Technical Document

# Abstract MQTT Driver Guide

**October 16, 2017**

niagara4

# Abstract MQTT Driver Guide

**Tridium, Inc.**
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

## Confidentiality



## Trademark notice



## Copyright and patent notice

# Contents

# Preface

## About this Guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

### Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. In order to make the most of the information in this book, readers should have some training or previous experience with Niagara 4 or NiagaraAX software, as well as experience working with JACE network controllers.

### Document Content

This document describes how to create an Abstract MQTT (MQ Telemetry Transport) client connection to the Abstract MQTT broker for a Niagara 4.2 (or later) station for the purpose of publishing or subscribing the "lightweight" messaging protocol into the Niagara Framework.

## Document change log

Changes to this document are listed in this topic.

- Updated: October 16, 2017

  Added information about 'Bql Query Builder' in "Discovering AbstractMqttPoints" topic.

  Added two new topics "Discovering AbstractMqttPoint" and "Adding Points to the database".

  Updated the component topics and few more topics in the chapter "Setup an AbstractMqttDriverNetwork".

- Updated: July 10, 2017

  Updated for initial product release.

## Related Documentation

The following documents relate to the AbstractMqtt Driver.

- Niagara Drivers Guide
- Niagara 4 Platform Guide

# Chapter 1 Introduction

**Topics covered in this chapter**

♦ MQTT Requirements

MQTT is a publish-subscribe-based "lightweight" messaging protocol for use on top of the TCP/IP protocol driver installation.

### Description

MQTT (MQ Telemetry Transport) driver installation is an extremely simple and lightweight messaging protocol. Its publish/subscribe architecture is designed to be open and easy to implement, with up to thousands of remote clients capable of being supported by a single server.

### Publish/Subscribe

The MQTT protocol is based on the principle of publish and subscribe the patterns. Multiple clients connect to a broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. It restricts one subscriber to one topic.

### Client

In this documentation, the term the client refers to the MQTT client. That is, either a Publisher or a Subscriber. The MQTT client can be both a publisher and subscriber at the same time. An MQTT client is any device from a micro controller up to a full fledged server, that has an MQTT library running and is connecting to an MQTT broker over any type of network. The Niagara solution for MQTT uses only the client, and not the broker. Datatypes supported by the client are Boolean, String, Numeric and Enum.

### Broker

You may install an applicable broker. Depending on the actual implementation, a broker may handle many concurrently connected MQTT clients. The broker is primarily responsible for :

- Receiving all messages
- Filtering them
- Deciding who is interested in it
- Sending the message to respective subscribed clients.

Another responsibility of the broker is the authentication and authorization of clients. Most of the times a broker is also extensible, which allows you to easily integrate custom authentication, authorization and integration into back-end systems. Integration is an especially important aspect because often the broker is the component which is directly exposed on the Internet and handles a lot of clients and then passes messages along to downstream analyzing and processing systems. The Niagara MQTT client supports One way SSL connection. Whenever a client tries to connect with a broker using a login credentials, it is done over SSL.

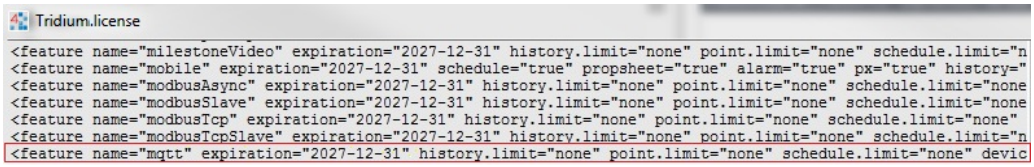
## MQTT Requirements

This topic describes the licensing and software requirements for using the Niagara Abstract MQTT Driver.

### License Requirements

The abstract Mqtt driver module is a licensed feature. You can check to see if your software installation is licensed for Mqtt by opening the license file from the License Manager view. The feature name is present only if your platform is licensed for Mqtt.

**Figure 1** Example Mqtt license with point limit attribute values set to "none".




## Software Requirements

The Niagara Abstract MQTT Driver is available for Niagara 4.2 (or later).

# Chapter 2 Setup an AbstractMqttDriverNetwork

**Topics covered in this chapter**

- Adding an AbstractMqttDriverNetwork
- Adding an AbstractMqttDriverDevice
- AbstractMqttDriverDevice configuration
- About AbstractMqttPoints

This section provides the procedures and descriptions that are commonly required and used with the **abstractMqttDriver** in typical online scenarios.

To configure the AbstractMqttDriverNetwork, first add and open the **abstractMqttDriver** palette and then add the AbstractMqttDriverNetwork under your station Drivers container.

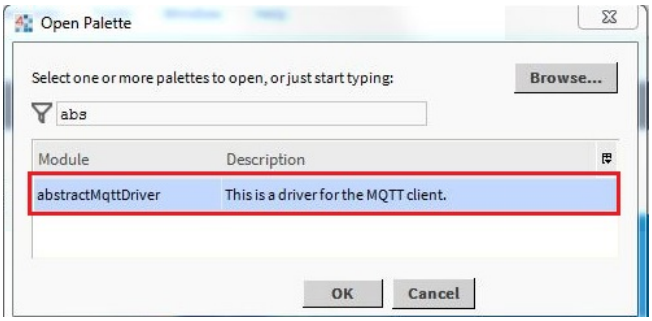
## Adding an AbstractMqttDriverNetwork

Use the following procedure to add an AbstractMqttDriverNetwork component under a station **Drivers** container.

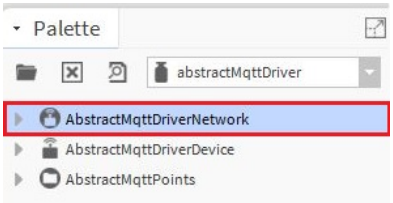
Step 1 Open the **Palette** side bar by selecting **Window→Side Bars→Palette** from the **Menu** bar.

Step 2 Click on the **Open Palette** icon from the **Palette** side bar .

Step 3 Using the **Palette** side bar controls, open the **abstractMqttDriver** palette.




Step 4 Select the AbstractMqttDriverNetwork and add it from the **abstractMqttDriver** palette and copy-and-paste (or drag and drop) the AbstractMqttDriverNetwork object into the **Drivers** container. The **Name** dialog box appears.




Step 5 Name the AbstractMqttDriverNetwork, as desired and click the **OK** button.

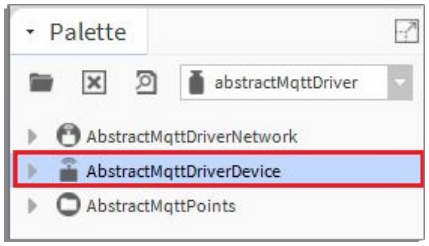
## Adding an AbstractMqttDriverDevice

Use the following procedure to add an **AbstractMqttDriverDevice** component under an AbstractMqttDriverNetwork.

Step 1 Double-click the AbstractMqttDriverNetwork. The **Mqtt Client Driver Device Manager** view appears in the view pane.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Select the **AbstractMqttDriverDevice** from the **abstractMqttDriver** palette and copy-and-paste (or drag and drop) the **AbstractMqttDriverDevice** under the AbstractMqttDriverNetwork or the **Mqtt Client Driver Device Manager** view. The **Name** dialog box appears.



