Azure

Summary Information

Communication Driver Name: MQTTAzure

Implementation DLL: T.ProtocolDriver.MQTTAzure.dll

Interface: TCPIP

Protocol: MQTT (Message Queuing Telemetry Transport) is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol

PC Hardware requirements: Ethernet board

Implemented Methods: Connect, Disconnect, Subscribe, Unsubscribe, and Publish

---

Channel Configuration

Protocol Options

Not used in this driver

Node Configuration

Station Configuration

Station syntax: <URL>;<Port>;<ClientID>;<Username>;<Password>;<SslProtocol>;[X509Certificate];<PayloadFormat>

Where:

- **<URL>** = MQTT Broker (Server) name. It must be the same name configured in the host Name parameter in the Azure IoT Hub
- **<Port>** = MQTT Broker port. It must be the same port configured as the listening port in the Broker. Default value is 8883
- **<ClientId>** = Device ID configured in the Azure IoT Hub
- **<Username>** = Username defined in the MQTT Broker and is a concatenation of the Host Name and the Client ID. Syntax: <URL>/<ClientId>
- **<Password>** = Password defined in the MQTT Broker. It is the SAS Token defined for the Device ID
- <**SslProtocol>** = IoT Hub uses Transport Layer Security (TLS) to secure connections from IoT devices and services. Three versions of the TLS protocol are currently supported, namely versions 1.0, 1.1, and 1.2. TLS 1.0 and 1.1 are considered legacy and are planned for deprecation. For more information, see Deprecating TLS 1.0 and 1.1 for IoT Hub. To avoid future issues, use TLS 1.2 as the only TLS version when connecting to the IoT Hub.
- **[X509Certificate]** = Optional. Path of the X509 Certificate using TLS v1.0. This must be the complete path of the X509 certificate in the client computer. The certificate must be installed in the computer. One of the easiest ways to install the certificate on the client computer is to use the wizard to import the certificates through the "Internet Options" and import the certificate in the "Trusted Root Certification Authorities". You need to import the certificate in DER format.
- **<Payload Format>** = Format of the message sent to the Broker. By default, the payload is set to follow JSON/SparkplugB format.

Station Examples

URL= CustomHub-MQTT.azure-devices.net

Port= 8883

Client ID= MQTT_Device

Username= CustomHub-MQTT.azure-devices.net/ MQTT_Device

Password= SharedAccessSignature sr=CustomHub-MQTT.azure devices.net&sig=ygIt%2FG8fIFAXhUwGXBlm6WKDK5IhbmVHkUs4atsbXFU%3D&skn=iothubowner&se=1612 031715

Ssl Protocol= <TLS>

X059 Certificate=

Payload Format= JSON/SparkplugB

For more information on how to set up a connection with Azure IoT Hub and find the required parameters, see Append.

Point Configuration

The MQTT and Azure protocol supports Write Commands only.

Address

The Address syntax is: [Payload]:<Group >;<Node>;<Device>;

Where:

- [**Payload**]= It is a fixed parameter that assists the user in understanding that the address configuration (group_id, edge_node_id, and devised) are the user-defined parameters in the Payload structure, similar to MQTT + SparkplugB protocol.
- <**Group**> = Provides a logical grouping of Edge Nodes
- <**Node**> Identifies the ID of the Edge Node
- <**DeviceId**> Identifies the ID of the Device from the Edge Node. This field can be empty while accessing the main Edge Node

  E.g.: Payload:GroupID;NodeID;DeviceID;
  Payload:AzureGroup;Factory1;Motor

Troubleshoot

The status of the driver execution can be observed through the diagnostic tools, which are:

• Trace window
• Property Watch
• Module Information

The above tools indicate if the operations have succeeded or have failed. A status of 0 (zero) means communication is successful. Negative values indicate internal driver errors, and positive values are protocol error codes.

Common Errors

This section details some errors you might see in your connection to the Azure IoT Hub via the MQTT protocol and discusses the possible causes of the errors.

Error: Exception connecting to the broker

If you see this message on the Trace Window Logs (with the Debug and Devices modules enabled), check the port number parameter in Devices > Nodes > PrimaryStation.

Device.Node.Status = -1

If you get this error code in Devices.Node.<NodeName>.Status, check if the URL defined in Devices > Nodes > PrimaryStation is correct.

