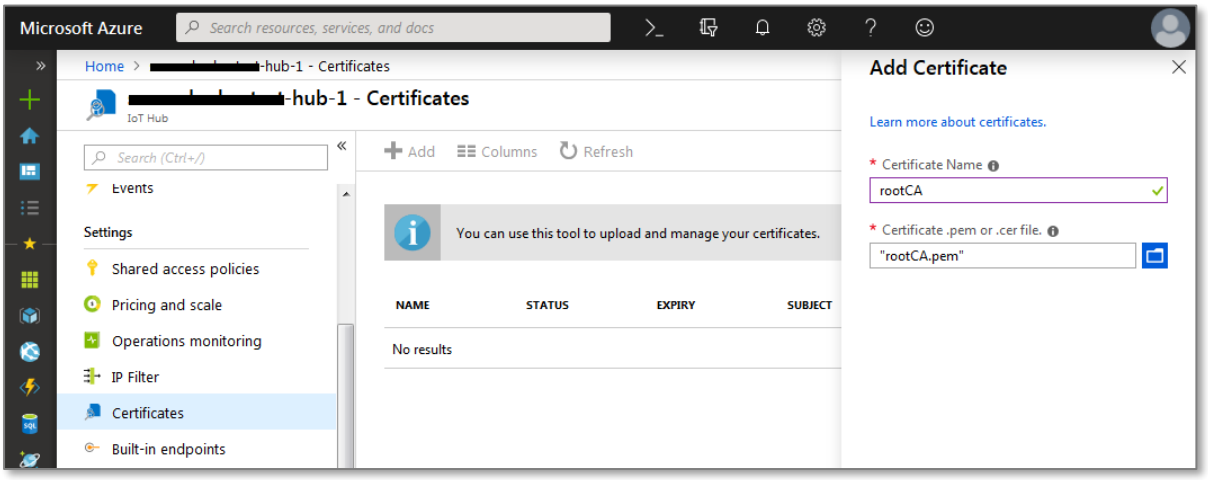
3.2.2.2 Sign a client side-certificate for the module

Generate a client-side certificate for your module (see u-connectXpress Wi-Fi security application note [4]), where Common Name (CN) is set to the Device ID of the device.

☞ This client-side certificate (CC) and its corresponding private key (PK) are also used later in the configuration.

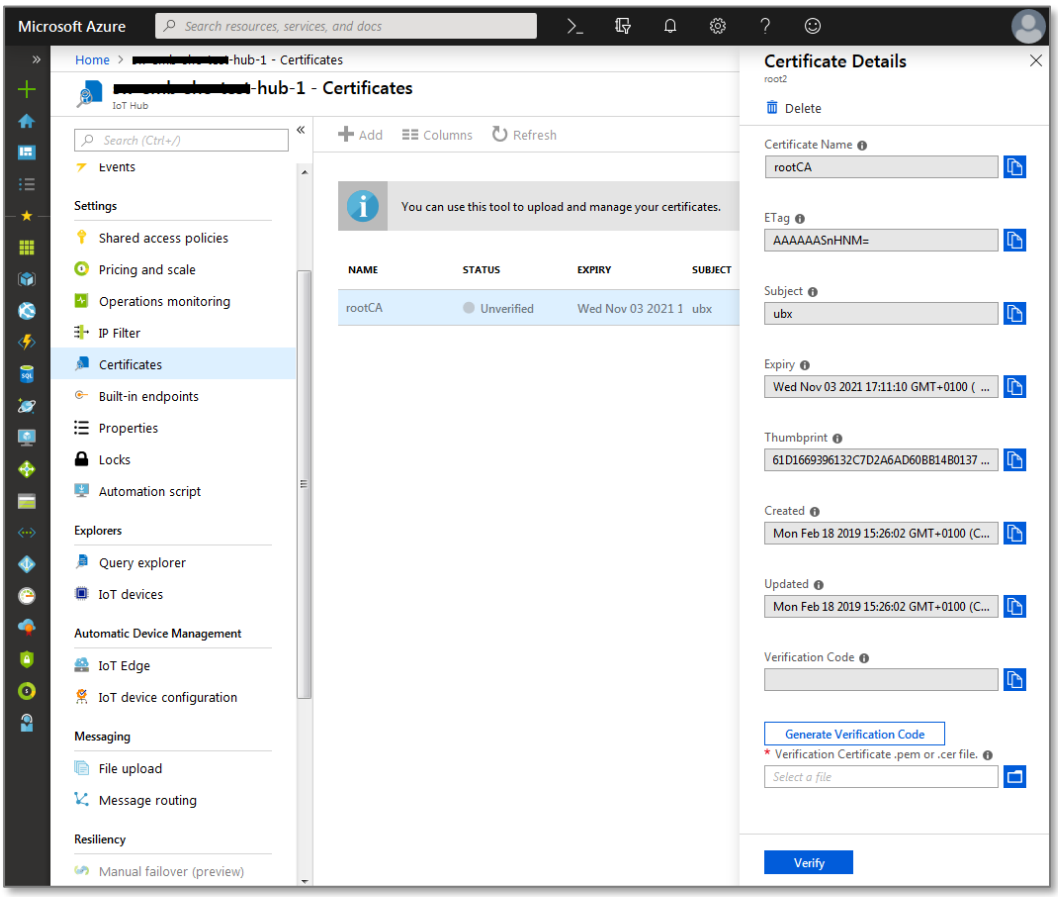
3.2.2.3 Upload and verify the CA used to sign the client-side certificate

1. To upload the CA certificate from section 3.2.1 above, go to the “Certificates” settings and select **Add**.

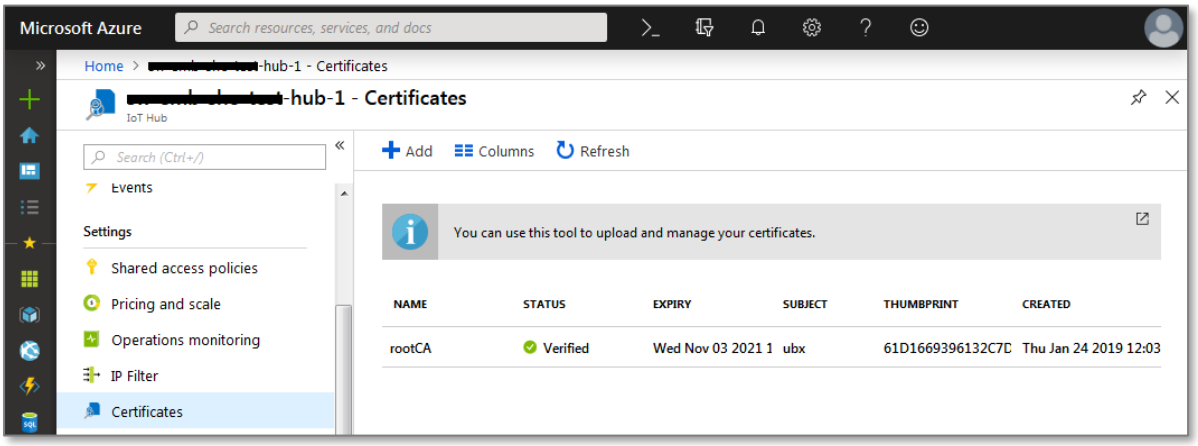


2. Select the certificate added in the previous step. To generate the verification code needed to generate a new certificate, select **Generate Verification Code** in the “Certificate Details” pane.

3. Generate the new client-side certificate using the same CA used previously in section 3.2.1, but this time use the Verification Code as the Common Name (CN). To generate a client-side certificate, see the procedures given in u-connectXpress Wi-Fi security application note, reference [4].



4. Upload the certificate generated in the previous step to the Verification Certificate in the Certificate Details pane and select “Verify”. The STATUS of the CA certificate changes to Verified.



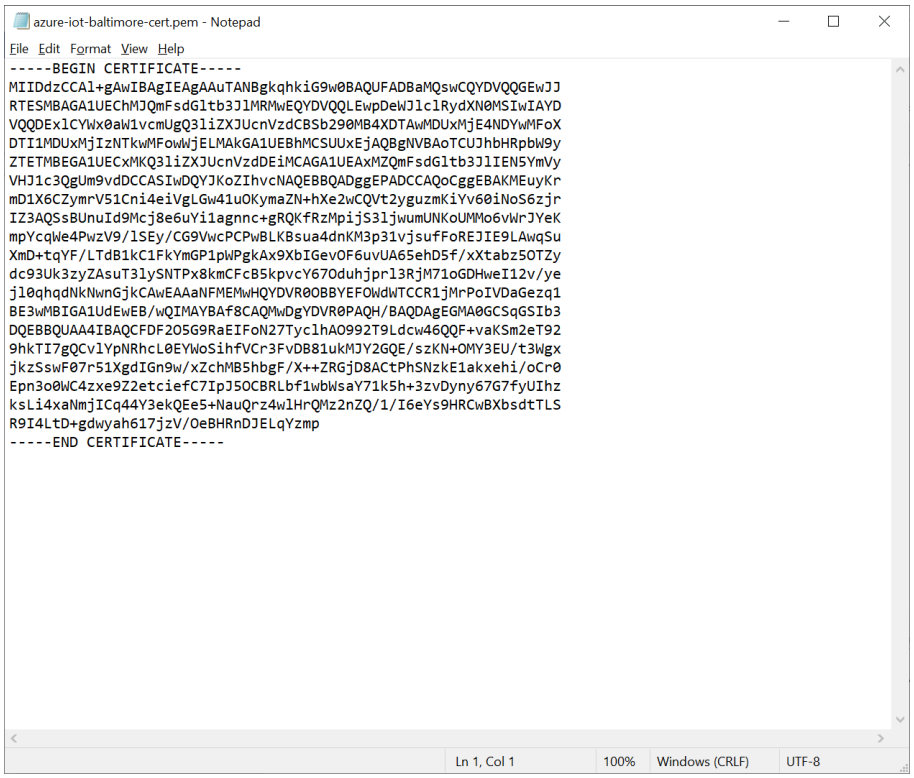
3.2.3 Prepare module certificates

1. Obtain the latest Azure IoT Hub server certificate. The certificate can found in a `.c-file` in the `certs` directory of the `Azure-iot-sdk-c` repository:

   https://github.com/Azure/azure-iot-sdk-c/blob/master/certs/certs.c

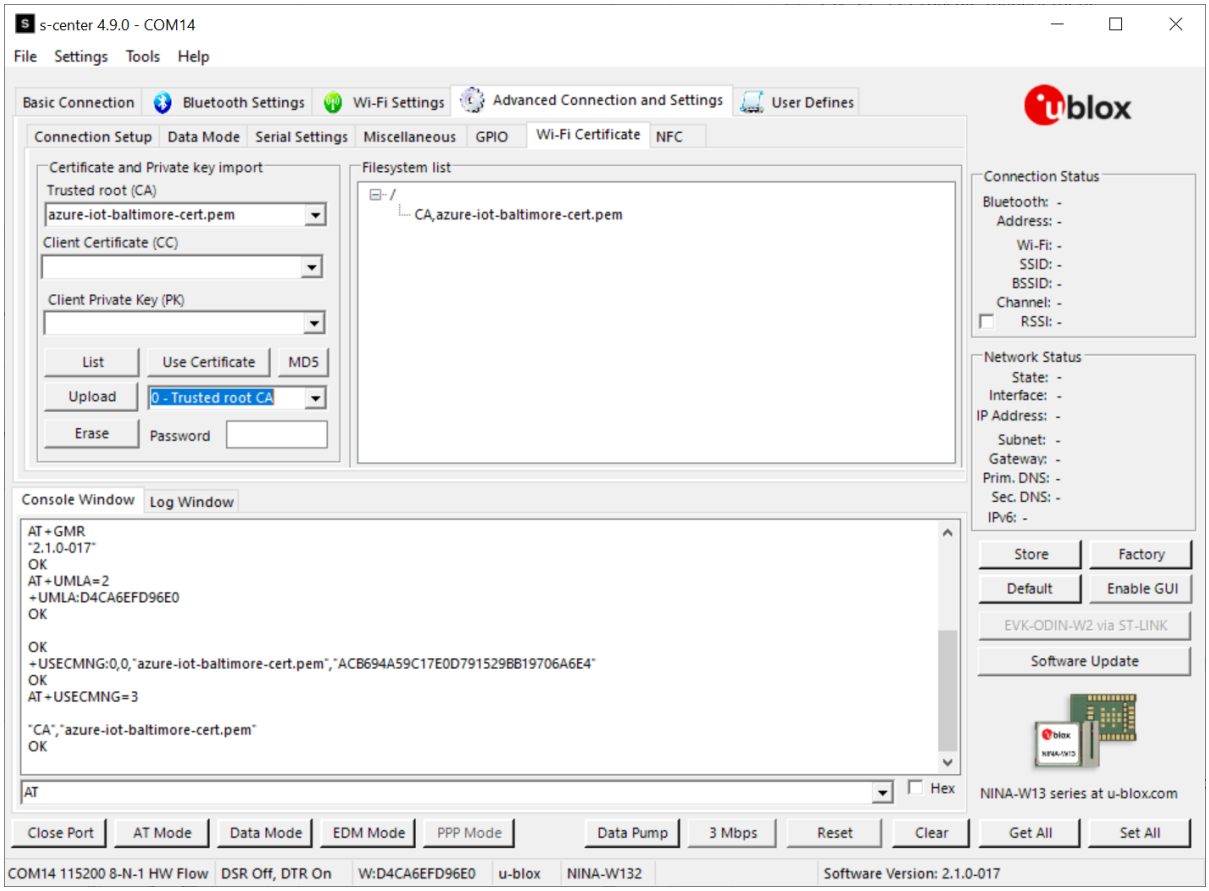
2. To convert the c-file to a local Azure certificate file containing the DigiCert Baltimore Root certificate, copy the certificate information from `certs.c` to an text editor. Include the lines: `----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`, and remove the quotation marks (`"`) at the beginning and end of every line. Also remove the `\r\n` characters at the end of every line.

3. Save the file as, for example, `azure-iot-baltimore-cert.pem`.



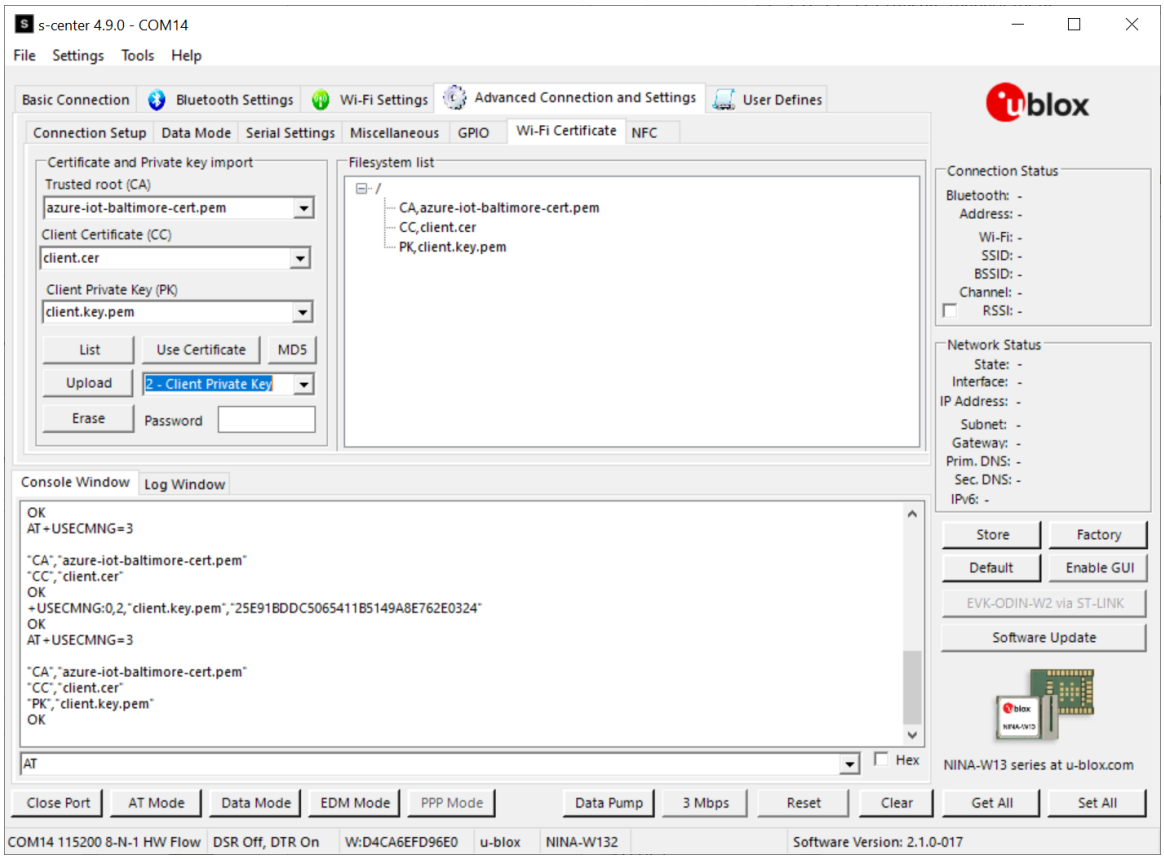
3.2.4 Install certificates on the module

1. Upload the local Azure certificate file to the module as a trusted root (CA) certificate using s-center, or the command `AT+USECMNG=0,0`.



☞ For further information about `AT+USECMNG`, see the u-connect AT commands manual [1] and u-connectXpress user guide [3].

2. Upload the client certificate (CC) and the client private key (PK) generated in section 0, using either:
   - AT commands `AT+USECMNG=0,1` (client certificate) and `AT+USECMNG=0,2` (private key).
   - s-center



☞ As s-center requires a file suffix `.crt`, `.cer`, or `.pem`, the files may need to be renamed.

3.2.5 Connect to the cloud

1. Use s-center or the command AT+UWSC and AT+UWSCA to set up a network connection to the internet on the module. See the u-connectXpress user guide [3] for details.
2. To publish and subscribe to events, connect to Azure IoT Hub using the commands AT+UDCP and MQTT. For further information about the mqtt scheme, see the u-connectXpress MQTT Application Note [2] The format of the AT+UDCP command for Azure is as follows:

```
AT+UDCP=mqtt://iothubhostname:8883/?client=device_id&user=iothubhostname/device_id
&ca=server_cert_name&cert=device_cert_name&privKey=device_key_name
&pt=devices/device_id/messages/events/
&st=devices/device_id/messages/devicebound/#
```

☞ In the URL, replace the *marked* sections above, as shown below:

Item	Description
iothubhostname	From the Hostname section in the Overview page in Azure IoT Hub.
device_id	The Device ID set to the device in Azure IoT Hub.
server_cert_name	The internal name given to the Azure IoT Hub server certificate when uploaded to the module.
device_cert_name	The internal name given to the client certificate when uploaded to the module.
device_key_name	The internal name given to the client private key when uploaded to the module.

Example

```
at+udcp=mqtt://iot-xxxx-test-hub-1.azure-devices.net:8883/?client=device1
&user=iot-xxxx-test-hub-1.azure-devices.net/device1
&ca=azure-iot-baltimore-cert.pem&cert=client.cer&privKey=client.key.pem
&pt=devices/device1/messages/events/
&st=devices/device1/messages/devicebound/#
```

3. Switch to Data mode and send data to/from the Azure IoT server; monitor the transferred data as mentioned in Appendix B.3.

☞ The # wildcard is not supported by ODIN-W26-7.0.0, or NINA-W13/NINAW15 prior to version 3.0.

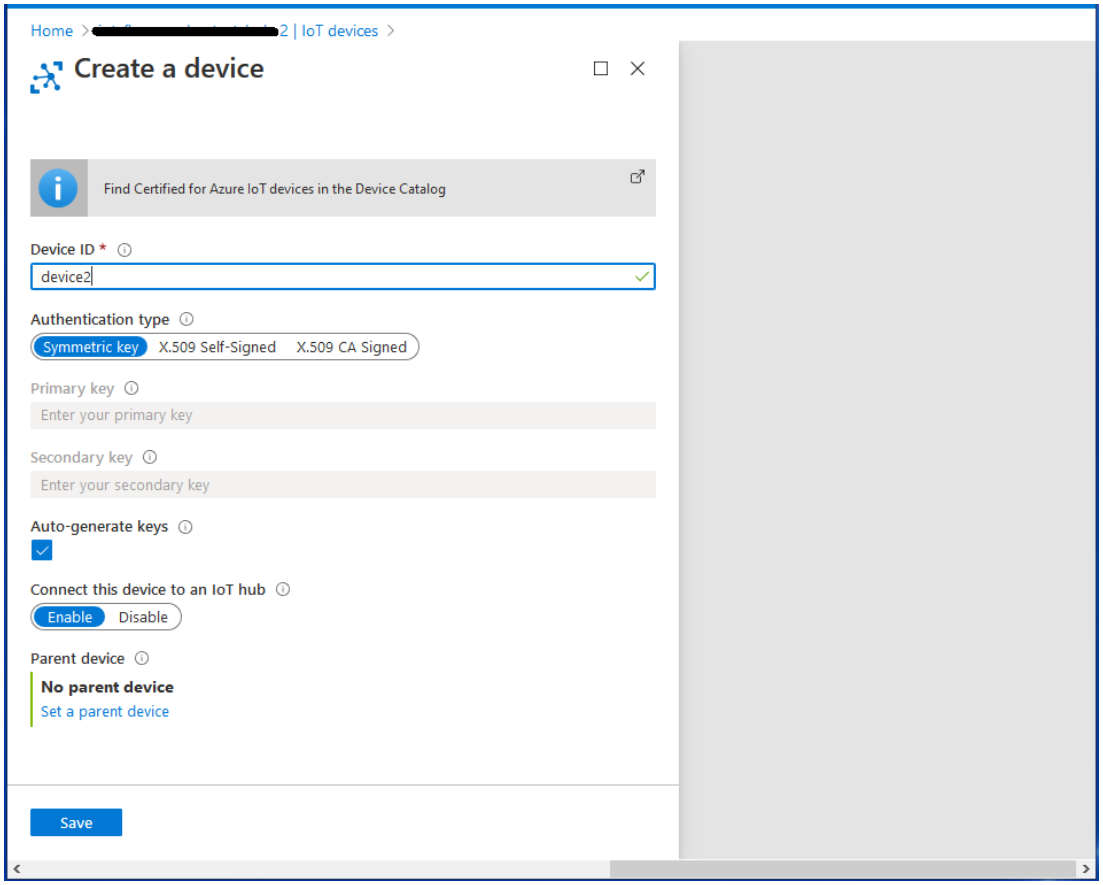
3.3 Configure a module with a Symmetric Key/SAS Token

☞ SAS Tokens are supported from on NINA-W13 and NINA-W15 starting with version 3.0.