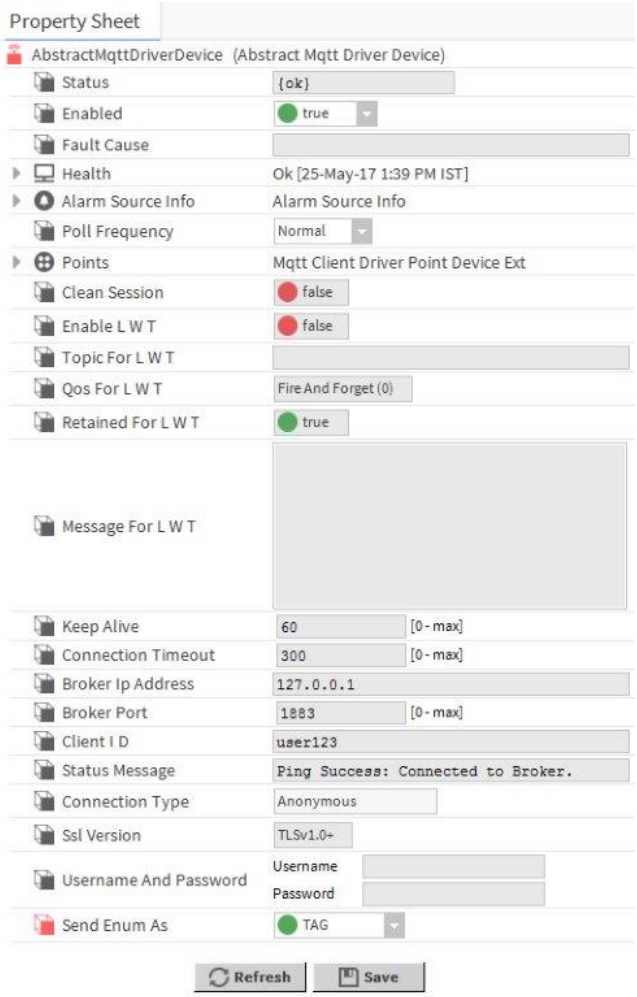

Step 4 Name the **AbstractMqttDriverDevice**, as desired and click the **OK** button.

## AbstractMqttDriverDevice configuration

You place an **AbstractMqttDriverDevice** objects under the AbstractMqttDriverNetwork to represent actual Abstract Mqtt Driver Devices that you want to communicate with the broker. You can configure the **AbstractMqttDriverDevice** object by setting its properties to match the settings for the broker.

By selecting the **Property Sheet** view of the **AbstractMqttDriverDevice** object, you can view and configure the following **AbstractMqttDriverDevice** properties that apply to the entire network. In one network, you can add multiple devices which are connected to the different brokers.

**NOTE:** If the **AbstractMqttDriverDevice** is connected to the broker and there is network issue like network outage, network connectivity problem or broker issue then the device will be disconnected, however it will keep on retrying until the device is connected. In this case, if the device has to connect again using different parameter, user has to disconnect (by selecting the disconnect action) enter the new connection parameters then connect. This will make new connection with the broker.




A description of the **AbstractMqttDriverDevice** specific properties follows (displayed in the **Property Sheet** view).

## Actions

The **AbstractMqttDriverDevice** object has three visible actions that are available when you right-click on the device property or device node in the **Nav** tree:

- Ping

  This action manually initiates a "ping" (check device status) on the actual **AbstractMqttDriverDevice**.

- Connect

  This action establishes a connection with the broker.

- Disconnect

  This action waits for the Abstract MQTT client to finish any work and for or the TCP/IP session to disconnect.

**NOTE:** For the Connect and Disconnect actions to work, the Ip address, Client ID, and port number of the broker must be entered. Also the Username and Password must be entered, if the **`Connection Type`** property is set to `User Login Over SSL`.

### Clean Session

A persistent session can be requested by the client on connection establishment with the broker. The client can control whether or not the broker stores the session by using the Clean Session flag. If the clean session is set to `true` then the client does not have a persistent session and all information is lost when the client disconnects for any reason. When clean session is set to `false`, a persistent session is created and it will be preserved until the client requests a clean session again. If there is already a session available then it is used and queued messages will be delivered to the client if available.

### Enable L W T

When a client connects to the server this property can define a topic and a message that needs to be published automatically when an ungraceful disconnection occurs.

### Topic For L W T

This is a topic where a message for Last will and testament (L W T) will be available.

### Qos For L W T

MQTT defines three levels of Quality of Service (Qos). The QoS defines how hard the broker/client will try to ensure that a message is received. Messages may be sent at any Qos level, and clients may attempt to subscribe to topics at any Qos level. The three levels are:

- 0: The broker/client will deliver the message once, with no confirmation
- 1: The broker/client will deliver the message at least once, with confirmation required
- 2: The broker/client will deliver the message exactly once by using a four step handshake.

### Retained For L W T

Set this property to `true` to determine if the message will be saved by the broker for the specified topic as last known good value. This is useful for newly connected subscribers to receive the last retained message on that topic immediately after subscribing. This is extremely helpful for status updates of components or devices on individual topics.

### Message For L W T

This property shows the message for L W T when an unexpected disconnect occurs.

### Keep Alive

The value of this property specifies an interval for the longest possible period of time that a broker - client connection can exist without sending a message. The broker must disconnect a client, which doesn't send a PINGREQ or any other message in user's specified time of the Keep Alive. This provides Abstract MQTT with a work-around to the issue of half-open connection. The keep alive functionality assures that the connection is still open and both broker and client are connected to one another. The client specifies a time interval in seconds and communicates it to the broker during the establishment of the connection. The interval is the longest possible period of time, which the broker and client can endure without sending a message.

- The client should ensure that the interval between Control Packets being sent does not exceed the Keep Alive value.
- In the absence of sending any other Control Packets, the client must send a PINGREQ Packet.
- As long as messages are exchanged frequently and the keep alive interval is not exceeded, there is no need to send an extra message to ensure that the connection is still open.
- If the client doesn't send any messages during the period of the keep alive it must send a PINGREQ packet to the broker to confirm its availability and also make sure the broker is still available.
- The broker must disconnect a client, which doesn't send PINGREQ or any other message in user's specified time of the Keep Alive.

- Also, the client should close the connection if the response from the broker isn't received in a reasonable amount of time.
- After receiving a PINGREQ the broker must reply with a PINGRESP packet to indicate its availability to the client. Similar to the PINGREQ the packet doesn't contain any payload.
- If the broker doesn't receive a PINGREQ or any other packet from a particular client, it will close the connection and send out the last will and testament message.
- If the keep alive interval is set to 0, the keep alive mechanism is deactivated.

### Connection Timeout

This property specifies the maximum amount of time (in seconds) to wait for a response after sending the request to the broker.

### Broker Ip Address

This property specifies the Ip address of the broker. If the broker is installed on the same platform, the default **`Broker Ip Address`** is `127.0.0.1`. If the broker is installed in some other platform, the platform Ip address must be entered in the **Broker Ip Address** field.

### Broker Port

This property specifies the port number of the broker.

### Client I D

The Abstract MQTT protocol defines a "client identifier" that uniquely identifies a client in a network. In simple terms, when connecting to a server a client needs to specify a unique string that is not used currently and will not be used by any other client that will connect to the server. If the two devices are connected with same ID, the **`Status`** sets to `down` and fault message occurs in the **Fault Cause** field. Two clients can not have same client ID, but it is possible if they are connecting from the different stations. In this case the behavior of both the clients will not be as expected and there will be issues in communication between the client and broker.

### Status Message

This property shows whether the broker is connected or disconnected to the client.

### Connection Type

- `Anonymous`

  Select this property to establish an anonymous connection with the broker. The connection type **`Anonymous`** is non secure, choosing the **`Anonymous`** user authentication mode will generate a warning message on the security risk involved. User or administrator has to agree the warning message and the same will be logged in the Niagara system for audit purpose.

- `Anonymous Over SSL`

  The connection is established over SSL, using one-way SSL, where the broker is required to be configured with certificates and keys. Though the connection over SSL is secure, the connection type anonymous is non secure, choosing the **`Anonymous`** user authentication mode will generate a warning message on the security risk involved. User or administrator has to agree the warning message and the same will be logged in the Niagara system for audit purpose.

- `User Login Over SSL`

  This property requires login credentials. You must enter the username and password for the broker if you select this property. In Niagara the default and recommended connection type will be **`User Login Over SSL`**.

### Username And Password

- Username: This property is used when connecting to the broker (User Login Over SSL connection). This may or may not be required depending on the broker.
- Password: This property is used when connecting to the broker (User Login Over SSL connection). This may or may not be required depending on the broker.

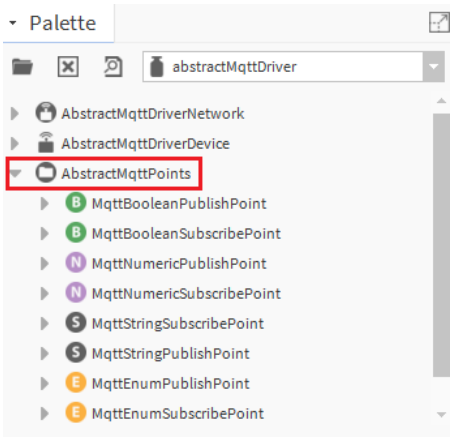
### Send Enum As

Set this property value to `TAG` or `ORDINAL`. This property is configured at **AbstractMqttDriverDevice**. As per this property's value, the enum publish or subscribe points sends or receives the data. All the enum publish or subscribe points of the device use this property to publish or subscribe data to/from the respective topic.

