Base station MQTT server

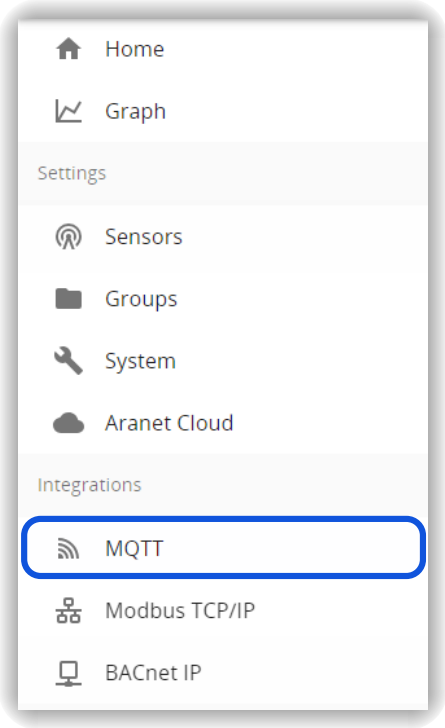
Seamlessly integrating with MQTT, the industry-standard communication protocol, our Base station empowers you to effortlessly connect your Aranet sensors to existing automation and control systems, enhancing efficiency and streamlining operations like never before.

Overview

MQTT is a lightweight, efficient messaging protocol designed for communication between devices in constrained environments. It follows a publish-subscribe pattern, where devices communicate through a central broker. Publishers send messages to specific topics, and subscribers receive those messages based on their subscriptions.

Configuration settings in base station WebGUI

To enable the MQTT server feature on the Aranet base station, in the WebGUI navigate to the main menu sidebar item shown in the screenshot below. Please note that use of the MQTT server feature requires an appropriate licence file to be uploaded on the base station.



The following items are shown in the MQTT configuration panel as indicated in the screenshot:

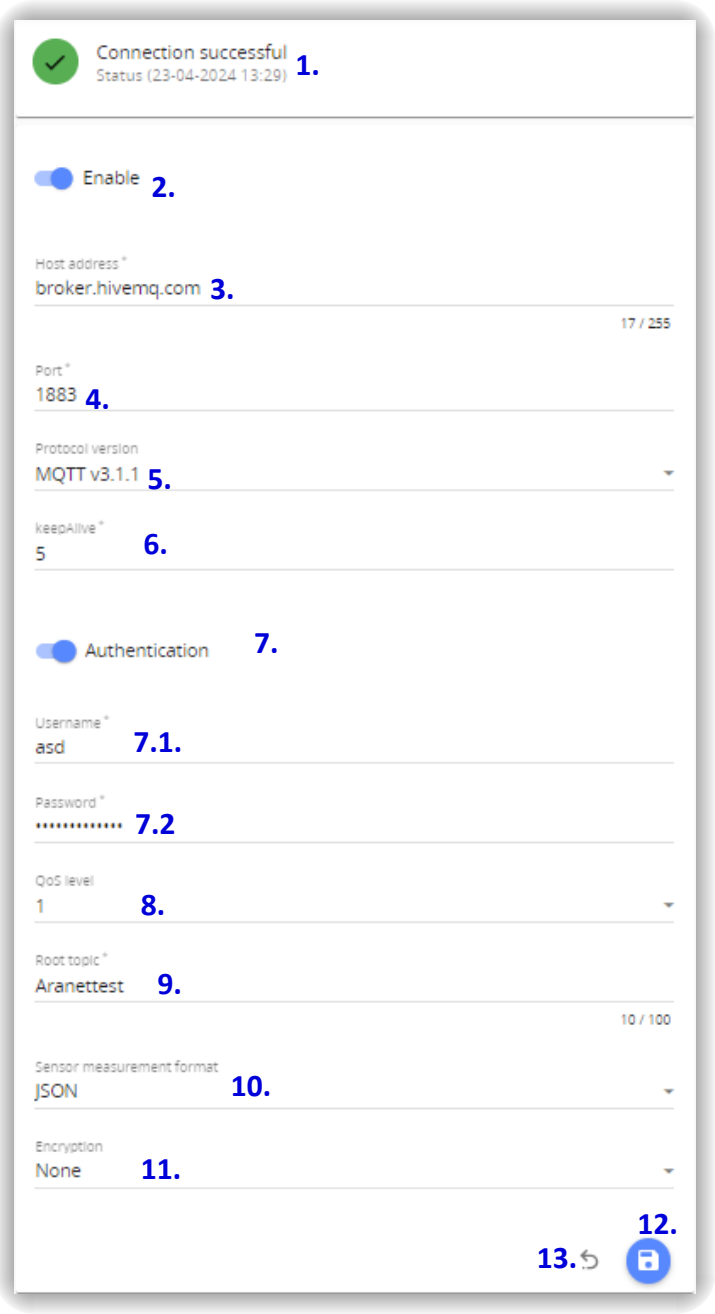
1. Status of the MQTT connection
2. Enable/disable the MQTT data transmission
3. A field to specify the *Host address*
4. The *TCP port* used for the connection to the MQTT broker.

NOTE: The most common ports are 1883 or 8883

5. The *MQTT protocol version* used for the connection to the MQTT broker

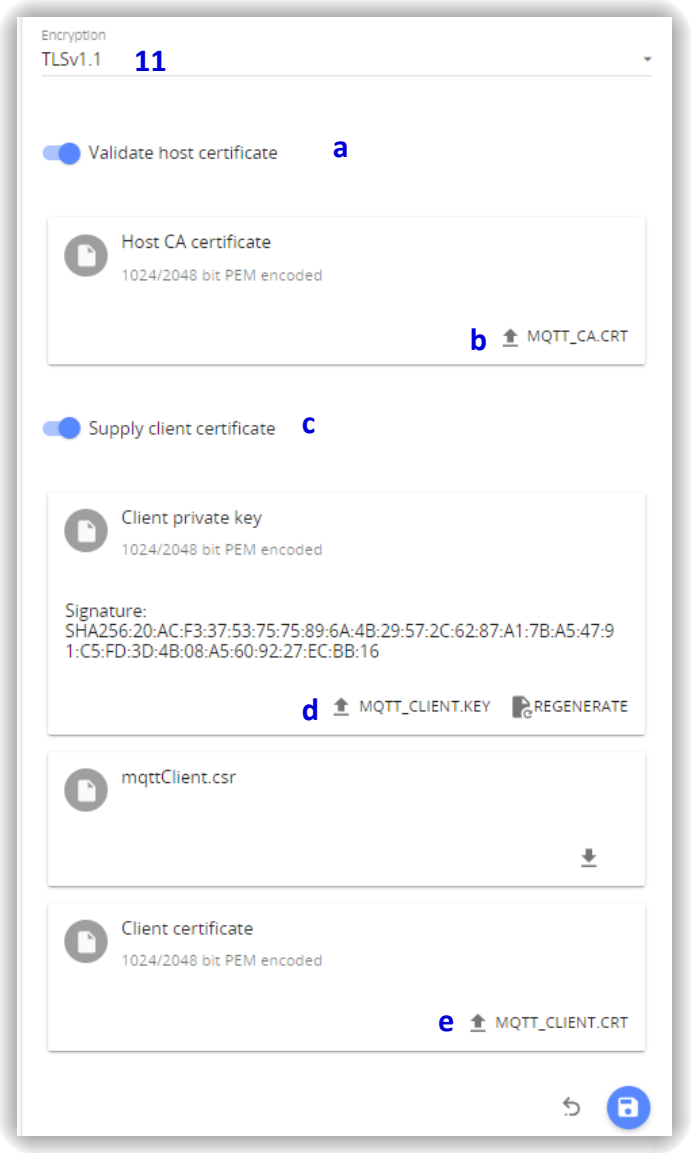
NOTE: The broker should support this version

6. A field to specify the *keepAlive interval*
7. Allows enabling of additional authentication for the connection to the MQTT broker
   (a) Username for the Authentication
   (b) Password for the Authentication
8. *Quality of Service level* (0, 1 or 2) for MQTT message delivery to the MQTT broker
9. The *root topic* name with which MQTT messages will be published from the Aranet PRO base station.
10. The data format *raw*, *JSON* or *Azure* in which messages will be published
11. Additional *encrypted certificates* (TLS version 1.1, 1.2 or 1.3) to be used for a more secure connection to the MQTT broker;
12. A button to save configuration changes.
13. A button to revert to the previous configuration without saving the changes.



When selecting an encryption, the following screen will show up:

a. Validate host certificate–enable to upload the necessary secure connection certificates;
b. `MQTT_CA.CRT`—press to upload the root *CA certificate* in a PEM format for the MQTT broker;
c. Supply the *client certificate*—enable to upload the device *public certificate* and *private key* for a secure connection to MQTT broker;
d. `MQTT_CLIENT.KEY`—press to upload the Aranet PRO base station *private key* for a secure connection to MQTT broker;
e. `MQTT_CLIENT.CRT`—press to upload the Aranet PRO base station *public key* for a secure connection to MQTT broker.



General MQTT structure

MQTT runs on top of `TCP/IP` using a *PUSH/SUBSCRIBE* topology. In MQTT architecture, there are two types of systems: clients and brokers. A broker is the server that the clients communicate with. The broker receives communications from publishers and sends those communications on to other clients.

Aranet PRO MQTT publisher topic structure

Here are some differences in the topic structures in the MQTT publisher topic structure depending on the data format.

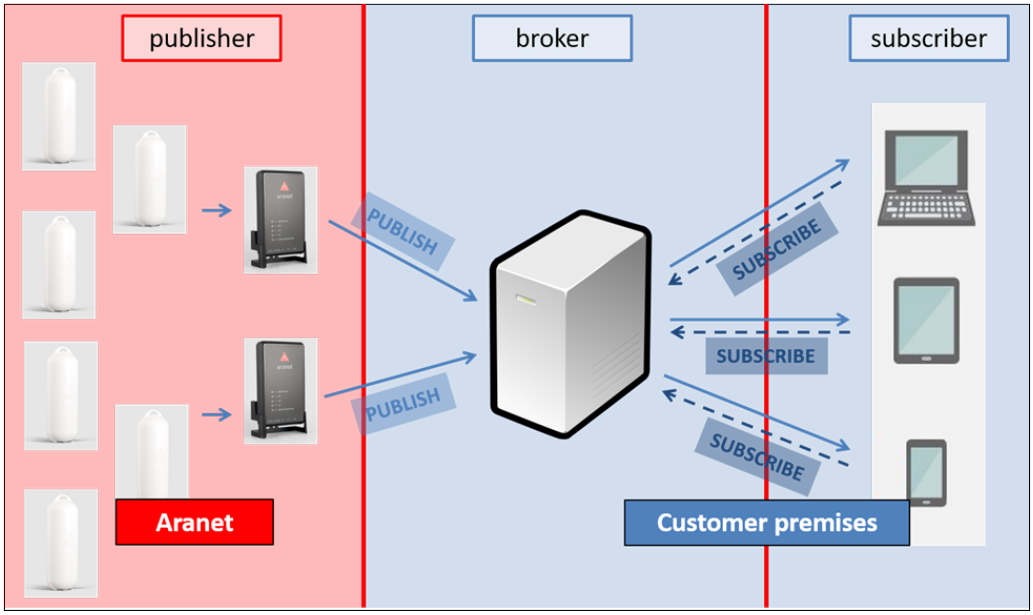


Figure 1: General MQTT network structure

Figure 2: RAW format.

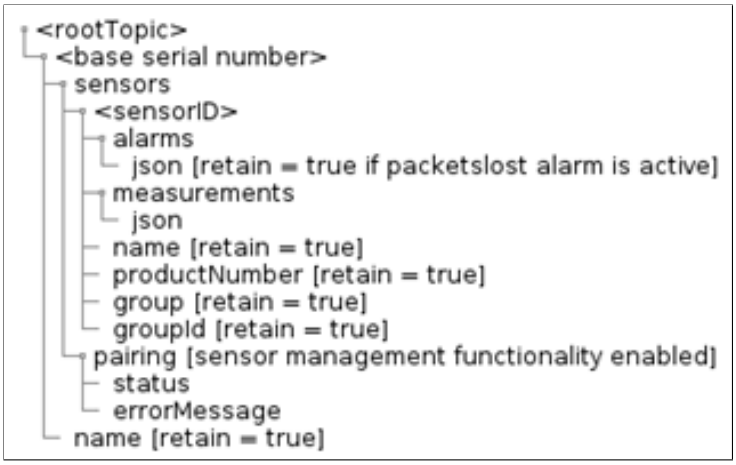


Figure 3: JSON format.

Sensor measurement data messages from the PRO base can be published on the MQTT broker in 3 following formats (hierarchy):

Raw measurement types

In topic structure `<root topic name>/<PRO base serial number>/sensors/<sensorID>/measurements/<measurementtype>` where:

- **`<root topic name>`**—Aranet PRO base station MQTT message identification name which should be configured

on the base MQTT page *Root topic* field. For more details see below Aranet PRO base station configuration interface

- **<PRO base serial number>**—serial number of PRO base station;
- **<sensor ID>**—6 HEX digit sensor ID where the first digit is the sensor segment (for details see Segments for sensors document) and remaining 5 digits are from sensor marking from the physical label on the sensor body which can be seen also in PRO base station graphical user interface;
- **<measurement type>**, can be one of the following:
  - **temperature:** data is given in [C] (degrees Celsius);
  - **humidity:** relative humidity data is given in [%] (percentage);
  - **co2:** carbon dioxide concentration level data given in [ppm] (parts per million);
  - **co2Abc:** shows whether CO2 manual (0.000000) or automatic (1.000000) calibration mode is enabled for the sensor;
  - **atmosphericpressure:** atmospheric pressure data are given in [Pa] (Pascal);
  - **voltage:** data are given in [V] (Volts);
  - **current:** electric current data are given in [A] (Ampere);
  - **weight:** tarred weight in [kg] (kilogram);
  - **weightraw:** untarred weight in [kg] (kilogram);
  - **illuminance:** data from LUX sensor given in [lx] (lux);
  - **distance:** data are given in [m] (meters);
  - **vwc:** volumetric water content data of soil/substrate given as [/] (the fraction of one whole);
  - **bec:** bulk electric conductivity data are given in [S/m] (Siemens per meter);
  - **pec:** pore water electrical conductivity data are given in [S/m] (Siemens per meter);
  - **dp:** dielectric permittivity data of soil or substrate given in absolute numbers;
  - **ppfd:** photosynthetic photon flux density data are given in $[\frac{\mathrm{umol}}{\mathrm{m}^2}\mathrm{s}]$ (micromol per square meters multiplied by seconds);
  - **pulses:** pulses in each sensor measurement interval in absolute numbers [pulses];
  - **derivedp:** pulses measurement in each sensor measurement interval once the sensor conversion rule is applied in user-defined units;
  - **pulsescumulative:** cumulative pulses in absolute numbers [pulses];
  - **derivedpc:** cumulative pulses measurement once the sensor conversion rule is applied in user-defined units;
  - **co:** carbon monoxide concentration level data are given in [ppm] (parts per million);
  - **differentialpressure:** data are given in [Pa] (Pascal);
  - **motorseconds:** operational/switched-on (AC connection or contact closed) time of the connected device to the sensor in each sensor measurement interval in [s] (seconds);

- **motorsecondscumulative:** cumulative or total operational/switched-on (AC connection or contact closed) time of the connected device to the Aranet hour meter sensors in [s] (seconds);
- **derived:** derived measurements in user-defined units;
- **rssi:** received signal strength data given in [dBm];
- **battery:** battery charge level which is given as [/] (the fraction of one whole);
- **time:** measurement time in Unix epoch format

Additionally measurement units for the sensor data according to measurement type is published in topics: `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/measurements/<measurement type>/units`

Examples

JSON measurments

Only measurement values are sent, but no sensor measurement units and alarm messages in the topic structure: `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/json/measurements`

```
{
        "humidity": "44.0",
        "temperature": "34.200",
        "rssi": "-30",
        "time": 1626434635,
        "battery": "0.17"
}
```

JSON alarms

Sensor alarm messages from PRO base is published on the MQTT broker in following hierarchy (format): `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/alarms/+`

```
{
        "temperature": {
                "value": "34.20",
                "diff": "66.80",
                "activeSince": "1626426827"
        },
        "humidity": {
                "value": "44.00",
                "diff": "38.00",
                "activeSince": "1626426827"
        }
}
```

Azure format for sensor data publishing to Azure IoT Hub platform

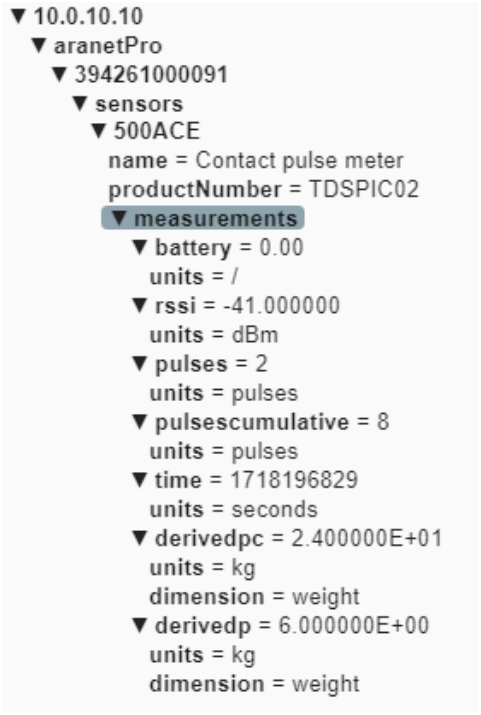
```
{
        "sensors": [
        {
                "measurments": [
                {
                        "measurment": "humidity",
                        "units": "%",
                        "value": "38.0"
                },
                {
                        "measurment": "temperature",
                        "units": "C",
                        "value": "21.850"
                },
                {
                        "measurments": "rssi",
                        "units": "dBm",
                        "value": "-47"
                },
                {
                        "measurment": "time",
                        "value": 1618691111
                },
                {
                        "measurment": "battery",
                        "units": "/",
                        "value": "0.90"
                }
                ],
                "uid": "10136"
        }
        ]
}
```


Differences between the sensor measurement formats (‘raw’ and ‘JSON’)

Sensor measurements MQTT message topic structure

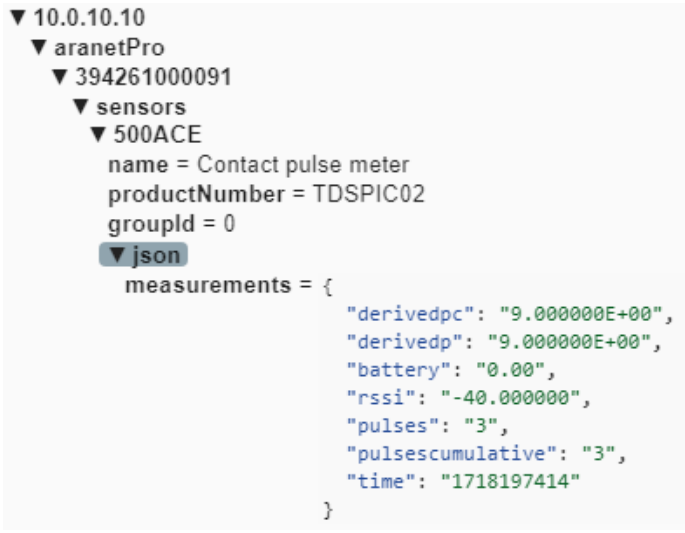
RAW

<RootTopic>/<SerialNumber>/sensors/
<SensorID>/measurements/<measurement>



JSON

<RootTopic>/<SerialNumber>/sensors/
<SensorID>/json/measurements



Examples of MQTT subscription topics for measurement data reception

- **Example 1 - Receive sensor measurements from all the bases if `<RootTopic>` consists of a single level topic**
  - **RAW:** `+/+/sensors/+/measurements/+`
  - **JSON:** `+/+/sensors/+/json/measurements`
- **Example 2 - Receive all the sensor measurements from the base with S/N 394261000688 if `<RootTopic>` is set to 'Aranet'**
  - **RAW:** `Aranet/394261000688/sensors/+/measurements/+`
  - **JSON:** `Aranet/394261000688/sensors/+/json/measurements`
- **Example 3 - Receive measurements from the sensor with ID 100118 paired to the base with S/N 394261000688**

if <RootTopic> is set to two-level topic 'Riga/Warehouse'

- **RAW:** `Riga/Warehouse/394261000688/sensors/100118/measurements`
- **JSON:** `Riga/Warehouse/394261000688/sensors/100118/json/measurements`

Additional notes

MQTT Last Will and Testament message support

Additional topic "online" (for raw and JSON measurement formats) and message type "baseOnline" (for Azure IoT Hub messaging format) has been added on which broker also sends Aranet PRO "Last Will and Testament" message in case if Aranet PRO MQTT publisher disconnects from the broker. On successfull connection Aranet PRO MQTT publisher notifies about the online state, but on gracefull or ungracefull disconnect MQTT broker notifies subscribers about the Aranet PRO MQTT publisher's offline state.