If not already done, Create a new IoT Hub.

3.3.1 Create the first device

1. Go to the IoT Devices tool in the IoT hub Explorer section and select **New** or **Add** to register a new device.
2. Enter a unique Device ID, for example “device2”, the device’s serial number, or the result of `AT+UMLA=2` for your module.
3. Set the Authentication type to “Symmetric Key”



4. Select **Save** to include your new device in the list of IoT devices.

The device page on IoT Hub now contains the auto-generated primary and secondary keys, including the connection strings to use.

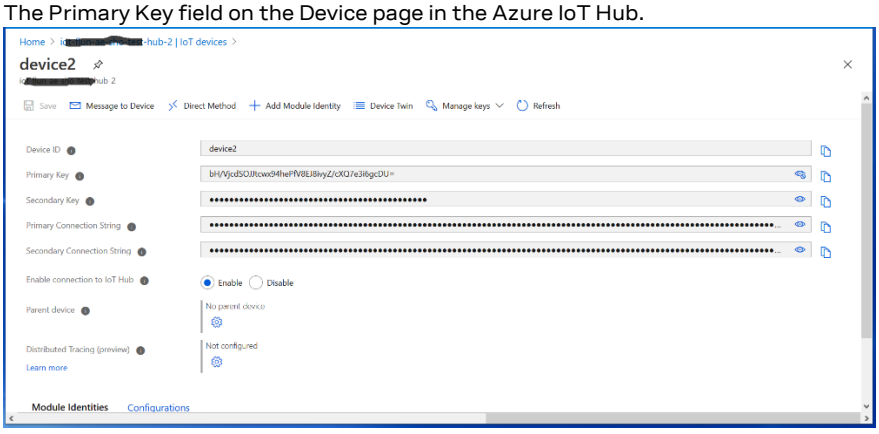
3.3.2 Connect to the cloud

1. Generate a SAS token using the algorithm defined by the code snippets included in the Microsoft Azure documentation https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security#security-tokens. The returned string is the complete SAS Token.

   A python program which generates a SAS Token could be written as:

```
from base64 import b64encode, b64decode
from hashlib import sha256
from time import time
from urllib import parse
from hmac import HMAC

IOTHUBHOSTNAME = "iothubhostname"
PRIMARY_KEY    = "primary_key"

def generate_sas_token(uri, key, policy_name, expiry=3600):
    ttl = time() + expiry
    sign_key   = "%s\n%d" % ((parse.quote_plus(uri)), int(ttl))
    signature  = b64encode(HMAC(b64decode(key), sign_key.encode('utf-8'),
                 sha256).digest())

    rawtoken   = {
        'sr'   :  uri,
        'sig'  : signature,
        'se'   : str(int(ttl))
    }

    if policy_name is not None:
        rawtoken['skn'] = policy_name

    return 'SharedAccessSignature ' + parse.urlencode(rawtoken)

res = generate_sas_token(IOTHUBHOSTNAME, PRIMARY_KEY, "")
print(res)
```

☞ In the above program, replace the *marked* sections as follows:

Item	Description
iothubhostname	From the Hostname section in the Overview page in Azure IoT Hub.
primary_key	The Primary Key field on the Device page in the Azure IoT Hub. 

Example

Invocation of:

```
generate_sas_token("iot-xxxx-test-hub-2.azure-devices.net",
"bH/VjcdSOJJtcwx94hePfV8EJ8ivyZ/cXQ7e3i6gcDU=", "")
```

Returns the SAS Token:

```
SharedAccessSignature
sr=iot-xxxx-test-hub-2.azure-devices.net
&sig=ix4nP1LaU%2FZq2Nk3OaBwJ0MikCPIxGxz2uiZ3KUxecM%3D&se=1591876214&skn=
```

☞ The SAS Token ***does*** contain both spaces, ampersands, equal and percent signs.

2. Use s-center or the commands `AT+UWSC` and `AT+UWSCA` to set up a network connection to the internet on the module. For further information, see the u-connectXpress user guide [3].
3. Use the command `AT+UDUV` to set an URL Value to the SAS Token:

```
AT+UDUV=0,"SharedAccessSignature          sr=iot-xxxx-test-hub-2.azure-devices.net
&sig=ix4nP1LaU%2FZq2Nk3OaBwJ0MikCPIxGxz2uiZ3KUxecM%3D&se=1591876214&skn="
```

4. Enable WiFi and connect to an internet-connected AP.
5. Use the command `AT+UDCP` command and MQTT to connect to Azure IoT Hub, publish and subscribe to events. For further information about the `mqtt:` scheme, see the u-connectXpress MQTT Application Note [2]. The format of the `AT+UDCP` command for Azure using a SAS Token is as follows:

```
AT+UDCP="mqtt://iothubhostname:8883/?ca=<ca certificate>
&client=device_id
&user=iothubhostname/device_id
&passwd=%%url_value_index
&pt=devices/device_id/messages/events/
&st=devices/device_id/messages/devicebound/#"
```

☞ In the URL, replace the *marked* sections above as follows:

Item	Description
iothubhostname	From the Hostname section in the Overview page in Azure IoT Hub.
device_id	The Device ID set to the device in Azure IoT Hub.
url_value_index	The URL Value index given as parameter to `AT+UDUV`.

Example

```
at+udcp="mqtt://iot-xxxx-test-hub-2.azure-devices.net:8883/
?ca=<ca certificate>
&client=device2
&user=iot-xxxx-test-hub-2.azure-devices.net/device2
&passwd=%%0
&pt=devices/device2/messages/events/
&st=devices/device2/messages/devicebound/#"
```

6. Switch to Data mode and send data to/from the Azure IoT server; monitor the transferred data as mentioned in Appendix B.3.

3.4 Device Provisioning Service (DPS)

To get started with the Azure IoT service, it is possible to use the IoT Hub device provisioning service (DPS) that enables zero-touch provisioning and configures the device connection to the cloud without requiring human intervention, allowing customers to configure multiple devices in a secure and scalable manner.

3.4.1 Create and Prepare the Azure IoT Hub Device Provisioning Services

When connected to the Azure DPS, the device is assigned to the IoT Hub created in Create a new IoT Hub. To create a new IoT Hub Device Provisioning Service using the Azure Portal:

1. In the Azure portal, select **+ Create a resource**.
2. From the **Categories** menu, select **Internet of Things** then **IoT Hub Device Provisioning Service**.
3. Select **Create** and fill the required information accordingly to your Azure plan and resources.
4. Select **Review + Create** to validate your provisioning service.
5. Select **Create**.

To Link the IoT hub and your Device Provisioning Service:

1. In the **Settings** menu, select **Linked IoT hubs**.
2. Select **+ Add**.
3. On the **Add link to IoT hub** panel, provide the following information:
4. **Access Policy:** Select **iothubowner** as the credentials for establishing the link with the IoT hub.
5. Select **Save**.

See also Azure Quick Start [6].

3.4.2 Configure Enrollment group

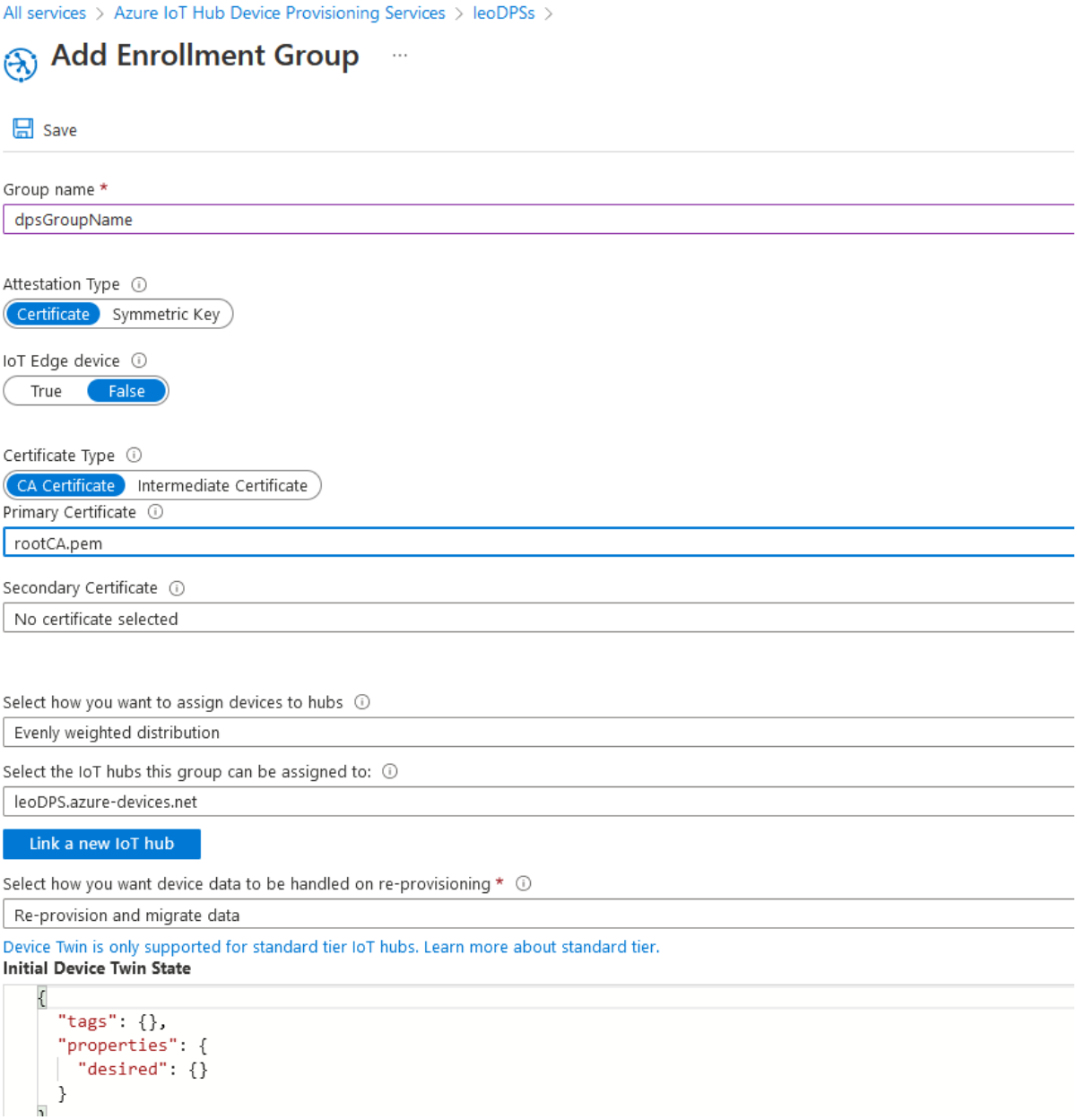
An Enrollment group is a group of devices that share a specific attestation mechanism. Enrollment groups support X.509 certificate or symmetric key attestation. Devices in an X.509 Enrollment group present X.509 certificates that have been signed by the same root or intermediate Certificate Authority (CA). The common name (CN) of each device's end-entity (leaf) certificate becomes the registration ID for that device.