## About AbstractMqttPoints

The AbstractMqttPoints must be located under the **Points** container.

The AbstractMqttPoints are available in the `AbstractMqttPoints` folder of the **abstractMqttDriver** palette.




### MqttBooleanPublishPoint

This point publishes only boolean data.

### MqttBooleanSubscibePoint

This point subscribes only boolean values.

### MqttNumericPublishPoint

This point publishes only numeric data.

### MqttNumericSubscibePoint

This point subscribes only numeric values.

### MqttStringPublishPoint

This point publishes only string data.

### MqttStringSubscibePoint

This point subscribes only string values.

### MqttEnumPublishPoint

This point publishes only enum data.

### MqttEnumSubscibePoint

This point subscribes only enum values.

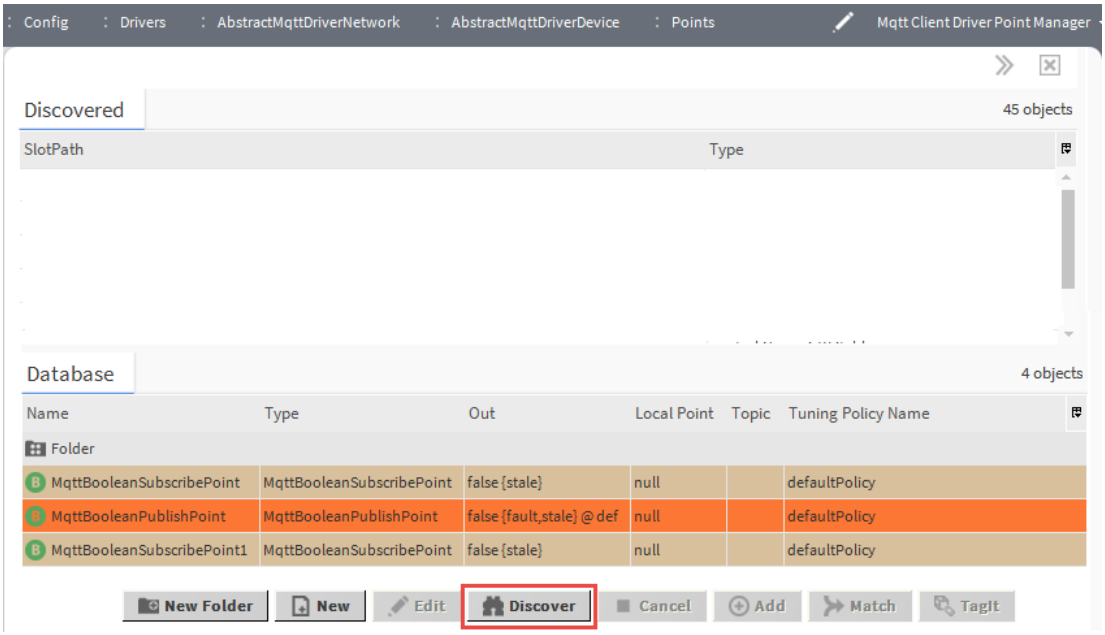
## Discovering AbstractMqttPoints

This procedure describes how to discover AbstractMqttPoints in the AbstractMqttDriverNetwork.

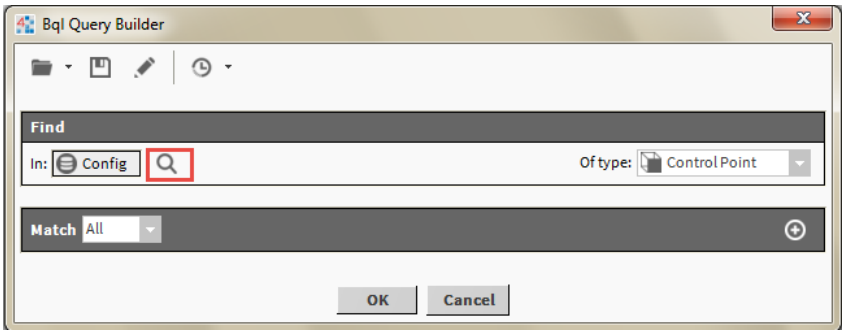
Step 1 In the Nav Tree, expand the **Station→Drivers→AbstractMqttDriverNetwork→AbstractMqttDriverDevice** node.

Step 2 Right-click on the **Points** node and select **Views→Mqtt Client Driver Point Manager** or double click on**Points** node to open the view.

Step 3 In the **Mqtt Client Driver Point Manager** view, click on the **Discover** button.




Step 4 When you click on the **Discover** button, **Bql Query Builder** window pops up.




In the **Bql Query Builder** window, do as follows:

- Click on  icon, the **Choose Root** window pops up.

- In the **Choose Root** window, select the path of points which are available in the station to discover.
- In the **Of Type** drop down list, select `Control Point` option and click the **OK** button to go to the **Bql Query Builder** window.

  **NOTE:**

  You can also select below options in the **Of Type** drop down list:

  - `Boolean Point`
  - `Numeric Point`
  - `Enum Point`
  - `String Point`.
- Click the **OK** button to discover the selected points.

## Adding Points to the database

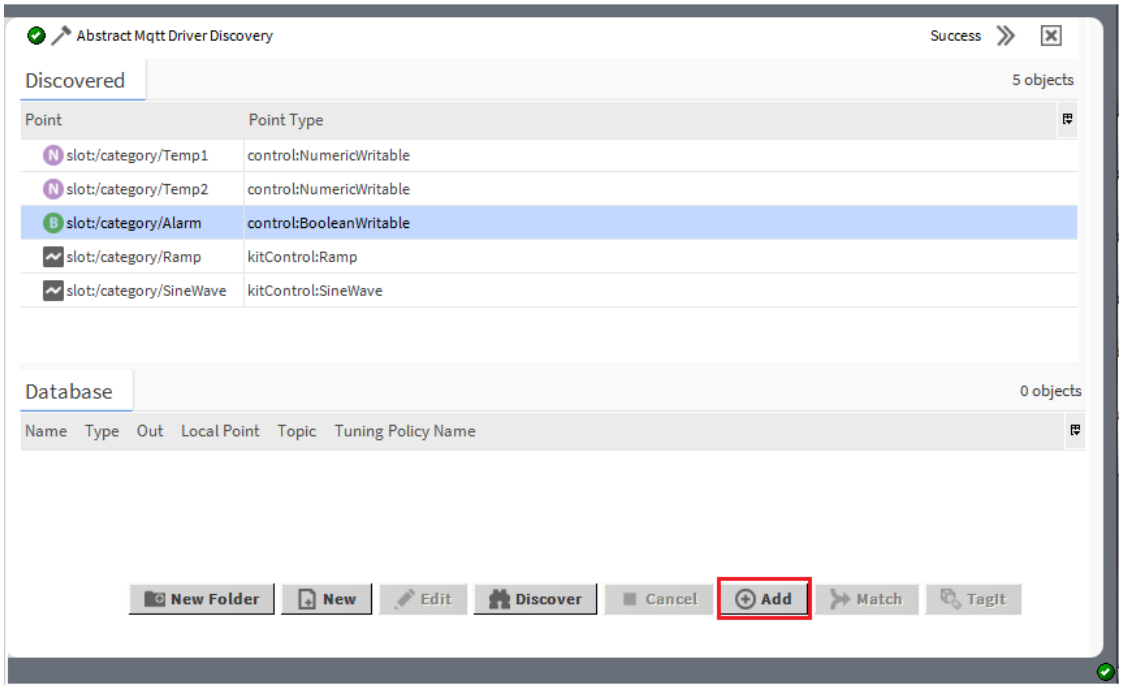
This procedure describes how to add control points to the station for selected discovered objects. This approach is useful when existing station points needs to be exposed as mqtt publisher/subscriber points.

