

# MQTT And Azure IoT

For more information on these topics, see [MQTT Protocol](#) and [Introduction to Microsoft IoT Hub](#).

[Quick video tutorial](#) (no audio)

## System Requirements

The requirements for a successful configuration of MQTT and Azure are listed below:

- MQTT Driver version 0.2.1
- Microsoft Azure IoT Hub Account
- Visual Studio Code with an Azure IoT Hub

---

## 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. Visual Studio Code can be downloaded [here](#).

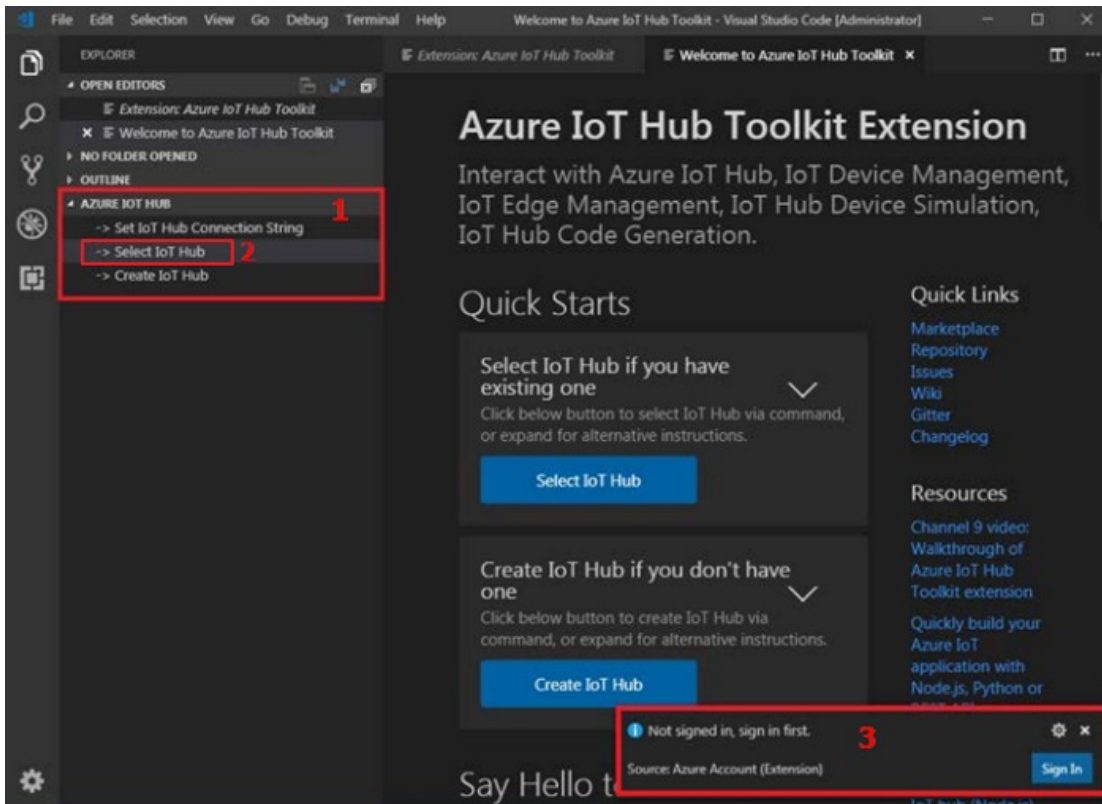
To download the extension that allows interactions to Visual Studio, Azure IoT Hub, and IoT Device Management, click [here](#).