The registration ID is used to uniquely identify a device registration with the Device Provisioning Service. The registration ID must be unique in the provisioning service ID scope. Each device must have a registration ID. The registration ID is a case-insensitive string (up to 128 characters long) of alphanumeric characters plus the special characters: '-', '.', '_', ':'. The last character must be alphanumeric or dash ('-').

1. From your DPS in Azure portal, select the “Manage Enrollments” tab. then select the “Add Enrollment” group button at the top.

2. In the Add Enrollment Group panel, enter the following information, then select Save.

   The device certificate can be signed with the root CA or an intermediate CA, this document uses the root CA.



For more information, see “Create an enrollment group” in the “Provision multiple X.509 devices” tutorial [5].

3.4.3 Configure IoT device connection to Device Provisioning Service (DPS)

For X.509-based attestation, the registration ID is set to the common name (CN) of the device certificate. For this reason, the common name must adhere to the registration ID string format.

1. Create client private key and certificate signing request (CSR):

```
$ openssl req -newkey rsa:2048 -nodes -keyout nina.key.pem -out nina.csr
```

   The common name (CN) of each device's end-entity (leaf) certificate becomes the registration ID for that device.

2. Sign the CSR with the client root CA registered on Azure during the Upload and verify the CA used to sign the client-side certificate procedure:
```
$ openssl x509 -req -in nina.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial
-out nina.pem -days 500 -sha256
```
3. Upload the client Private Key and the client certificate created above to the module, as described in Install certificates on the module.
4. Use the command AT+UDCP command and MQTT to connect to Azure IoT Hub Device Provisioning Service (DPS), publish and subscribe to events. For further information about the mqtt: scheme, see the u-connectXpress MQTT Application Note [2]. The format of the AT+UDCP command for Azure using a SAS Token is as follows:
```
AT+UDCP="mqtt://iothubhostname:8883/?
client=registration_id
&user=idScope/registrations/registration_id/api-version=2019-03-31
&pt=$dps/registrations/PUT/iotdps-register/?$rid={request_id}
&st=$dps/registrations/res/#”
&encr=3
&ca=<ca certificate>
&cert=<device certificate>
&privKey=<device private key>
```
5. In the URL, replace the *marked* sections above as follows:

Item	Description
iothubhostname	From the Hostname section in the Overview page in Azure IoT Hub.
registration_id	The Device ID set to the device in Azure IoT Hub.
idScope	is the ID Scope from the Azure IoT Hub Device Provisioning Service.

The example below is pseudo-code illustrating the flow for using DPS with u-blox u-connectXpress. It is assumed that the module is already provisioned with the certificates and connected to the Wi-Fi with internet connection:

```
AT+UDCP=mqtt://global.azure-devices-provisioning.net:8883/?
client=ninadps
&user=0ne0067E45C/registrations/ninadps/api-version=2019-03-31
&pt=$dps/registrations/PUT/iotdps-register/?$rid={request_id}
&st=$dps/registrations/res/#
&keepAlive=60
&encr=3
&ca=ca.pem
&cert=ninadps.pem
&privKey=ninadps.key.pem

+UDCP:1
OK

+UUDPC:1,2,6,0.0.0.0,0,40.113.176.170,8883
ATO1
OK
```

// Publish the following message:

```
"{\"payload\":\"\",\"registrationId\":\"registration_id\"}"

{"operationId":"5.b084dab098f0a900.5f5b44b2-ae20-4957-bd06-
40f9e8dcec3f","status":"assigning"}
```

3.4.4 Monitoring DPS status

In order to poll the DPS status, it is possible to publish to the following topic, where `operationId` is returned on the previous message, for example:

```
$dps/registrations/GET/iotdps-get-
operationstatus/?$rid={request_id}&operationId=5.b084dab098f0a900.5f5b44b2-ae20-4957-
bd06-40f9e8dcec3f
```

This example demonstrates the process using u-connectXpress in Data Mode. As it is not possible to start a new peer while in Data Mode, it is necessary to enter Command mode before creating the new peer. To enter Command mode, you send the escape sequence +++. Because the topic has an escape character “&”, the URL needs to be used with a placeholder:

```
AT+UDUV=0,"$dps/registrations/GET/iotdps-get-
operationstatus/?$rid={request_id}&operationId=5.b084dab098f0a900.5f5b44b2-ae20-4957-
bd06-40f9e8dcec3f”

AT+UDCP=mqtt://global.azure-devices-
provisioning.net:8883/?client=ninadps&user=0ne0067E45C/registrations/ninadps/api-
version=2019-03-
31&pt=%%0&st=$dps/registrations/res/#&keepAlive=60&encr=3&ca=ca.pem&cert=ninadps.pem&pri
vKey=ninadps.key.pem

+UUDPC:1,2,6,0.0.0.0,0,40.113.176.170,8883
ATO1
OK
```

Publish the following message “get operationstatus” to the status topic.

```
{\"operationId\":\"5.54dcdee4e6de2c9a.26a7ed00-28fd-4e98-bb5c-
b464a00262fd\",\"status\":\"assigned\",\"registrationState\":{\"x509\":{\"enrollmentGrou
pId\":\"enrolGroup\"},\"registrationId\":\"dpsdev\",\"createdDateTimeUtc\":\"2022-07-
01T16:46:50.830016Z\",\"assignedHub\":\"leoDPS.azure-
devices.net\",\"deviceId\":\"dpsdev\",\"status\":\"assigned\",\"substatus\":\"initialAss
ignment\",\"lastUpdatedDateTimeUtc\":\"2022-07-
01T16:46:51.0919787Z\",\"etag\":\"IjhkMGJlMTE3LTAwMDAtMDEwMC0wMDAwLTYyYmYyNGZiMDAwMCI=\"
}}
```

In the received message, "`status":"assigned`" indicates that the device has been transferred to the configured IoT Hub. In this example, `"assignedHub":"leoDPS.azure-devices.net"` provides the necessary confirmation. The device can also be seen in **Devices** under the **Azure IoT Hub**:

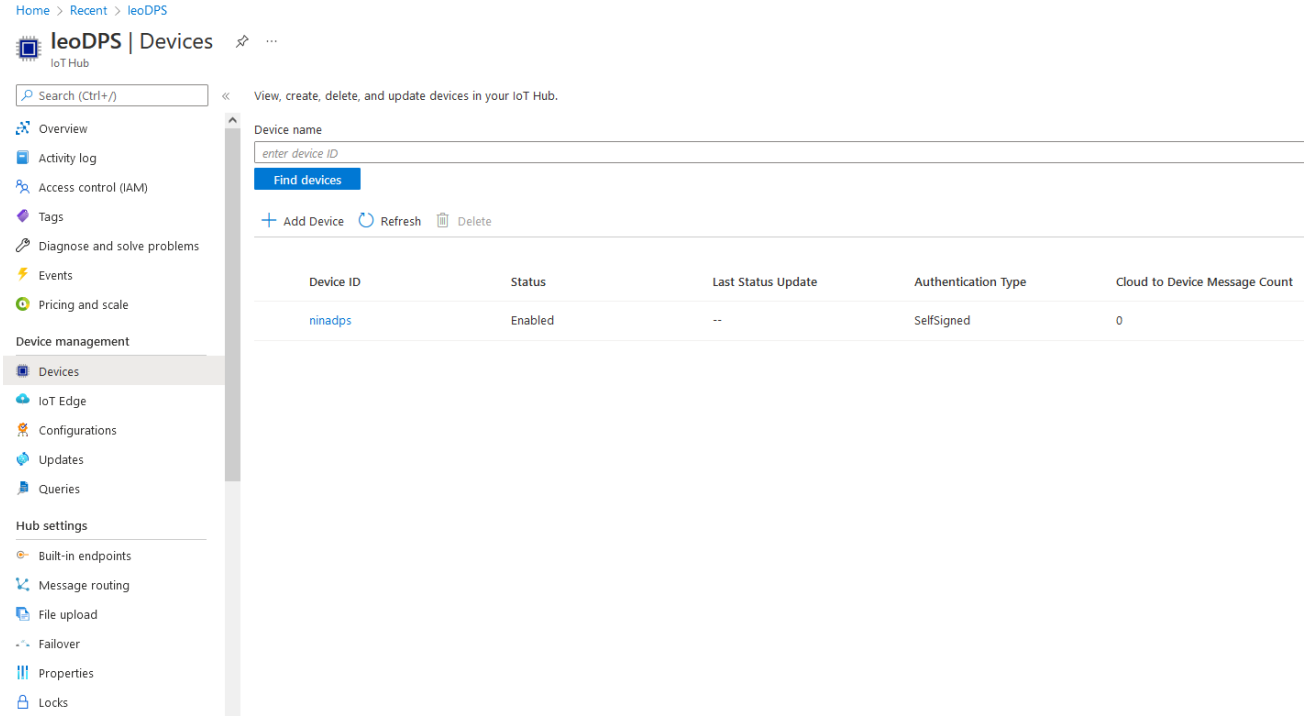


Figure 1: Monitoring DPS status example

The following example includes the log from the DPS process using NINA-W15 and the python script example found at https://github.com/u-blox/u-connectXpress_azure_device_provisioning_services

```
$ python .\azure_dps.py COM44

com44 open
 18ms -> AT+UFACTORY
 400ms <- AT+UFACTORY
OK
 401ms -> AT+CPWROFF
 416ms <- AT+CPWROFF
OK
 2804ms -> AT+USECMNG=0,0,ca.pem,1261
 3836ms <- AT+USECMNG=0,0,ca.pem,1261
>
 3836ms -> -----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
 5859ms <- +USECMNG:0,0,"ca.pem","ACB694A59C17E0D791529BB19706A6E4"
OK
 5860ms -> AT+USECMNG=0,1,cert.pem,1224
 6923ms <- AT+USECMNG=0,1,cert.pem,1224
>
 6923ms -> -----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
 8938ms <- +USECMNG:0,1,"cert.pem","9A1F95B8D512FA0ED075DD2B794ECB2A"
OK
 8941ms -> AT+USECMNG=0,2,key.pem,1704
 10033ms <- AT+USECMNG=0,2,key.pem,1704
>
 10034ms -> -----BEGIN PRIVATE KEY-----
...
-----END PRIVATE KEY-----
```

```
 12057ms <- +USECMNG:0,2,"key.pem","695D5D1EBB1DDBD0C6F2C128BAEB7F34"
OK
 12057ms -> ATO2
 12067ms <- ATO2
OK
 14076ms -> 0xAA00120044 'AT+UWSC=0,2,SSID\r' 0x55
 14089ms AT response: OK
 14091ms -> 0xAA00100044 'AT+UWSC=0,5,2\r' 0x55
 14105ms AT response: OK
 14106ms -> 0xAA001F0044 'AT+UWSC=0,8,PASSWORD\r' 0x55
 14121ms AT response: OK
 14122ms -> 0xAA000F0044 'AT+UWSCA=0,3\r' 0x55
 14153ms AT response: OK
 17017ms AT event: +UUWLE:0,802AA8035ADE,11
 17065ms AT event: +UUNU:0
 19092ms AT event: +UUNU:0
 22100ms -> 0xAA011F0044 'AT+UDCP=mqtt://global.azure-devices-
provisioning.net:8883/?client=ninadps&user=0ne0067E45C/registrations/ninadps/api-
version=2019-03-31&pt=$dps/registrations/PUT/iotdps-
register/?$rid={request_id}&st=$dps/registrations/res/#&encr=3&keepAlive=60&ca=ca.pem&ce
rt=cert.pem&privKey=key.pem\r' 0x55
 22440ms AT response: +UDCP:2
OK
 24712ms Connect event IPv4
        Channel id: 0
 25726ms AT event: +UUDPC:2,2,6,0.0.0.0,0,20.43.44.164,8883
 25726ms -> 0xAA002C0036 '\x00{"payload":"","registrationId":"ninadps"}' 0x55
 26168ms Data event:
        Channel id: 0
        Data: {"operationId":"5.b084dab098f0a900.076859d0-7180-40b8-9d06-
bf611e398ec0","status":"assigning"}

 26169ms -> 0xAA00950044 'AT+UDUV=0,$dps/registrations/GET/iotdps-get-
operationstatus/?$rid={request_id}&operationId=5.b084dab098f0a900.076859d0-7180-40b8-
9d06-bf611e398ec0\r' 0x55
 26197ms AT response: OK
 26197ms -> 0xAA00CD0044 'AT+UDCP=mqtt://global.azure-devices-
provisioning.net:8883/?client=ninadps&user=0ne0067E45C/registrations/ninadps/api-
version=2019-03-31&pt=%%0&encr=3&keepAlive=60&ca=ca.pem&cert=cert.pem&privKey=key.pem\r'
0x55
 26234ms AT response: +UDCP:4
OK
 26235ms Connect event IPv4
        Channel id: 1
 27245ms AT event: +UUDPC:4,2,6,0.0.0.0,0,20.43.44.164,8883
 27245ms -> 0xAA00160036 '\x01get operationstatus' 0x55
 27466ms Data event:
        Channel id: 0
        Data: {"operationId":"5.b084dab098f0a900.076859d0-7180-40b8-9d06-
bf611e398ec0","status":"assigned","registrationState":{"x509":{"enrollmentGroupId":"enro
lGroup"},"registrationId":"ninadps","createdDateTimeUtc":"2022-07-
04T08:28:35.8694014Z","assignedHub":"leoDPS.azure-
devices.net","deviceId":"ninadps","status":"assigned","substatus":"initialAssignment","l
astUpdatedDateTimeUtc":"2022-07-
04T08:28:36.154464Z","etag":"IjYwMDA4OTJjLTAwMDAtMDEwMC0wMDAwLTYyYzJhNGI0MDAwMCI="}}

Successfully Assigned
 27468ms -> 0xAA000D0044 'AT+CPWROFF\r' 0x55
 27482ms AT response: OK
```

3.5 Integration with Azure IoT Explorer

IoT Explorer is a tool provided by Azure to set up the connection and monitoring of the module to Azure Cloud applications. The software is provided as a free download and is available at https://github.com/Azure/azure-iot-explorer/releases.

The main features are:

- Simple Azure connection configuration
- Device twin and direct method functions
- Real-time Azure monitoring and alarm management
- Advanced diagnostics tools

For more information about the IoT Explorer functions, see also the official Azure IoT Explorer page.

3.5.1 IoT Hub connection

The first time that Azure IoT Explorer is executed, the application is prompted for the user's IoT hub connection string. After providing the connection string, select **Connect**.

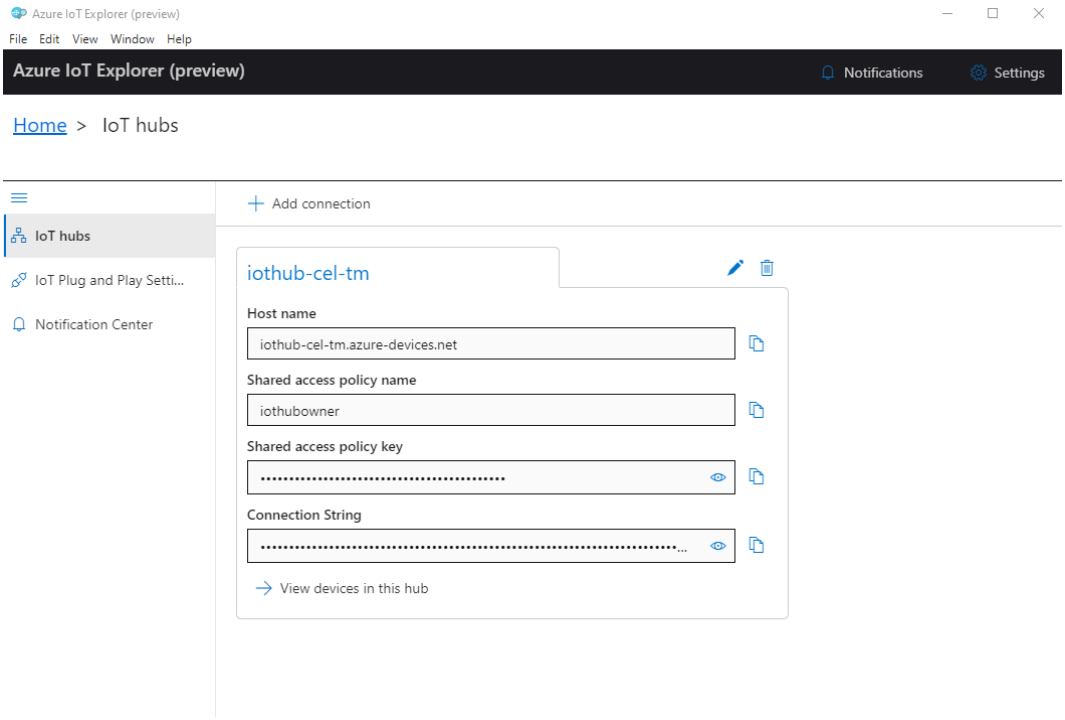


Figure 2: IoT hubs

3.5.2 Viewing Devices

When the tool is connected to the IoT hub, all device identities registered with the IoT hub are displayed in the **Devices** dialog, as shown in Figure 3. Select the Device ID for further information about the device.

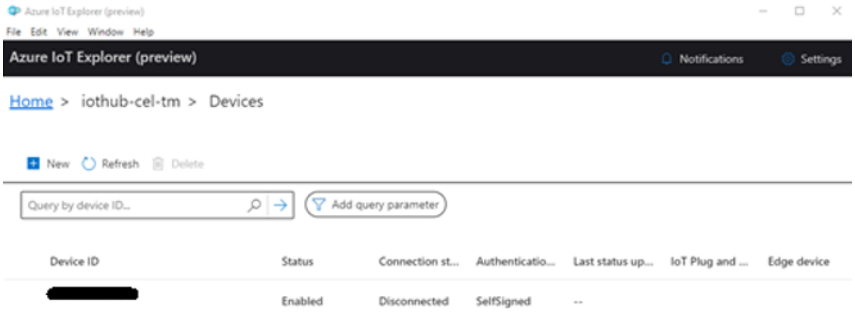


Figure 3: Azure IoT Explorer – Devices dialog

3.5.3 Registering a new device

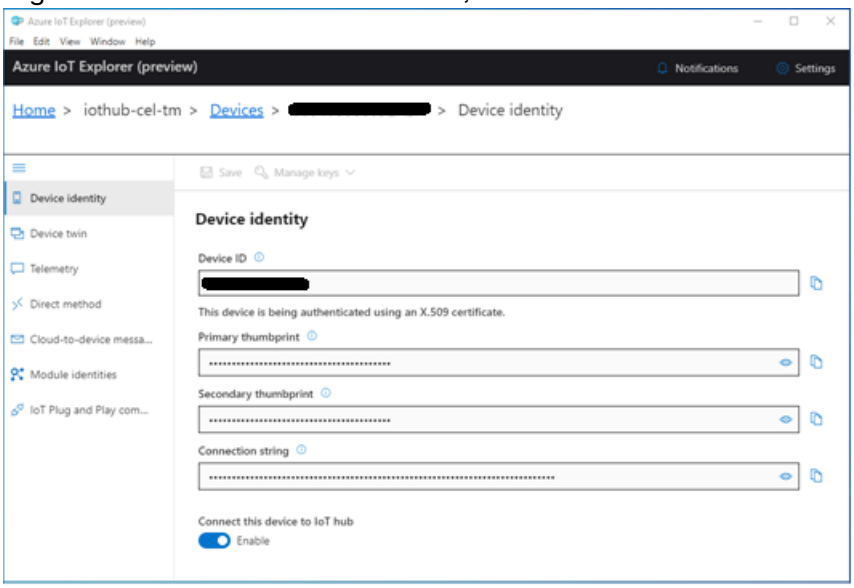
1. Select **New** to register a new device with the Azure IoT hub.
2. Enter a **Device ID**.
3. Using the default device settings, select auto-generate authentication keys and enable the connection to the IoT hub.

3.5.4 Deleting a device

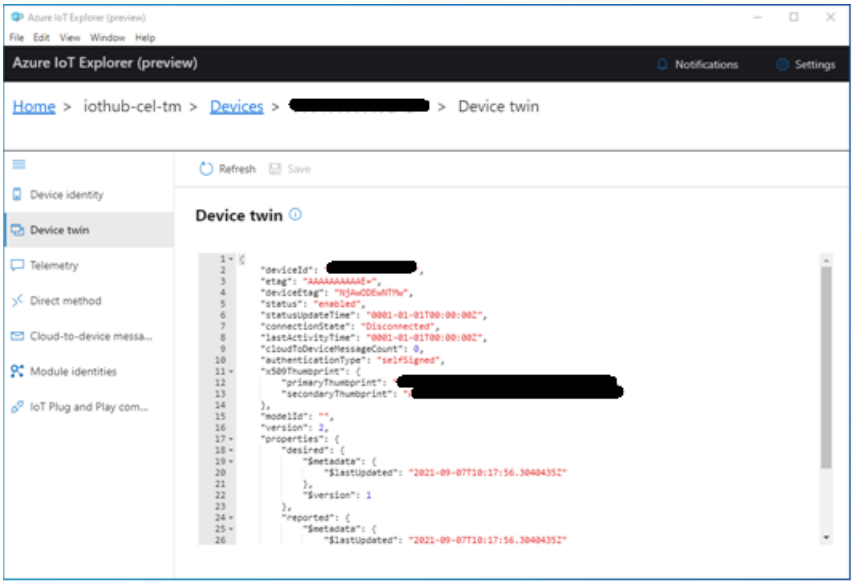
1. Select the device to be deleted and check the details of the device are correct.
2. Select Delete.

