Copyright (c) Microsoft Corporation. All rights reserved.

Licensed under the MIT license.

Microsoft msMqtt Niagara 4 Module for IoT Hub Publishing

# Abstract

Tridium released a driver called AbstractMqttDriver which is capable of publishing data using the MQTT protocol in December of 2017.

Microsoft developed the msMqtt Niagara module and its component(s) tool to streamline the publishing of BACnet data points to IoT Hub by automating the programming necessary to publish data through the new Tridium AbstractMqttDriver with minimal effort.

# Requirements

#### Software:

* Niagara Workbench (Has to be a full install, web workbench will not work)
* IoT Hub DeviceExplorer  
  download from <https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer>

#### Niagara Modules:

* AbstractMqttDriver-rt   
  (this may need to be downloaded and installed from Niagara-Central or your through your vendor support chain.)
* driver-rt
* alarm-rt
* bql-rt
* control-rt
* kitControl-rt
* bacnet-rt
* ndriver-rt
* converters-rt

#### Azure IoT Hub Subscription

* IoT Hub  
  <http://azure.microsoft.com> | <http://portal.azure.com>

#### Network

* Open connectivity outbound on port 8883

# About the Tridium AbstractMqttDriver

Tridium released the first version of the AbstractMqttDriver for Niagara 4.2. Your Niagara version must be at least 4.2 to utilize the AbstractMqttDriver and the msMqtt module. Earlier versions such as including Niagara AX are not compatible with the AbstractMqttDriver.

# msMqtt Module Licensing

The msMqtt module does not cost anything or require any special Niagara licensing files to be installed.

Note: The msMqtt module enhances workflow to funnel data into the Tridium AbstractMqttDriver which does use utilize Global Capacity Licensing which does affect your device and point counts.

# AbstractMqttDriver Licensing

The Tridium developed driver utilizes Niagara Global Capacity Licensing.

Since the msMqtt BacnetToMqttUtil adds devices and points to the Tridium AbstractMqttDriver the resulting impact to licensing consumes one Device by adding a AbstractMqttDriver MqttDevice and one proxy point for each BACnet point published to IoT Hub.

See the supporting documentation form the AbstractMqttDriver and Global Capacity Licensing for more information about how Tridium has implemented licensing for the AbstractMqttDriver.

# Obtain IoT Hub Sas Security Token

Follow the instructions in the following link to obtain a SaS security token for use in steps further into this document.

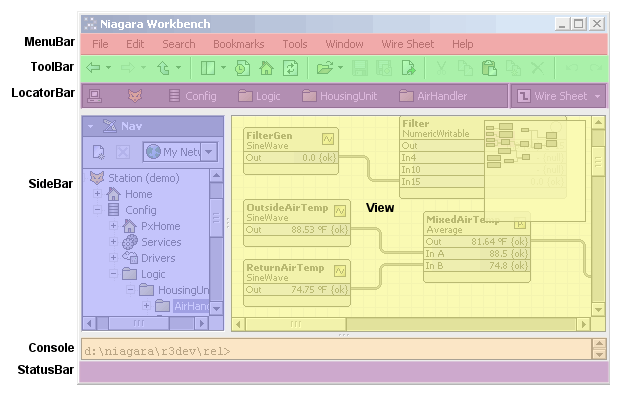
<https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer>