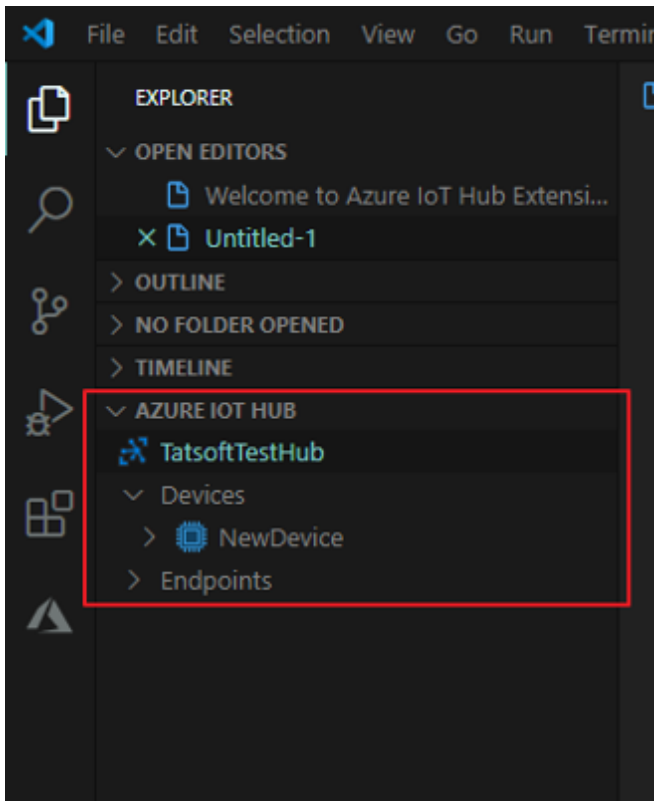
This guide assumes that an IoT Hub already exists within the Azure Portal account.

After installing the extension, open the VSCode application. In the explorer pane of the VS Code, click the "Azure IoT Hub" tab in the bottom left corner (1 below), and click "Select IoT Hub" (2) in the context menu.

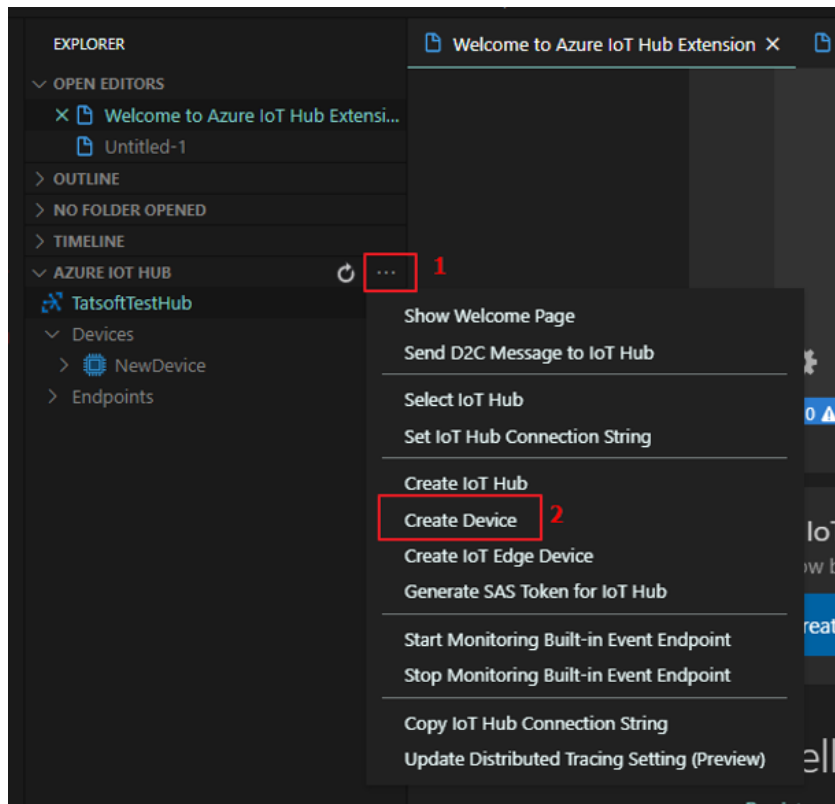
If you are not signed into Azure, a pop-up will appear in the bottom right corner to prompt you sign in to Azure (3).



Your Azure Subscription list will appear after you sign in. Select the Azure Subscription and IoT Hub. After a few seconds, devices and endpoints will appear in the Azure IoT Hub tab.