Error Connecting to Broker. Status: 5

If you see this message on the Trace Window Logs (with the Debug and Devices modules enabled), check the following parameters in Devices > Nodes > PrimaryStation.

- Username
- ClientID

MQTT driver connection lost: System.EventArgs

If you see this message on the Trace Window Logs (with the Debug and Devices modules enabled), check the QoS setting in Devices > Nodes > PrimaryStation. As stated in a previous section, the IoT Hub will automatically disconnect your client if the QoS value is 2 (Exactly Once).

Another reason your client might be disconnected is that someone else tried to connect to your IoT Device using the same ClientID. IoT Hub only supports one active MQTT connection per device. Any new MQTT connection with the same device ID causes IoT Hub to drop the existing connection.

Append – How to Set up Azure IoT Hub

The requirements for a successful configuration are:

- Microsoft Azure IoT Hub Account
- Visual Studio Code with an Azure IoT Hub extension

Configuration Settings

Visual Studio Code

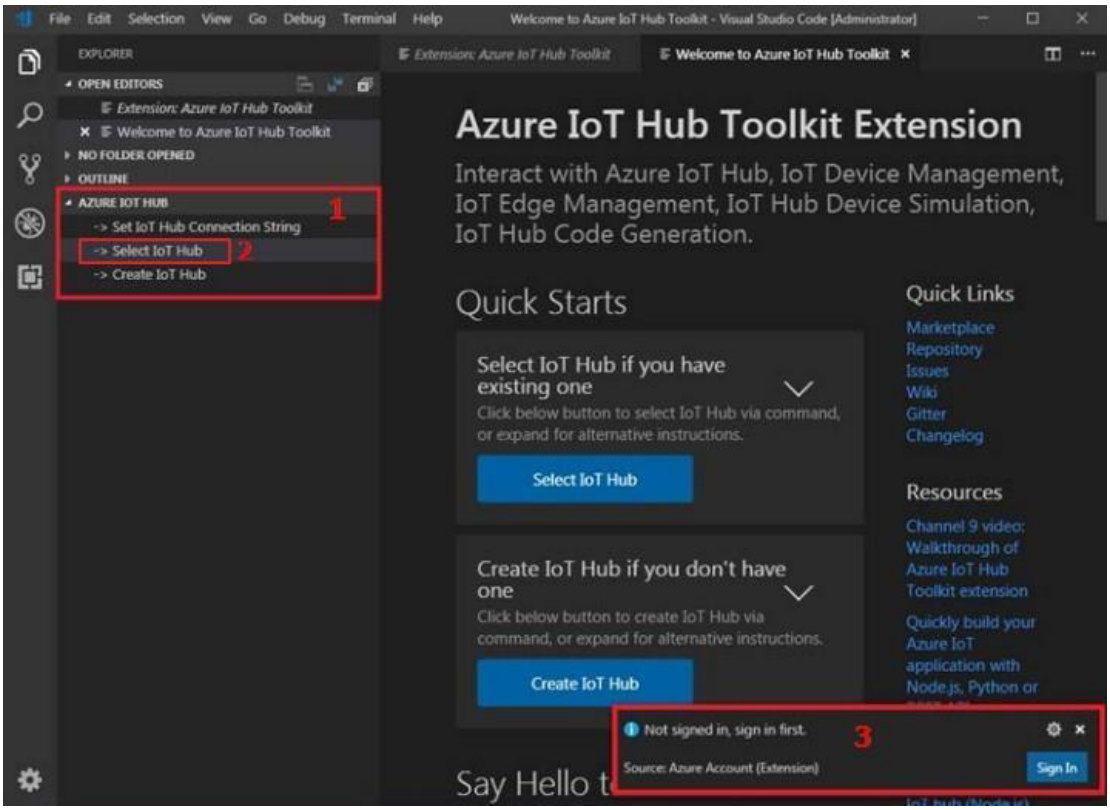
Visual Studio Code is an open-source, streamlined code editor with support for development operations like debugging, task running, and version control. It can be downloaded here.

To download the extension that allows interaction with Azure IoT Hub and IoT Device Management, click here.