**Prerequisites:**

- Discovered points are visible in the **Mqtt Client Driver Point Manager** view.

Step 1 In the **Mqtt Client Driver Point Manager** view, select the point in the **Discovered** pane to add.

Step 2 Click the **Add** button (or double-click on the selected object). The **Add** dialog box appears.




In the **Add** dialog box, you can edit the point properties before each point is added in the **Database** pane.

In the **Add** dialog, do the following:

a. In the **Name** field, name the point, as desired.

b. In the **Type** field, you can select the type of the point. There are two types of point: publish (read) and subscribe (write).

c. You can keep the **Local Point** field as default or null.

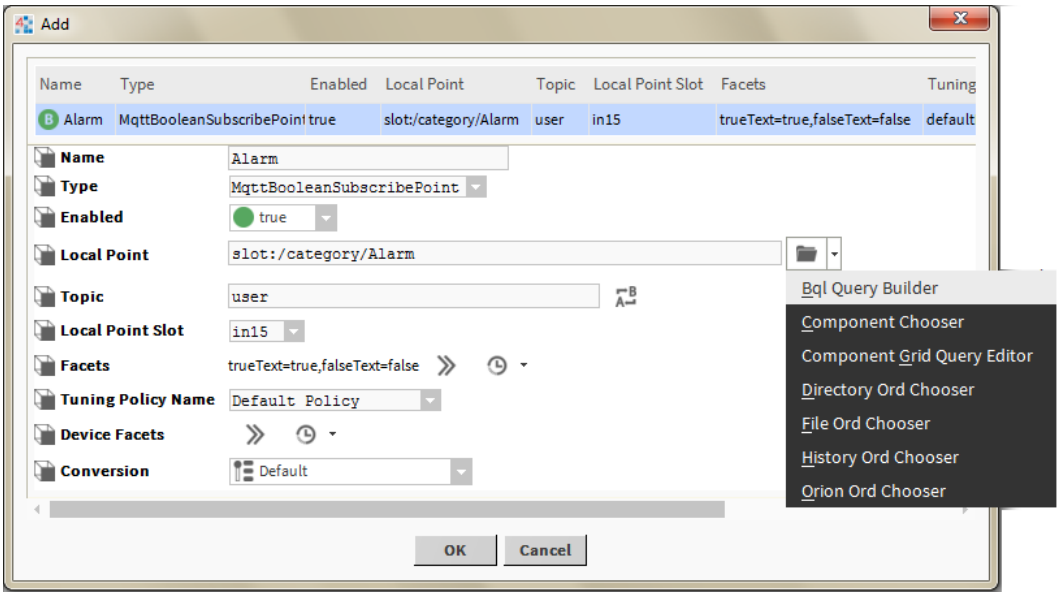
**NOTE:** If the point is added from the discovered points, the local point refers to the ord of the point by default. If it is null, then the point is not linked to any point by default. You have to link the point manually, if needed.

d. In the **Topic** field, enter the topic name, as desired.

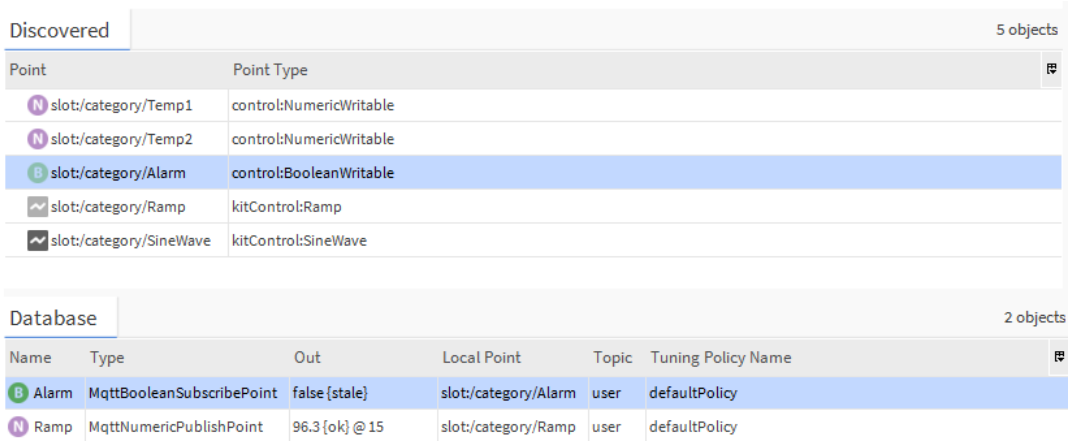
e. In the **Local Point Slot** field, select the desired input point slot which will directly link to the functional blocks.

**NOTE:** The **Local Point Slot** options are available only for subscribe point. If it is a publish point, the default slot is "out".

f. Click **OK** button to add the point to **Database** pane.




The added point begins updating with the real-time values, as shown here.




## Creating the MqttBooleanPublishPoint

This topic describes how to add MqttBooleanPublishPoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttBooleanPublishPoint and copy-and-paste (or drag and drop) the MqttBooleanPublishPoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttBooleanPublishPoint, as desired and click the **OK** button.

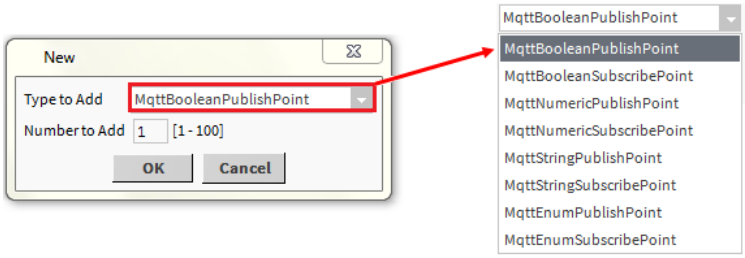
You can also add the MqttBooleanPublishPoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

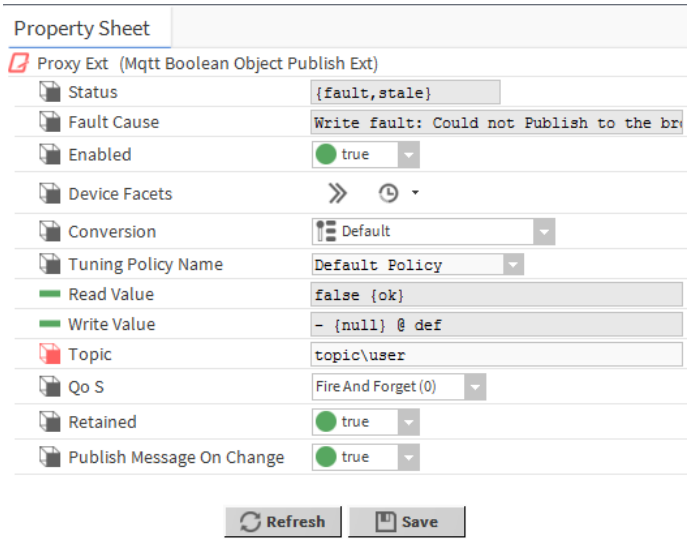
Step 6 Expand the MqttBooleanPublishPoint and double click on the **Proxy Ext** node. The **Mqtt Boolean Object Publish Ext** view appears.

Step 7 In the **Mqtt Boolean Object Publish Ext** view, do the following:

a. In the **Topic** property, type “topic\user”.

**NOTE:** Enter the topic name, “\” in the **Topic** property, and “user” can be any name or word.

b. Click the **Save** button.




**NOTE:** The **`Publish Message On Change`** default property is `true`.

Step 8 Go to the **Mqtt Boolean Object Publish Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt Boolean Object Publish Ext** view and select **Action→Publish** to publish the topic.

Publish returns immediately to the application thread after passing the request to the Abstract MQTT client.

## Creating the MqttBooleanSubscribePoint

This topic describes how to add MqttBooleanSubscribePoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttBooleanSubscribePoint and copy-and-paste (or drag and drop) the MqttBooleanSubscribePoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttBooleanSubscribePoint, as desired and click the **OK** button.