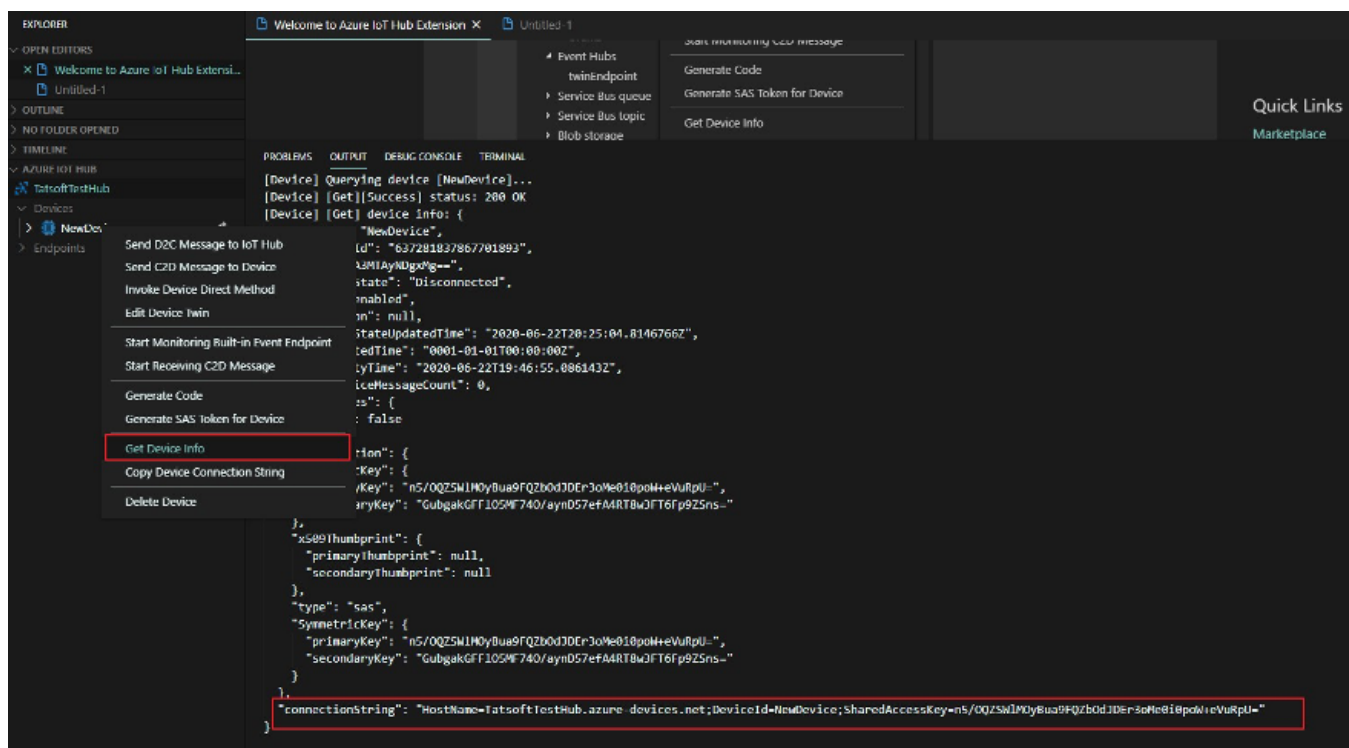


New IoT Devices can be created through the Azure IoT Hub extension. Select the context menu (1 below), click on Create Device (2), and enter a Device ID for the new IoT device.



Now that your device is created, you need to setup your Connection String and SAS Token.

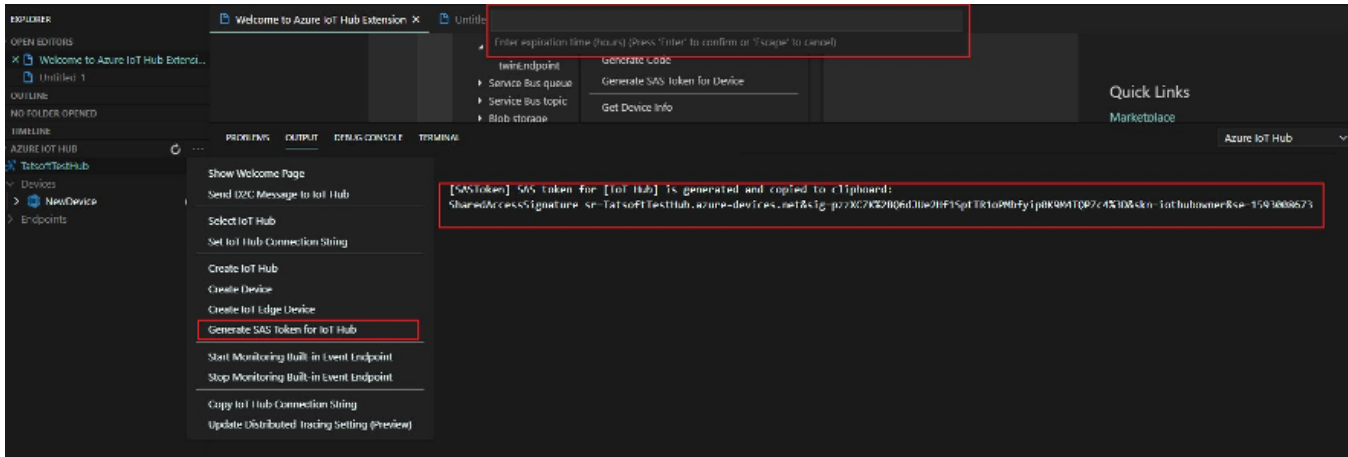
To do this, right click on your device and select Get Device info. You should see information displayed in the output window.