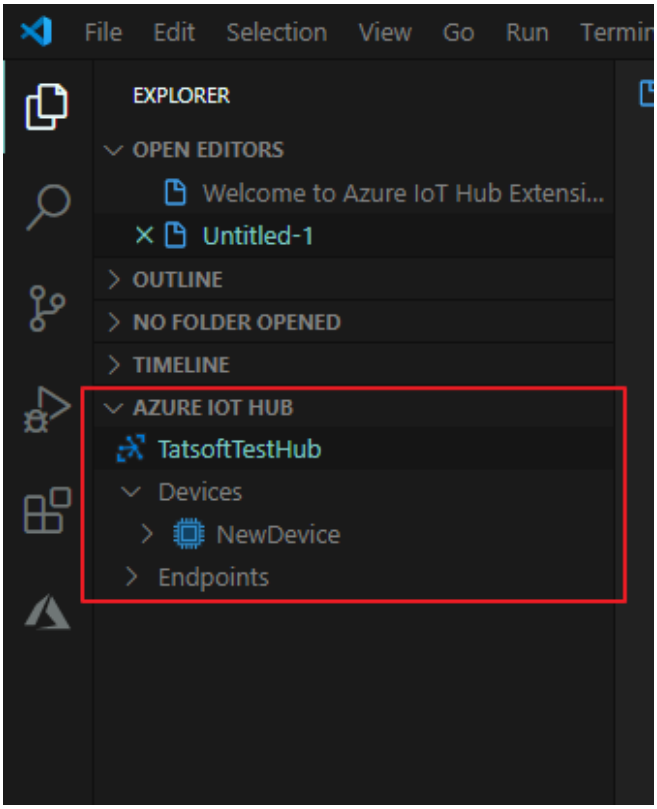
> **Note Item**
>
> This guide assumes that an IoT Hub already exists within the Azure Portal account.

After installing the extension, open the VSCode application. In the VSCode's explorer, click on the ”Azure IoTHub” tab in the bottom left corner (1), and click ”Select IoT Hub” (2) in the context menu.

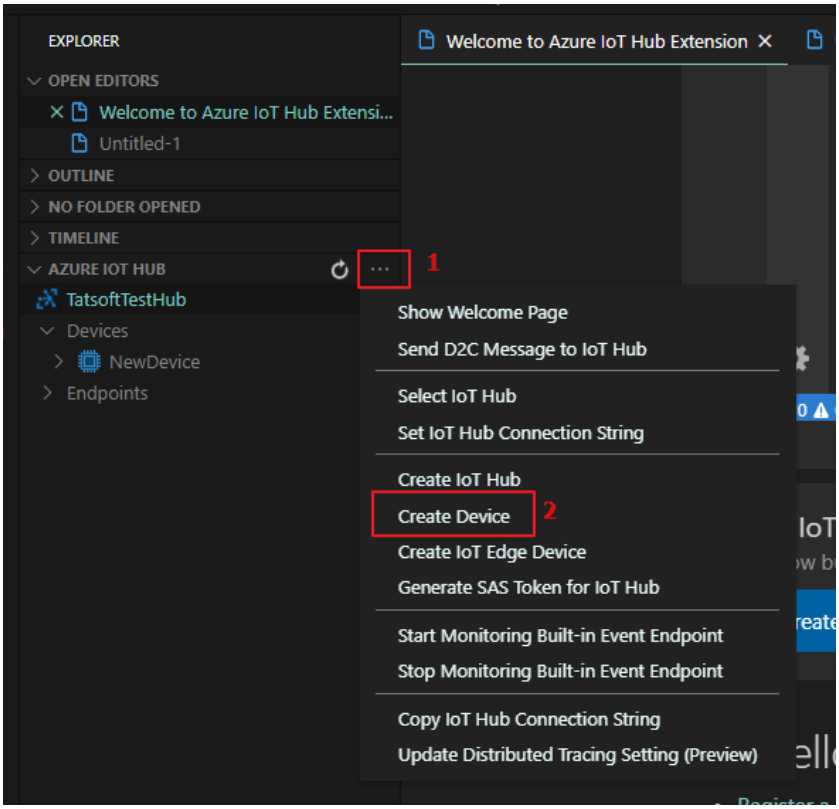
If you have not signed in to Azure, a pop-up will show in the bottom right corner to let you sign into Azure (3).



After you sign in, your Azure Subscription list will be shown. Select Azure Subscription and IoT Hub. The devices and endpoints list will be shown in the ” Azure IoT Hub” tab in a few seconds.