# Obtain and Install the msMqtt Module for Workbench

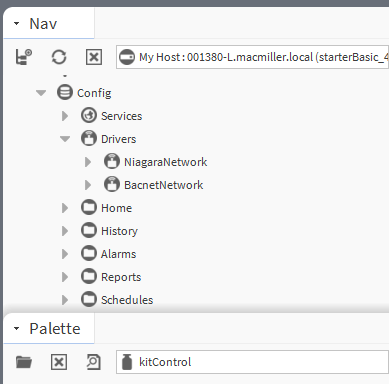
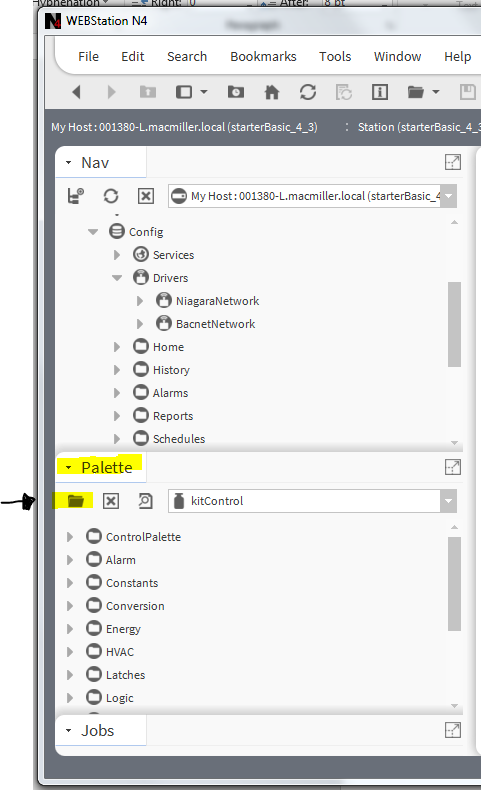
1. **Download** a copy of the ***msMqtt-rt.jar*** from **Microsoft** **(location to download from???)**
2. **Download** a copy of the ***abstractMqttDriver-rt***, and ***abstractMqttDriver-wb*** module files from ***Niagara Central***, <http://www.niagara-central.com>, or obtain them through your ***Niagara solutions provider***.
3. **Copy** the **modules** you obtained to the ***modules*** folder in the ***install directory*** of your Niagara instance. This is usually similar to ***C:\Niagara\Workbench-4.2.36.34\modules***.
4. If Niagara is running, **close and reopen the workbench**.
5. If you have a local station running restart it if you intend to use the module on your locally.

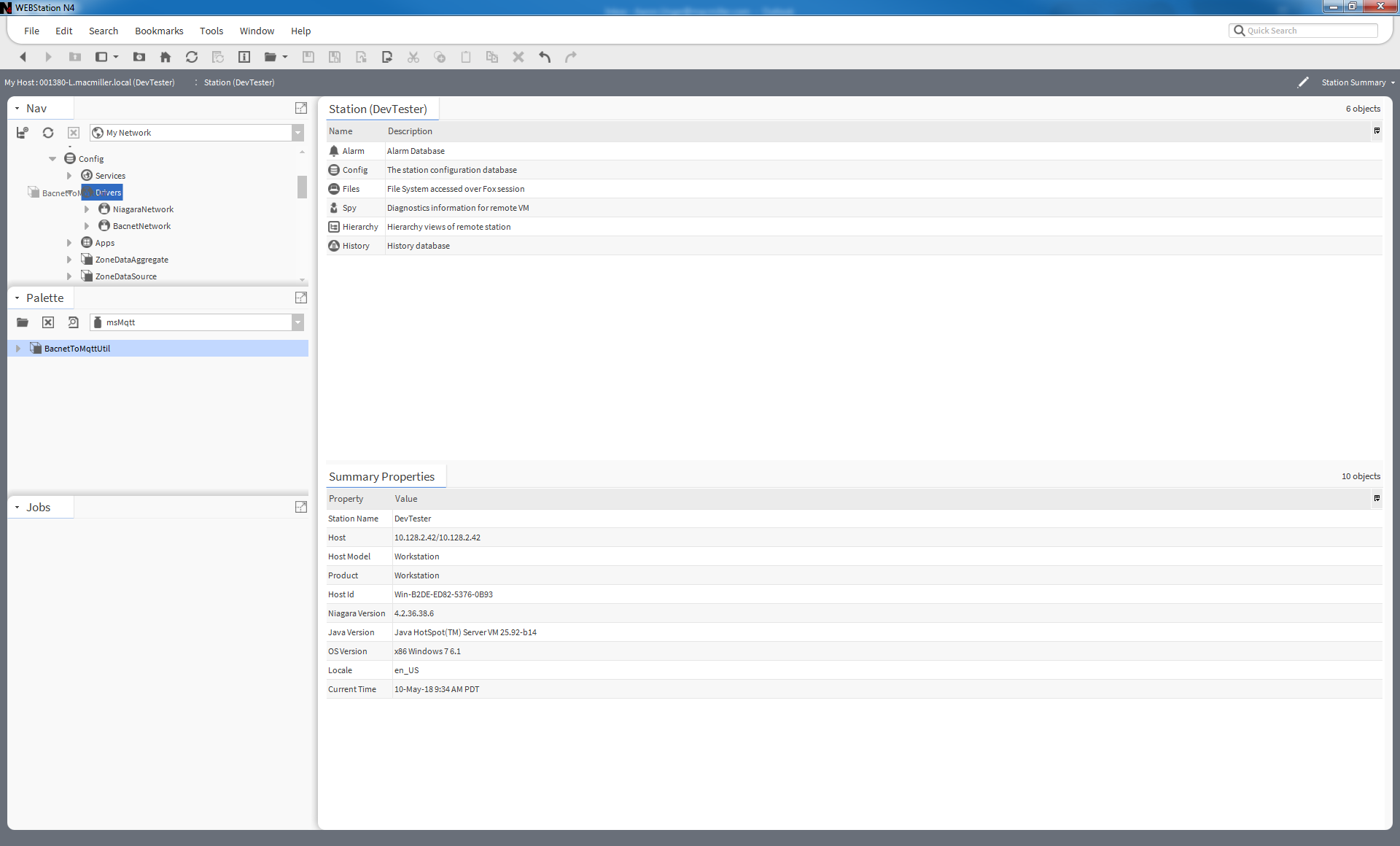
# Deploy the msMqtt module onto your JACE