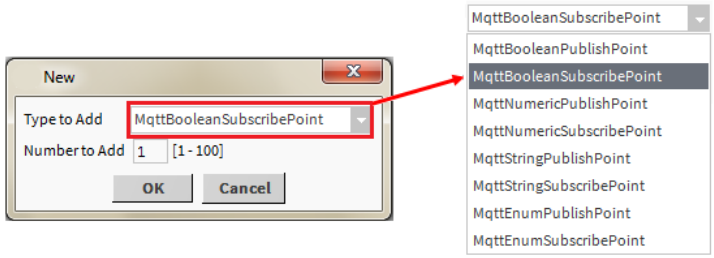
You can also add the MqttBooleanSubscribePoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

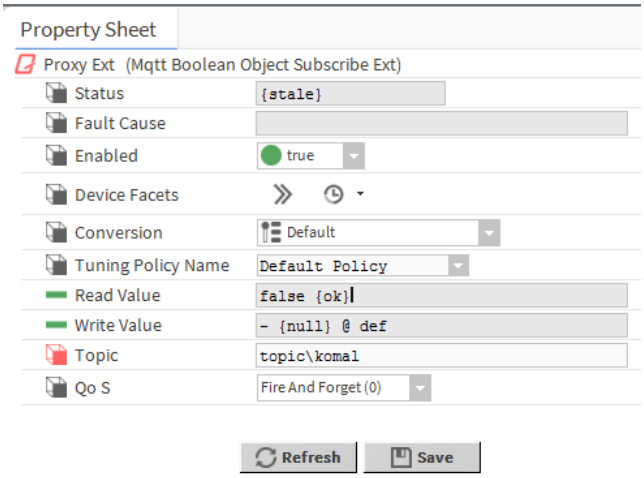
Step 6 Expand the MqttBooleanSubscribePoint and double click on the **Proxy Ext** node. The **Mqtt Boolean Object Subscribe Ext** view appears.

Step 7 In the **Mqtt Boolean Object Subscribe Ext** view, do the following:

a. In the **Topic** property, type the "topic\user".

**NOTE:** Enter the topic name, "\" in the **Topic** property, and "user" can be any name or word.

b. Click the **Save** button.




Step 8 Go to the **Mqtt Boolean Object Subscribe Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt Boolean Object Subscribe Ext** view and select **Action→Subscribe** to subscribe the topic.

**NOTE:** To unsubscribe the client from the topic select **Action→Unsubscribe**.

## Creating the MqttNumericPublishPoint

This topic describes how to add the MqttNumericPublishPoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttNumericPublishPoint and copy-and-paste (or drag and drop) the MqttNumericPublishPoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttNumericPublishPoint, as desired and click the **OK** button.

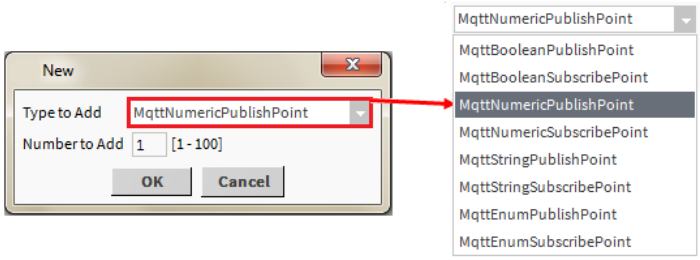
You can also add the MqttNumericPublishPoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

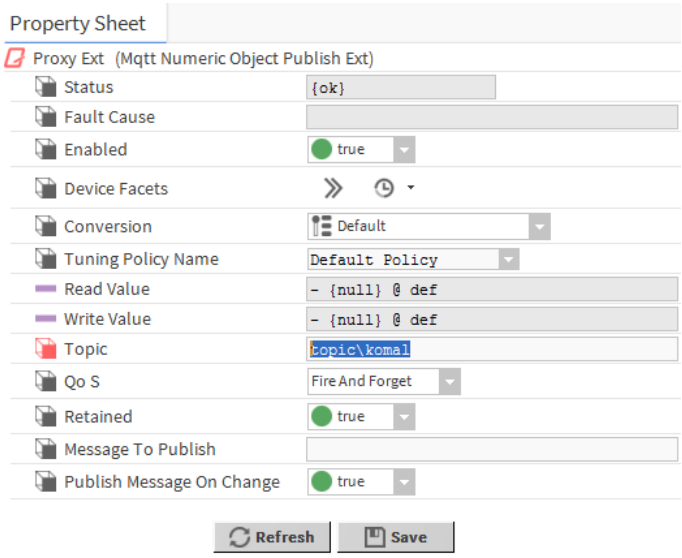
Step 6 Expand the MqttNumericPublishPoint and double click on the **Proxy Ext** node. The **Mqtt Numeric Object Publish Ext** view appears.

Step 7 In the **Mqtt Numeric Object Publish Ext** view, do the following:

a. In the **Topic** property, type “topic\user”.

**NOTE:** Enter the topic name, “\” in the **Topic** property, and “user” can be any name or word.

b. Click the **Save** button.




**NOTE:** The **`Publish Message On Change`** default property is `true`.

Step 8 Go to the **Mqtt Numeric Object Publish Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt Numeric Object Publish Ext** view and select **Action→Publish** to publish the topic.

Publish returns immediately to the application thread after passing the request to the Abstract MQTT client.

## Creating the MqttNumericSubscribePoint

This topic describes how to add the MqttNumericSubscribePoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttNumericSubscribePoint and copy-and-paste (or drag and drop) the MqttNumericSubscribePoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttNumericSubscribePoint as desired and click the **OK** button.

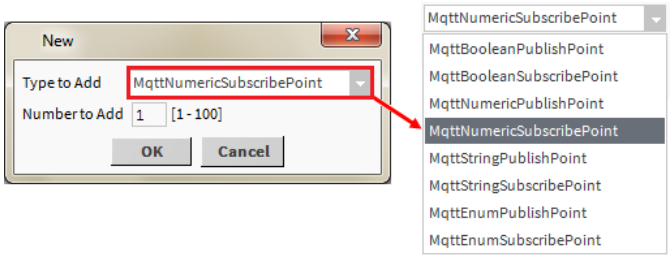
You can also add the MqttNumericSubscribePoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

Step 6 Expand the MqttNumericSubscribePoint and double click on the **Proxy Ext** node. The **Mqtt Numeric Object Subscribe Ext** view appears.

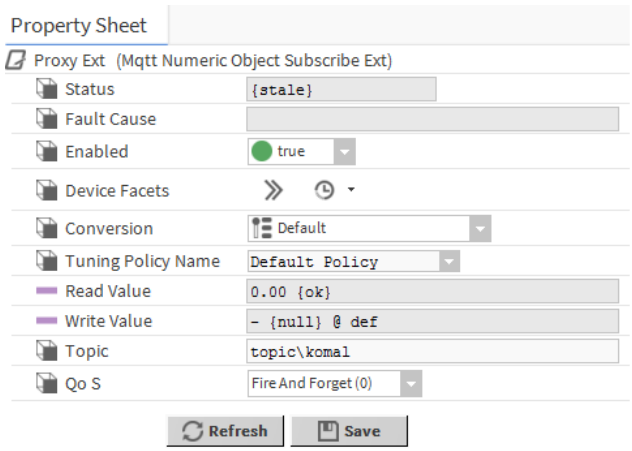
Step 7 In the **Mqtt Numeric Object Subscribe Ext** view, do the following:

a. In the **Topic** property, type “topic\user”.

**NOTE:** Enter the topic name, “\” in the **Topic** property, and “user” can be any name or word.

You can use only one subscribe point per user. However, you can use a wildcard (#) entry to add more subscribe points to the topic. For example, `topic\#`.

b. Click the **Save** button.




Step 8 Go to the **Mqtt Numeric Object Subscribe Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt Numeric Object Subscribe Ext** view and select **Action→Subscribe** to subscribe the topic.

**NOTE:** You can request the server unsubscribe the client from the topic by selecting **Action→Unsubscribe**.

## Creating the MqttStringPublishPoint

This topic describes how to add the MqttStringPublishPoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttStringPublishPoint and copy-and-paste (or drag and drop) the MqttStringPublishPoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttStringPublishPoint, as desired and click the **OK** button.

