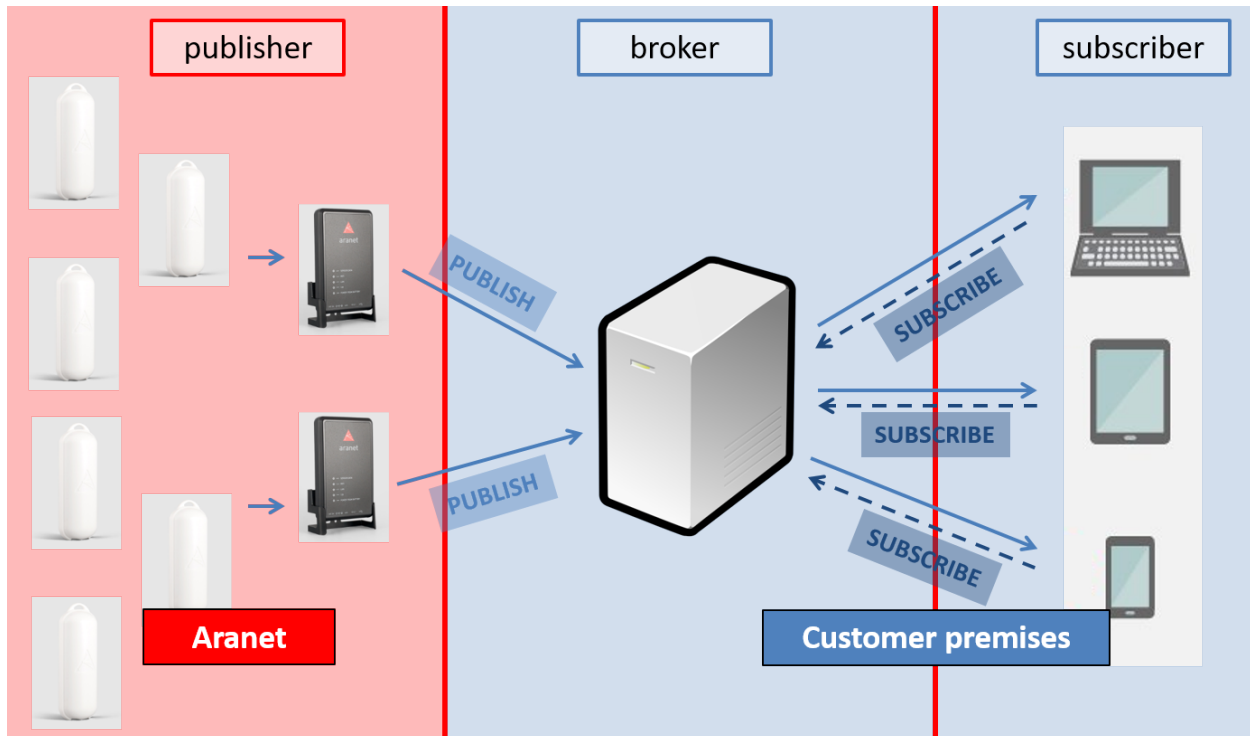


Aranet MQTT functionality and integration with Amazon AWS

1. General MQTT network structure:



2. MQTT message format

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

1) raw

in topic structure `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/measurements/<measurement type>` 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);