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

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

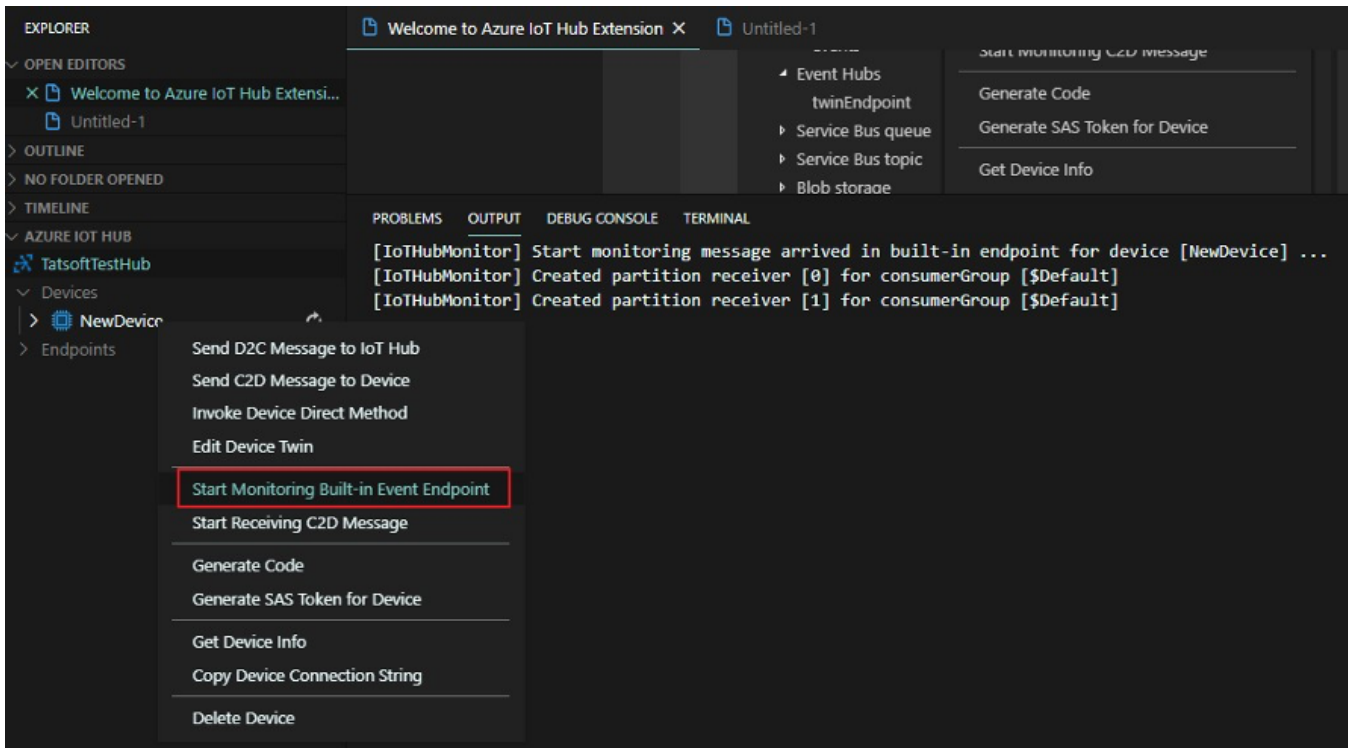
Right click on the device, select Generate SAS Token for IoT Hub, and enter the expiration time. You should see the information displayed in the output window.