MQTT Last Will and Testament message examples in different messaging formats.

Online notification on successful connect sent by Aranet PRO:

Raw/JSON

- *Topic:* `<rootTopic>/<base serial number>/online`
- *Payload example:* "true"
- *Notes:* Retained message. String value.

Azure IoT Hub

- *Topic:* `devices/<base serial number>/messages/devicebound/#`
- *Payload example:*

```
{
        "body": {
                "online": "true"
        },
        "enqueuedTime": "2024-11-4T14:46:30.064Z",
        "properties": {
                "msgType": "baseOnline"
        }
}
```

- *Notes:* Retained message. Message can be identified by the `"msgType"` value `"baseOnline"`."

LWT notification on Aranet PRO disconnection sent by MQTT broker:

Raw/JSON

- *Topic:* `<rootTopic>/<base serial number>/online`

- *Payload example:* ”false”
- *Notes:* Retained message. String value.

Azure IoT Hub

- *Topic:* `devices/<base serial number>/messages/devicebound/#`
- *Payload example:* N/A
- *Notes:* Only Azure Event Grid broker supports LWT message sending functionality on Aranet PRO MQTT publisher disconnect event (which requires usage of raw or JSON messaging format instead of Azure IoT Hub message format).

Sensors and P/N grouped by measurments

Measurement (Unit)	Sensor Name (P/N)
Voltage (V)	0-10 V transmitter (TDSPVM01.010, TDSPVM02) 0-10 V transmitter with 12 VDC power supply (TDSVT102) 0-10 V transmitter with 24 VDC power supply (TDSVT202) Stem diameter sensor (TDSPSD02)
Humidity (%)	T/RH Sensor with radiation shield (TDSPT009, TDSPT509) T/RH Sensor IP68 (TDSPT801) RSSI sensor (TDSPTT01, TDSPTT02) T/RH Probe (TDSPT309) T/RH Probe (ammonia resistant) (TDSPT409) T/RH IP67 sensor (TDSPT802) T/RH IP42 sensor (TDSPT001) Aranet4 PRO (TDSPC003) Aranet w/o display (TDSPC205) Aranet2 PRO (TDSPRH02)

Measurement (Unit)	Sensor Name (P/N)
Temperature (C)	RSSI sensor (TDSPTT01, TDSPTT02) T/RH Sensor IP68 (TDSPT801) T/RH Sensor with radiation shield (TDSPT009, TDSPT509) T/RH Probe (TDSPT309) T/RH Probe ammonia resistant (TDSPT409) T/RH IP67 Sensor (TDSPT802) PT100 EXT sensor (TDSPTE06) PT100 sensor (TDSPT006) PT100 transmitter (TDSPT106, TDSPTA06) PT1000 transmitter (TDSPT206, TDSPT506) PT100 thermometer (TDSPT306) T probe (TDSPT002) T compact (TDSPT204) Medical T (TDSPTK01) Soil moisture, EC and T sensor (TDSPHE01) Soil sensor Teros-12 (TDSPHE02) T/RH IP 42 sensor (TDSPT001) Aranet4 PRO (TDSPC003) Aranet w/o display (TDSPC205) Aranet2 PRO (TDSPRH02) Soil sensor WET150 (TDSPHW02) CO2 and T sensor (TDSPC005) IR Plant temperature sensor (TDSIR001)
Derived (<user-defined>)	Stem diameter sensor (TDSPSD02) 0-10 V transmitter (TDSPVM01.010, TDSPVM02) 0-10 V transmitter with 12 VDC power supply (TDSVT102) 0-10 V transmitter with 24 VDC power supply (TDSVT202) Ultrasonic distance sensor (TDSPDM01, TDSPDM02) 4-20 mA transmitter (TDSPCL01.010, TDSPCL02) 4-20 mA transmitter with 12 VDC power supply (TDSCT102) 4-20 mA transmitter with 24 VDC power supply (TDSCT202)
Weight (kg)	Weight Sensor 50 kg (TDSPSV01.050, TDSPSV02.050) Weight sensor 100 kg (TDSPSV01.100, TDSPSV02.100)
Weight_raw (kg)	Weight Sensor 50 kg (TDSPSV01.050, TDSPSV02.050) Weight sensor 100 kg (TDSPSV01.100, TDSPSV02.100)
Current (A)	4-20 mA transmitter (TDSPCL01.010, TDSPCL02) 4-20 mA transmitter with 12 VDC power supply (TDSCT102) 4-20 mA transmitter with 24 VDC power supply (TDSCT202)
pulses (count)	Dry Contact pulse counter (TDSPIC01.010, TDSPIC02)
pulsescumulative (count)	Dry Contact pulse counter (TDSPIC01.010, TDSPIC02)

Measurement (Unit)	Sensor Name (P/N)
derived_CPP (<user-defined>)	Dry Contact pulse counter (TDSPIC01.010, TDSPIC02)
derived_CPC (<user-defined>)	Dry Contact pulse counter (TDSPIC01.010, TDSPIC02)
Distance (m)	Ultrasonic distance sensor (TDSPDM01, TDSPDM02)
PPFD ($\mu mol/m^2s$)	PAR Sensor (TDSKAR01, TDSKAR02, TDSPPA02)
CO2 (ppm)	CO2 Sensor (TDSPC001, TDSPC004) Aranet4 PRO (TDSPC003) Aranet w/o display (TDSPC205) Aranet2 PRO (TDSPRH02) CO2 and T sensor (TDSPC005)
VWC (fraction 0.0 - 1.0)	Soil VWC sensor (TDSPSM02) Soil VWC, EC and T sensor (TDSPHE01) Soil sensor Teros-12 (TDSPHE02) Soil sensor WET150 (TDSPHW02)
bec (S/m)	Soil VWC, EC and T sensor (TDSPHE01) Soil sensor Teros-12 (TDSPHE02) Soil sensor WET150 (TDSPHW02)
pec (S/m)	Soil VWC, EC and T sensor (TDSPHE01) Soil sensor Teros-12 (TDSPHE02) Soil sensor WET150 (TDSPHW02)
dp (N/A)	Soil VWC, EC and T sensor (TDSPHE01) Soil sensor Teros-12 (TDSPHE02) Soil sensor WET150 (TDSPHW02)
Motor Seconds (s)	AC Hour sensor (TDSPAC01) Dry contact Hour meter (TDSPHM01)
Motor Seconds Cumulative (s)	AC hour sensor (TDSPAC01) Dry contact Hour meter (TDSPHM01)
Differential Pressure (pascal)	Differential Pressure Sensor (TDSPDP01, TDSPDP02)
Illuminance (lux)	LUX Sensor (TDSKLM01)

NOTE: Only EU P/N shown

Alarms

Name	Description	Repetitive	Retain
`Battery`	Sensor's battery charge level is low	Yes	No
`Channel`	Sensor is using a different radio channel than the base station	Yes	No
`Errorflags`	Sensor malfunction detected	Yes	No
`Packetslost`	Measurement from the sensor was not received in the estimated time	No	Yes
`<measurment>`	Measurement value has reached user defined threshold	Yes	No

Public Aranet PRO MQTT publisher messages for demo purposes

We recommend using MQTT Explorer to view the MQTT structures for yourself using the public broker service https://www.mqtt-dashboard.com/*HiveMQ Public Broker* and subscribe to messages published by Aranet PRO MQTT demo publisher.

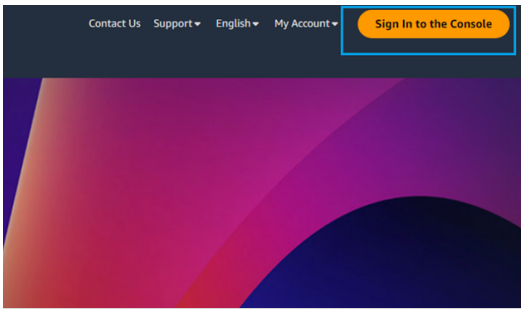
Host address	https://www.mqtt-dashboard.com/
Port	1883
Protocol version	MQTT v3.1.1
Authentication	Disabled
Encryption	None
Root topic	`Aranetest`
Subscription topic	`Aranetest/394260700033/#`

MQTT connection configuration with Amazon AWS platform

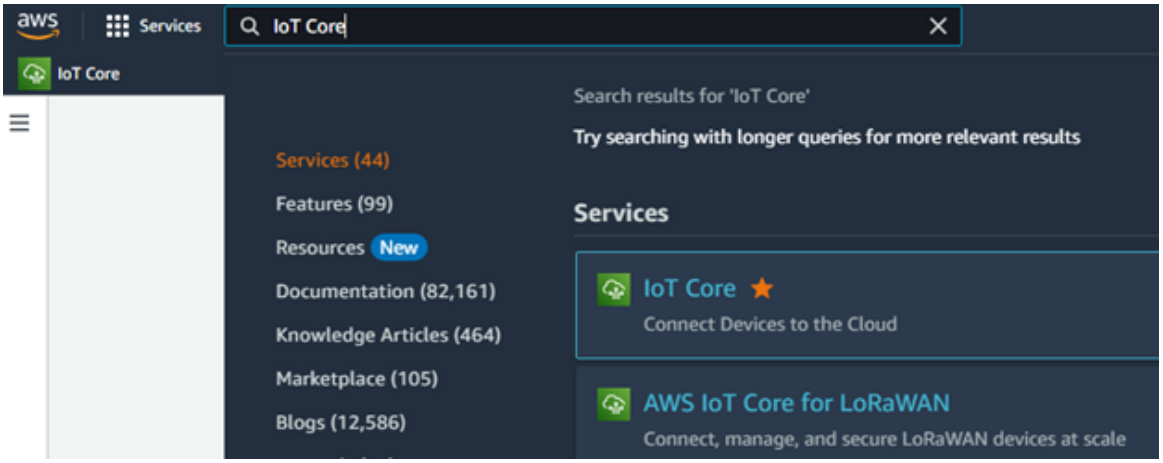
Access AWS IoT Core console

Aranet PRO base station allows all sensor data publishing directly to *AWS IoT Core*, but here base only should have a firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of the Aranet PRO base station in the graphical user interface section *System* → *FIRMWARE* and if it is older than 2.5.17, then update to the latest version available from https://aranet.com/downloads/ section of our webpage:

1. Open the AWS page https://aws.amazon.com/ in a web browser, sign in to the Console:



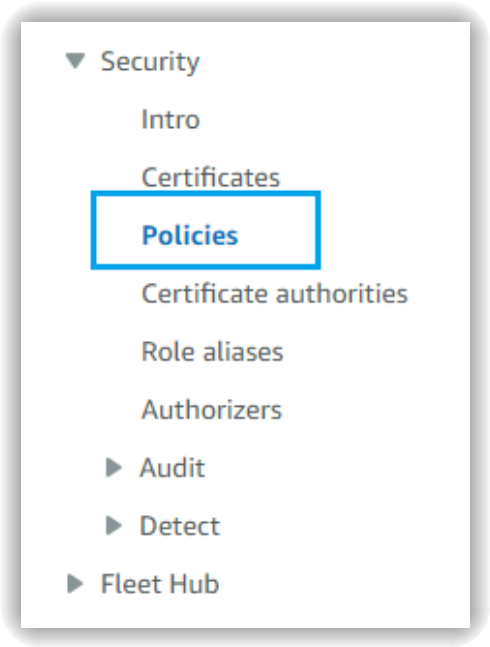
2. In console search for *IoT Core* service:



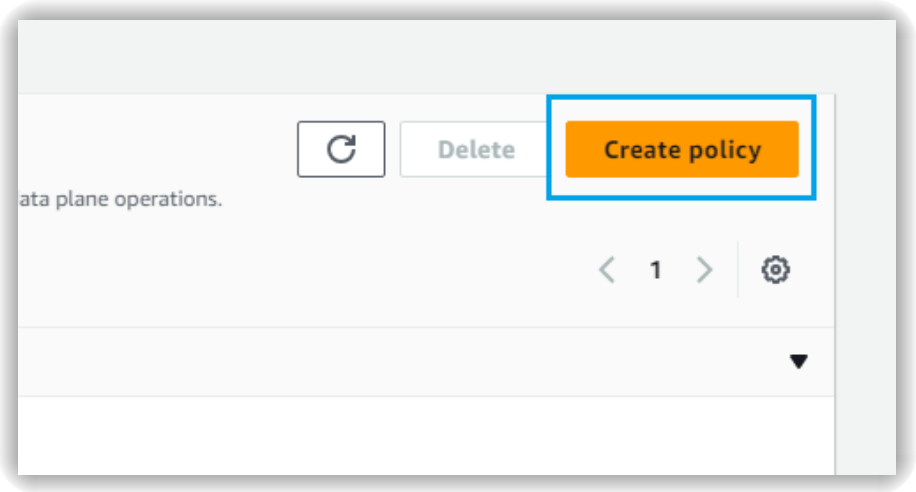
Create a policy for MQTT connect/publish/subscribe actions.

1. In the main menu (left side in the IoT Core console) open Security → Policies:

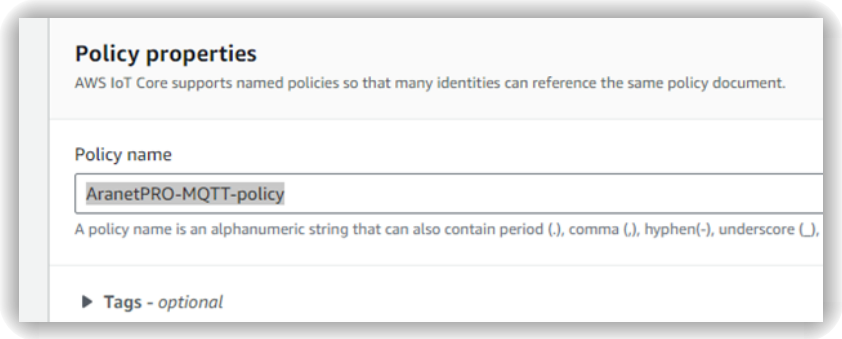
NOTE: A policy will be required later when a new *thing* will be created. This procedure must be performed once (to create a policy) unless there is a reason to have multiple policies.