- b. **humidity** – relative humidity data is given in [%] (percentage);
- c. **co2** – carbon dioxide concentration level data given in [ppm] (parts per million);
- d. **co2Abc** – shows whether CO2 manual (0.000000) or automatic (1.000000) calibration mode is enabled for the sensor;
- e. **atmosphericpressure** – atmospheric pressure data are given in [Pa] (Pascal);
- f. **voltage** – data are given in [V] (Volts);
- g. **current** – electric current data are given in [A] (Ampere);
- h. **weight** – tarred weight in [kg] (kilogram);
- i. **weightraw** – untarred weight in [kg] (kilogram);
- j. **illuminance** – data from LUX sensor given in [lx] (lux);
- k. **distance** – data are given in [m] (meters);
- l. **vwc** -volumetric water content data of soil/substrate given as [/] (the fraction of one whole);
- m. **bec** – bulk electric conductivity data are given in [S/m] (Siemens per meter);
- n. **pec** - pore water electrical conductivity data are given in [S/m] (Siemens per meter);
- o. **dp** – dielectric permittivity data of soil or substrate given in absolute numbers;
- p. **ppfd** - photosynthetic photon flux density data are given in [umol/(m² s)] (micromol per square meters multiplied by seconds);
- q. **pulses** – pulses in each sensor measurement interval in absolute numbers [pulses];
- r. **derivedp** – pulses measurement in each sensor measurement interval once the sensor conversion rule is applied in user-defined units;
- s. **pulsescumulative** - cumulative pulses in absolute numbers [pulses];
- t. **derivedpc** – cumulative pulses measurement once the sensor conversion rule is applied in user-defined units;
- u. **co** – carbon monoxide concentration level data are given in [ppm] (parts per million);
- v. **differentialpressure** – data are given in [Pa] (Pascal);
- w. **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);
- x. **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);
- y. **derived** – derived measurements in user-defined units;
- z. **rssi** – received signal strength data given in [dBm];
- aa. **battery** – battery charge level which is given as [/] (the fraction of one whole);
- bb. **time** – measurement time in Unix epoch format:
<https://www.freeformatter.com/epoch-timestamp-to-date-converter.html>

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

```

▼ broker.hivemq.com
  ▼ Aranetest
    ▼ 394260700033
      ▼ sensors
        ▼ 100051
          productNumber = TDSPT001
          ▼ measurements
            ▼ humidity = 42.0
              units = %
            ▼ temperature = 19.950
              units = C
            ▼ rssi = -74
              units = dBm
            time = 1618671102
            ▼ battery = 0.07
              units = /

```

- 2) **JSON** (only measurement values are sent, but no sensor measurement units and alarm messages)
 in topic structure <root topic name>/<PRO base serial number>/sensors/<sensor ID>/json/measurements

```

▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = 2021B7
          productNumber = TDSPT306
          ▼ json
            measurements = { "temperature": "21.800", "rssi": "-74", "time": 1618671680, "battery": "0.93" }

```

- 3) **Azure** format for sensor data publishing to Azure IoT Hub platform:

▼ devices/349681000816/messages/events/msgType=sensorMeasurements&uid=101306

```
{
  "sensors": [
    {
      "uid": "101306",
      "measurements": [
        {
          "measurement": "humidity",
          "value": "38.0",
          "units": "%"
        },
        {
          "measurement": "temperature",
          "value": "21.850",
          "units": "C"
        },
        {
          "measurement": "rssi",
          "value": "-47",
          "units": "dBm"
        },
        {
          "measurement": "time",
          "value": 1618691111
        },
        {
          "measurement": "battery",
          "value": "0.90",
          "units": "/"
        }
      ]
    }
  ]
}
```

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**/ +

- a. **battery/activeSince** – showing time in Unix epoch format when low battery charge alarm appeared in the sensor:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 30014E
          productNumber = TDSPC004
          name = 30014E
            ▼ alarms
              ▼ battery
                activeSince = 1618650883
```

- b. **channel/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded the event when sensor started using different radio channel than configured on the base itself:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = 2021B7
          productNumber = TDSPT306
            ▼ alarms
              ▼ channel
                activeSince = 1618673067
```

- c. **packetsLost/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded that measurement data from some sensor is not received/missing:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 101306
          ▼ alarms
            ▼ packetsLost
              activeSince = 1618673049
```

d. **errorFlags/**

- a. **value** - showing number error code value when instead of measurements error message was received from the sensor;
- b. **activeSince** - showing time in Unix epoch format when instead of the measurement error message was received from the sensors:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 1022FF
          name = 1022FF
          productNumber = TDSPT409
            ▼ alarms
              ▼ errorFlags
                value = 33
                activeSince = 1618676856
```

- e. **<measurement>** - shows for which measured parameter configured alarm threshold was breached;
- a. **value** – shows measurement value that generated the alarm;
 - b. **diff** – shows value by what configured alarm threshold was breached. It is positive when the upper threshold was breached and negative when the lower threshold is breached;
 - c. **activeSince** - shows time in Unix epoch format when alarm threshold was breached:

```

▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 1022FF
          name = 1022FF
          productNumber = TDSPT409
          ► measurements (9 topics, 45 messages)
          ▼ alarms
            ▼ temperature
              value = 24.65
              diff = 18.25
              activeSince = 1618677267