3.5.5 Interacting with a device

1. In the **Device identity** dialog, select a value in the **Device ID** column to view details about the registered device. For each device, there are two main sections: **Device** and **Digital Twin**.



2. Select the **Device twin** tab to access information about the device.



Appendix

A Glossary

Abbreviation	Definition
API	Application Programming Interface
AWS	Amazon Web Services
CA	Certificate Authority
CN	Common Name
CSR	Certificate Signing Request
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
TLS	Transport Layer Security

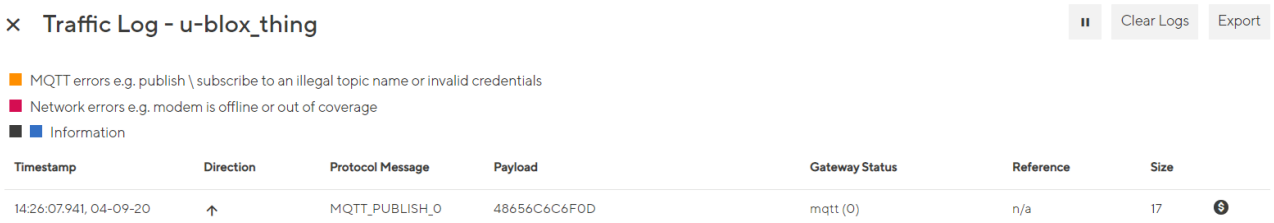
Table 1: Explanation of the abbreviations and terms used

B Monitoring messages to/from the cloud

B.1 In Thingstream platform

B.1.1 Device-to-cloud

Monitoring connection events in Things / [thing_name] / Thing Details / Traffic Logs:



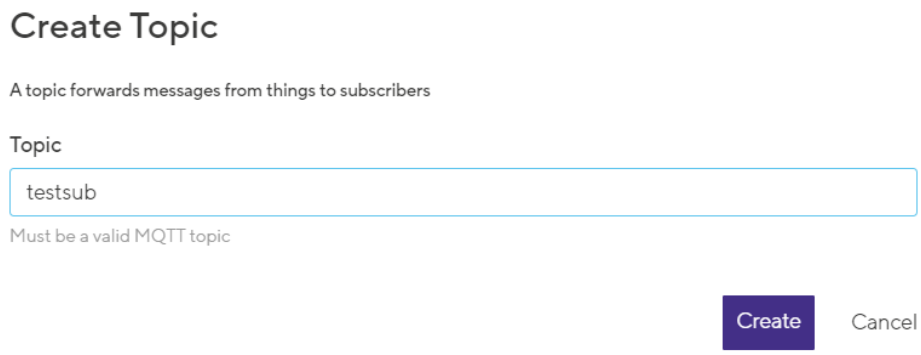
The received data (publish topic) payload is displayed in Traffic Log.

On the received data it is possible to so more advanced things, this can be made by creating a flow that handles the received data and performs things on the received data.

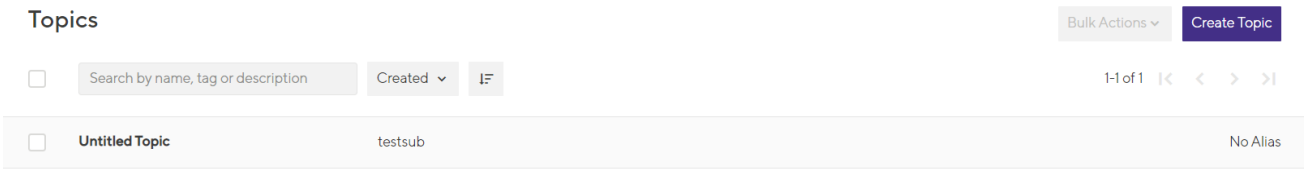
To learn more about flows, see https://developer.thingstream.io/guides/platform-portal/flows

B.1.2 Cloud-to-device

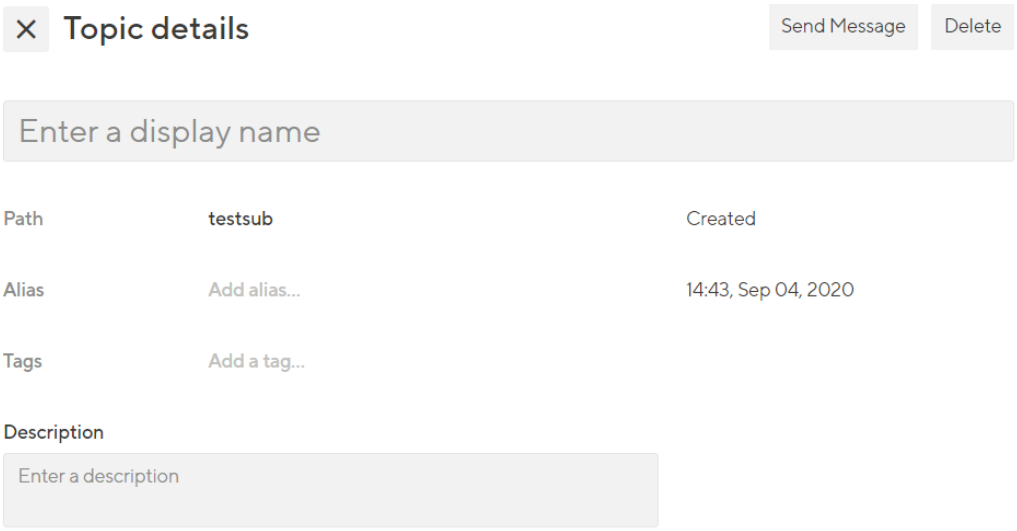
1. Send a message using Topic and select Create Topic:



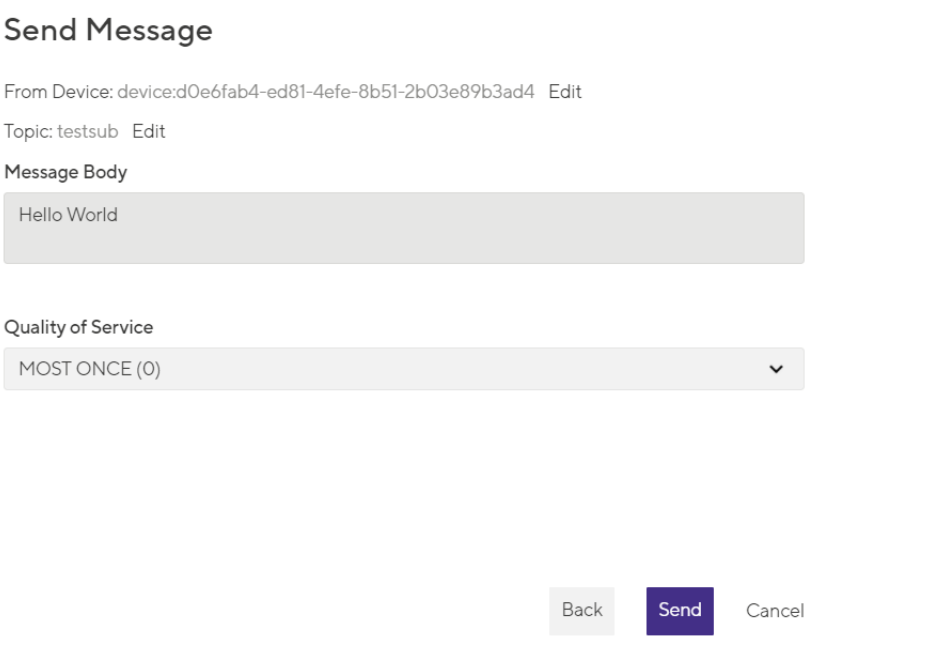
2. Use a valid Subscription topic for the connection, for example “testsub”

