

Technical Document

Niagara MQTT Integration with Google Cloud Platform Guide

August 3, 2022



Niagara MQTT Integration with Google Cloud Platform Guide

Tridium, Inc.

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

Confidentiality

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

Trademark notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and patent notice

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2022 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

Contents

About this guide (Mqtt to GCP Integration)	5
Document change log (Mqtt to GCP Integration)	5
Related documentation	5
Chapter 1 Setting up GCP Cloud IoT	7
Creating a device	9
Creating a gateway	10
Binding and unbinding a device	11
Creating a topic for publish and subscribe	14
Chapter 2 Setting up a Google gateway in Niagara	15
About the Google Mqtt Gateway component mixins	21
GCP IoT registry messages	25
GCP IoT device messages	25
Chapter 3 About gateway actions	27
Gateway actions	27
Glossary	31
Index	33

About this guide (Mqtt to GCP Integration)

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. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document describes how to integrate your existing Niagara Abstract Mqtt driver with the Google Cloud Platform (GCP). Sections in this guide include chapters about setting up the GCP IoT, setting up google gateway in Niagara, and performing gateway actions.

Document change log (Mqtt to GCP Integration)

Changes to this document are listed in this topic.

August 3, 2022

- New version description details are updated in the "Gateway Data" and "BacnetDevice1 Data" in the "About Gateway Actions" chapter.
- Updated the gateway metadata state "version" in the "Publish Gateway State" and "Publish Device State".

July 20, 2022

- Updated new property sheet for the "GoogleMQTTGateway".
- Updated new property sheet for the "DefaultDeviceExportMarker".
- Added new property "Model" to the property description table.

June 4, 2022

Initial document release.

Related documentation

Additional information is available in the following documents.

- *Abstract MQTT Driver Guide*
- *Niagara Drivers Guide*

Chapter 1 Setting up GCP Cloud IoT

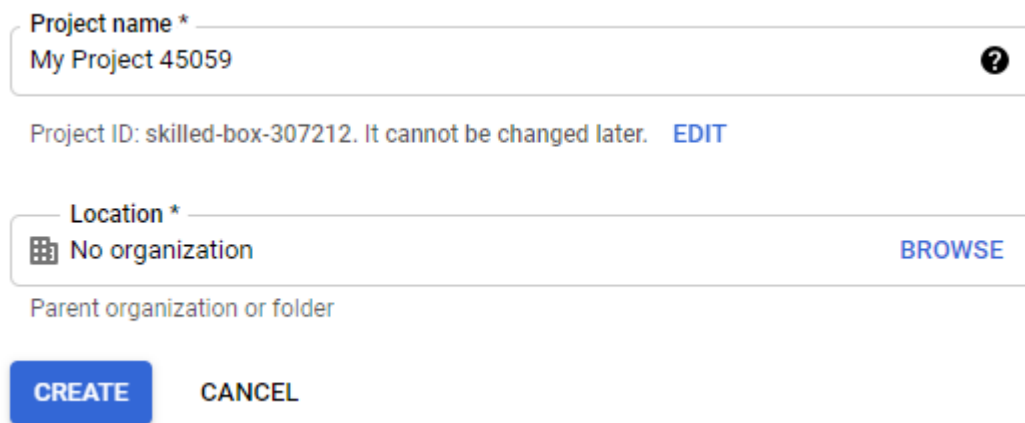
Topics covered in this chapter

- ◆ Creating a device
- ◆ Creating a gateway
- ◆ Binding and unbinding a device
- ◆ Creating a topic for publish and subscribe

You need to create a project on the Google platform before you can connect to an MQTT driver. This topic describes how to create a project on the Google platform.

Prerequisites: You have a current valid account on the Google platform.

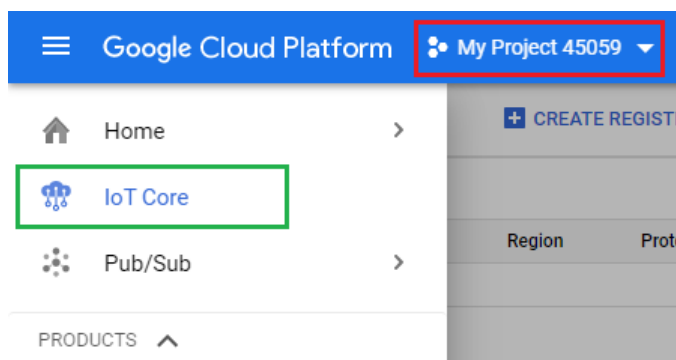
Step 1 Log in to your account on <https://console.cloud.google.com> and create a project, as follows.