```

Aranet PRO base station publishes also:

- 1) name that is assigned to the sensor on the Aranet PRO base station in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/name and
- 2) product number of the sensor in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/productNumber:

```

▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = Name of the sensor
          productNumber = TDSPT306

```

- 3) name of the Aranet PRO base station itself in topic <root topic name>/<PRO base serial number>/name:

```

▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      name = Name of the Base

```

- 4) name of the group that is assigned to the sensor in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/group and
- 5) numeric identifier of this sensor group in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/groupid:

```

▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 300240
          name = CO IP67
          groupId = 0
          group = Custom Group name
          productNumber = TDSPC005
          ▼ measurements

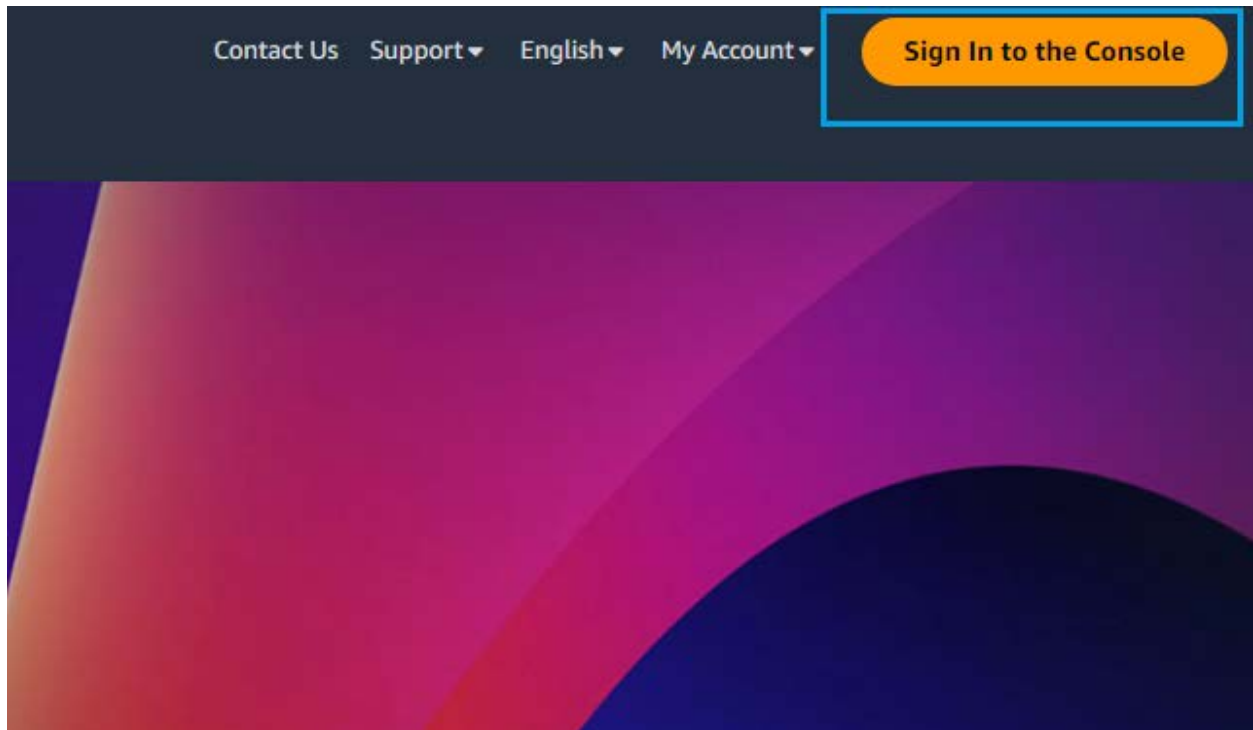
```