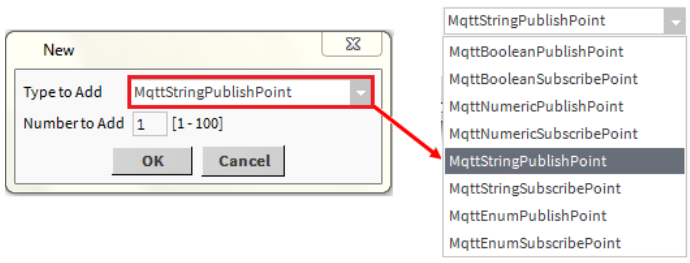
You can also add the MqttStringPublishPoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.



d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

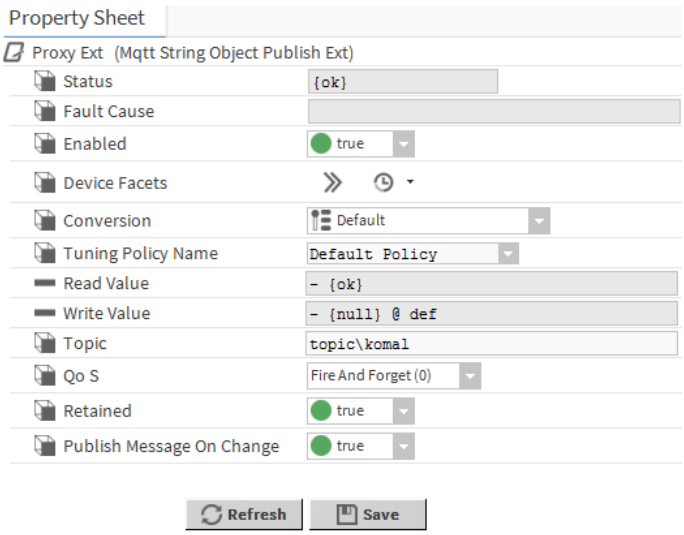
Step 6 Expand the MqttStringPublishPoint and double click on the **Proxy Ext** node. The **Mqtt String Object Publish Ext** view appears.

Step 7 In the **Mqtt String Object Publish Ext** view, do the following:

a. In the **Topic** property, type "topic\user".

**NOTE:** Enter the topic name, "\" in the **Topic** property, and "user" can be any name or word.

b. Click the **Save** button.




**NOTE:** The **`Publish Message On Change`** default property is `true`.

Step 8 Go to the **Mqtt String Object Publish Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt String Object Publish Ext** and select **Action→Publish** to publish the topic.

Publish returns immediately to the application thread after passing the request to the Abstract MQTT client.

## Creating the MqttStringSubscribePoint

This topic describes how to add the MqttStringSubscribePoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttStringSubscribePoint and copy-and-paste (or drag and drop) the MqttStringSubscribePoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttStringSubscribePoint, as desired and click the **OK** button.

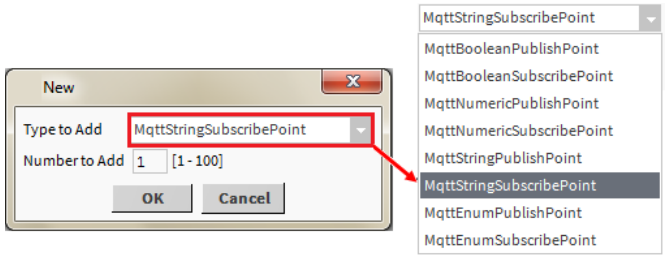
You can also add the MqttStringSubscribePoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

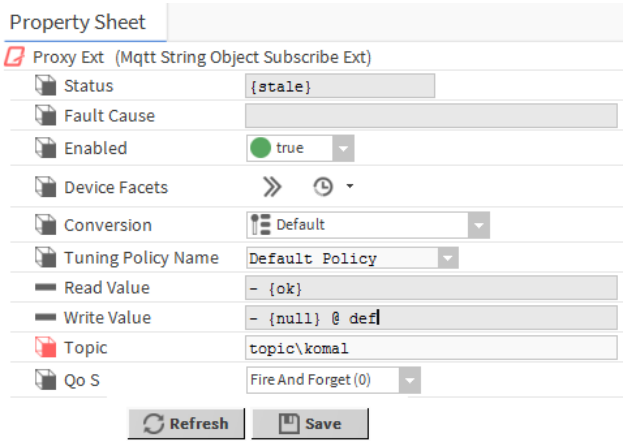
Step 6 Expand the MqttStringSubscribePoint and double click on the **Proxy Ext** node. The **Mqtt String Object Subscribe Ext** view appears.

Step 7 In the **Mqtt String Object Subscribe Ext** view, do the following:

a. In the **Topic** property, type the “topic\user”.

**NOTE:** Enter the topic name, “\” in the **Topic** property, and “user” can be any name or word.

b. Click the **Save** button.




Step 8 Go to the **Mqtt String Object Subscribe Ext** view on the property sheet.

Step 9 Right-click on the **Mqtt String Object Subscribe Ext** view and select **Action→Subscribe** to subscribe the topic.

**NOTE:** To unsubscribe the client from the topic select **Action→Unsubscribe**.

## Creating the MqttEnumPublishPoint

This topic describes how to add the MqttEnumPublishPoint manually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttEnumPublishPoint and copy-and-paste (or drag and drop) the MqttEnumPublish-Point under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttEnumPublishPoint, as desired and click the **OK** button.

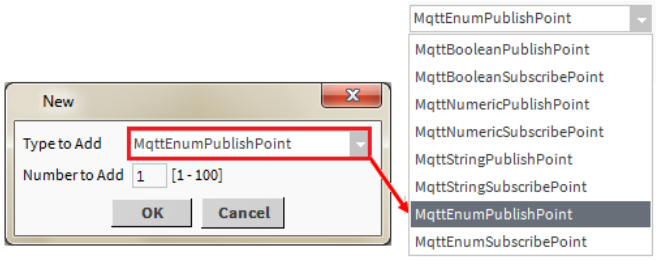
You can also add the MqttEnumPublishPoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

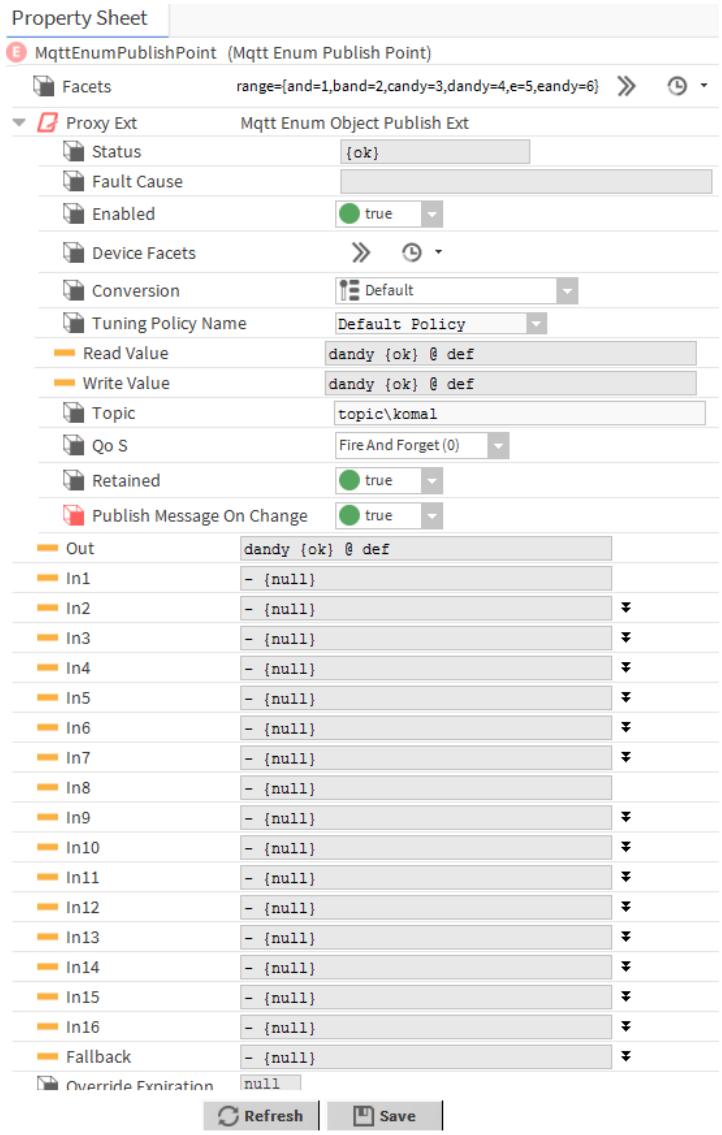
b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

Step 6 Expand the MqttEnumPublishPoint and double click on the **Proxy Ext** node. The **Mqtt Enum Object Publish Ext** view appears.