2. Create a new policy (press on *Create policy*):

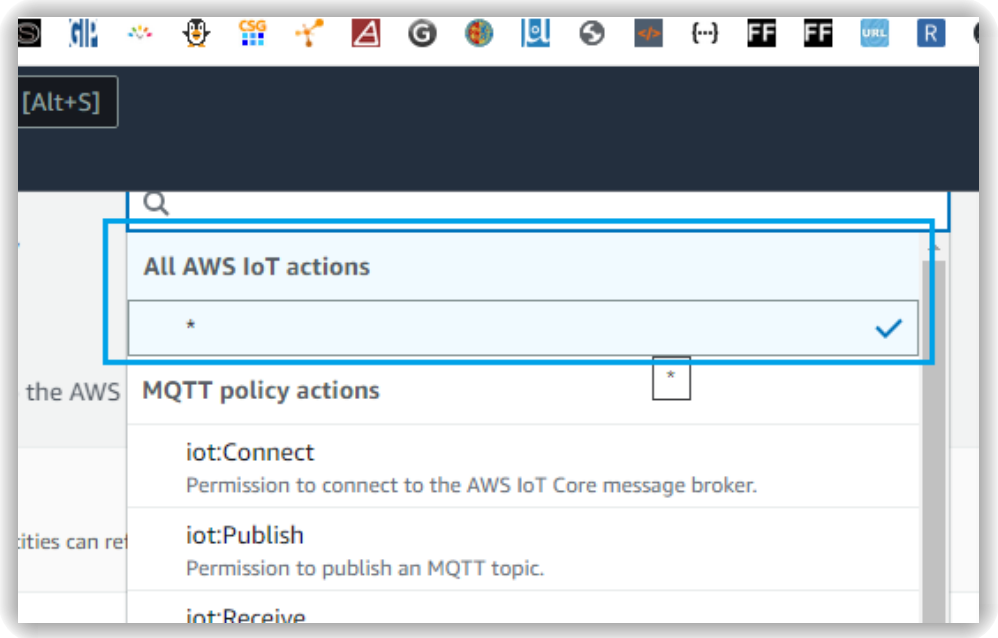


3. Enter the name for the policy:



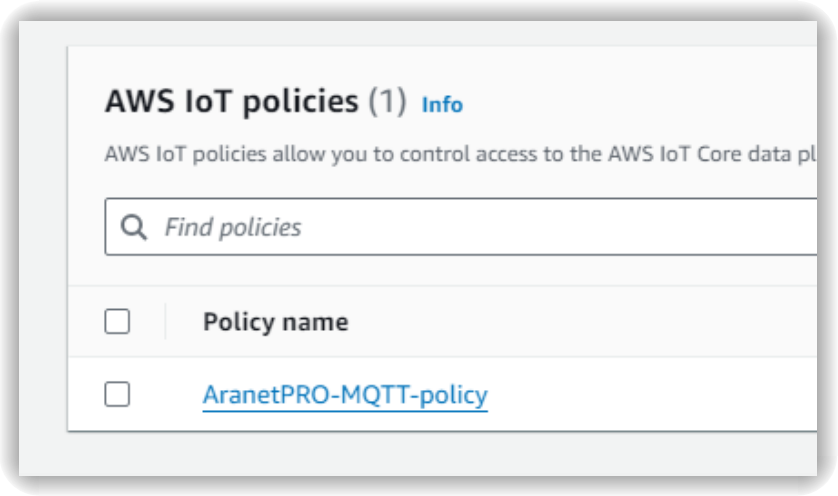
4. In the *Policy document* enter following properties:

   - *Policy effect*: Allow
   - *Policy action*: select *All IoT actions*



   - *Policy resource*: enter '*'

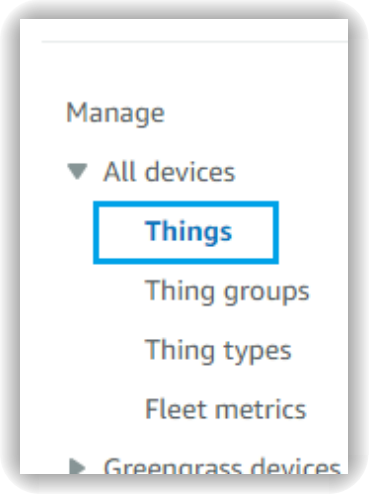
5. Press *Create*. A new policy which allows to perform all MQTT protocol actions on all MQTT topics has been created.



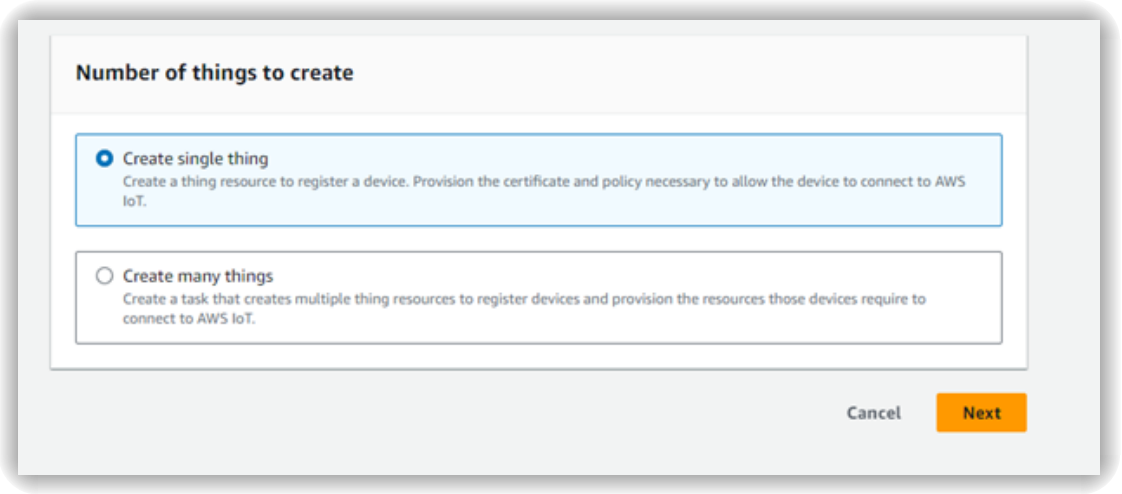
Create a *thing* in the IoT Core.

1. In the main menu (left side in the IoT Core console) open *Manage* → *Things*:

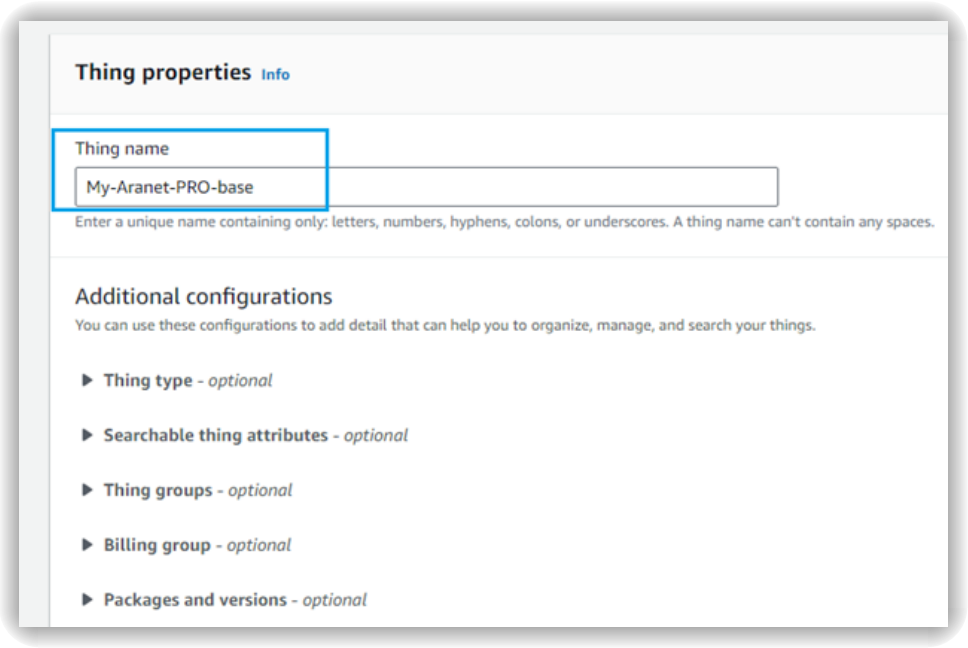
NOTE: This procedure can be performed as many times as required (once per each AranetPRO base station) as it guides through the steps of how to create a new*thing* in IoT Core service. It will require a policy which was created in the section “Create a policy for MQTT connect/publish/subscribe actions”



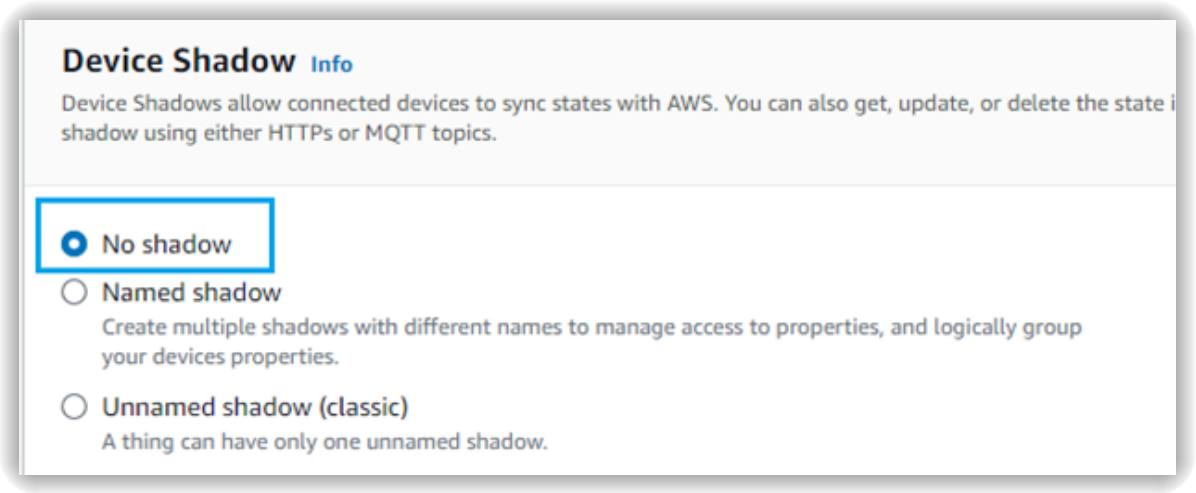
2. Press on *Create things*, then select *Create single thing* and press *Next”*



3. Enter the name for a *thing*:

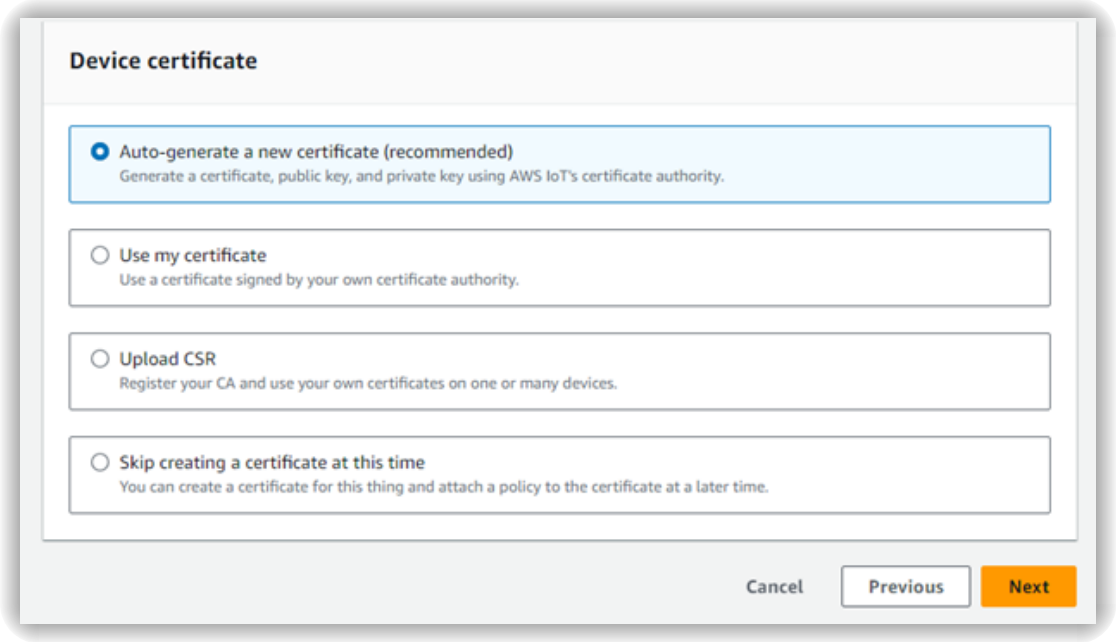


4. For *Device Shadow* select *No shadow*:

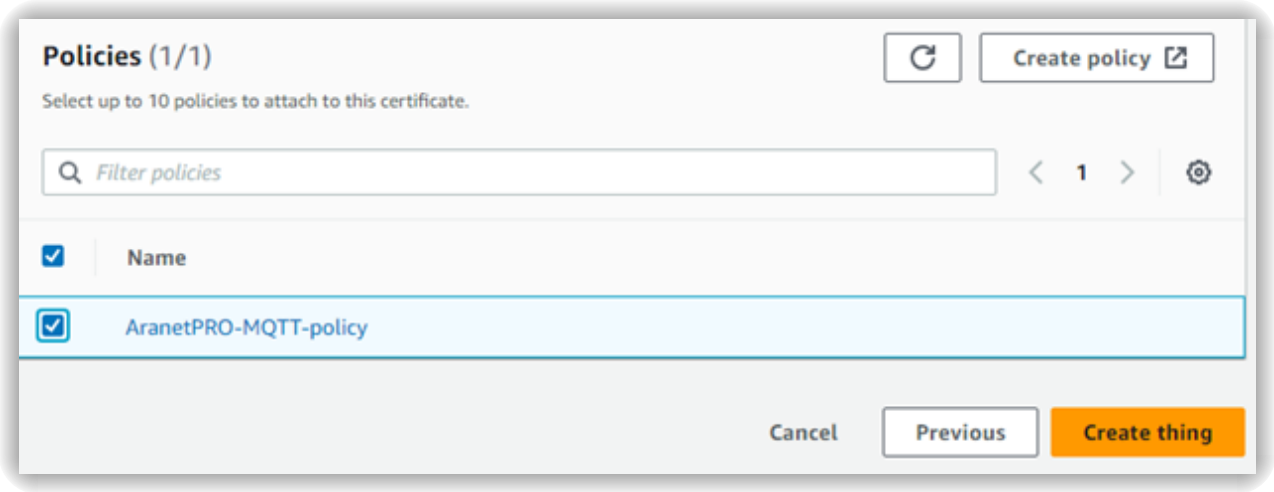


5. *Device certificate* – there are multiple options. In this example first option: *Auto-generate a new certificate* will be

used. Press *Next*.



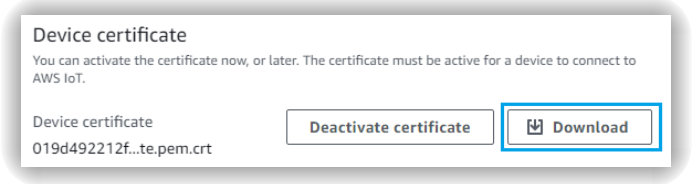
6. Select the policy which was created previously, and press *Create thing*:



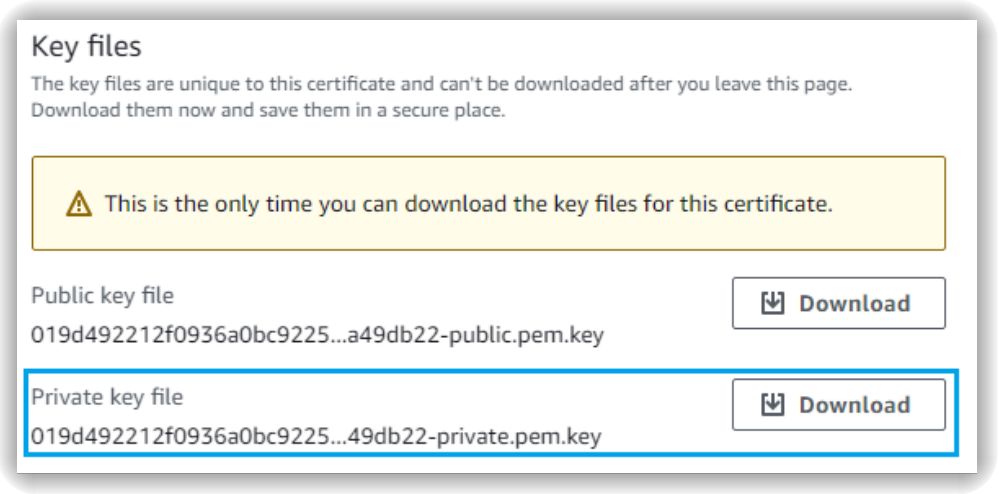
7. Download the certificates and keys.

NOTE: Theses files will be required later when the configuration on base station for MQTT will be performed. Downloaded certificate files must be stored in a secure place. Download and rename the files accordingly:

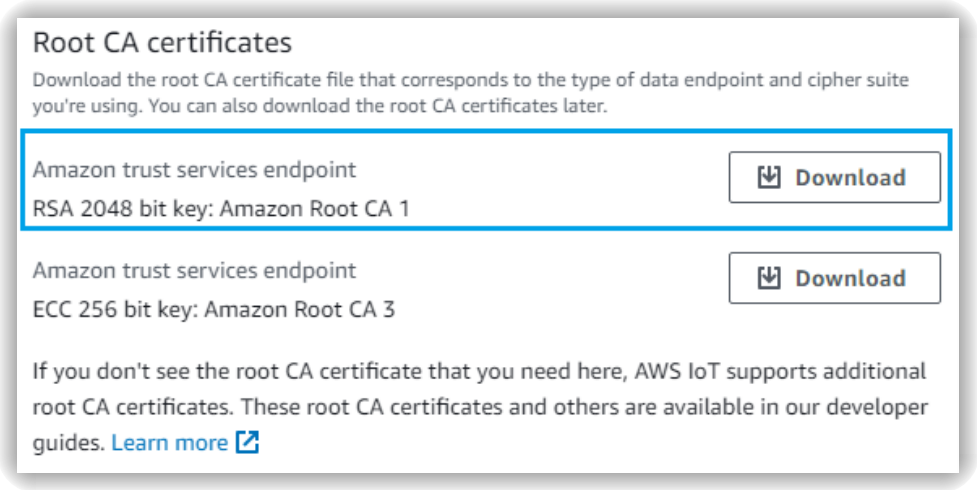
a. Download the *Device certificate* file and rename it as `aranet-pro-base.crt`



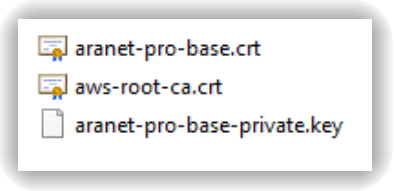
b. Download the *Private key file* and rename it as `aranet-pro-base-private.key`



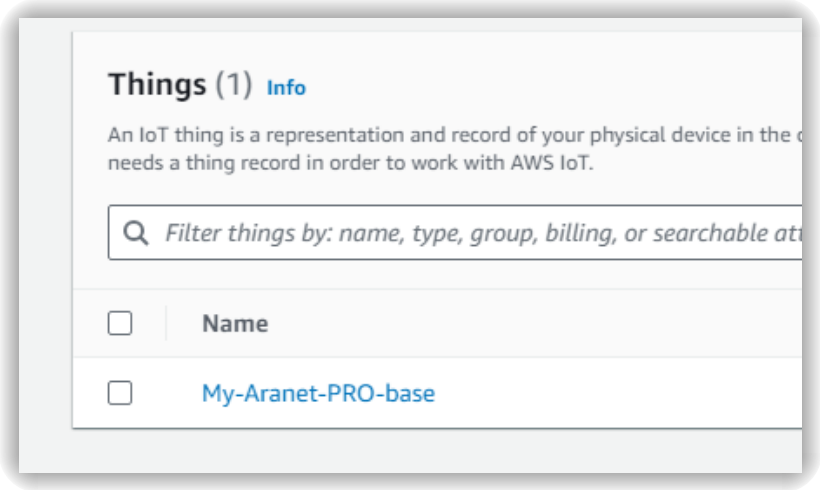
c. Download the *CA Root certificate* file and rename it as `aws-root-ca.crt`



d. As a result there must be three downloaded files:



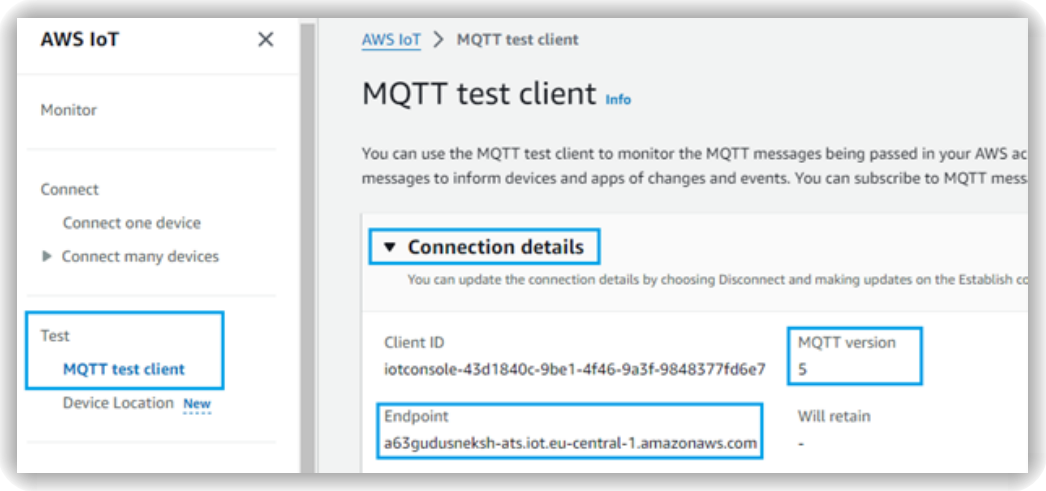
e. Also download the *public key* (it will not be required) and press *Done*. A new *thing* has been created.