3. MQTT connection configuration with Amazon AWS platform

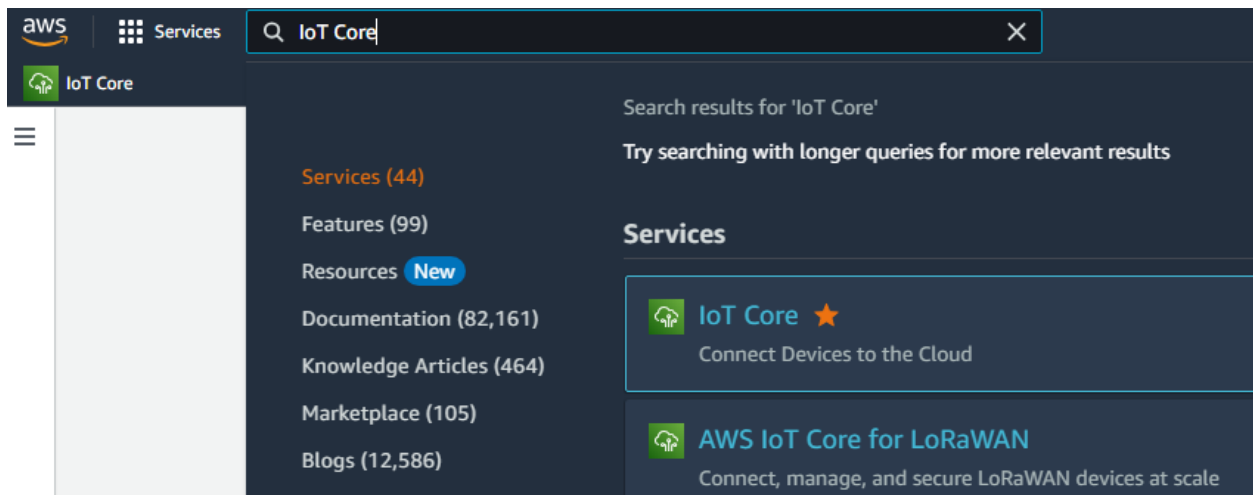
Access AWS IoT Core console

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) In web browser open AWS page <https://aws.amazon.com/>, sign in to the Console:



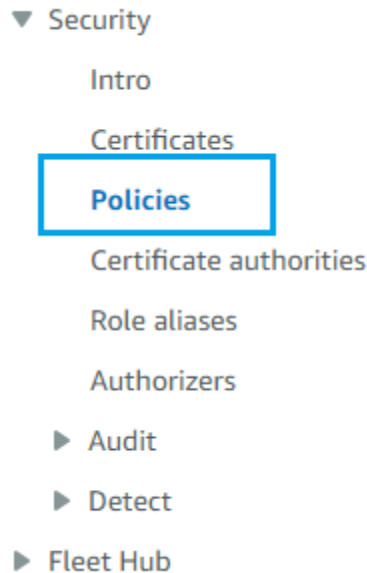
- 2) In console search for "IoT Core" service:



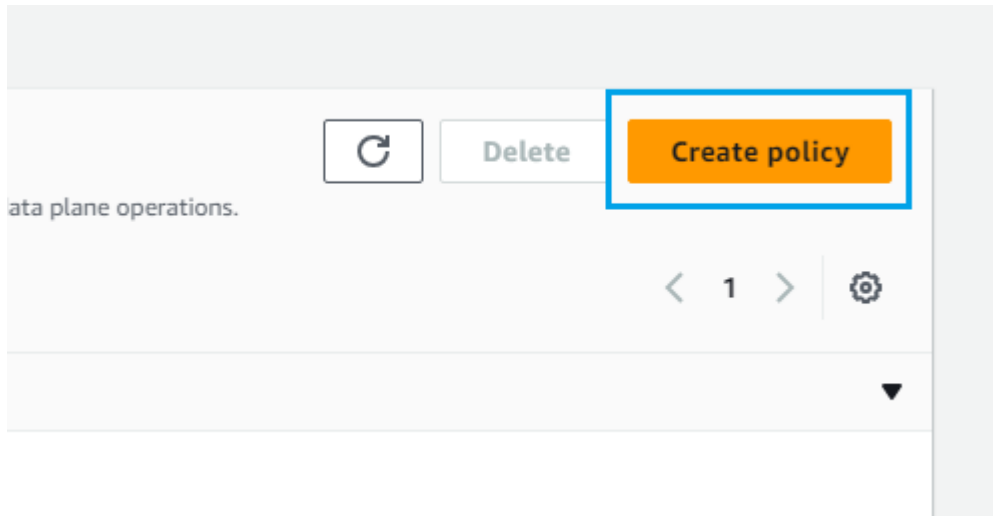
Create a policy for MQTT connect/publish/subscribe actions

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.