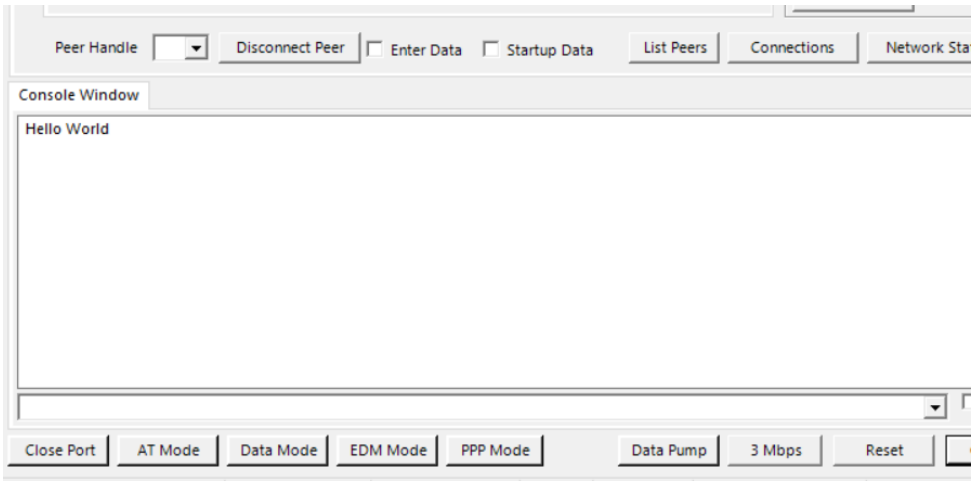


3. Select Topic



4. Select your Quality of Service and the Message Body then click on Send Message



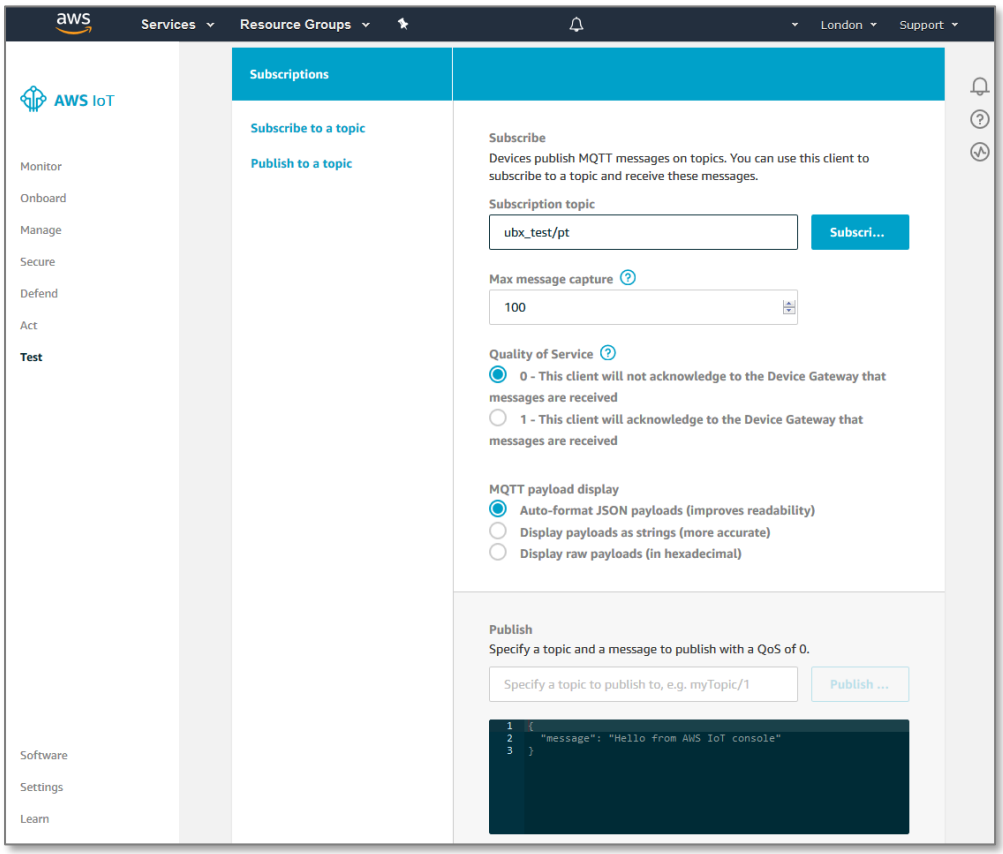


B.2 In AWS IoT Core

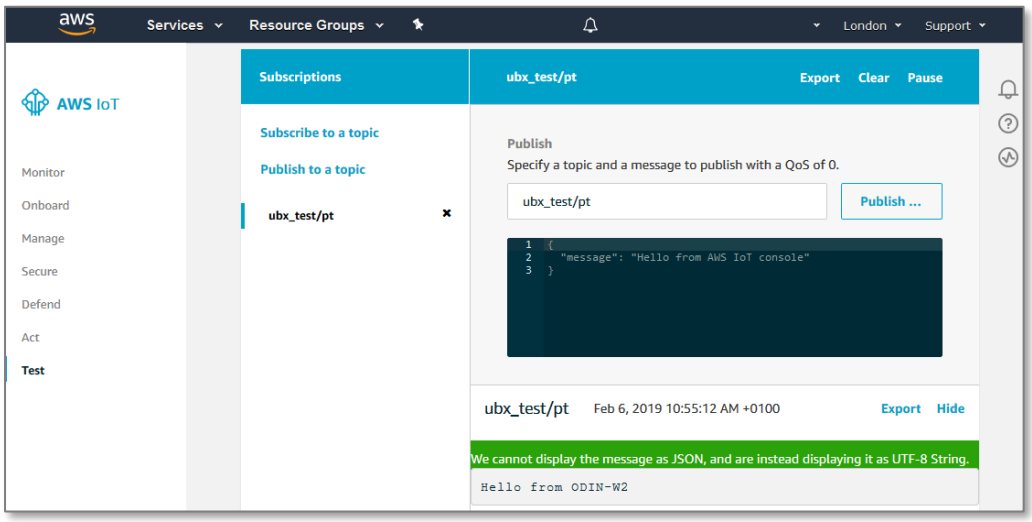
AWS IoT includes an MQTT client that can be used to monitor the MQTT messages sent by a connected Thing – as well as send messages to a connected Thing.

B.2.1 Device-to-cloud

1. Go to “Test” and connect the MQTT client to the AWS IoT server. Subscribe to the topic on which the Thing publishes, for example `ubx_test/pt`.

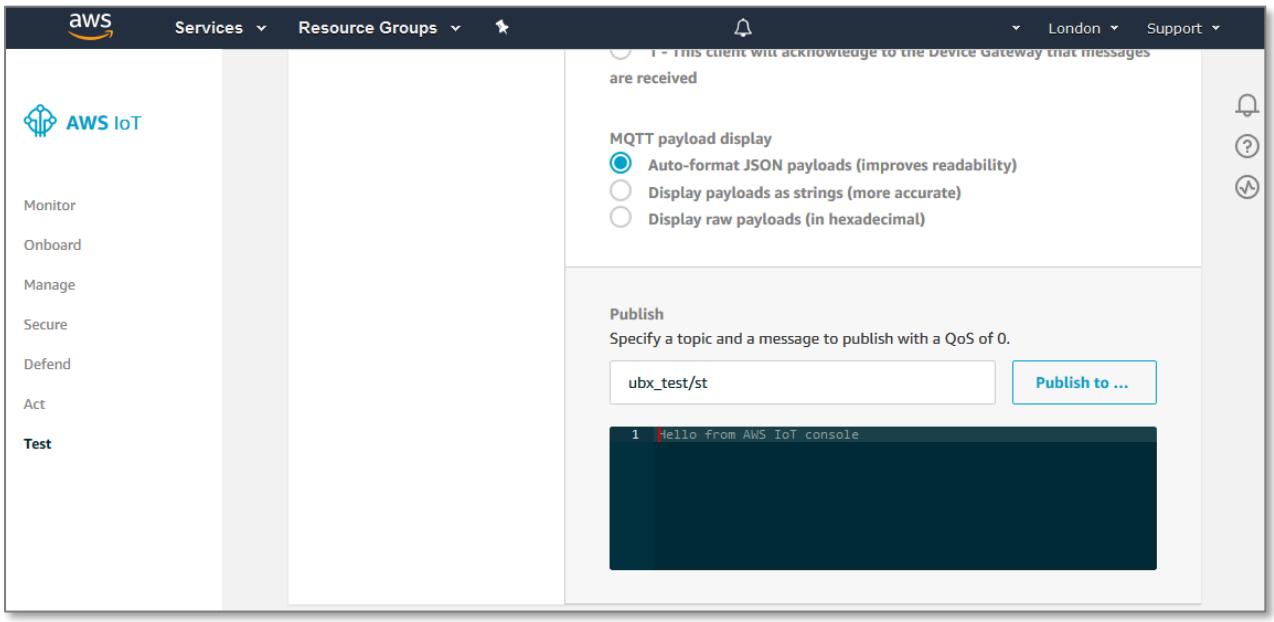


2. Monitor the device-to-cloud messages.

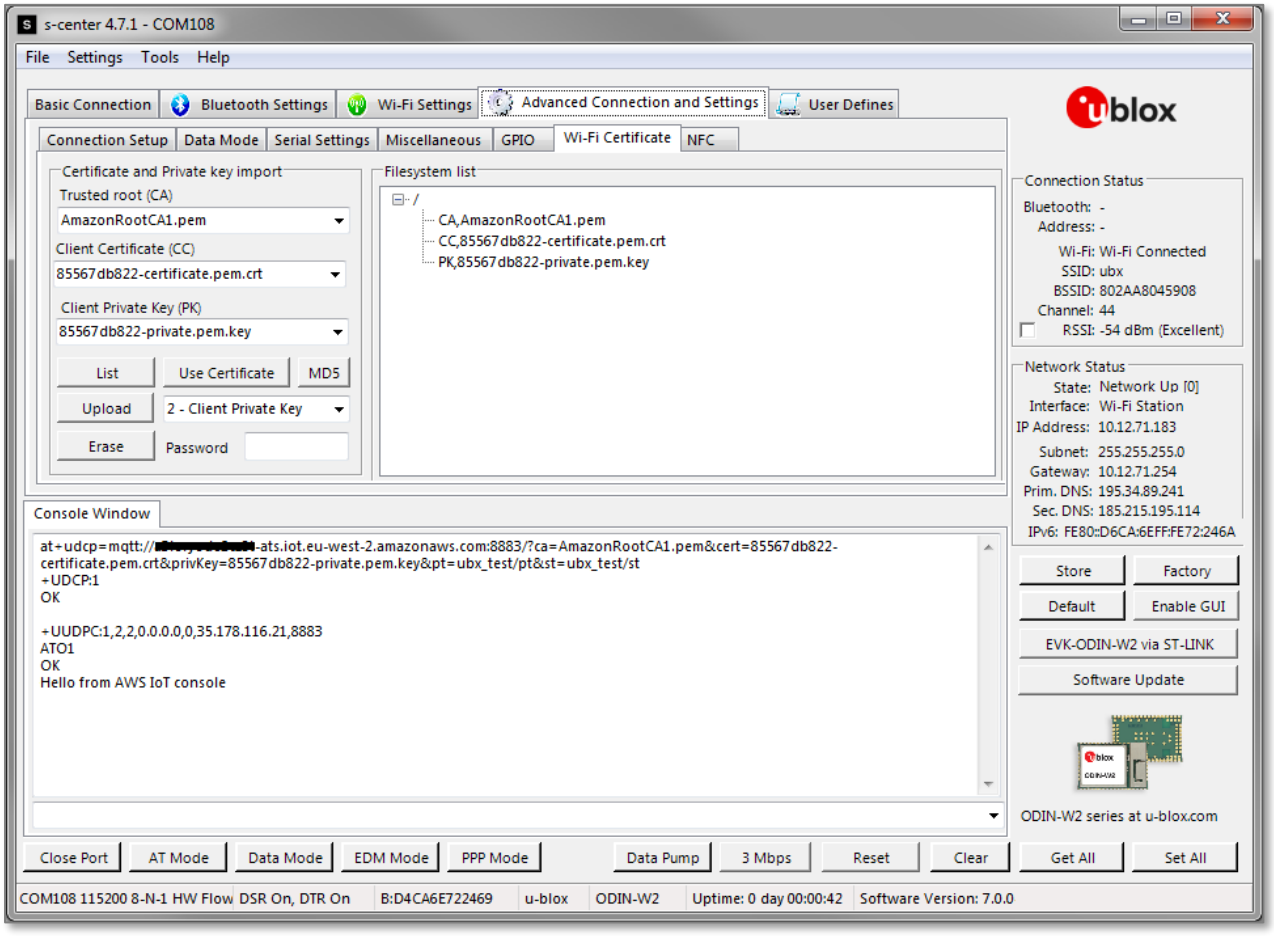


B.2.2 Cloud-to-device

1. Use the AWS IoT console to publish a message. Go to "Test" and start/connect the MQTT client to the AWS IoT server.
2. Define the topic to which the Thing subscribes, e.g., `ubx_test/st`, and select **Publish to...**.