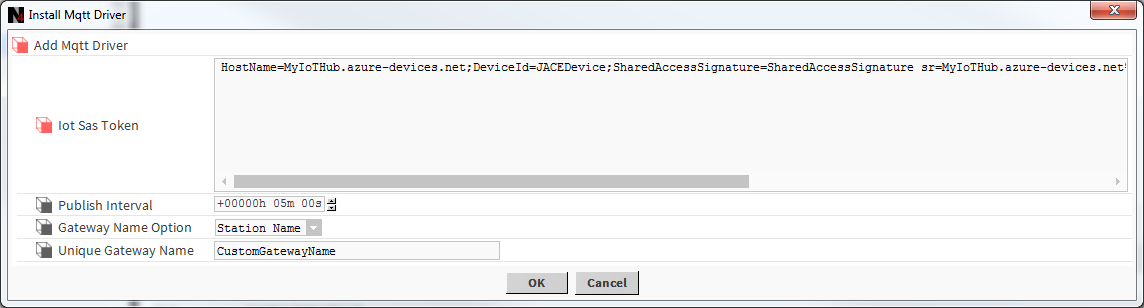
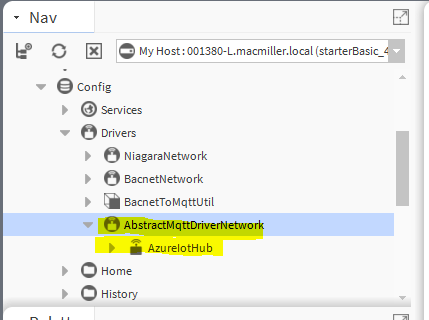
1. **Open** ***Niagara Workbench*** on your computer.
2. **Connect** to the ***Platform*** of the JACE
   1. From the ***File menu*** select ***Open*** -> ***Open Platform***
   2. Enter the ***host name or IP*** of your JACE in the in the **Host** field.
   3. Click Ok.
3. Use the **Software Manager** to Install the msMqtt module.
   1. Double click ***Software Manager*** in the right View pane and wait for the module list to populate the list of installed and available modules.
   2. **Right click** ***each of the modules*** listed below and choose ***Install***.
      1. msMqtt-rt
      2. AbstractMqttDriver-rt
      3. AbstractMqttDriver-wb
      4. driver-rt
      5. alarm-rt
      6. bql-rt
      7. control-rt
      8. kitControl-rt
      9. bacnet-rt
      10. ndriver-rt
      11. converters-rt
   3. After selecting install on all of the required modules browse to the bottom of the Software Manger and **press** the ***Commit*** ***button*** at the bottom of the Software Manager. You may be asked if you want to restart the JACE. Choose ***Yes***.