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



- 2) Create a new policy (press on “Create policy”):



3) Enter the name for the policy:

Policy properties
AWS IoT Core supports named policies so that many identities can reference the same policy document.

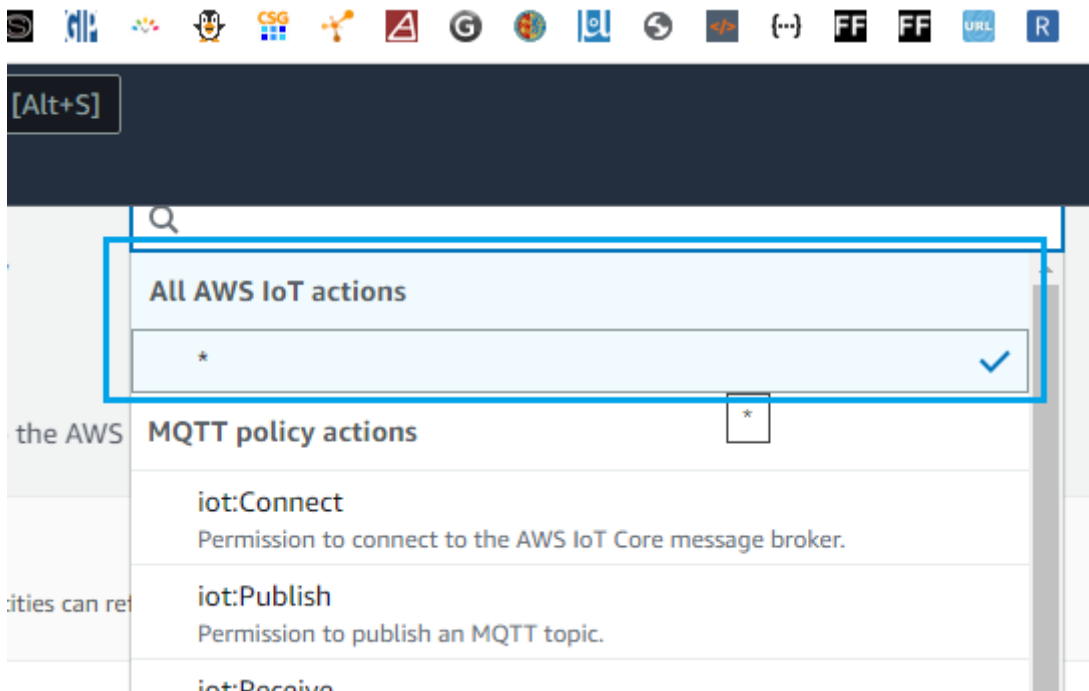
Policy name

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_),

► **Tags - optional**

4) In the “Policy document” enter following properties:

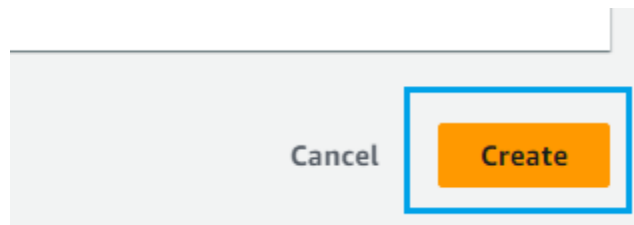
- a. “Policy effect”: “Allow”,
- b. “Policy action”: select “All IoT actions”,