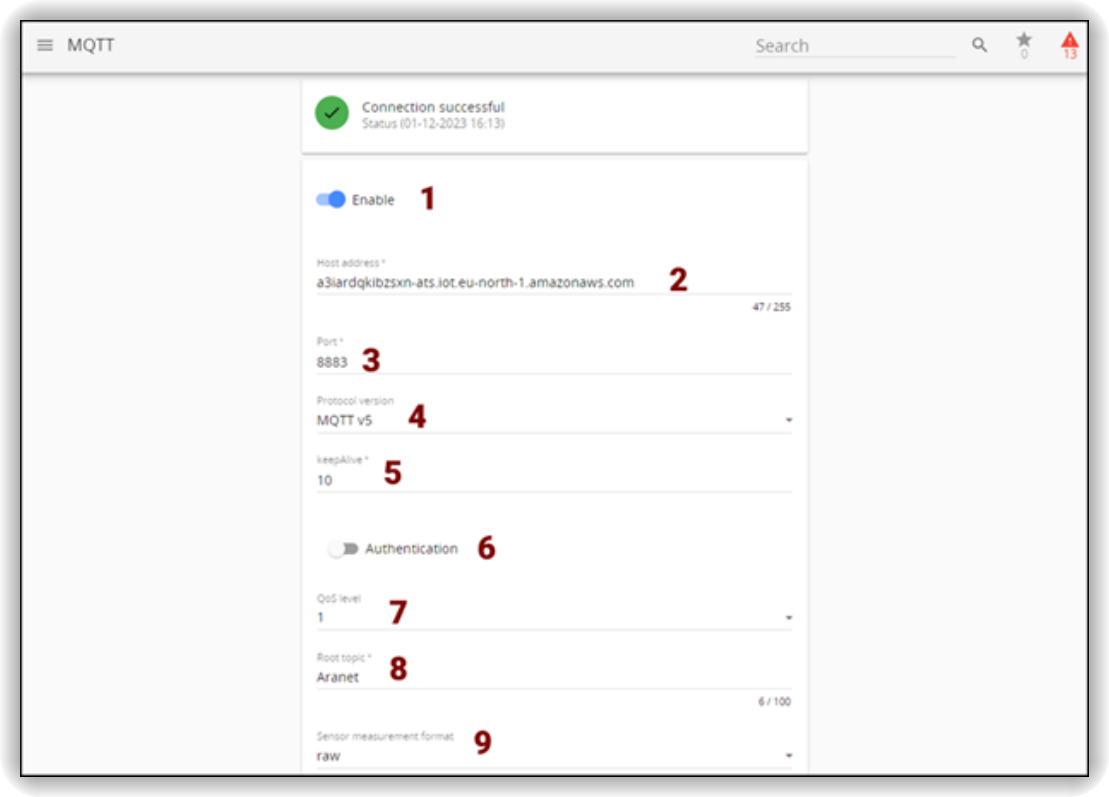
Configure Aranet PRO base station's MQTT to connect to AWS IoT Core.

NOTE: The following step will be used to determine what is the *Host address* to which the Aranet PRO bases station will connect to and the MQTT protocol version. In order to connect the Aranet Pro base station to AWS IoT Core using MQTT, a *thing* must be created in AWS IoT Core service. Follow the steps described in create a thing in the IoT Core

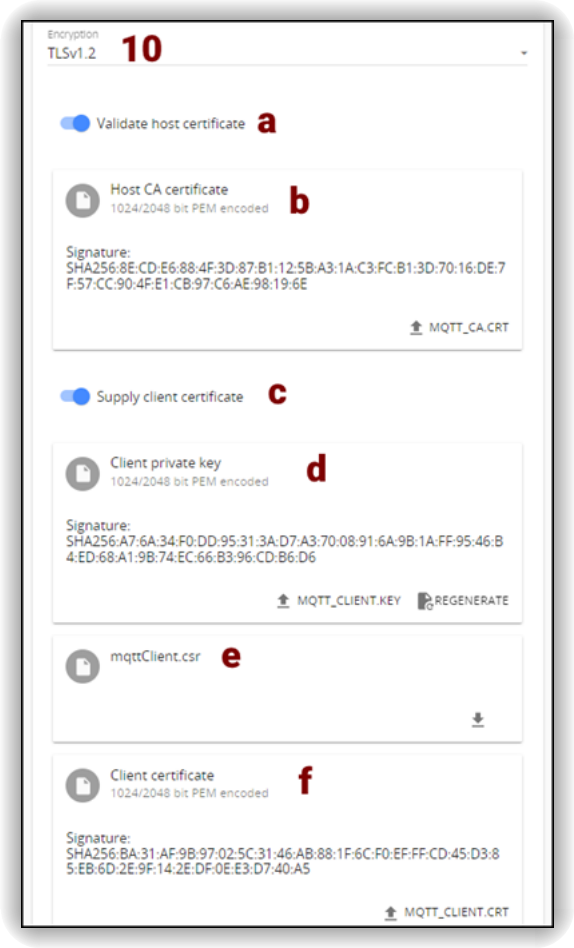
- In AWS IoT Core main menu open *Test* → *MQTT test client* and press on *Connection details*:



- An *Endpoint* will be used for *Host address* (in the Aranet PRO base) and *MQTT version* for *Protocol version* accordingly.
- Go to the Aranet PRO base station MQTT configuration interface and set it up as so:



1. *Enable* — enable the MQTT;
2. *Host address* — use *Endpoint* address from the previous Chapter;
3. *Port* — enter `8883`;
4. *Protocol version* — set MQTT version from the previous Chapter – in our example *MQTT v5*;
5. *Keepalive* — use `10`;
6. *Authentication* — disabled;
7. *QoS level* — use `1`;
8. *Root topic* — use “Aranet”
9. *Sensor measurement format* — use `raw`;
10. *Encryption* — Amazon AWS requires sertificate validation. Use *TLSv1.2*.



a. *Validate host certificate* — enable to upload necessary secure connection certificates;

b. — use `aws-root-ca.crt` file saved from Amazon AWS (see previous Chapter for details);

c. *Supply client certificate* — enable to upload the devices *public certificate* and *private key* for a secure connection to the MQTT broker

d. — use `aranet-pro-base-private.key` file saved from Amazon AWS (see previous Chapter for details);

e. — use `aranet-pro-base.crt` file saved from Amazon AWS (see previous Chapter for details);

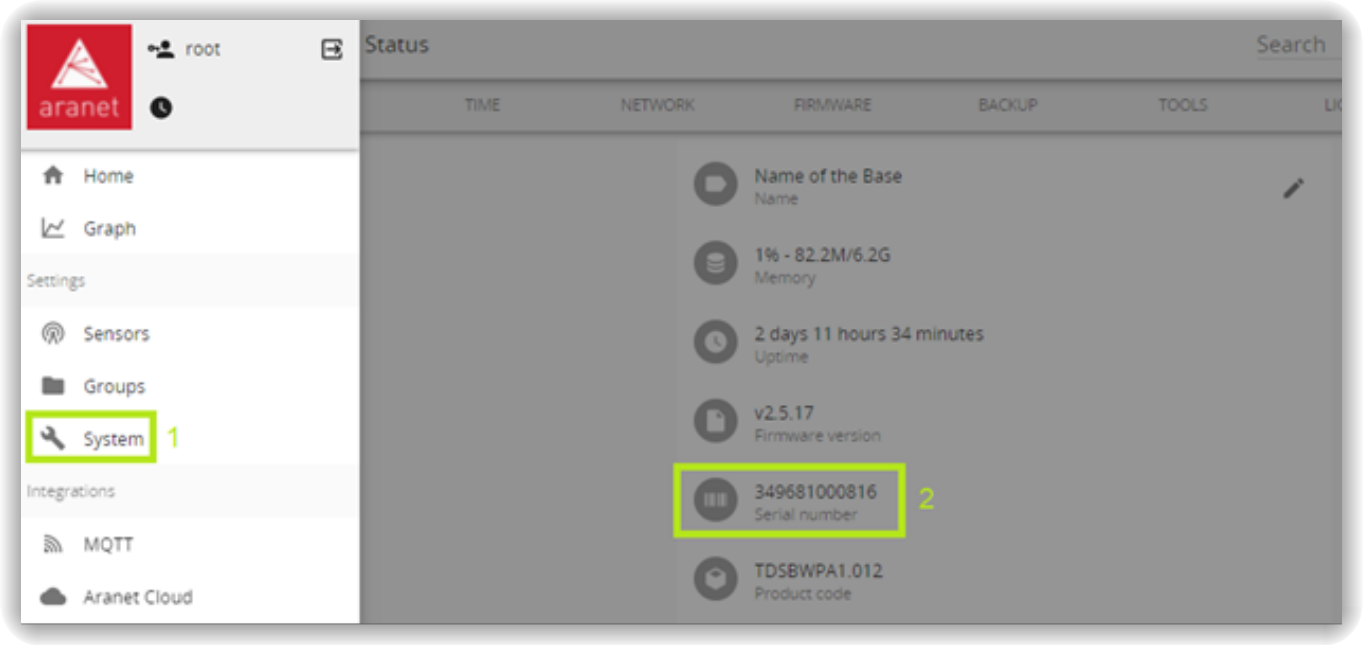
* When all the necessary configuration parameters are entered, they should be saved by pressing the blue Save icon . If the configured MQTT connection is successful, then a Connection successful message will be shown on the top of the page showing also the precise time when the connection was established.

MQTT connection configuration with Azure IoT Hub

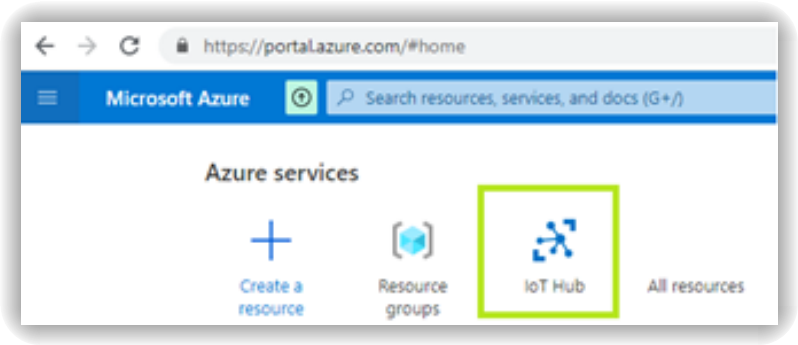
Aranet PRO base station also allows all sensor data publishing directly to Azure IoT Hub, but here the base should also have the firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of the Aranet PRO base station in the graphical user interface section *System* → *FIRMWARE* and if it is older than 2.5.17, then update to the latest version available from https://aranet.com/downloads/ section of our webpage:

1. Check and save for further use the Serial number of the Aranet PRO base station you have. It can be seen on the

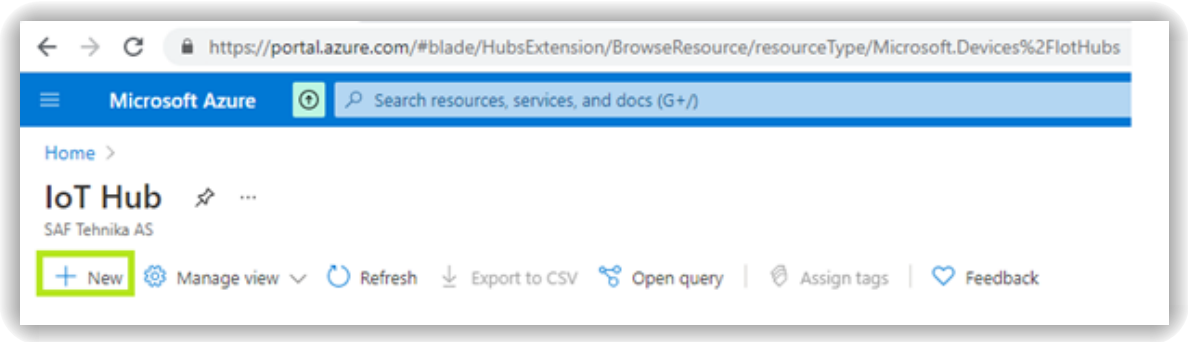
label on the back of the physical device or from the graphical user interface section *System* → *STATUS*:



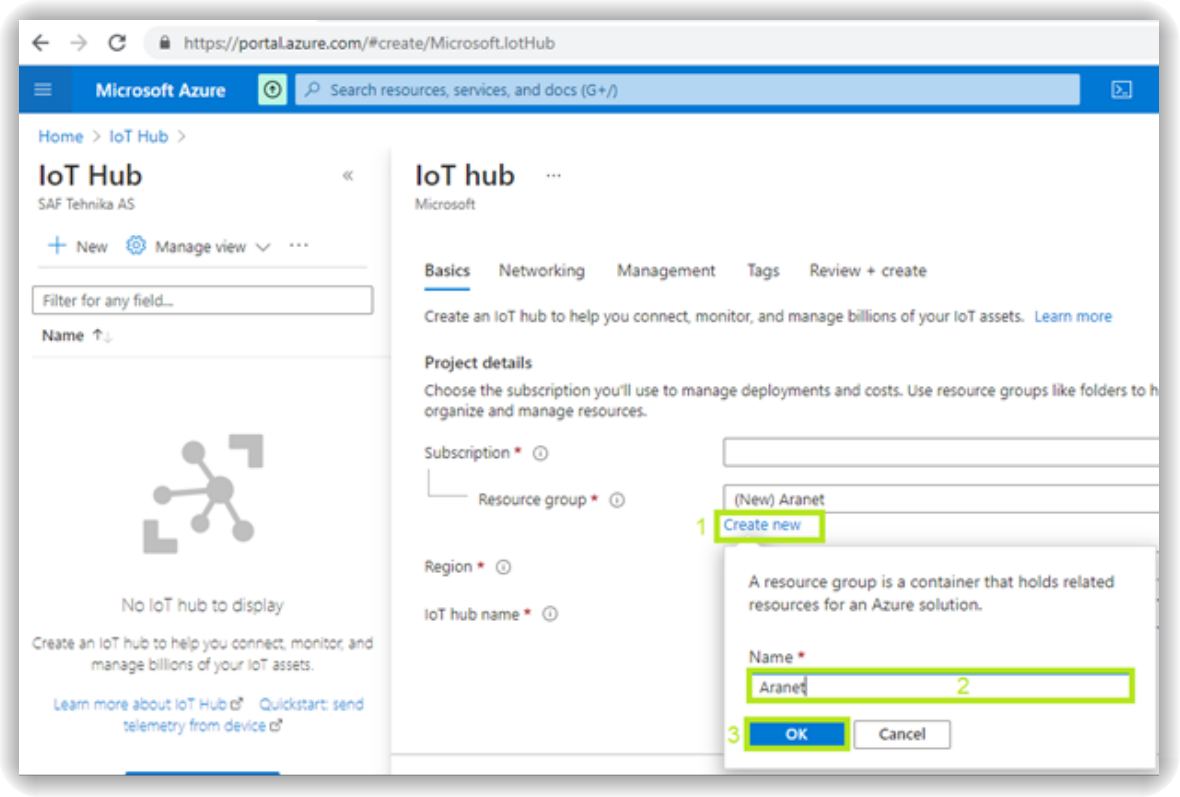
2. Log in to Your Azure account and create a new *IoT Hub resource*:



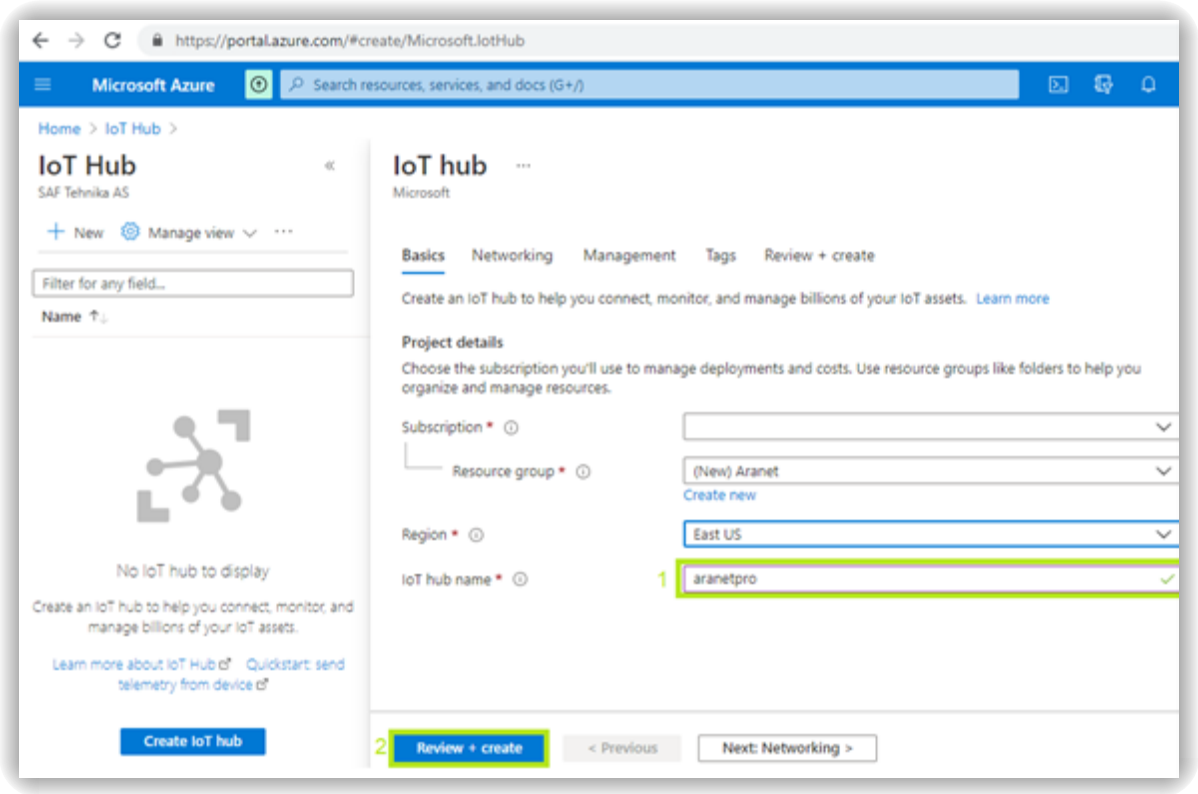
3. Click on *New*:



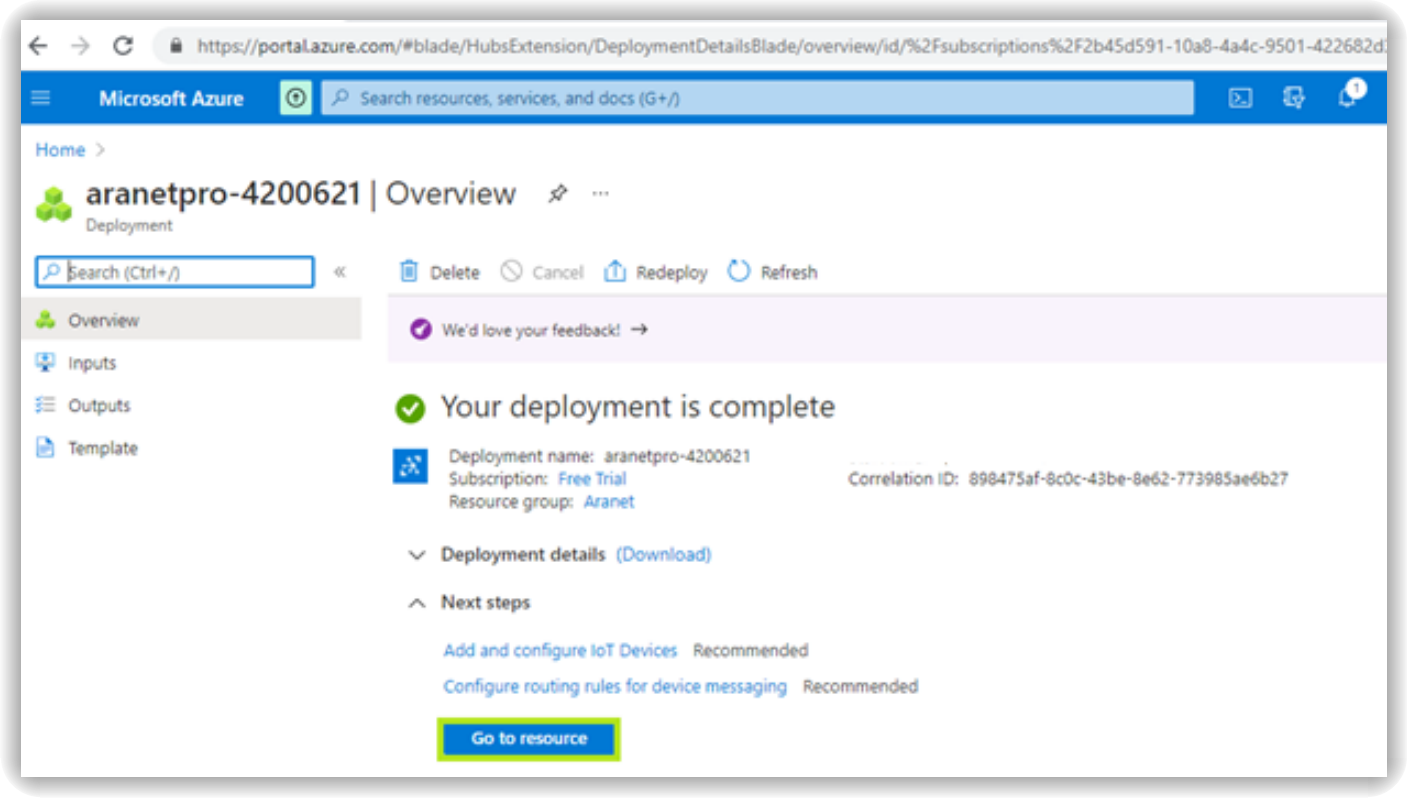
4. In the next window click on *Create new* for a *Resource group*, enter the necessary *Name* and press the *OK* button:



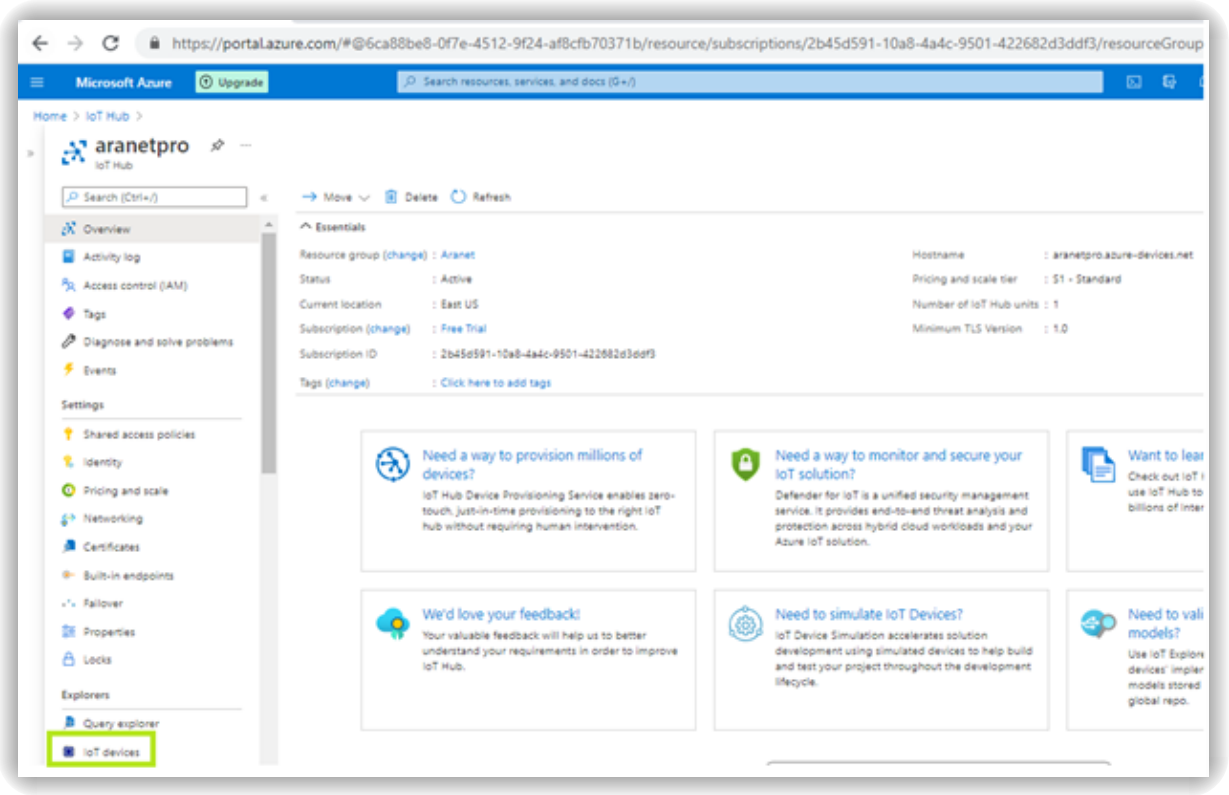
5. In the same window enter the necessary *IoT hub name* and press the *Review + create* button. Then press the *Create* button again::



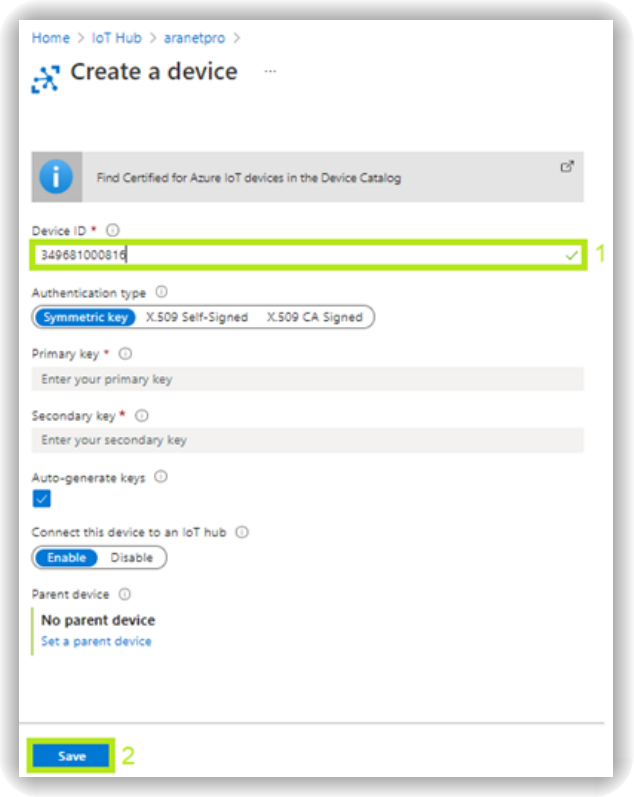
6. Wait for the Azure system to deploy the new *resource* and when it is done press on the *Go to resource* button:



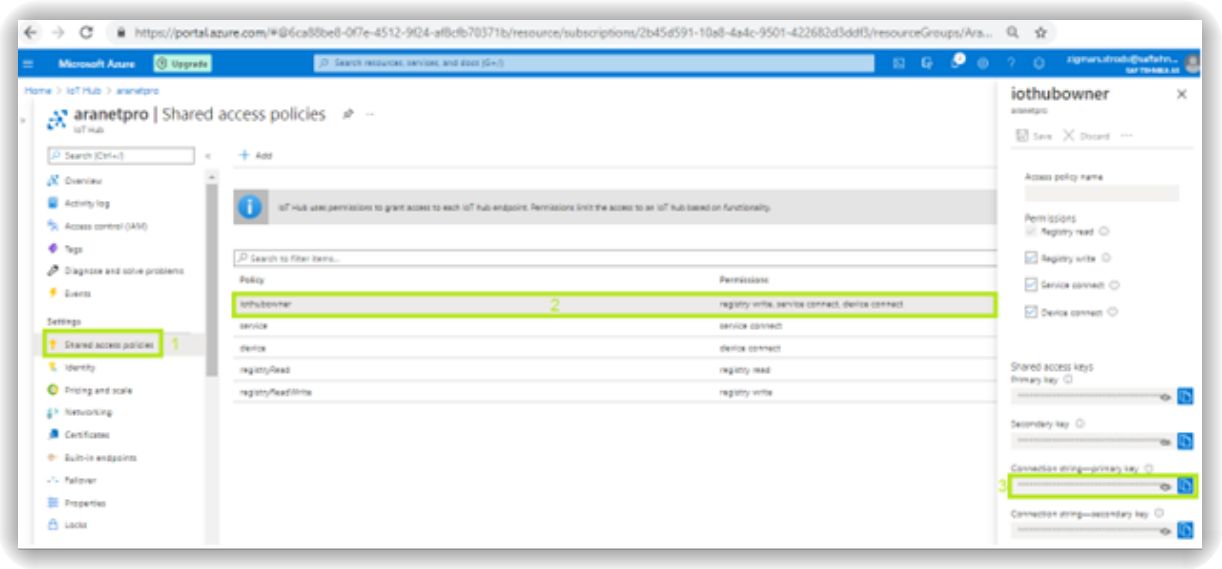
7. Then select the *IoT devices* section from the left side menu and press on *New*::



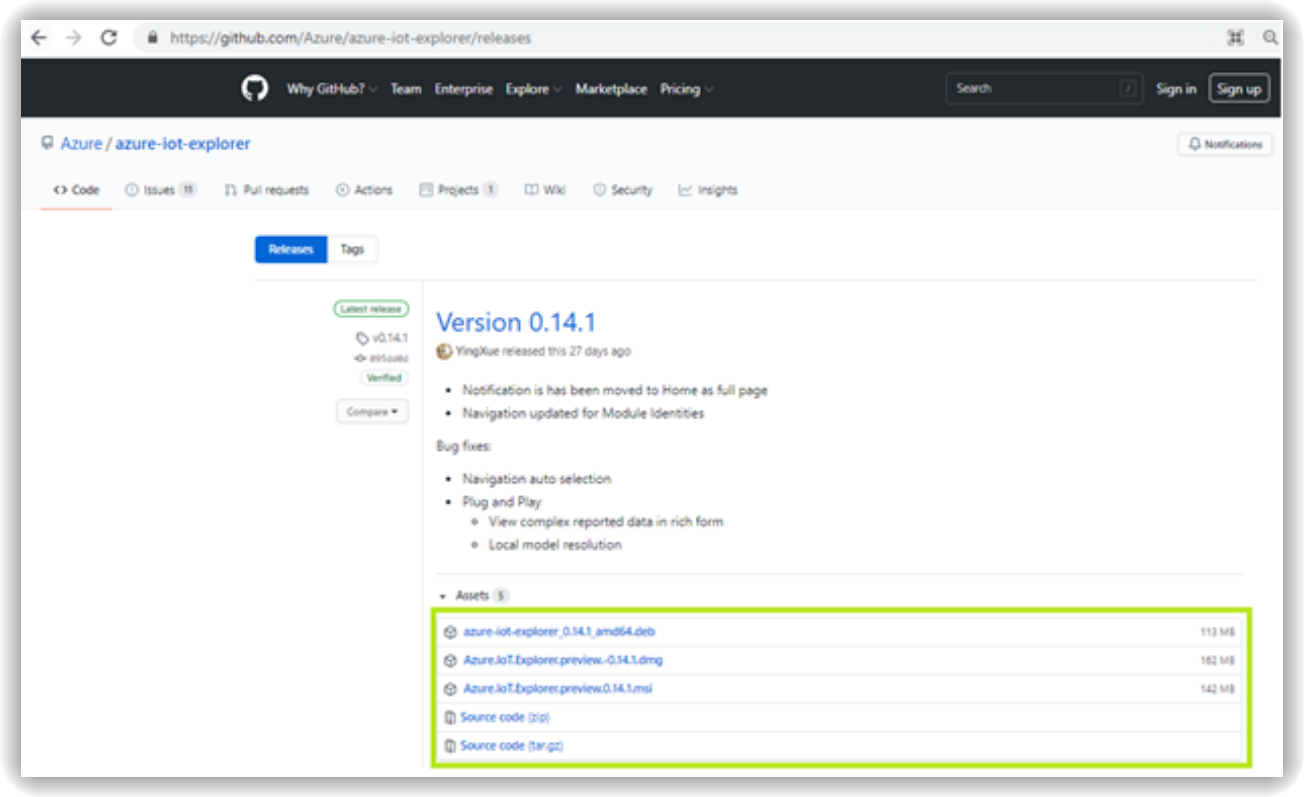
8. Just enter or copy the serial number of the Aranet PRO base station from step 1 in the *Device ID* field without any additional symbols and characters and press the *Save* button:



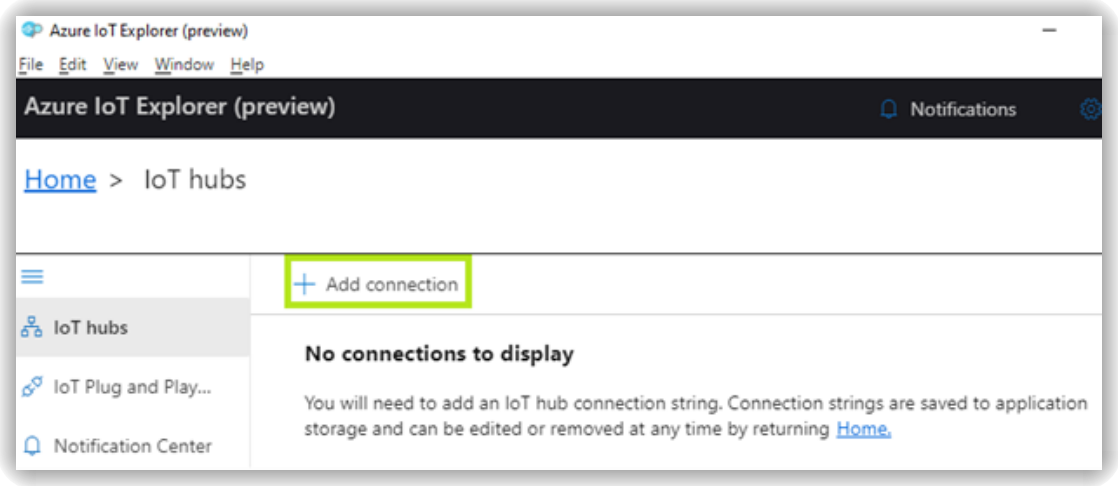
9. Next go to section *Shared access policies*, click on *iothubowner* record and copy the information from the *Connection string—primary key* field:



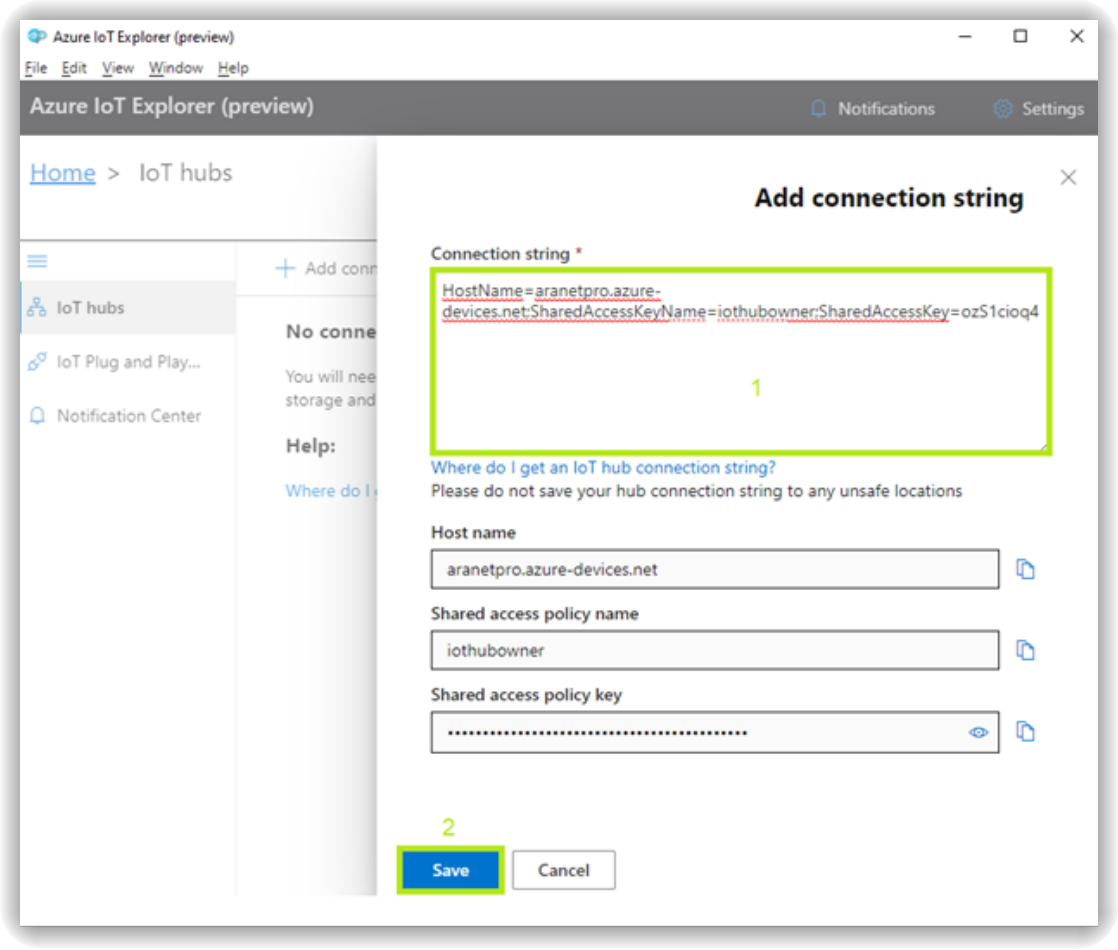
10. Now go to https://github.com/Azure/azure-iot-explorer/releases and install the *Azure IoT explorer* on Your operating system:



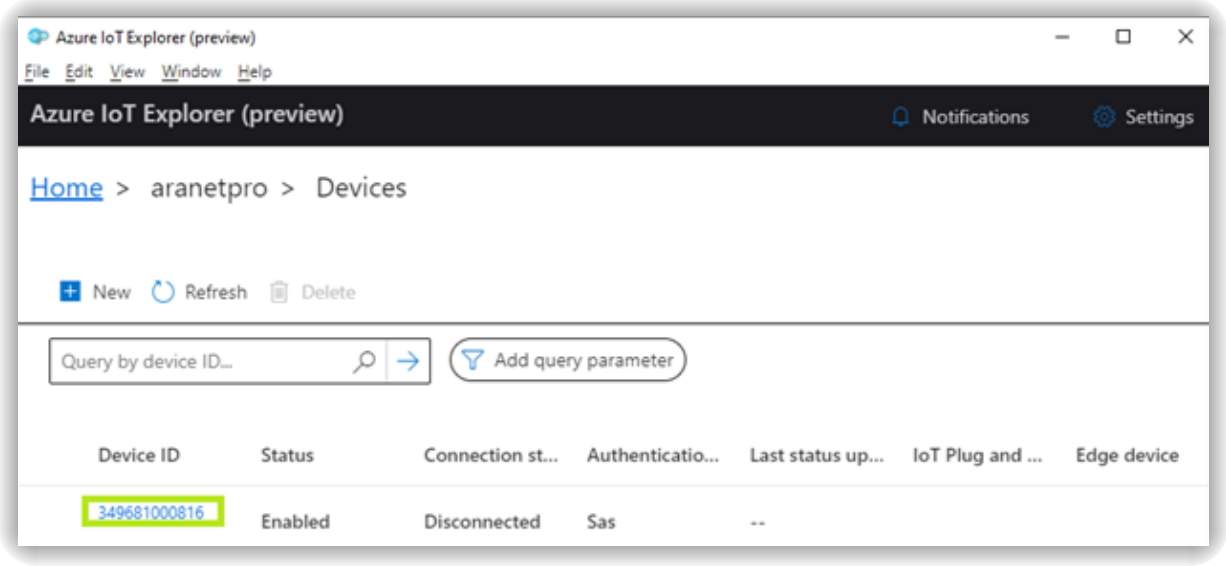
11. Launch *Azure IoT explorer* application and press on *Add connection*



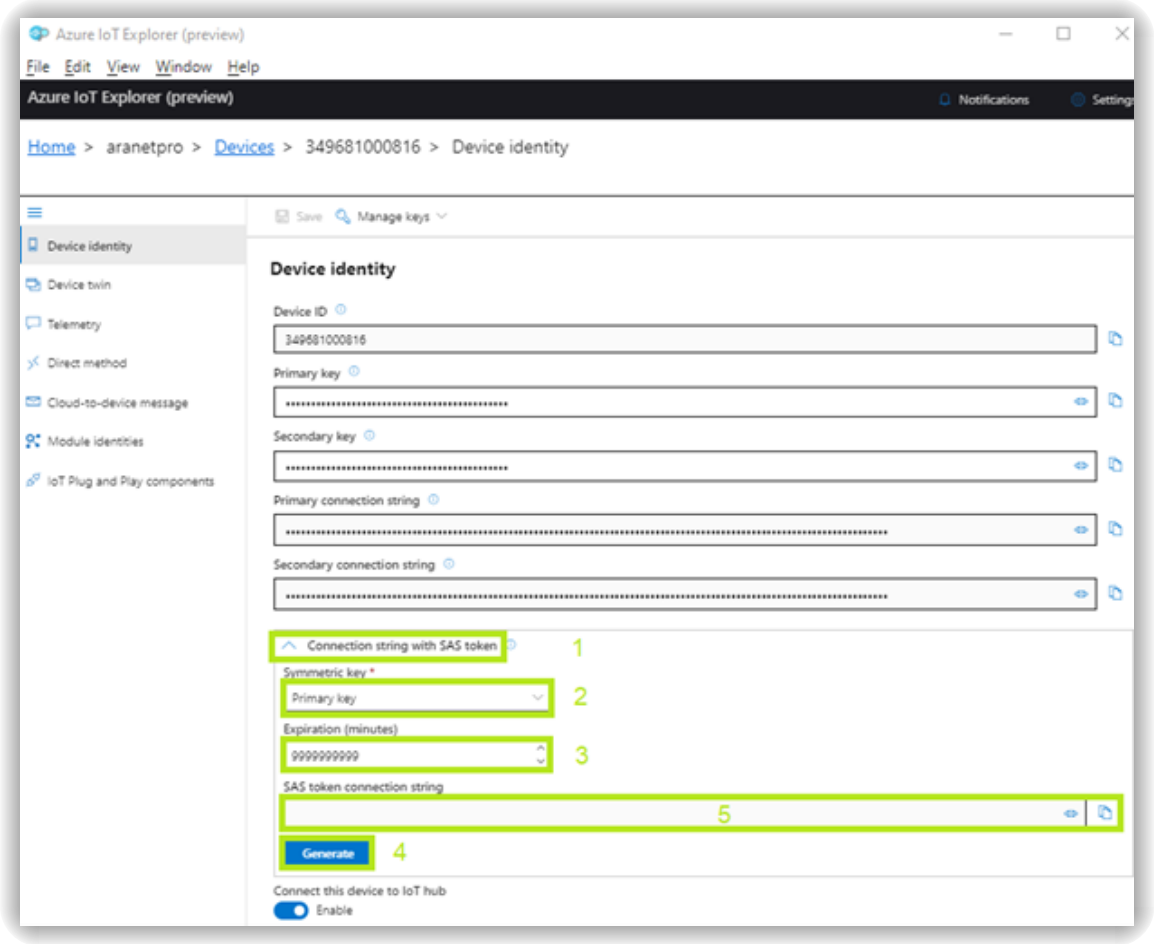
12. In the opened window paste the string from step 9 and press the *Save button*:



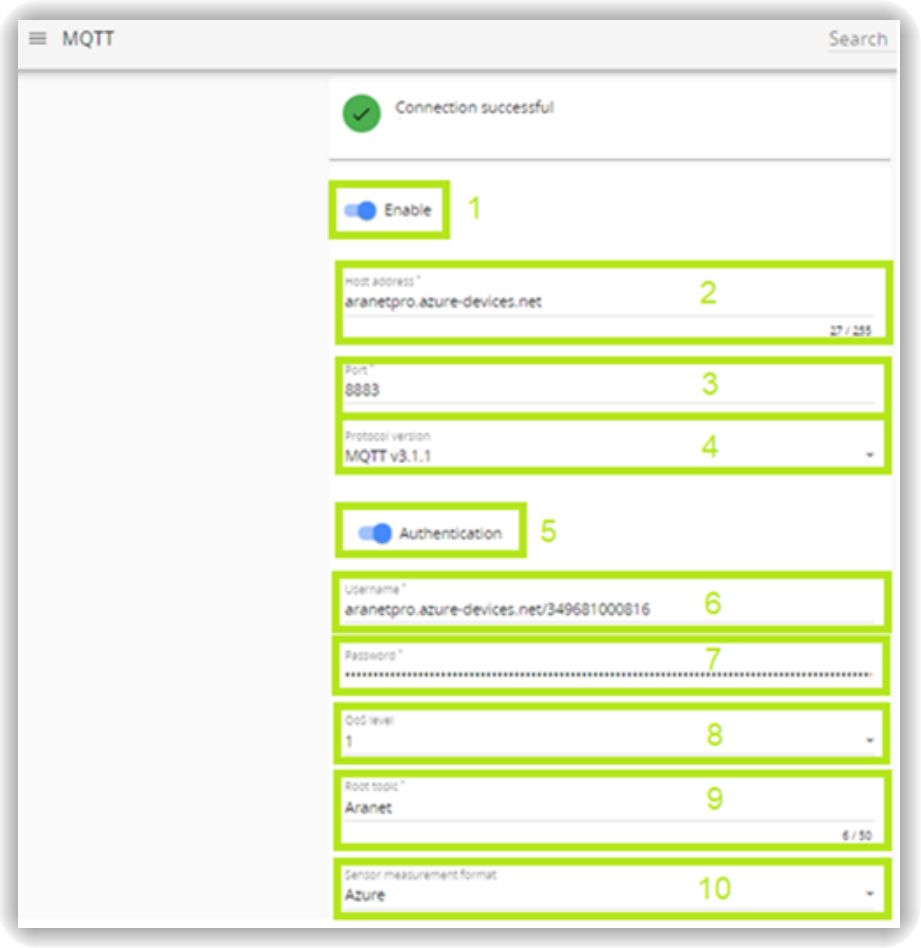
13. Now click on the previously created Aranet PRO base station *IoT devices object*:



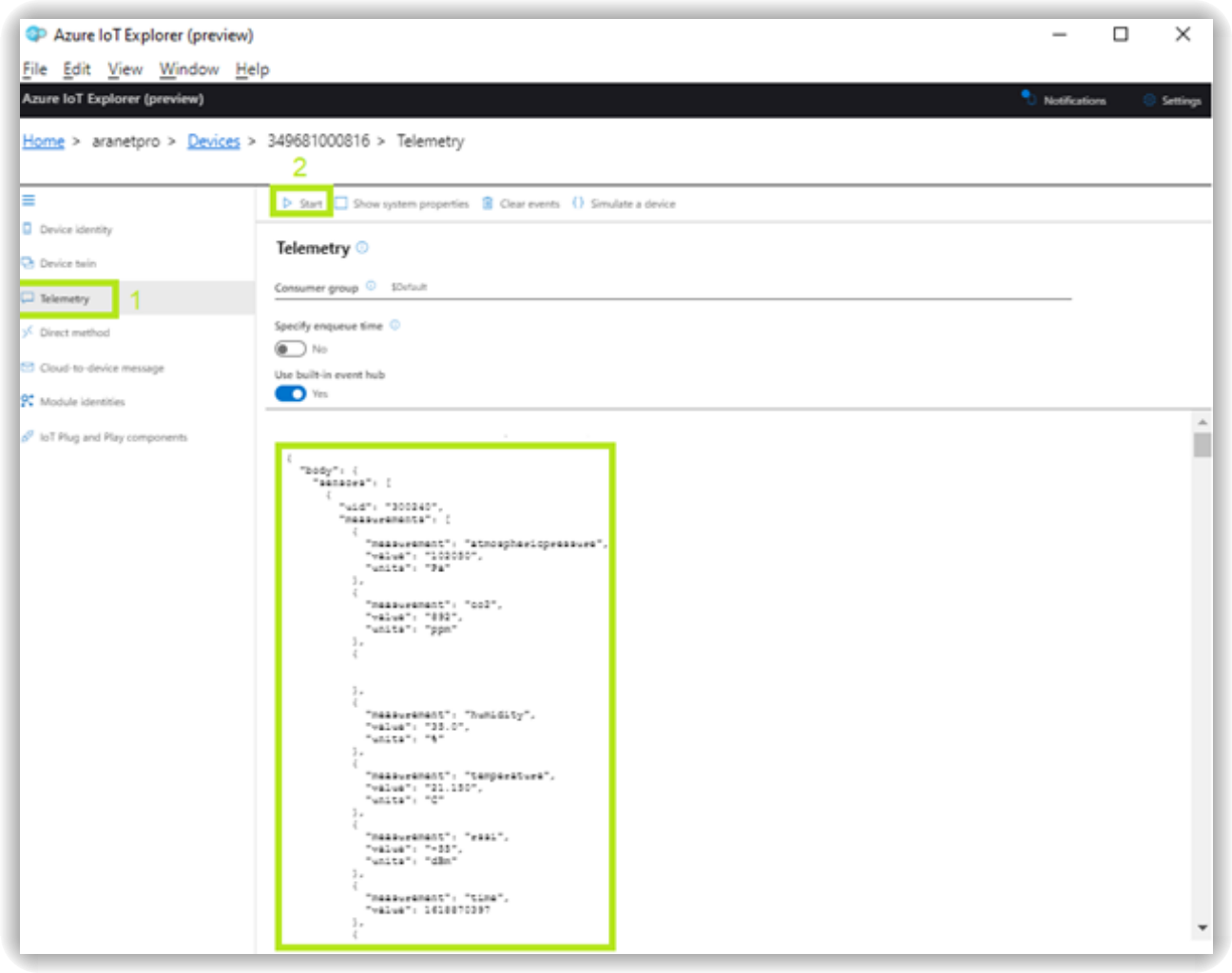
14. Click on *Connection string* with the *SAS token* subsection, select *Primary key* as a *Symmetric key*, enter the necessary *Expiration (minutes) time*, for example, 9999999999, and press the *Generate* button and then copy the generated *SAS token connection string*:



15. Paste the copied string into any text editor, for example, *Notepad* or *Microsoft Word*:
16. Now open the Aranet Pro base stations graphical user interface, section MQTT and:
    1. *Enable* the MQTT connection;
    2. paste the *HostName=* value from the text editor in to the *Host address* field (in our example `aranetpro.azure-devices.net`);
    3. select the *Port* as `8883`;
    4. select the *Protocol version* as *MQTT v3.1.1*;
    5. Enable *Authentication*;
    6. in the *Username* field paste *HostName=* value again, then slash "/" and the serial number of the Aranet PRO base station (in our example, `aranetpro.azure-devices.net/349681000816`);
    7. in the *Password* field paste all of the text string that comes after *SharedAccessSignature=* (in our example, SharedAccessSignature `sr=aranetpro.azure-de`…) etc.
    8. select the *QoS level* as `0` or `1` (2 is not supported by Azure);
    9. enter any *Root topic* value as You want;
    10. select the *Sensor measurement format* as *Azure* (sensor measurements will not be accepted by the Azure platform if the format is selected as *raw* or *JSON*);



17. select the *Encryption* as *TLSv1.2*;
18. and finally press on the *Save* icon:
19. If the connection to the Azure platform is successful, then the corresponding success message will be shown on top of the MQTT page in Aranet PRO base station graphical user interface. Additionally, a user can check what data is published on the Azure IoT hub from the Azure IoT explorer application by clicking on the Telemetry section and then pressing on Start button:



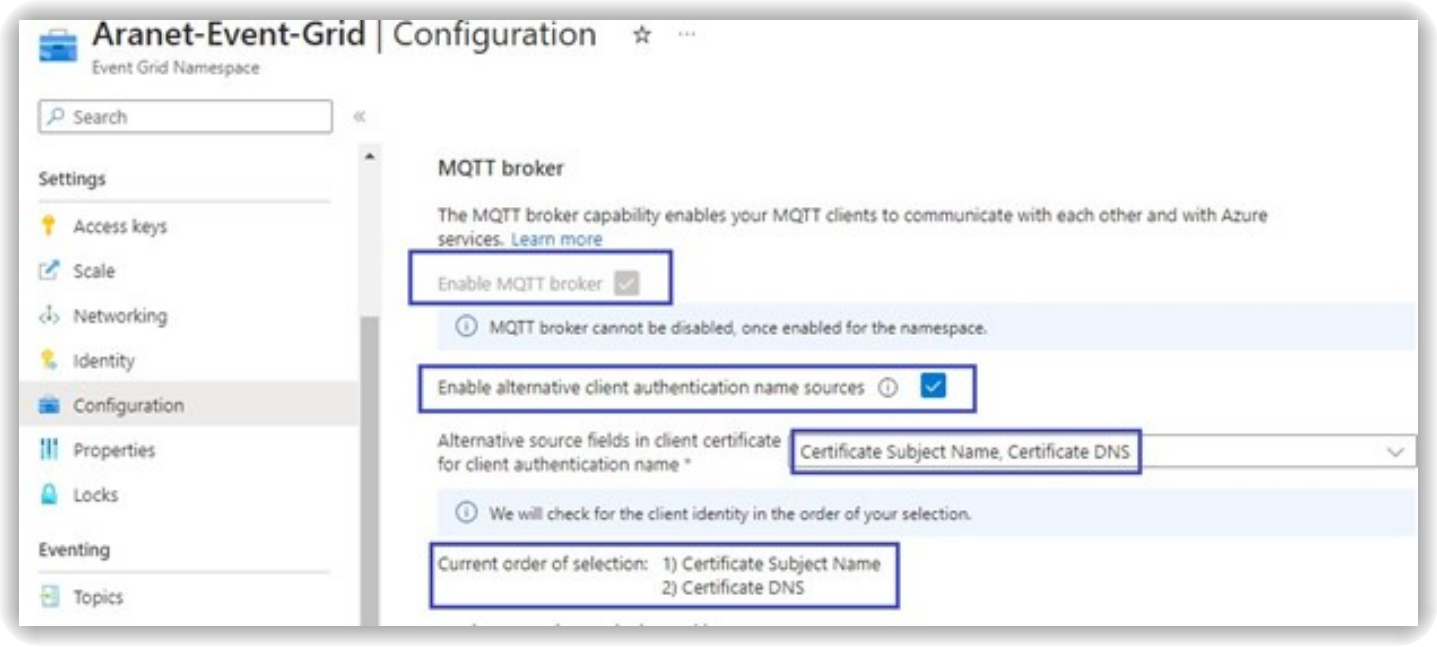
How to Connect Aranet PRO to Azure Event Grid MQTT broker

This example will describe Event Grid Namespace MQTT broker and Aranet PRO base station MQTT configuration details for Client authentication using self-signed CA certificate chain.