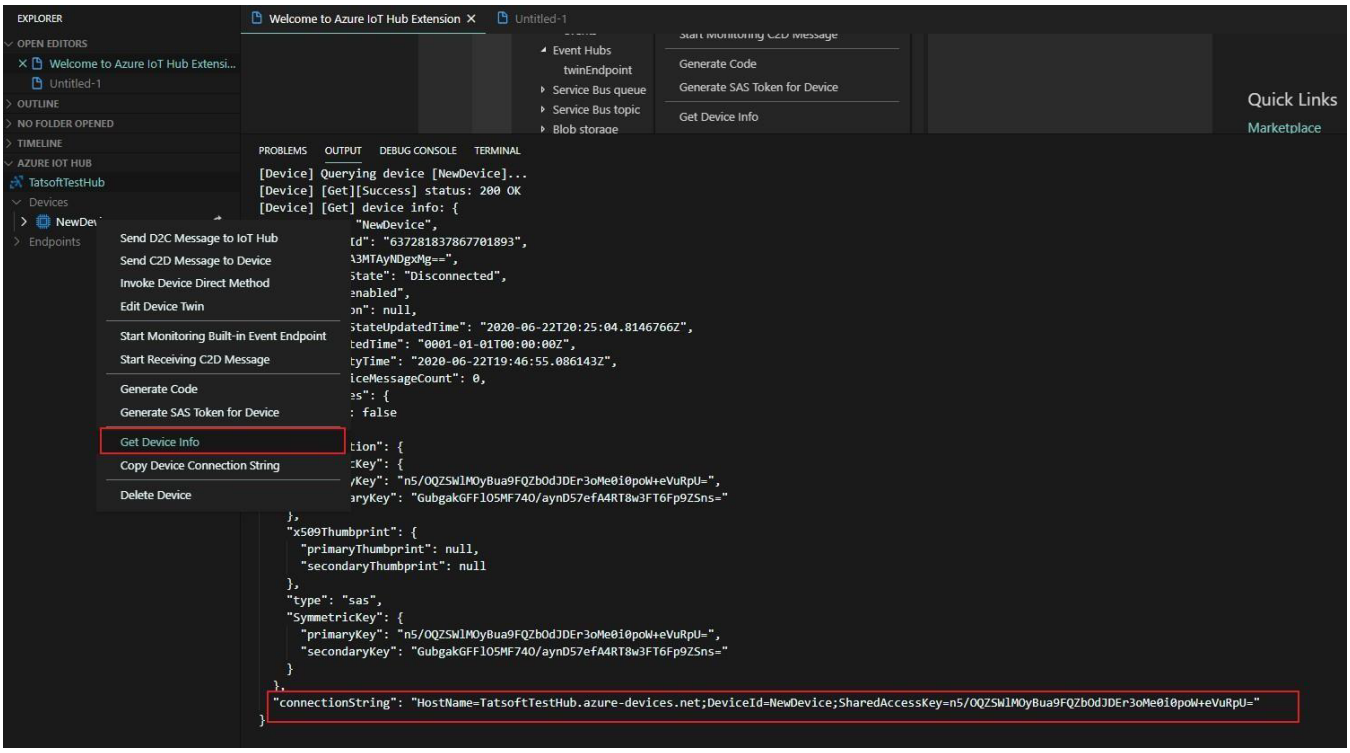


New IoT devices can be created through this extension. From the context menu (1), click Create Device (2). Enter a Device ID for the new IoT device.



Now that we have our Devices created, we will need to get our Connection String and SAS Token.