Record the following pieces of information from this string:

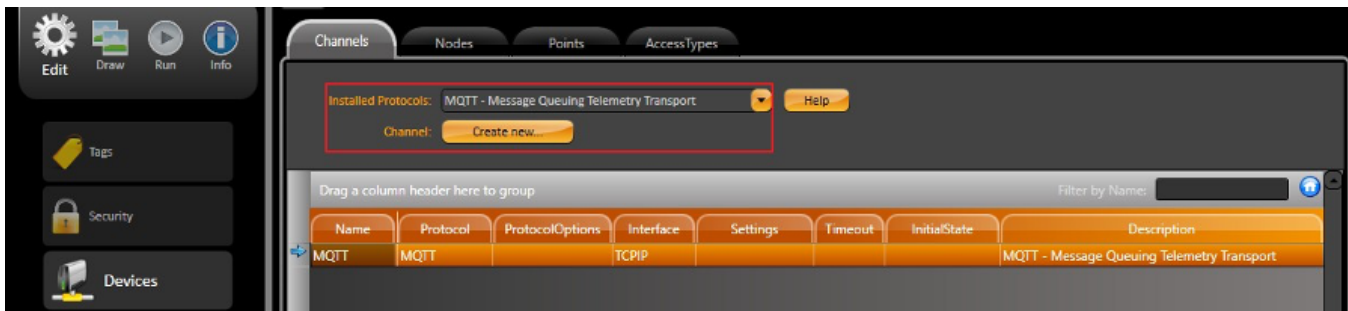
```
Password = SharedAccessSignature sr=TatsoftTestHub.azure-devices.net&sig=pzzXCZK%2BQ6dJUe2Hf1SptTR1oPMBfyip0K9M4TQPZc4%3D&skn=iothubowner&se=1593008673
```

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



## MQTT Driver

In your project's Engineering environment, navigate to **Devices > Channels** and add a new MQTT channel by selecting it from the Installed Protocols menu.



In **Devices > Nodes**, add a node for the newly created MQTT Channel. The primary station's configuration is made as follows:

```
URL = <Host Name> Port = <Port Number>
Client ID = <Device Id>
Username = <Host Name>/<Device Id> Password = <SAS Token>
TLS Version = <TLSv1.0>
X059 Certificate = <Certificate thumbprint> (optional) QoS = <Quality of Service>
Keep Alive = <Message sent to Broker to prevent the link from being broken> (in seconds)
```

For the IoT Hub and the device configured in VSCode, the parameters are as follows.

```

URL = TatsoftTestHub.azure-devices.net Port = 8883
Client ID = NewDevice
Username = TatsoftTestHub.azure-devices.net/NewDevice
Password = SharedAccessSignature sr=TatsoftTestHub.azure-devices.net&sig=pzzXCZK%2
BQ6dJUe2Hf1SptTR1oPMBfyip0K9M4TQPZc4%3D&skn=iothubowner&se=1593008673

TLS Version = <TLSv1.0> X509 Certificate =
QoS = AtMostOnce Keep Alive = 1

```

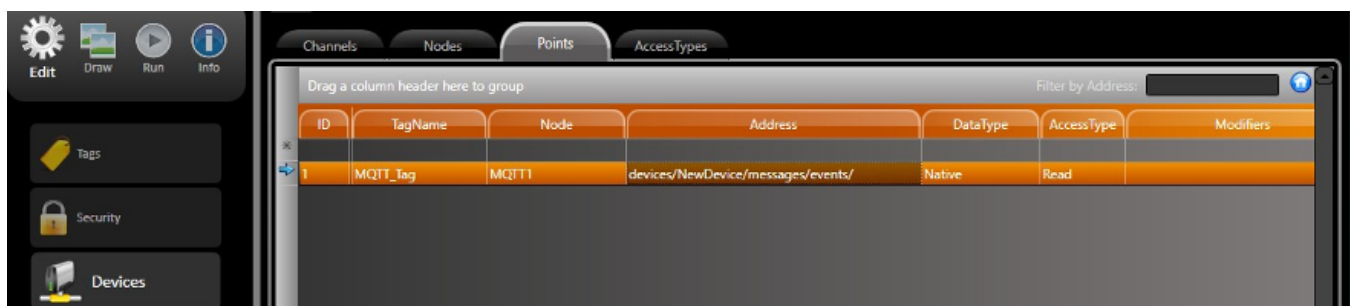


Finally, in **Devices > Points**, fill in the address information according to the syntax below:

```
devices/<Device Id>/messages/events/
```

For this example device, the correct topic is:

```
devices/NewDevice/messages/events/
```





Once the device configuration is complete, go to **Run > Startup** and execute the project. Enable the Debug and Devices options by clicking on settings and selecting the corresponding boxes. Open the TraceWindow and monitor the communication exchange.