Before configuration, make sure you have access to **Event Grid** Namespace and it has *MQTT broker enabled, alternative client authentication name sources* enabled and for client authentication

Alternative source fields in client certificate set in following order:

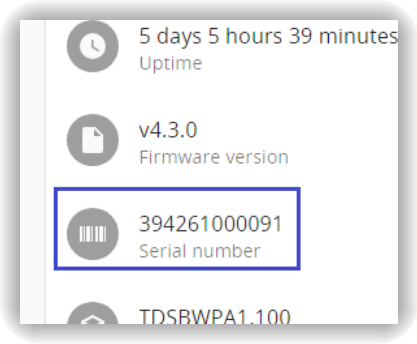
1. Certificate Subject name
2. Certificate DNS



In this example will be used a *self-signed CA certificate file*, *Client certificate file* and *Client key file*.

For *CA certificate's Common Name MQTT hostname* of your Event Grid Namespace will be used.

For *Client certificate's Common Name* the Aranet base station serial number will be used. It will be also used later in Event Grid *Clients* configuration as *Client Authentication Name*.



Generate certificate and key files

In order to generate the certificate and key files, OpenSSL utilities will be used. In this example OpenSSL v3.0.2 is being used.

Additional configuration in OpenSSL configuration file openssl.cnf is required:

Configuration section	Section variable and value pairs
`v3_ca`	`basicConstraints = critical,CA:TRUE` `subjectAltName = @alt_names`
`req`	`x509_extensions = v3_ca,v3_req` `req_extensions = v3_req`
`alt_names`	`DNS.1 =` <Event Grid Namespace MQTT hostname>

Steps of generating the files

1. Generate the *CA key*:

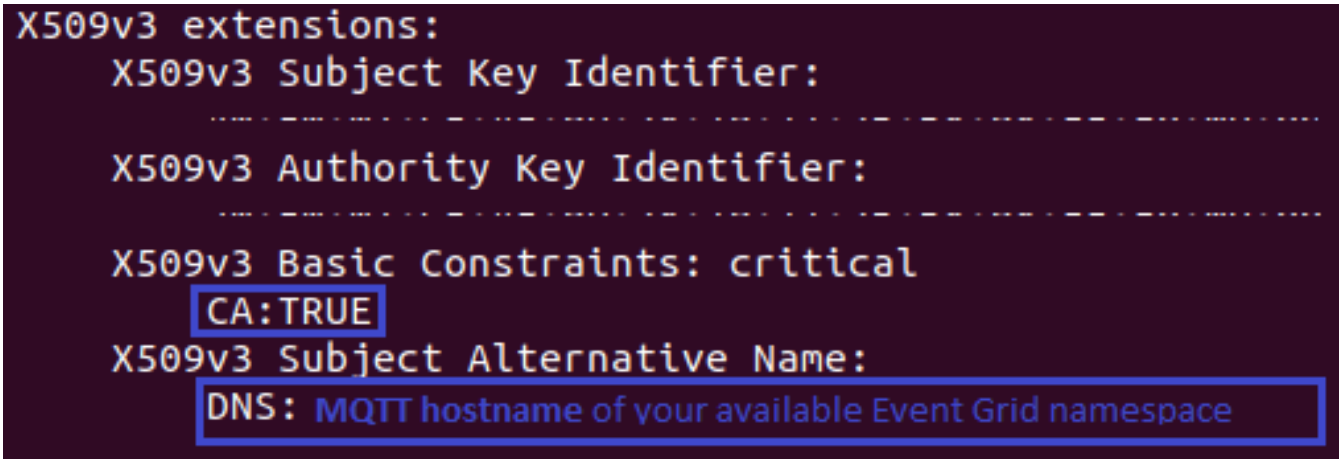
   `openssl genrsa -aes256 -out ca.key 4096`

2. Generate the *CA certificate file* using previously created *CA key file* (for the *Common Name* use the *MQTT hostname* of your available Event Grid Namespace):

   `openssl req -new -x509 -sha256 -days 2525 -key ca.key -out ca.crt - extensions v3_ca`

3. Verify that the created *CA certificate file* contains the necessary `X509v3` extension key - value pairs:

   `openssl x509 -in ca.crt -text`



4. Generate *client key file*:

   `openssl genrsa -out client.key 4096`

5. Generate *client certificate* signing request file for signing the *client certificate file* (for the *Common Name* use the):

   `openssl req -new -sha256 -out client.csr -key client.key -extensions v3_req`

6. Generate client certificate file using previously generated *CSR file* and *CA certificate* and key files:

   `openssl x509 -req -sha256 -in client.csr -CA ca.crt -CAkey ca.key - CAcreateserial -out client.crt -days 2525`

7. Self-signed *CA* and *client certificate* and key files are now created. List of files:

   a. `ca.crt`*root CA certificate* which was used to sign client certificate file (will be used in Event Grid *CA certificate* configuration);

   b. `ca.key`–*CA privatye key* file;

   c. `client.crt`–client certificate file signed with CA certificate (will be used in Aranet PRO base station MQTT configuration);

   d. `client.key`–client private key (will be used in Aranet PRO base station MQTT configuration);

   e. `client.csr`–can be removed (will not be used).

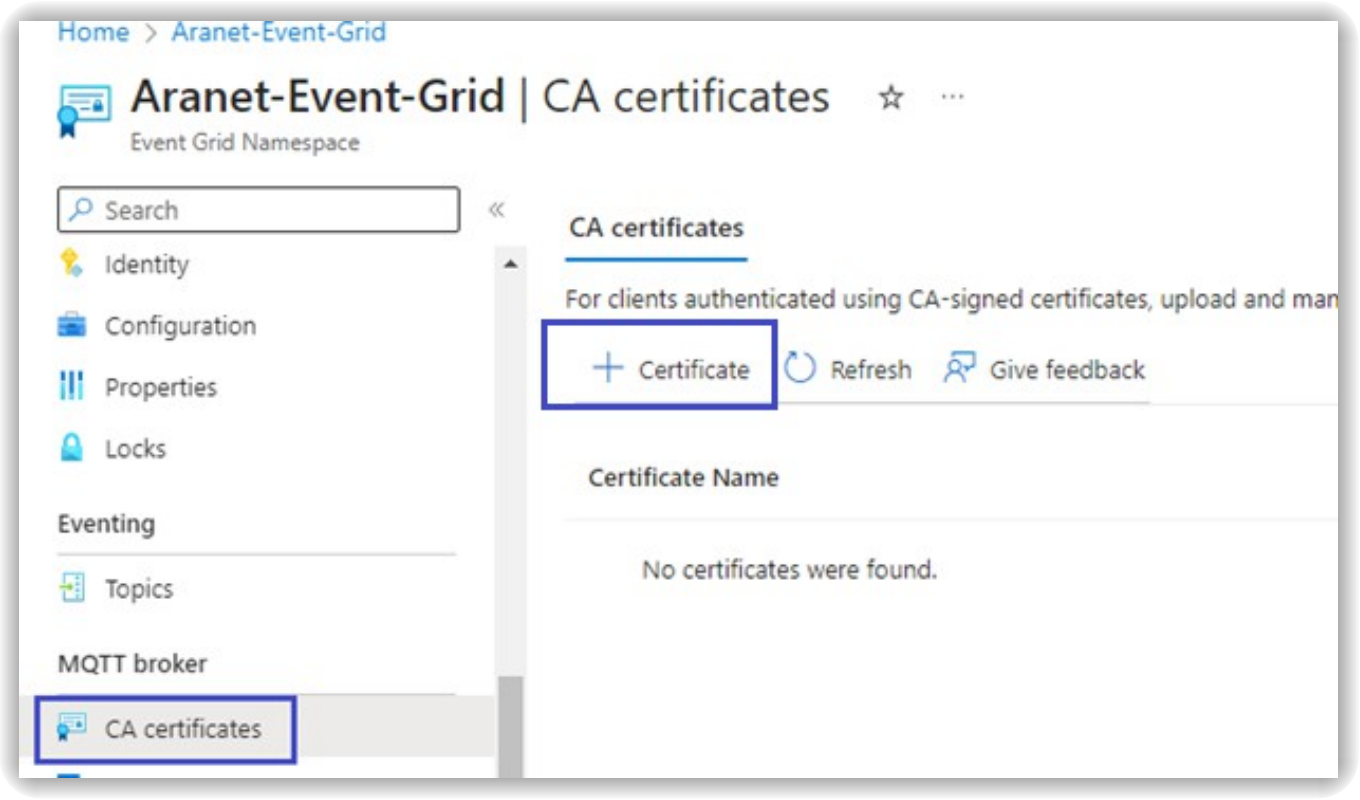
NOTE: if more than one pair of *client key* and *certificate files* are needed (in case if two or more Aranet PRO base stations

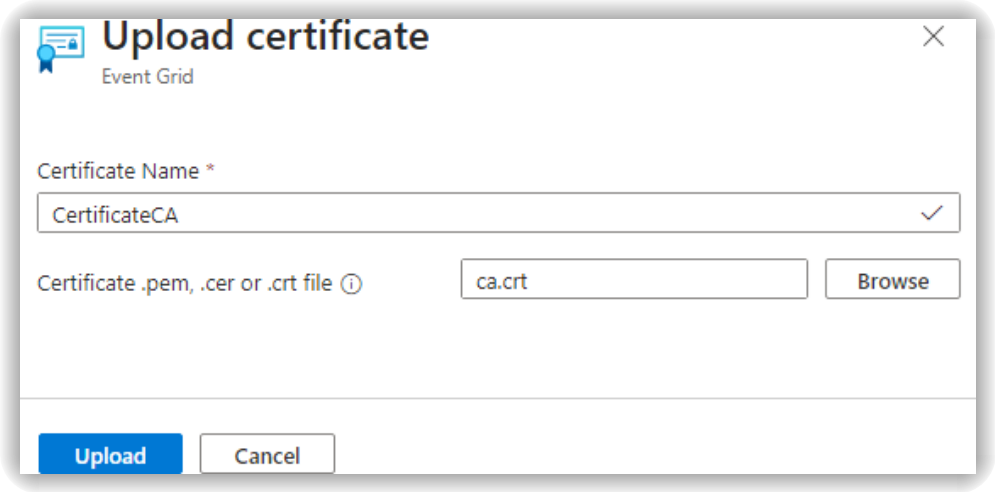
are planned to be connected) then the same pair of *CA certificate* and key files can be used to sign other pairs of *client certificate* and key files as long as these clients are planned to be connected to the same Event Grid Namespace using the same *MQTT hostname* set in the *CA certificate's Common Name*.

Proceed with Event Grid configuration steps.

Event Grid configuration – add CA certificate

In Event Grid Namespace section *MQTT broker* open *CA certificates* and upload `ca.crt` file. Set name as `CertificateCA`.

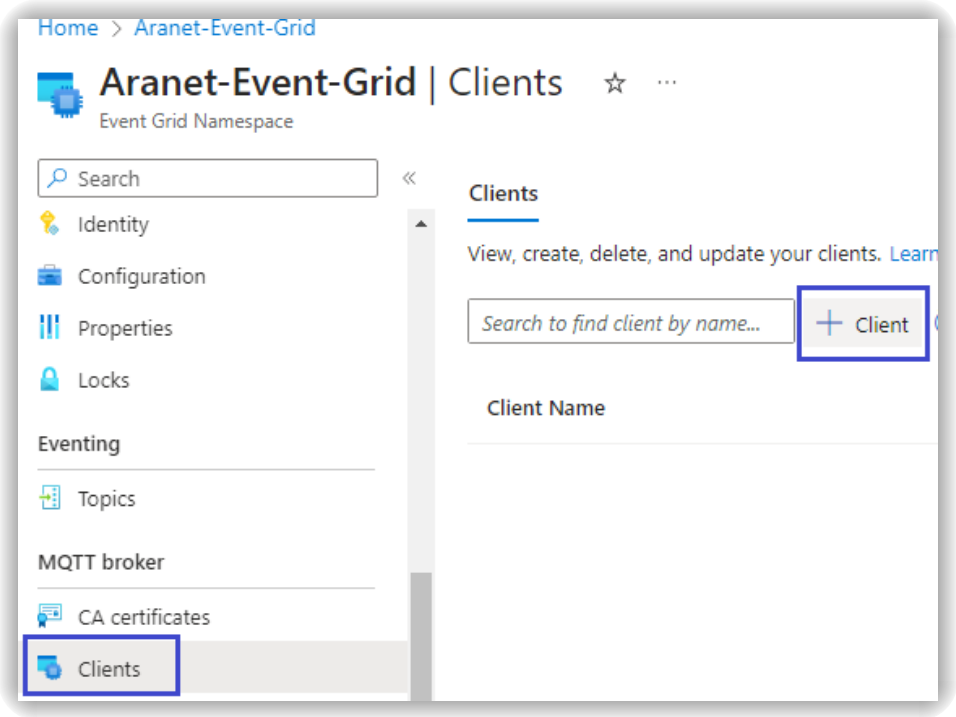




IMPORTANT: It has been noticed that connections from the base station are failing if there are multiple CA certificate files added.

Event Grid configuration – create MQTT broker client

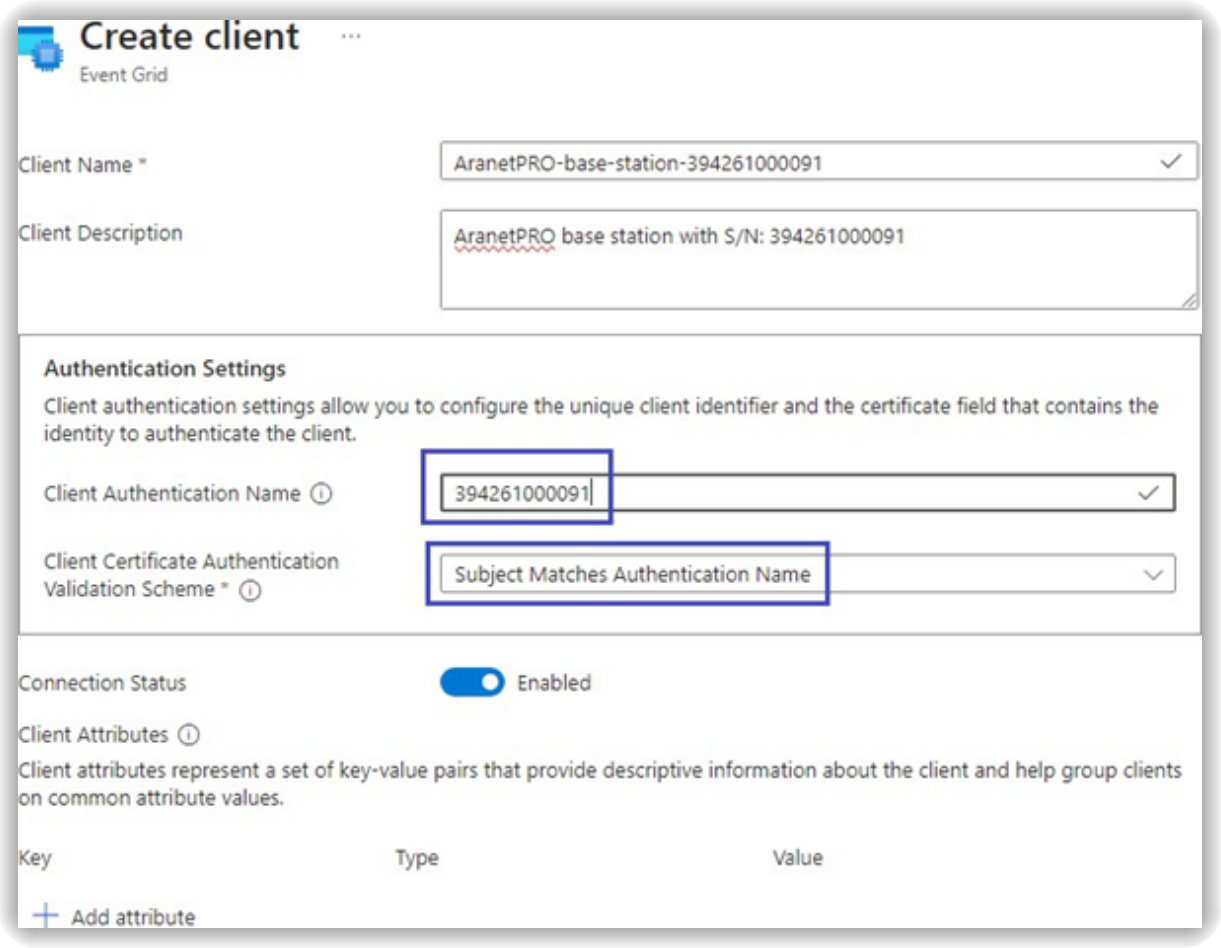
In the same Event Grid Namespace navigate to the *MQTT broker* section *Clients* and press on +" to add a new Client.