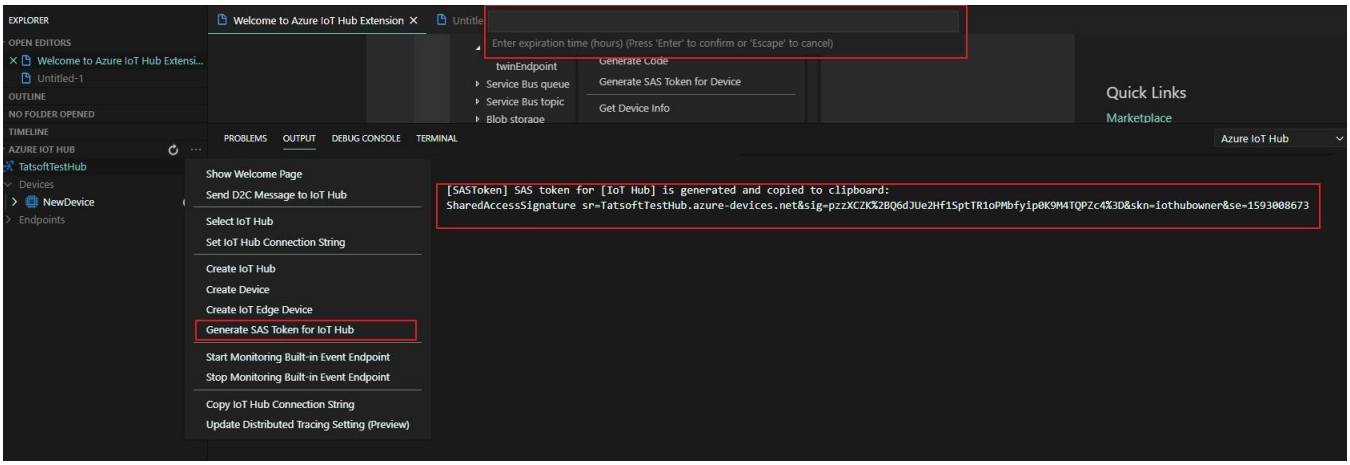
Right-Click on the Device. Select ”Get Device Info”. You should see some information displayed in the Output Window.



Review the Connection String created for the device, and record the following pieces of information from this string:

- **HostName**= TatsoftTestHub.azure-devices.net
- **DeviceId**= NewDevice

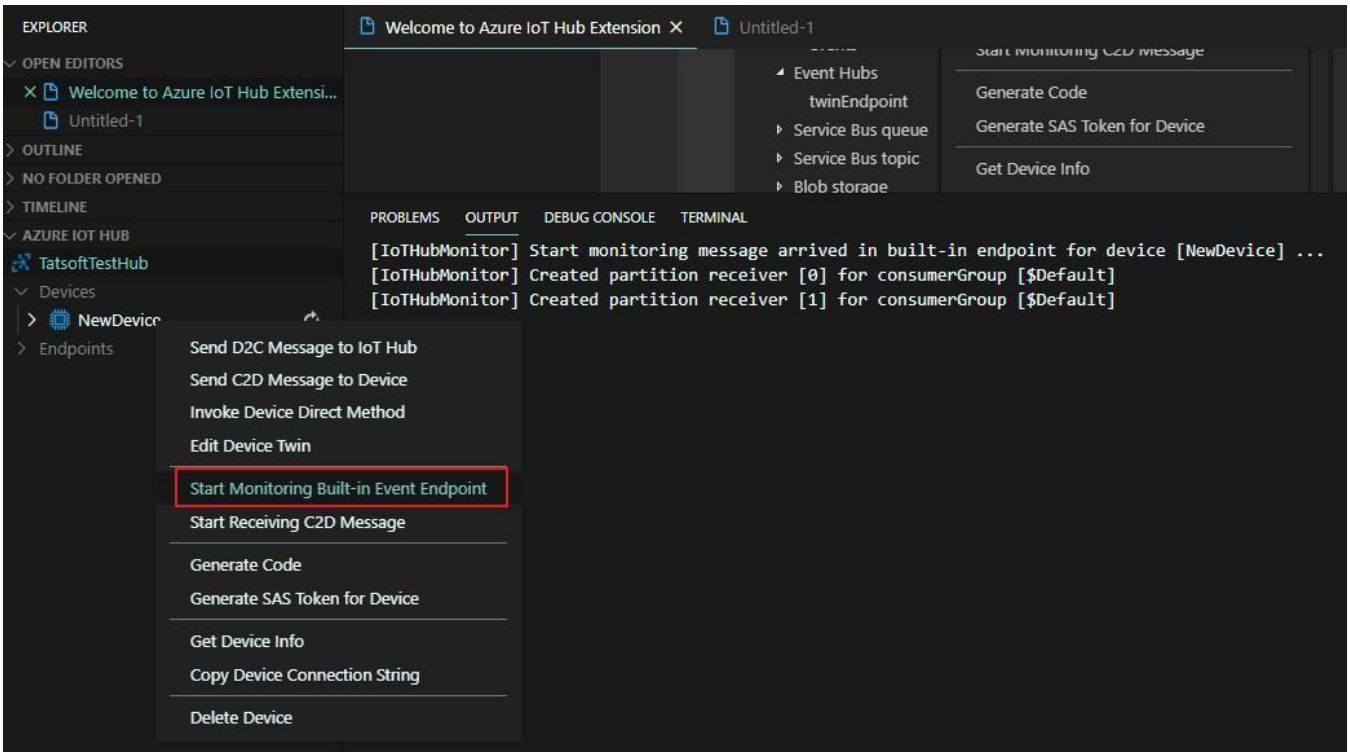
Again, right-click the device, and select ”Generate SAS token for device”. Enter the expiration time. You should see some information displayed in the Output Window.