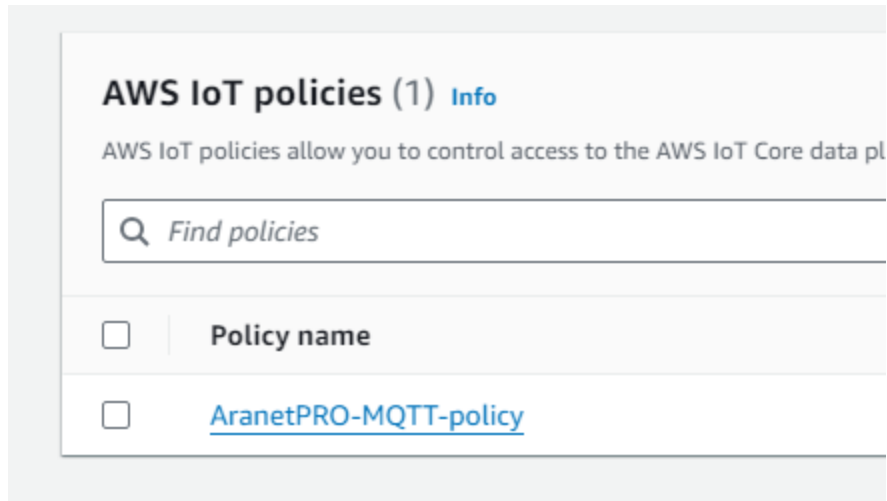
- c. "Policy resource": enter "*"

Policy resource

- 5) Press "Create".



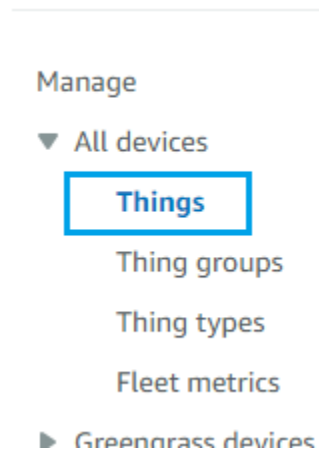
- 6) A new policy which allows to perform all MQTT protocol actions on all MQTT topics has been created.



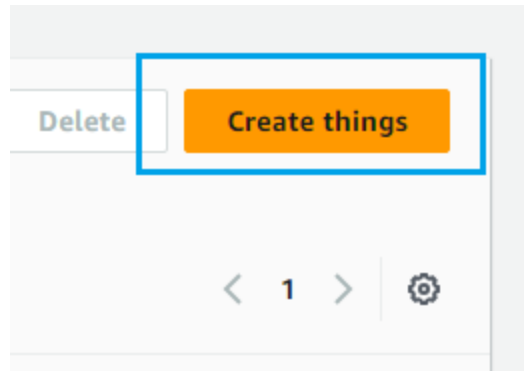
Create a thing in the IoT Core

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 section “**Create a policy for MQTT connect/publish/subscribe actions**”

- 1) In main menu (left side in the IoT Core console) open Manage -> Things :



- 2) Press on “Create things”:



3) Select “Create single thing” and press “Next”

Number of things to create

☒ **Create single thing**
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

☐ **Create many things**
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Cancel **Next**

4) Enter the name for a thing:

Thing properties [Info](#)

Thing name

My-Aranet-PRO-base

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ **Thing type** - *optional*
- ▶ **Searchable thing attributes** - *optional*
- ▶ **Thing groups** - *optional*
- ▶ **Billing group** - *optional*
- ▶ **Packages and versions** - *optional*

5) For “Device Shadow” select “No shadow”:

Device Shadow [Info](#)

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state i shadow using either HTTPs or MQTT topics.

☒ **No shadow**

☐ **Named shadow**

Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

☐ **Unnamed shadow (classic)**

A thing can have only one unnamed shadow.

6) Device certificate – there are multiple options. In this example first option: “Auto-generate a new certificate” will be used. Press “Next”.

Device certificate

☒ **Auto-generate a new certificate (recommended)**
 Generate a certificate, public key, and private key using AWS IoT's certificate authority.

☐ **Use my certificate**
 Use a certificate signed by your own certificate authority.

☐ **Upload CSR**
 Register your CA and use your own certificates on one or many devices.

☐ **Skip creating a certificate at this time**
 You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel
Previous
Next

7) Select the policy which was created previously, and press "Create thing":

Policies (1/1)

Select up to 10 policies to attach to this certificate.