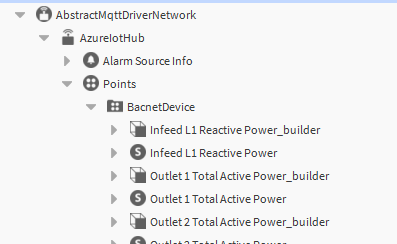


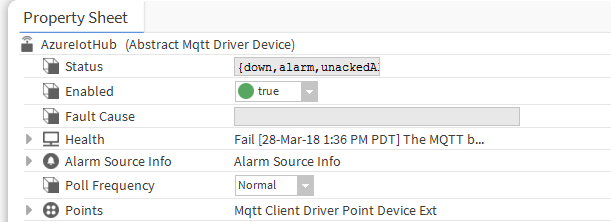
# Install, Configure and Publish Points to IoT Hub using BacnetToMqttUtil

1. **Open** your ***Station*** in the **Niagara Workbench**.
2. If you haven’t already opened your Niagara Station, from the ***File menu*** select ***Open*** -> ***Open Station***
3. Enter the ***host name or IP*** of your JACE in the in the **Host** field.
4. Click Ok.
5. **Expand** the ***Config*** and ***Drivers*** nodes **Navigation** **Tree** so you can see all the installed drivers in your station.  
   
6. **Open** the ***msMqtt*** module in the **Palette**.
   1. If the Palette is not already open, in the **Menu Bar**, go to **Window->Sidebars** and select ***Pallette***.
   2. **Click** the ***Folder Icon*** in the upper left of the **Palette** and **type** in ***msMqtt*** filter text box.
   3. **Select** the msMqtt module from the filtered list and click OK.  
        
      
7. **Drag** the ***BacnetToMqttUtil*** component from the **Palette** to **Drivers** in the **Nav Tree**.



1. **Right click** on the **BacnetToMqttUtil** in under **Drivers**, **hover** over the **Actions** menu and select ***Install Mqtt Driver***.  
   
   1. **Paste** your ***Sas Token*** **into** the **Iot Sas Token** field. This was obtained by steps earlier in this document.
   2. The Gateway Name Option defines how your JACE is going to be identified in the IoT Hub.
      1. The default option uses the name of the Niagara station as the identifier.
         1. You can alternatively choose to use the JACE Host ID instead. Keep in mind that HostIds an change sometimes when hardware is replaced.
         2. The final option is to type in your own name.
            1. **Select** ***Custom*** from the **Gateway Name Option** then **enter** your preferred identifier in the **Unique Gateway Name** field.
   3. **Click** the ***OK*** to install AbstractMqttDriver and the AzureIotHub device.
      1. If you have previously run this wizard, it will re-configure your network and device with your new settings you entered when running the Install MqttDriver action.  
           
           
         
2. Publish points to from the BacnetNetwork to the AzureIotHub Mqtt Device
   1. Right Click on the BacnetToMqttUtil, hover over the Actions menu, and select Export to Mqtt.
      1. This action submits a Niagara job that iterates through all BACnet devices in your BacnetNetworks publishes their points in the AzureIotHub Mqtt Device.
      2. Each BACnet device will have a folder in the AzureIotHub device points folder and each folder will contain the points from the device with the same name.