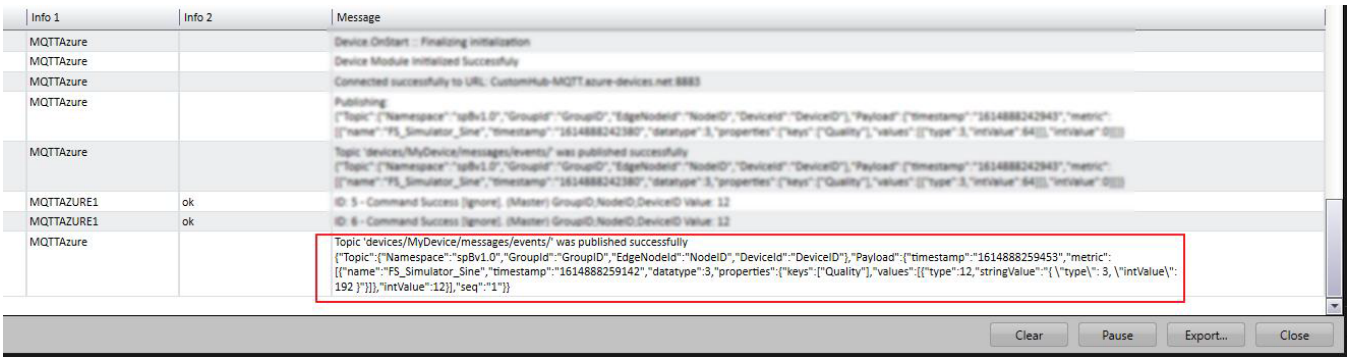
Record the following piece of information from this string:

- **Password**= SharedAccessSignature sr=TatsoftTestHub.azure-devices.net&sig=pzzXCZK%2BQ6dJUe2Hf1SptTR1oPMbfyip0K9M4TQPZc4%3D&skn=iothubowner&se=1593008673

To verify that data is flowing from the MQTT Driver to the cloud-based Azure IoT Hub, right-click the device entry, and select ”Start Monitoring Built-In Event Endpoint”.


EXPLORER
Welcome to Azure IoT Hub Extension
Untitled-1
OPEN EDITORS
Welcome to Azure IoT Hub Extensi...
Untitled-1
OUTLINE
NO FOLDER OPENED
TIMELINE
AZURE IOT HUB
TatsoftTestHub
Devices
NewDevice
Endpoints
Event Hubs
twinEndpoint
Service Bus queue
Service Bus topic
Blob storage
Generate Code
Generate SAS Token for Device
Get Device Info
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
[IoTHubMonitor] Start monitoring message arrived in built-in endpoint for device [NewDevice] ...
[IoTHubMonitor] Created partition receiver [0] for consumerGroup [$Default]
[IoTHubMonitor] Created partition receiver [1] for consumerGroup [$Default]
Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Receiving C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device


[IoTHubMonitor] [5:04:19 PM] Message received from [MyDevice]:
{
"Topic": {
"Namespace": "spBv1.0",
"GroupId": "GroupID",
"EdgeNodeId": "NodeID",
"DeviceId": "DeviceID"
},
"Payload": {
"timestamp": "1614888259453",
"metric": [
{
"name": "FS_Simulator_Sine",
"timestamp": "1614888259142",
"datatype": 3,
"properties": {
"keys": [
"Quality"
],
"values": [
{
"type": 12,
"stringValue": "{ \"type\": 3, \"intValue\": 192 }"
}
]
},
"intValue": 12
}
],
"seq": "1"
}
}



For the correct topic, see the Address section.