< 1 >
⚙

<input checked="" type="checkbox"/>	Name
<input checked="" type="checkbox"/>	AranetPRO-MQTT-policy

Cancel
Previous
Create thing

8) Download certificates and keys. These files will be required later when configuration on base station for MQTT will be performed. NOTE: downloaded certificate files must be stored in a secure place. Download and rename files accordingly:

- Download "Device certificate" file and rename it as "**aranet-pro-base.crt**"


Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate

019d492212f...te.pem.crt


Deactivate certificate

 Download

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


Key files

The key files are unique to this certificate and can't be downloaded after you leave this page.
Download them now and save them in a secure place.

 This is the only time you can download the key files for this certificate.


Public key file

019d492212f0936a0bc9225...a49db22-public.pem.key

 Download

Private key file

019d492212f0936a0bc9225...49db22-private.pem.key

 Download


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

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint


RSA 2048 bit key: Amazon Root CA 1

 Download




Amazon trust services endpoint

ECC 256 bit key: Amazon Root CA 3

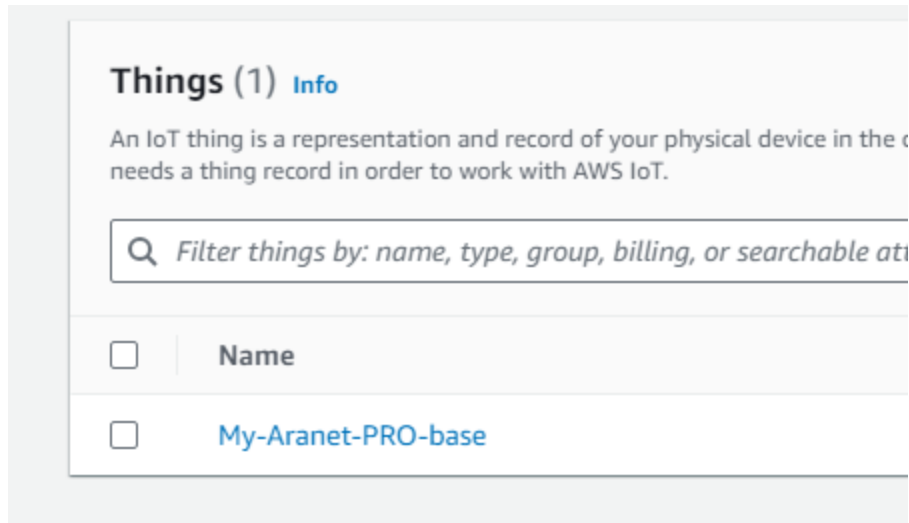
 Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. [Learn more](#) 

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

 **aranet-pro-base.crt**
 **aws-root-ca.crt**
 **aranet-pro-base-private.key**

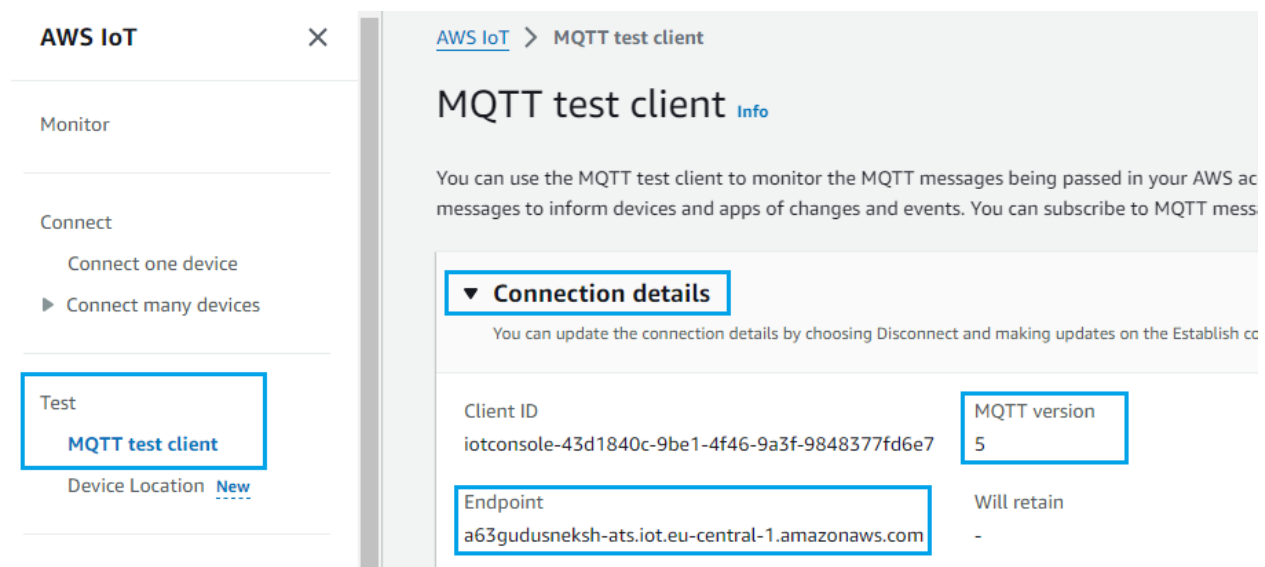
- e. Download also public key (it will not be required) and press “Done”. A new “thing” has been created.