3. Monitor the AWS-to-device messages.

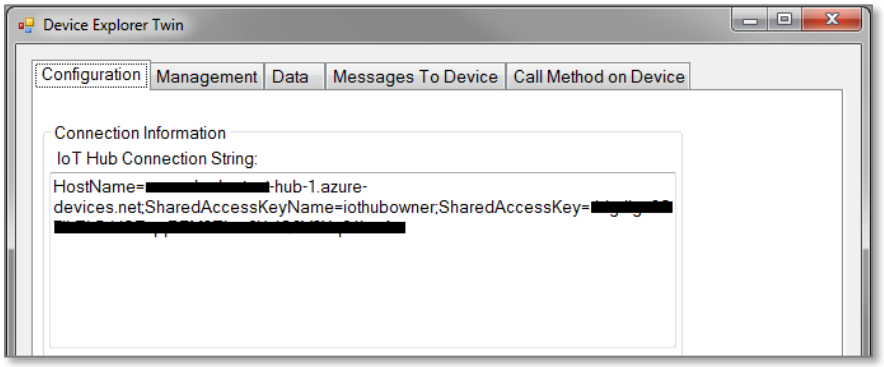


B.3 In Azure IoT Hub

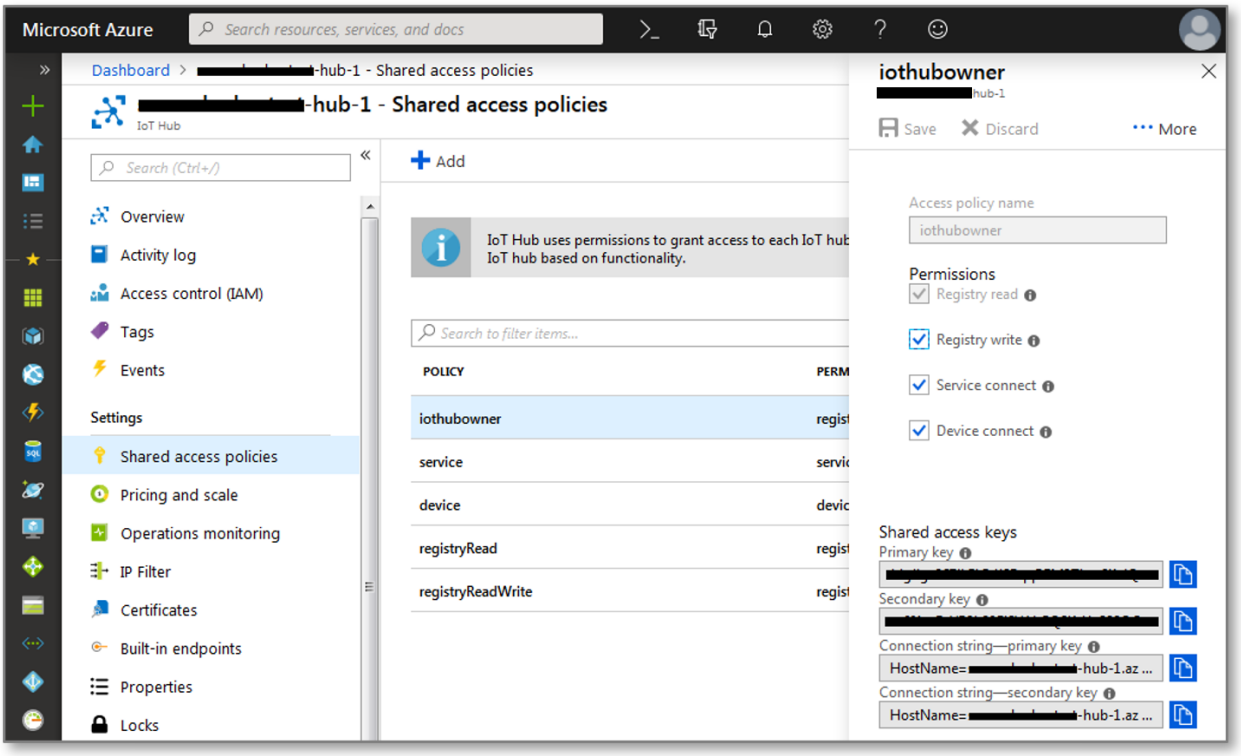
The Device Explorer tool is used to monitor messages between the device and the Azure IoT Hub. A pre-built version of the Device Explorer for Windows can be downloaded from: https://github.com/Azure/azure-iot-sdk-csharp/releases/tag/2019-9-11.

In this URL, scroll down for the SetupDeviceExplorer.msi

1. In the Device Explorer tool, go to the **Configuration** tab and add the Connection String for your Hub. The Connection String can be found in the IoT Hub.

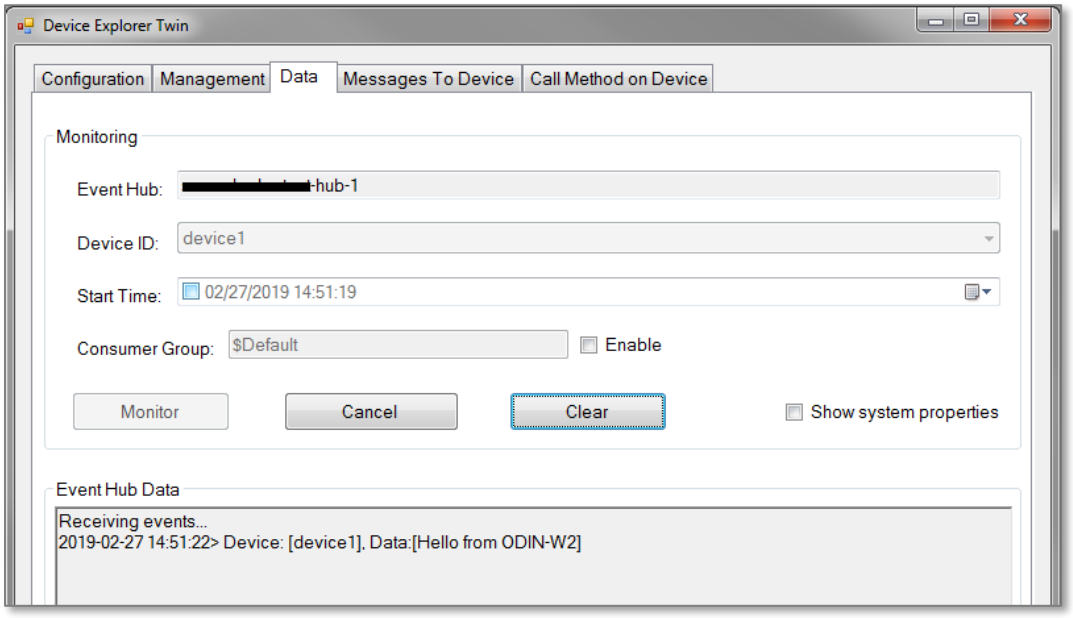


2. Go to **Settings / Shared access policies** and select the Policy **iothubowner**.



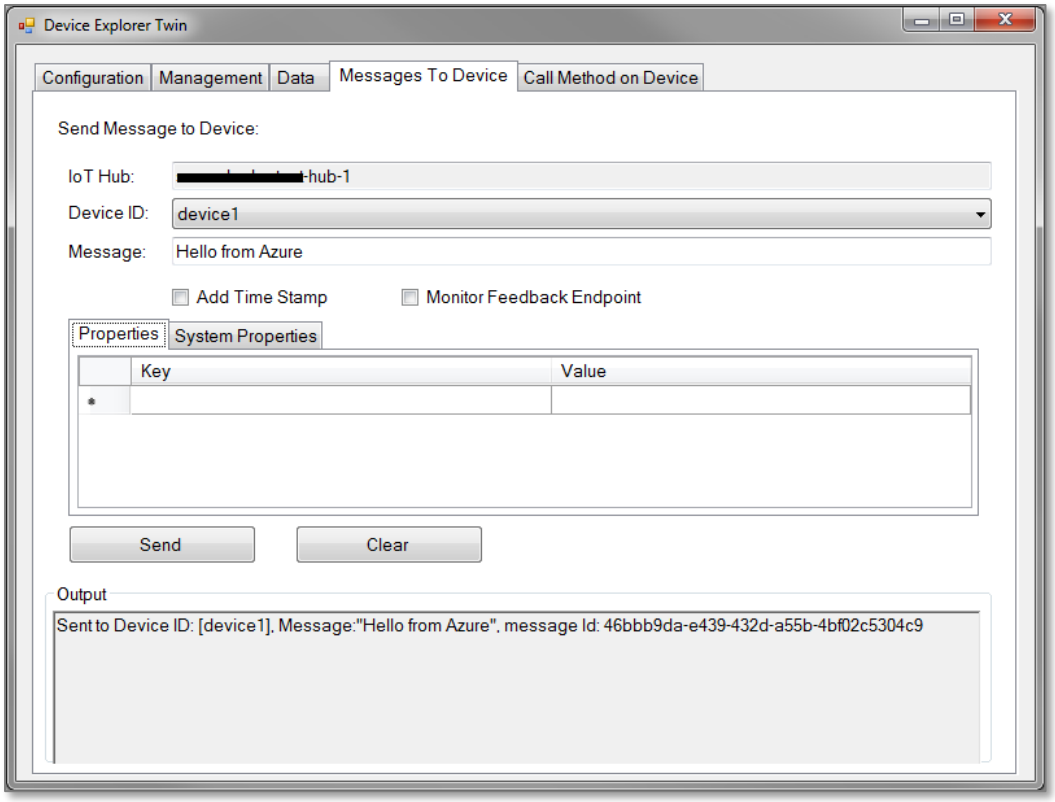
B.3.1 Device-to-cloud

In the Device Explorer tool, go to the Data tab and select the Device ID of the device to monitor (for example “device1”) and select **Monitor**.



B.3.2 Cloud-to-device

1. In the Device Explorer tool, go to the “Message to Device” tab and select the Device ID of the device to send message to (for example “device1”). Type a message in the Message text box and select **Send**.



2. Monitor the Cloud-to-Device messages.

Related documents

[1] u-connectXpress AT commands manual, UBX-14044127
[2] u-connectXpress MQTT application note, UBX-19005066
[3] u-connectXpress user guide, UBX-16024251
[4] u-connectXpress Wi-Fi security application note, UBX-20012830
[5] Tutorial: Provision multiple X.509 devices using enrollment groups
[6] Quickstart: Set up the IoT Hub Device Provisioning Service with the Azure portal

Revision history

Revision	Date	Name	Comments
R01	19-Mar-2019	cmag	Initial release.
R02	30-Oct-2019	flun	Included NINA-W13 v2.1.x, NINA-W15 v1.0.x as applicable products. Removed references to ODIN-W2 in the text, where also applicable to NINA-W1 products. Added links to related documents.
R03	12-Oct-2020	flun	Added support for Thingstream Updated chapter 2, Configuring Azure IoT Hub. Added support for SAS Tokens. Updated AWS policy example. Confirmed that NINA-W1 and ODIN-W2 modules have passed the AWS Device Qualification Program. Moved Appendix C to u-connectXpress Wi-Fi security application note [4].
R04	17-Feb-2021	flun	Included support for NINA-W156, ODIN-W263.
R05	9-Dec-2021	mhan	Removed IBM Watson IoT configuration chapter as IBM have now discontinued this service.
R06	15-Jul-2022	ldas	Updated Device Provisioning Service (DPS) and Integration with Azure IoT Explorer.

☞ For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Contact

For further support and contact information, visit us at www.u-blox.com/support.