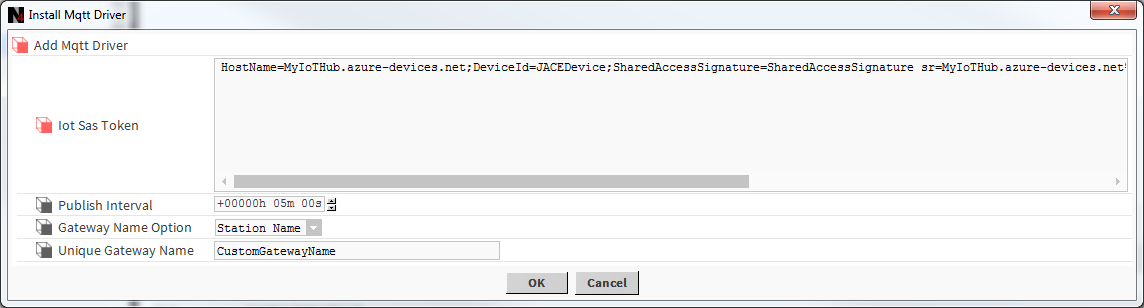


1. Double click on the AzureIotHub device and check the status. If the status is Down just after running the previous steps, sometimes the station will need to be rebooted.  
     
   
2. Restart the station then re-check the status. It should no longer be down. If restarting doesn’t resolve the device being offline there is probably a connectivity problem or an issue an issue with the SaS Token. Check that your network allows port 8883

# Behind the BacnetToMqttUtil Component Actions

#### The ‘Install Mqtt Driver’ Action

The Install Mqtt Driver action on the BacnetToMqttUtil component displays up the following popup wizard.



The fields in this wizard are used to configure the AbstractMqttDriver and the AzureIotHub device that are a result of running this wizard.

This configuration cascades into the other actions in the BacnetToMqttUtil and are used to configure additional components that are deployed to publish data to the IoT Hub.

When you click OK on the Install Mqtt Driver wizard:

* The AbstractMqttDriverNetwork is added to your station if it has not already been added.
* A device named AzureIotHub is added if to the AbstractMqttDriverNetwork if it does not already exist.
* Both the AbstractMqttDriver and AzureIotHub device are configured with the values you entered in the wizard form fields.

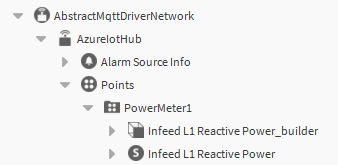
The following table expands on what happens with the data you enter into the wizard.

|  |  |  |
| --- | --- | --- |
| **Form Field** | **Affected Components** | **Explanation** |
| Iot Sas Token | **AzureIotHub** (device new or existing) | The connection string is parsed into its individual parts and configures your station in the following way:  AzureIotHub (device)   * BrokerIpAddress={HostName} * ClientId={DeviceId} * UsernameAndPassword={HostName}/{DeviceId} * Password={SharedAccessSignature} |
| Publish Interval | **AbstractMqttDriverNetwork** (driver new or existing) | The publish interval is written to the Default Tuning policy of the AbstractMqttNetwork for the Minimum and Maximum Write Time(s) |
| Gateway Name Option / Unique Gateway Name | **BacnetToMqttUtil**  and  **BacnetMessageBuilder**(s) | The Gateway name is included in the message sent the configured Iot Hub Device. It is the unique ID of your datasource. The default is the JACE but the Host ID or a unique name are also optional.  The gateway name is used by point published in the Point Folder beneath the configured AzureIotHub device.  Each source point has a two components in the AzureIotHub point folder. One to form the point message and the other a configured point to send the message to Azure Iot Hub.  The Mqtt ProxyPoint is added with the same name as the source point.  BAcnetMessageBuilder with the same name but extended with “\_builder” are used to send each single points data. |

#### The “Export to Mqtt” Action

The Export to Mqtt action finds all of the Bacnet proxy points in a Niagara station and creates a object hierarchy in the MQTT driver under the AzureIotHub device for publishing BACnet points to the Azure Iot Hub.

Under the AzureIotHub device, a folder is created with the same name as the source device for each BacnetDevice found in the station. Each BACnet point from the source device is published with two program objects. One Builder Object and StringWritable that has a MqttPublishProxyExt.



#### The “Change Gateway Name” Action

This is a quick option to change the source gateway name of MQTT messages if the name does not make sense or conflicts with another gateway that is sending data to the IoT Hub.