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

In order to connect 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”**.

- 1) Following step will be used to determine what is the “Host address” to which Aranet PRO base station will connect and the MQTT protocol version. In AWS IoT Core main menu open Test -> MQTT test client and press on “Connection details”:



- 2) “Endpoint” will be used for “Host address” (in Aranet PRO base) and “MQTT version” for “Protocol version” accordingly.

4. Aranet PRO base station configuration interface

Aranet PRO base station connection to MQTT broker is configured in the **MQTT** section of the graphical user interface. In the example below we will use configuration for connection to Hivemq public MQTT broker *broker.hivemq.com*:

The screenshot displays the MQTT configuration interface of the Aranet PRO base station. At the top, there is a 'MQTT' tab and a search bar. Below the tab, a green checkmark indicates a 'Connection successful' status with a timestamp of 'Status (01-12-2023 16:13)'. The configuration fields are as follows:

- 1** Enable: A toggle switch is turned on.
- 2** Host address *: a3iardqkibzsn-ats.iot.eu-north-1.amazonaws.com
- 3** Port *: 8883
- 4** Protocol version: MQTT v5
- 5** keepAlive *: 10
- 6** Authentication: A toggle switch is turned off.
- 7** QoS level: 1
- 8** Root topic *: Aranet
- 9** Sensor measurement format: raw

- 1) **Enable** – enable 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”;

Encryption
TLSv1.2
10

☒ Validate host certificate
a

Host CA certificate
1024/2048 bit PEM encoded
b

Signature:
SHA256:8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7
F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6E

MQTT_CA.CRT

☒ Supply client certificate
c

Client private key
1024/2048 bit PEM encoded
d

Signature:
SHA256:A7:6A:34:F0:DD:95:31:3A:D7:A3:70:08:91:6A:9B:1A:FF:95:46:B
4:ED:68:A1:9B:74:EC:66:B3:96:CD:B6:D6

MQTT_CLIENT.KEY
REGENERATE


mqttClient.csr
e



Client certificate
1024/2048 bit PEM encoded
f

Signature:
SHA256:BA:31:AF:9B:97:02:5C:31:46:AB:88:1F:6C:F0:EF:FF:CD:45:D3:8
5:EB:6D:2E:9F:14:2E:DF:0E:E3:D7:40:A5


MQTT_CLIENT.CRT

10) **Encryption** - Amazon AWS requires certificate validation. Use “TLSv1.2”.

- Validate host certificate** – enable to upload necessary secure connection certificates;
-  MQTT_CA.CRT - use “aws-root-ca.crt” file saved from Amazon AWS (see previous Chapter for details);
- Supply client certificate** - enable to upload the device public certificate and private key for secure connection to MQTT broker

- d.  MQTT_CLIENT.KEY - use “**aranet-pro-base-private.key**” file saved from Amazon AWS (see previous Chapter for details);
- e.  MQTT_CLIENT.CRT - use “**aranet-pro-base.crt**” file saved from Amazon AWS (see previous Chapter for details);

11) When all necessary configuration parameters are entered, they should be saved by pressing the

blue Save icon  . If configured MQTT connection is successful, then ***Connection successful*** message will be shown on the top of the page showing also the precise time when the connection was established.