Date Time	Type	Module Name	Info 1	Info 2	Message
23/06/2020 13:10:25.267	Info	Device	MQTT		Initializing Device Module
23/06/2020 13:10:25.268	Debug	Device	MQTT		Device.OnStart :: Reading channel configuration
23/06/2020 13:10:25.356	Debug	Device	MQTT		Device.OnStart :: Creating tag points list
23/06/2020 13:10:25.377	Debug	Device	MQTT		Device.OnStart :: Loading driver assembly
23/06/2020 13:10:25.380	Info	Device	MQTT		Protocol: MQTT loaded. (Version : 3.0.2.1)
23/06/2020 13:10:25.380	Debug	Device	MQTT		Device.OnStart :: Loading driver assembly
23/06/2020 13:10:25.384	Debug	Device	MQTT		Device.OnStart :: Creating list of COMAPIs
23/06/2020 13:10:25.403	Debug	Device	MQTT		Device.OnStart :: Creating group configuration
23/06/2020 13:10:25.433	Debug	Device	MQTT		Device.OnStart :: Creating events and fill groups
23/06/2020 13:10:25.449	Debug	Device	MQTT		Device.OnStart :: Creating threading pool
23/06/2020 13:10:25.450	Debug	Device	MQTT		Device.OnStart :: Finalizing initialization
23/06/2020 13:10:25.450	Info	Device	MQTT		Device Module Initialized Successfully
23/06/2020 13:10:26.415	Debug	Device	MQTT		Connecting to Broker OK.
23/06/2020 13:10:26.416	Debug	Device	MQTT		Initialization ok
23/06/2020 13:10:26.416	Debug	Device	MQTT		Treating Topic: devices/NewDevice/messages/events/
23/06/2020 13:10:26.429	Debug	Device	MQTT		Subscribing Topics
23/06/2020 13:10:26.435	Info	Device	MQTT1	ok	ID: 1 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:27.457	Info	Device	MQTT1	ok	ID: 2 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:29.473	Info	Device	MQTT1	ok	ID: 3 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:31.474	Info	Device	MQTT1	ok	ID: 4 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:33.473	Info	Device	MQTT1	ok	ID: 5 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:35.149	Debug	Device	MQTT		Sending Message: Topic: devices/NewDevice/messages/events/ Value: Message 1 Timestamp: 23/06/2020 13:10:35 -03:00
23/06/2020 13:10:35.149	Info	Device	MQTT1	ok	ID: 6 - Command Success. (Master) devices/NewDevice/messages/events/ Value: Message 1
23/06/2020 13:10:35.495	Info	Device	MQTT1	ok	ID: 7 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:37.479	Info	Device	MQTT1	ok	ID: 8 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:39.483	Info	Device	MQTT1	ok	ID: 9 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:41.507	Info	Device	MQTT1	ok	ID: 10 - Command Success. (Master) devices/NewDevice/messages/events/
23/06/2020 13:10:42.302	Debug	Device	MQTT		Sending Message: Topic: devices/NewDevice/messages/events/ Value: Message Test Timestamp: 23/06/2020 13:10:42 -03:00
23/06/2020 13:10:42.302	Info	Device	MQTT1	ok	ID: 11 - Command Success. (Master) devices/NewDevice/messages/events/ Value: Message Test
23/06/2020 13:10:43.505	Info	Device	MQTT1	ok	ID: 12 - Command Success. (Master) devices/NewDevice/messages/events/

Back on VSCode, you should see the received messages in the output window.

EXPLORER

- OPEN EDITORS
  - Welcome to Azure IoT Hub Extension...
  - Untitled-1
- OUTLINE
- NO FOLDER OPENED
- TIMELINE
- AZURE IOT HUB
  - TatsoftTestHub
    - Devices
      - NewDevice
    - Endpoints

OUTPUT

```
[IoTHubMonitor] Stopping built-in event endpoint monitoring...
[IoTHubMonitor] Built-in event endpoint monitoring stopped.
[IoTHubMonitor] Start monitoring message arrived in built-in endpoint for all devices ...
[IoTHubMonitor] Created partition receiver [0] for consumerGroup [$Default]
[IoTHubMonitor] Created partition receiver [1] for consumerGroup [$Default]
[IoTHubMonitor] [1:10:35 PM] Message received from [NewDevice]:
"Message 1"
[IoTHubMonitor] [1:10:42 PM] Message received from [NewDevice]:
"Message Test"
```