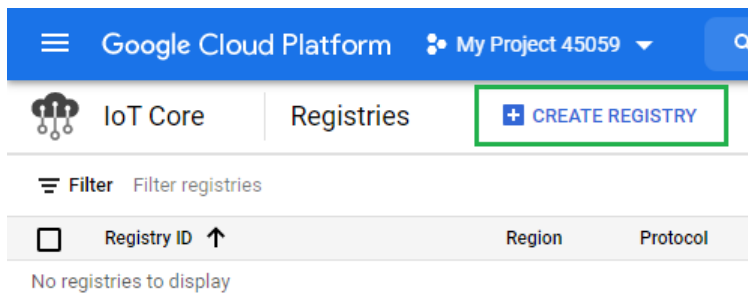
The screenshot shows the 'Create Project' form in the Google Cloud Platform console. The 'Project name' field is filled with 'My Project 45059' and has a help icon. Below it, the 'Project ID' is 'skilled-box-307212' with a note that it cannot be changed later and an 'EDIT' link. The 'Location' field is set to 'No organization' with a 'BROWSE' link. At the bottom are 'CREATE' and 'CANCEL' buttons.

- In the **Project Name** text field accept the default Project Id (automatically generated) or type your own.
NOTE: The project ID is a globally unique identifier for your project. You cannot change the project ID after the project is created.
- The **Location** field is optional. If you have an organization name to add to here, use the **BROWSE** link to select and add a value to this field.
- Click **Create** to initiate creation of the project on the [Google Cloud Platform, page](#) .

Step 2 In the **Project**, click the **menu button**  and select **IoT Core** from the menu pane.



The **IoT Core** view displays.



Step 3 On the **IoT Core** view, select the **Create Registry** menu.

The **Create Registry** view displays.

A screenshot of the 'Create a registry' form in the Google Cloud Platform console. The form includes a 'Registry ID' text field with the value 'niagara-reg' and a 'Region' dropdown menu set to 'asia-east1'. Below these fields, there is a section for 'Cloud Pub/Sub topics' with a dropdown menu set to 'None'. At the bottom, there is a '+ ADD ADDITIONAL TOPIC' button, a 'SHOW ADVANCED OPTIONS' link, and 'CREATE' and 'CANCEL' buttons.

Define how devices in this registry will send data to Cloud IoT Core. After you create your registry, you can start adding devices to it. [Learn more](#)

Registry properties

Registry ID
niagara-reg

Permanent identifier for your registry. 3-255 characters. Start with a letter. You can also include numbers and the following characters: + . % - _ ~

Region
asia-east1

Determines where data is stored for devices in this registry. Choice is permanent.

Cloud Pub/Sub topics

Cloud IoT Core routes device messages to Cloud Pub/Sub for aggregation. You can route messages to different topics and subfolders in Cloud Pub/Sub based on the type of data in the messages. [Learn more](#)

Select a Cloud Pub/Sub topic
None

Device telemetry events will be published to this topic by default.

+ ADD ADDITIONAL TOPIC

SHOW ADVANCED OPTIONS

CREATE CANCEL

- Step 4 In the **Create Registry** view, enter your registry ID in the **Registry ID** field, choose your appropriate region from the **Region** drop-down list and then click **SHOW ADVANCED OPTIONS**.

NOTE: You cannot edit **Registry ID** and **Region** later using the **Edit Registry** menu, however, you can edit the other fields later using the **Edit Registry** menu.

The **Advanced Settings** options display.

Select the protocols your devices will use to connect to Cloud IoT Core. [Learn more](#)

☒ MQTT

☒ HTTP

Cloud Logging

Set the default logging for all devices in this registry. You can apply a different setting or debug at the device level. [Learn more](#)

☒ Disabled
No device data stored.

☐ Error
Captures device errors, such as failed connection attempts and failed publishes. Does not include authentication errors.