Step 7 In the **Mqtt Enum Object Publish Ext** view, do the following:

a. In the **Topic** property, type "topic\user".

**NOTE:** Enter the topic name, "\" in the **Topic** property, and "user" can be any name or word.

**NOTE:** The **`Publish Message On Change`** default property is `true`.

Step 8 Set the **`Facets`** to the required enum value that you want to publish.

**NOTE:** The enum value publishes to the topic that depends on the **`Send Enum As`** property. That is either a `TAG` or `ORDINAL`.

Step 9 Go to the **Mqtt Enum Object Publish Ext** view on the property sheet.

Step 10 Right-click on the **.Ext** view and select **Action→Publish** to publish the topic.

Publish returns immediately to the application thread after passing the request to the Abstract MQTT client.

## Creating the MqttEnumSubscribePoint

This topic describes how to add the MqttEnumSubscribePointmanually.

Step 1 In the **Nav** side bar, under the station AbstractMqttDriverNetwork node, expand the **AbstractMqttDriverDevice** node and double-click on the **Points** container.

The **Mqtt Client Driver Point Manager** view displays.

Step 2 Go to the **abstractMqttDriver** palette.

Step 3 Expand the `AbstractMqttPoints` folder from the **abstractMqttDriver** palette.

Step 4 Select the MqttEnumSubscribePoint and copy-and-paste (or drag and drop) the MqttEnumSubscribePoint under the **Points** container or onto the **Mqtt Client Driver Point Manager** view. The **Name** dialog box appears.

Step 5 Name the MqttEnumSubscribePoint, as desired and click the **OK** button.

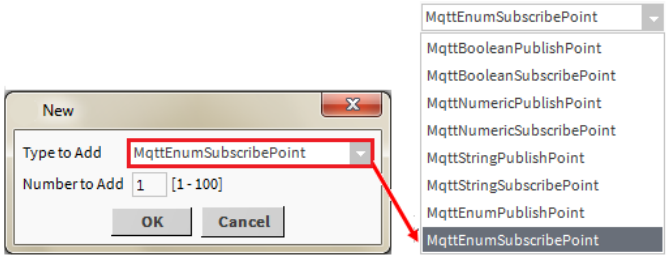
You can also add the MqttEnumSubscribePoint by clicking **New** button at the bottom of **Mqtt Client Driver Point Manager** view. The **New** dialog box appears.

In the **New** dialog box, do the following:

a. In the **Type to Add** field, select the type of proxy point that you want to add from the option list.

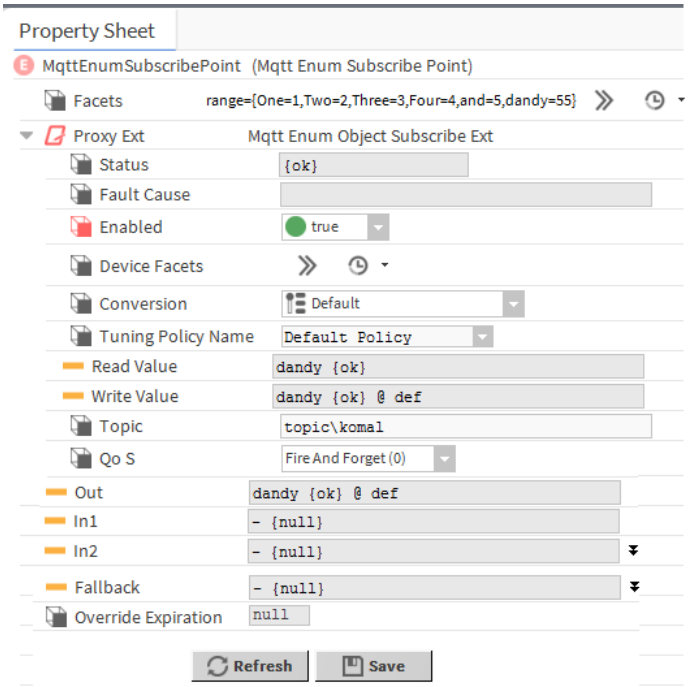
b. In the **Number to Add** field, type in a number to indicate the quantity of proxy points that you want to add.

c. Click the **OK** button. Another **New** dialog box appears.




d. In the **New** dialog box, you can edit proxy point properties before each point is added in the station.

Step 6 Expand the MqttEnumSubscribePoint and double click on the **Proxy Ext** node. The **Mqtt Enum Object Subscribe Ext** view appears.




Step 7 In the **Mqtt Enum Object Subscribe Ext** view, do the following:

a. In the **Topic** property, type "topic\user".

**NOTE:** Enter the topic name, "\" in the **Topic** property, and "user" can be any name or word.

Step 8 Set the **Facets** to the required enum value that you want to publish.

**NOTE:** The enum point sets as per he data on the subscribed topic, its **Facets** and **Send Enum As** property of the device.

- TAG: The data compares to tag of the **Facet**, matches the enum value and set as Enum Point.
- ORDINAL: The data receives the ordinal value, so the corresponding value from **Facet** selects and set as Enum Point.
- If data does not match with the tag and ordinal of the current Enum point, then an Invalid Data exception occurs.

Step 9 Go to the **Mqtt Enum Object Subscribe Ext** view on the property sheet.

Step 10 Right-click on the **Mqtt Enum Object Subscribe Ext** view and select **Action→Subscribe** to subscribe the topic.

**NOTE:** You can request the server unsubscribe the client from the topic by selecting **Action→Unsubscribe**.

## Publishing data using Mqtt

To publish data to the broker, you need to link a data point output value to an Mqtt Publish Point of the proper data type. This procedure describes how to link from a data point to an MqttNumericPublishPoint. The procedure for publishing the boolean, string or enum data point is the same except that you use the MqttBooleanPublishPoint, MqttStringPublishPoint or MqttEnumPublishPoint component to publish boolean, string or enum data respectively.

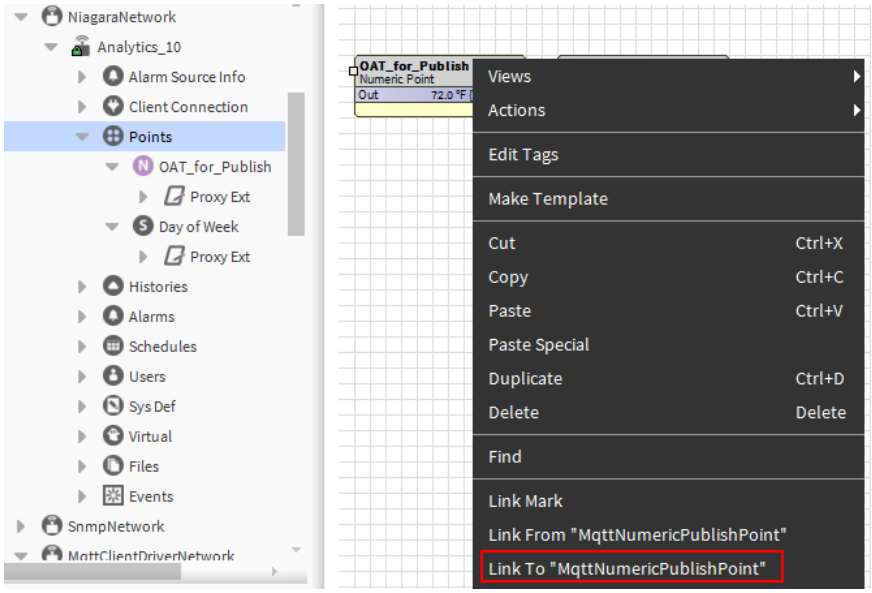
**Prerequisites:**

- Desired data point is available in a Workbench view through any valid network connection.

Step 1 Right-click on the **Points** node and select **Views→Wire Sheet**. The **Wire Sheet** view displays.