For the client following important Authentication Settings must be set:

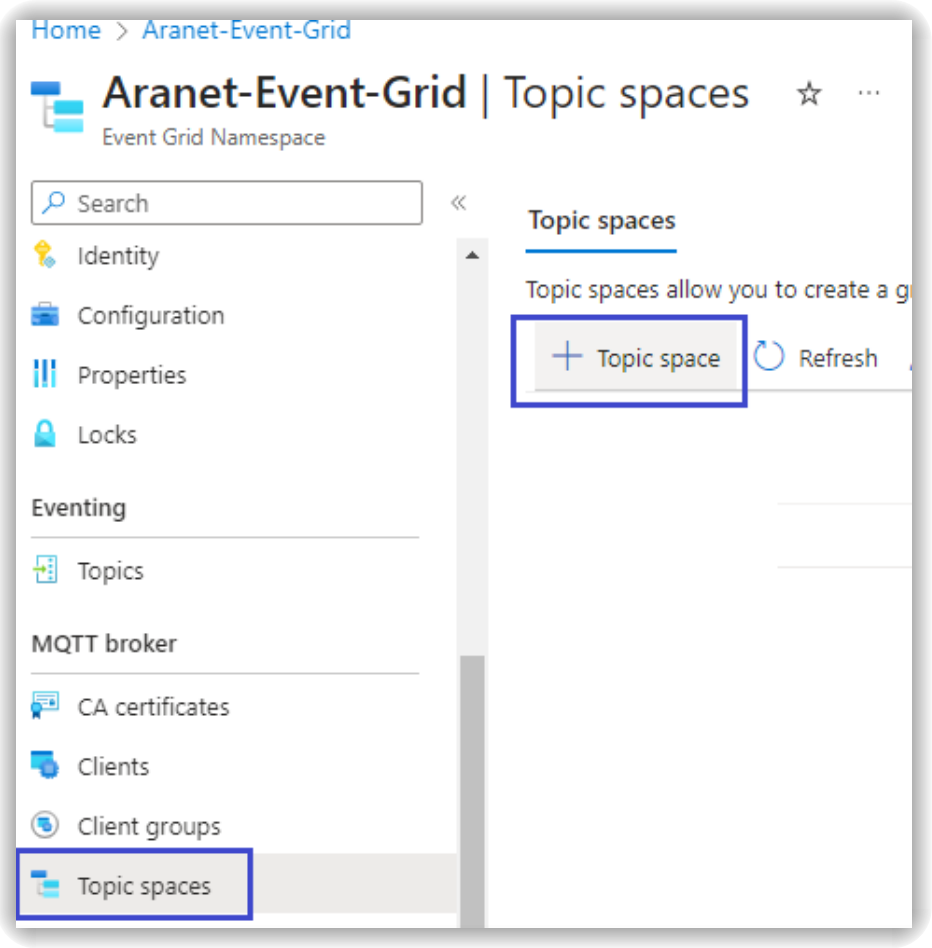
- Client Authentication Name: **<Aranet base station serial number>**
- Client Certificate Authentication Validation Scheme: **Subject Matches Authentication Name**

Set Connection Status to Enabled and press the *Create* button to add the new client for the Event Grid Namespace *MQTT broker*.

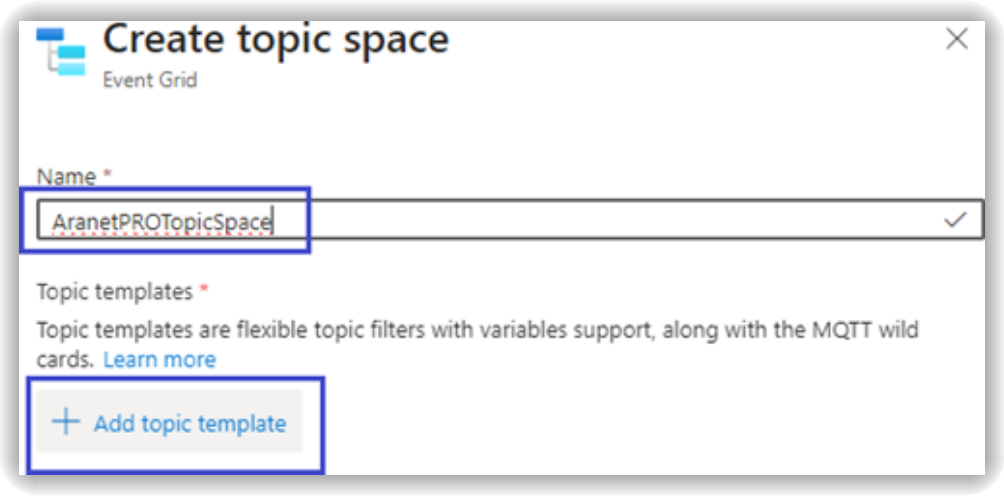


Create root topic for the Event Grid Namespace MQTT broker

In order to publish and subscribe to MQTT messages an additional configuration step – creation of the topic must be performed. For this proceed to the Event Grid Namespace *MQTT broker* section *Topic spaces* and create a new Topic space by pressing **+**.

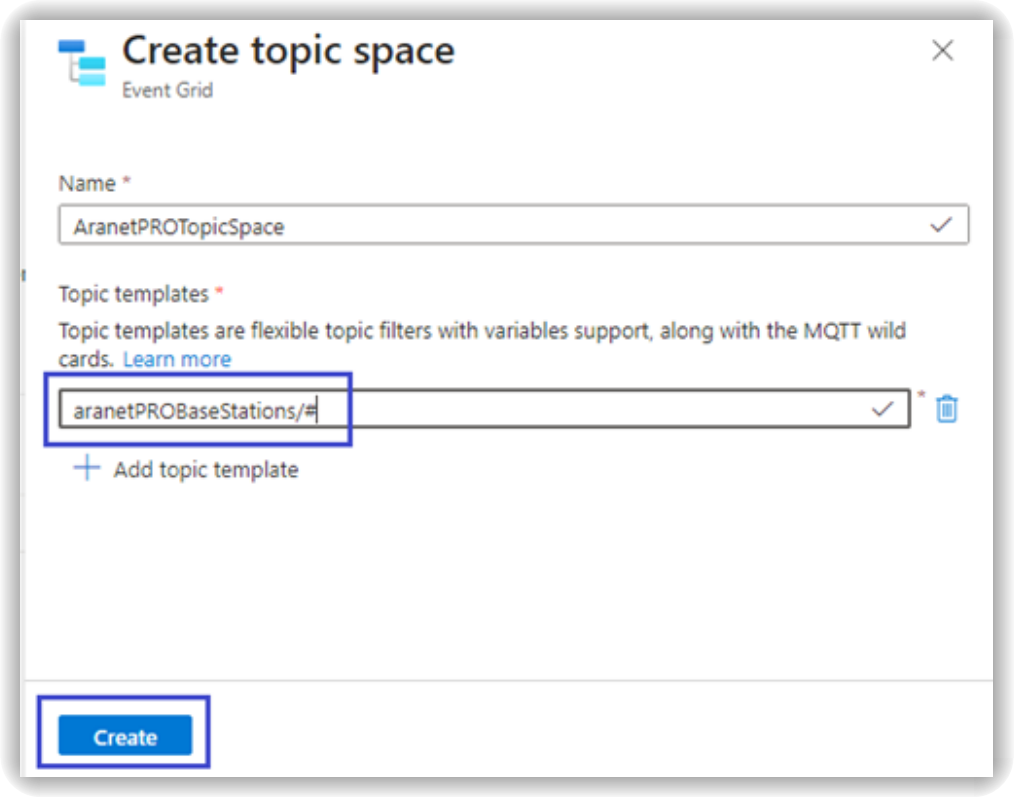


Give a Name for the Topic space. This name will be used later in the *Permission bindings* (publish and subscribe access rights) configuration. Press *Add topic template* to add a new topic template for this Topic space.



For the Topic template set the topic: `aranetPROBaseStations/#`

IMPORTANT: `aranetPROBaseStations` can be replaced with any other name. Topic template string must end with '/#'.



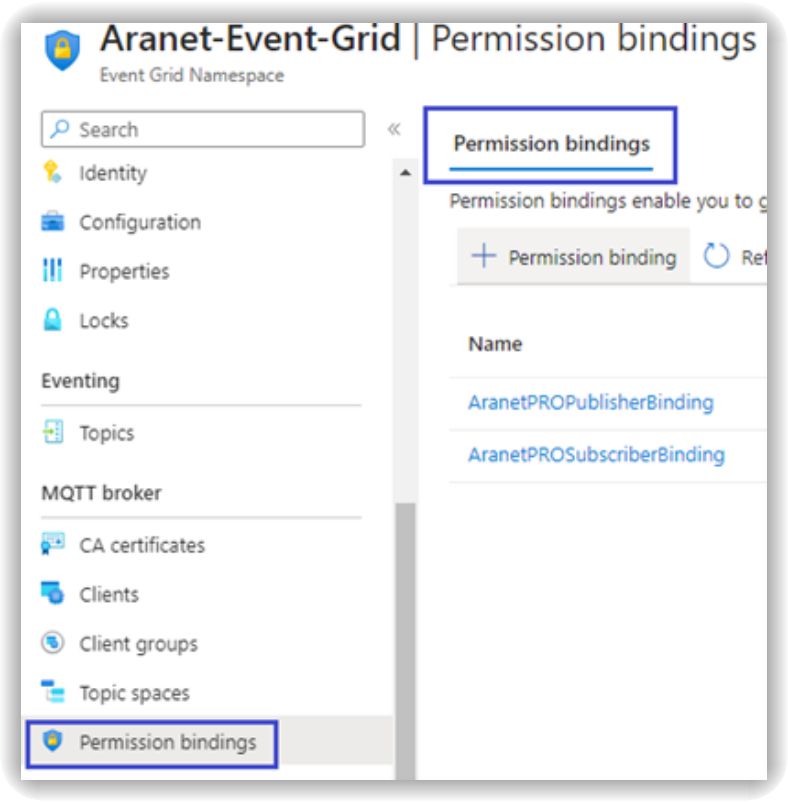
Later in this configuration example manual the topic template `aranetPROBaseStations` will be used in Aranet PRO MQTT configuration as the *Root topic* string.

Finally press "*Create*" button to add a new topic template.

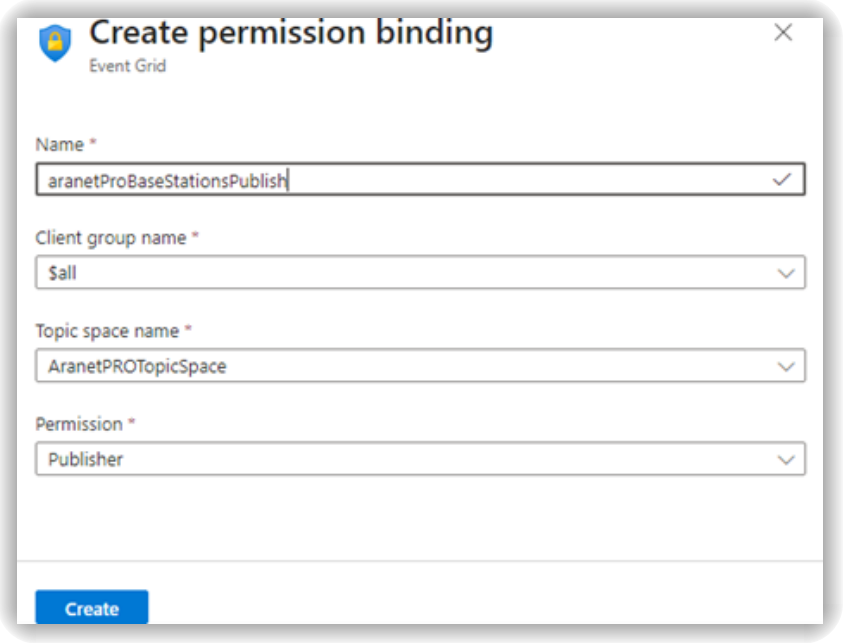
NOTE: in the scenario where multiple base stations are connected to the same MQTT broker hostname, the same topic template can be used across all base stations for publishing and subscribing to MQTT messages.

Setup topic Permission bindings for the Event Grid Namespace MQTT broker

Proceed to the Event Grid Namespace MQTT broker section *Permission bindings*. Press on + to create a new *Permission binding* for MQTT message publishing permission.

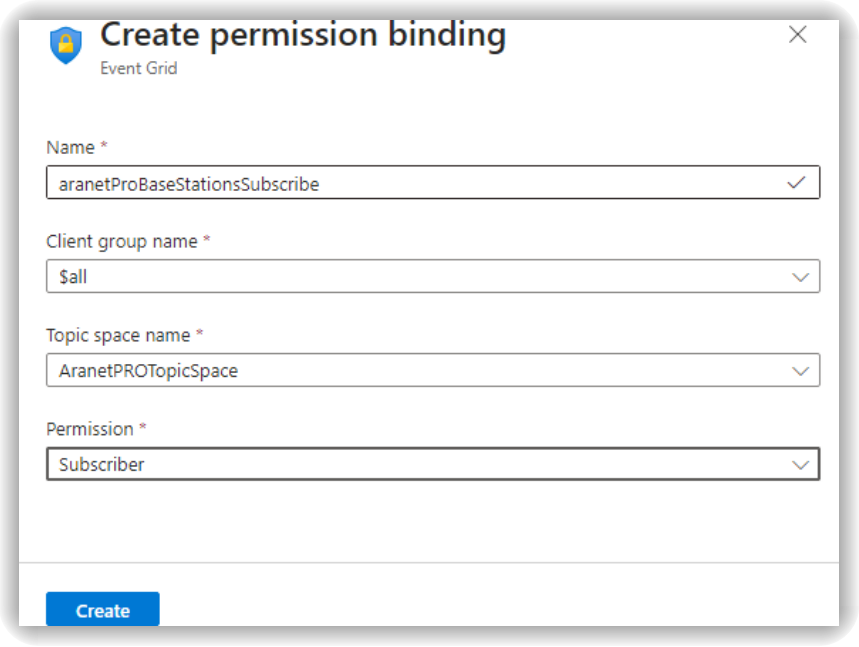


Create permission for the Publisher with following settings:



Press *Create*. Permission for Publisher has been created.

Now press on + to create a new *Permission binding* for MQTT message subscribe permission. Create permission for the Subscriber with following settings:



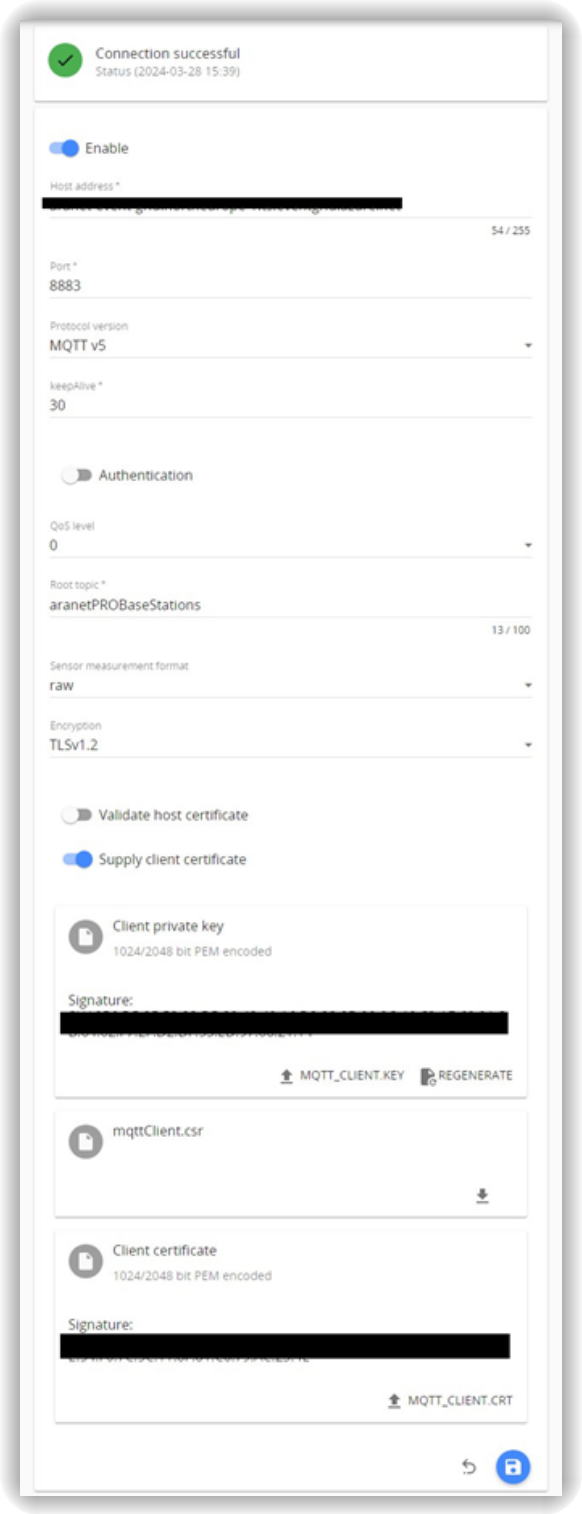
Press *Create* to add permission for the Subscriber.

Now Event Grid Namespace MQTT broker configuration is completed. Proceed to Aranet PRO base station to configure MQTT client settings in order to connect to Event Grid Namespace MQTT broker.

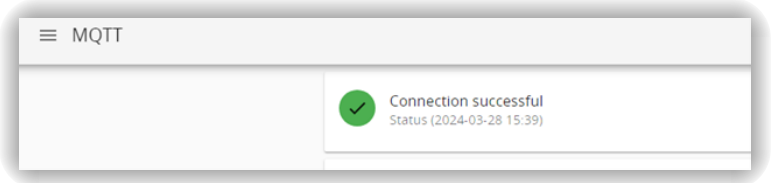
Setup Aranet PRO connection with Event Grid Namespace MQTT broker

Open web GUI of your Aranet PRO base station and proceed to Integrations: MQTT section. Use these settings:

- Enable: `ON`
- Host address: *MQTT hostname* of your available Event Grid Namespace
- Port: `8883`
- Protocol version: `MQTT v5`
- keepAlive: `30 seconds`
- Authentication: `OFF`
- QoS level: `0 or 2` (connection is dropped by using QoS: 1)
- Root topic: `aranetPROBaseStations`
- Sensor measurement format: `raw` or `JSON`
- Encryption: `TLSv1.2`
- Validate host certificate: `OFF`
- Supply client certificate: `ON`
- Client private key: upload `client.key` file
- Client certificate: upload `client.crt` file



Press to save the configuration. If all the steps were performed correctly Aranet PRO MQTT base station MQTT client process will report status 'Connection successful'.



Now messages of the Araent PRO base station MQTT client are being published to the Event Grid Namespace MQTT broker.