☐ Info
MQTT only. Captures device errors (except authentication errors) and includes all lifecycle events, such as device connections and disconnections.

☐ Debug
Captures all device activity in a highly verbose log statement. Recommended for device troubleshooting.

CA certificate (optional)

Input method

☒ Enter manually

☐ Upload

Certificate value

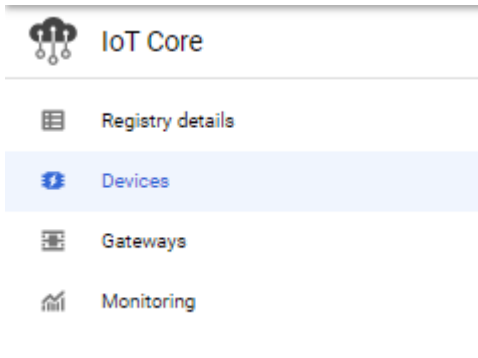
- Step 5 In the **Advanced Settings** section, choose your desired device **Connection Protocols**, select a **Cloud Logging** option and (optionally) provide your **CA certificate** manually or by upload.

Creating a device

This section describes how you can create a device.

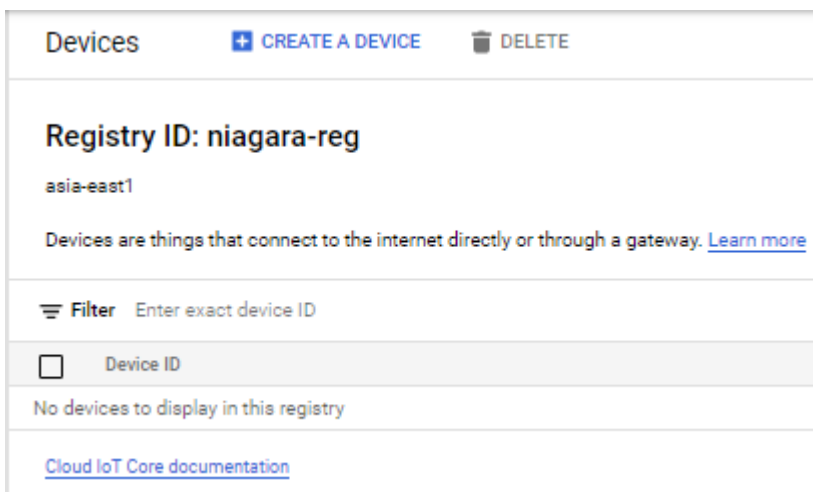
Prerequisites: You have created a project and a registry.

Step 1 From the **IoT Core** menu, select **Devices** to open the **Devices** view.



The **Devices** view displays a list of all registered devices.

Step 2 To create a new device, click **CREATE A DEVICE** under the created registry.



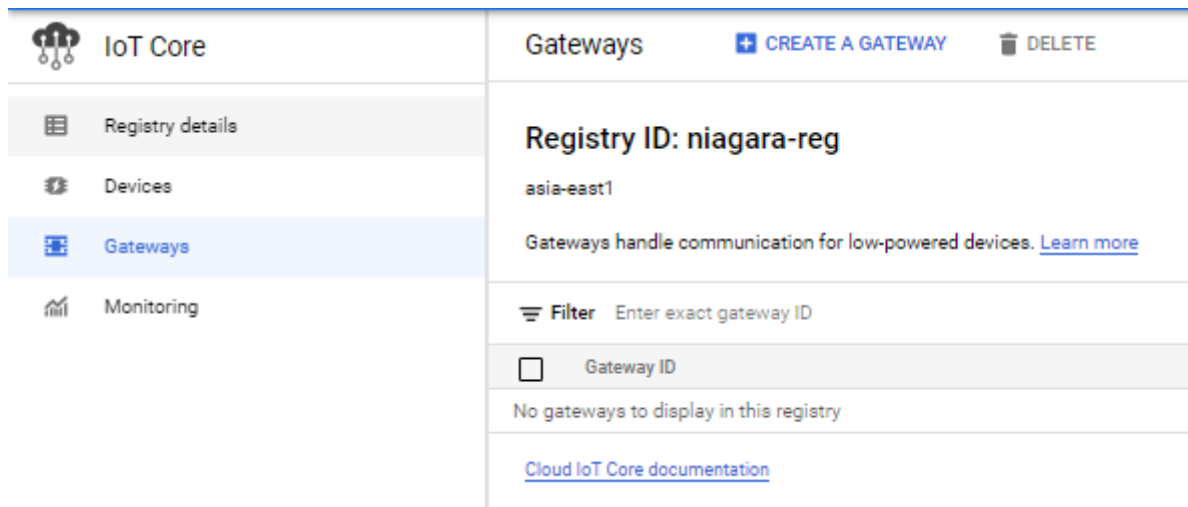
Step 3 Enter the **device ID**, keep the remaining default settings, and click **OK**.