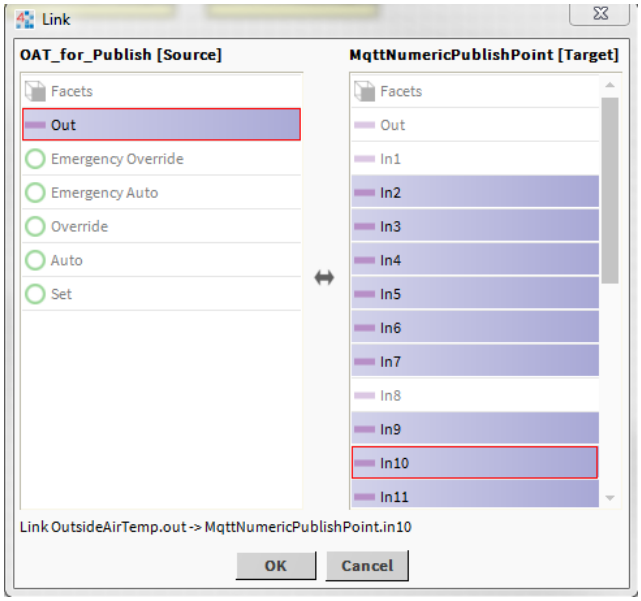
Step 2 Right-click on the MqttNumericPublishPoint and select a **Link Mark** option from the list.

Step 3 Navigate to the desired numeric data point, right-click on it and select the **Link To "MqttNumericPublishPoint"** from the menu.

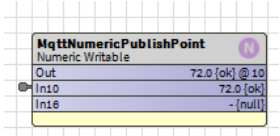



The **Link** window opens.

Step 4 Select the output of the data point to the input of the MqttNumericPublishPoint and click the **OK** button.




The data should now be transferring to the broker.



# Chapter 3 Plugins

**Topics covered in this chapter**

♦ Mqtt Client Driver Device Manager
♦ Mqtt Client Driver Point Manager

Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the **Nav** tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

For summary documentation on any view, select **Help→On View** (F1) from the menu or press F1 while the view is open.

## Mqtt Client Driver Device Manager

The **Mqtt Client Driver Device Manager** is the default manager view of the AbstractMqttDriverNetwork.

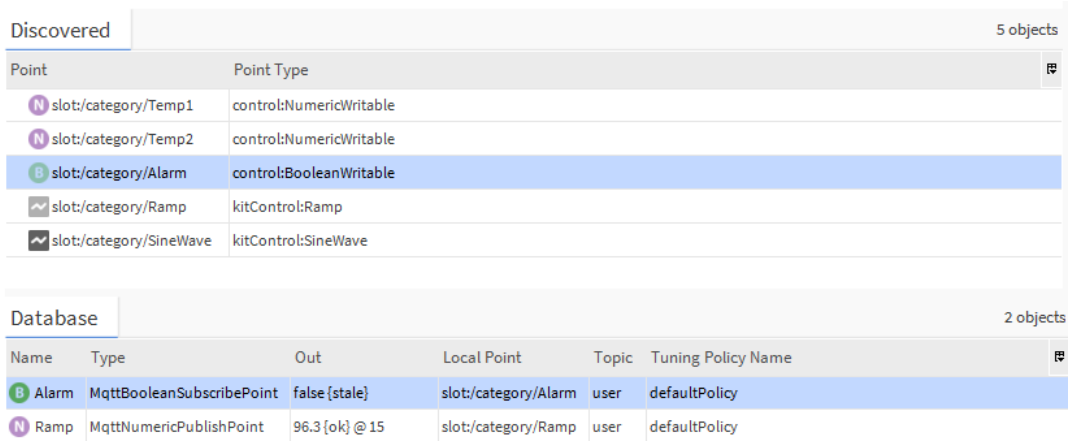
To open this view, right-click the AbstractMqttDriverNetwork and select **Views→ Mqtt Client Driver Device Manager**.

## Mqtt Client Driver Point Manager

The **Mqtt Client Driver Point Manager** is default view of the **Points** container.

To open this view, right-click **Points** container and select **Views→Mqtt Client Driver Point Manager**.

The **Mqtt Client Driver Point Manager** view splits into two panes when you select a **Discover** button.




### Discovered

This pane displays the points which are available in the device.

### Database

This pane displays the points that are added to your station database.

# Chapter 4 Components

**Topics covered in this chapter**

- AbstractMqttDriverNetwork
- AbstractMqttDriverDevice
- Mqtt Boolean Object Publish Point Ext
- Mqtt Boolean Object Subscribe Point Ext
- Mqtt Numeric Object Publish Ext
- Mqtt Numeric Object Subscribe Ext
- Mqtt String Object Publish Ext
- Mqtt String Object Subscribe Ext
- Mqtt Client Driver Point Device Ext
- Mqtt Enum Object Publish Point Ext
- Mqtt Enum Object Subscribe Point Ext

Components include services, folders and other model building blocks associated with a module. Each of the following Mqtt components are briefly described in this section.

This topic contains a short description of a component or the component plugin view.

## AbstractMqttDriverNetwork

The **`AbstractMqttDriverNetwork`** is the top-level container component for an AbstractMqttDriverNetwork in a station.

The **`AbstractMqttDriverNetwork`** component is located in the **abstractMqttDriver** palette. Like other drivers, you should install this component under the **Drivers** node of a station. The default view of this component is the **Mqtt Client Driver Device Manager** view where you can add new devices, remove or edit devices in a tabular layout.

## AbstractMqttDriverDevice

The **AbstractMqttDriverDevice** component should be added under the AbstractMqttDriverNetwork and has properties that you use to enable and configure the associated device, points and point device extensions.

The **AbstractMqttDriverDevice** is available in the **abstractMqttDriver** palette.

## Mqtt Boolean Object Publish Point Ext

The Mqtt Boolean Object Publish Point Ext is contained under the MqttBooleanPublishPoint and contains properties that you configure for publishing boolean data.

The Mqtt Boolean Object Publish Point Ext is available in the **abstractMqttDriver** palette.

## Mqtt Boolean Object Subscribe Point Ext

The Mqtt Boolean Object Subscribe Point Ext is contained under the MqttBooleanSubscribePoint and contains properties that you configure for subscribing boolean data.

The Mqtt Boolean Object Subscribe Point Ext is available in the **abstractMqttDriver** palette.

## Mqtt Numeric Object Publish Ext

The Mqtt Numeric Object Publish Ext is contained under the MqttNumericPublishPoint and contains properties that you configure for publishing numeric data.

The Mqtt Numeric Object Publish Ext is available in the **abstractMqttDriver** palette.

## Mqtt Numeric Object Subscribe Ext

The Mqtt Numeric Object Subscribe Ext is contained under the MqttNumericSubscribePoint and contains properties that you configure for subscribing numeric data.

The Mqtt Numeric Object Subscribe Ext is available in the **abstractMqttDriver** palette.

## Mqtt String Object Publish Ext

The Mqtt String Object Publish Ext is contained under the MqttStringPublishPoint and contains properties that you configure for publishing string data.

The Mqtt String Object Publish Ext is available in the **abstractMqttDriver** palette.

## Mqtt String Object Subscribe Ext

The Mqtt String Object Subscribe Ext is contained under the **`MqttStringSubscribePoint`** component and contains properties that you configure for subscribing string data.

The Mqtt String Object Subscribe Ext is available in the **abstractMqttDriver** palette as a child of the **`MqttStringSubscribePoint`** component.

## Mqtt Client Driver Point Device Ext

The Mqtt Client Driver Point Device Ext is container for Mqtt client driver points.

The Mqtt Client Driver Point Device Ext is contained under the **AbstractMqttDriverDevice** component.

## Mqtt Enum Object Publish Point Ext

The Mqtt Enum Object Publish Point Ext is contained under the MqttEnumPublishPoint and contains properties that you configure for publishing enum data.

The Mqtt Enum Object Publish Point Ext is available in the **abstractMqttDriver** palette.

## Mqtt Enum Object Subscribe Point Ext

TheMqtt Enum Object Subscribe Point Ext is contained under the MqttEnumSubscribePoint and contains properties that you configure for subscribing enum data.

The Mqtt Enum Object Subscribe Point Ext is available in the **abstractMqttDriver** palette.

# Index