Creating a gateway

This section describes how to create a gateway.

Prerequisites: You have set up a project on Google platform and created a device.

Step 1 From the **IoT Core** menu, select **Gateways** to see all the registered gateways in a list or to create new gateways from **Gateways** view.



Step 2 To create a new gateway under the established registry, click **CREATE A GATEWAY**.

Step 3 Enter the gateway ID, keep the remaining default settings and click **OK**.

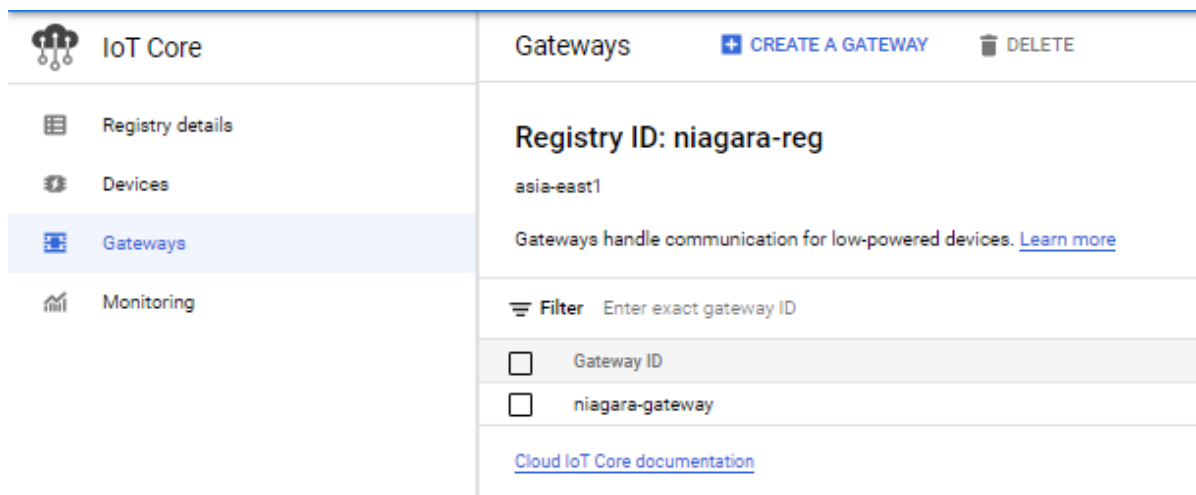
Binding and unbinding a device

This section describes how you can bind and unbind a device to and from the gateway.

Prerequisites: You have created a device and a gateway.

Step 1 After creating the gateway, you can bind devices to the gateway to publish and subscribe data.

Step 2 From the **IoT Core** menu, select **Gateways** to see all created gateways. In the following figure, only one gateway named “niagara-gateway” is present.



Step 3 To view the **Gateway details** page, click the name of the gateway (for example, “niagara-gateway”).

The screenshot shows the 'Gateway details' page for a gateway named 'niagara-gateway'. At the top, there is a navigation bar with links: 'EDIT GATEWAY', 'UPDATE CONFIG', 'SEND COMMAND', 'BLOCK COMMUNICATION', and 'DELETE'. Below the navigation bar, the gateway ID is displayed. A table provides details about the gateway's configuration:

Numeric ID	Registry	Cloud Logging	Communication
3294390922330430	niagara-reg	Registry default View logs	Allowed

Below the table, there are four tabs: 'DETAILS' (selected), 'BOUND DEVICES', 'CONFIGURATION & STATE', and 'AUTHENTICATION'. The 'Latest activity' section shows a list of events, all with a status of '-':

Event	Status
Heartbeat (MQTT only)	-
Telemetry event received	-
Gateway state event received	-
Config sent	-
Zone Config ACK (MQTT only)	-
Error	-

The 'Authorization method' is 'Association only'. The 'Gateway metadata' section states: 'You can add or edit metadata in gateway settings. [Edit gateway.](#)'

Step 4 Select the **BOUND DEVICES** tab to open the **Bound Devices** tab page.

The screenshot shows the 'Bound Devices' tab page for the same gateway. The 'BOUND DEVICES' tab is now selected. At the top, there are buttons for 'BIND DEVICES' and 'UNBIND'. Below these buttons, there is a filter section with a 'Filter' label and a text input field 'Enter exact device ID'. A table with one header row is shown:

<input type="checkbox"/> Device ID

Below the table, a message states: 'No devices are bound to the gateway. Bind devices to associate them with this gateway.'

Step 5 Click **BIND DEVICES** to display a list of devices. Select the checkbox associated with each device that you want to bind to the gateway, and click **BIND**.

Bind devices to gateway

Select devices from the "niagara-reg" registry to bind to the "niagara-gateway" gateway.

Binding devices to the gateway will allow them to exchange MQTT/HTTP messages with Cloud IoT Core. Limited to 10,000 devices per gateway.

Filter Enter exact device IDs separated by commas

<input type="checkbox"/>	Device ID	Communication	Last seen	Cloud Logging
<input type="checkbox"/>	BacnetDevice1	✓ Allowed	—	Registry default
<input type="checkbox"/>	BacnetDevice2	✓ Allowed	—	Registry default
<input type="checkbox"/>	BacnetDevice3	✓ Allowed	—	Registry default

No devices selected. Select 1 or more devices to bind to the gateway.

CANCEL BIND

A dialog box appears to confirm the binding status.

Gateway and device binding complete

✓ 2 bound successfully.
All the selected devices are now bound to the gateway "niagara-gateway".

OK

Step 6 Click **OK** to clear the dialog box and display the **BOUND DEVICES** tab page with a list of all the bound devices.

← Gateway details EDIT GATEWAY UPDATE CONFIG SEND COMMAND BLOCK COMMUNICATION DELETE

Gateway ID: niagara-gateway

Numeric ID	Registry	Cloud Logging	Communication
3294390922330430	niagara-reg	Registry default View logs	Allowed

DETAILS **BOUND DEVICES** CONFIGURATION & STATE AUTHENTICATION

BIND DEVICES UNBIND

Filter Enter exact device ID


<input type="checkbox"/>	Device ID	
<input type="checkbox"/>	BacnetDevice1	✓
<input type="checkbox"/>	BacnetDevice2	✓

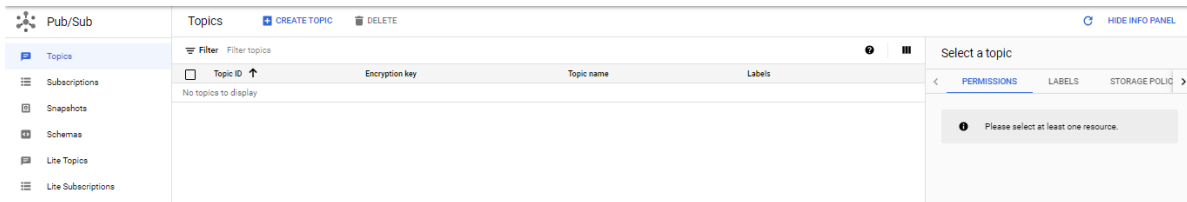
Step 7 To unbind a device from the gateway, select the checkbox of a bound device and then click **UNBIND**.

Creating a topic for publish and subscribe

This section describes how to create a topic for publish and subscribe.

Prerequisites: You have created a device and a gateway.

Step 1 To view the list of items, navigate to the left corner of your screen, click the **menu button**  and select **Pub/Sub**.



Step 2 To create a topic, click **Topics**→**CREATE TOPIC**.

Step 3 Provide an ID for the topic (for example, "state"), keep all remaining default settings, and click **CREATE TOPIC**.

The newly created topic appears in the topic list on the **Topics** page.

For mqttGateway, there are two topics:

- state-projects/{Project-Id}/topics/state
- events-projects/{Project-Id}/topics/events

Chapter 2 Setting up a Google gateway in Niagara

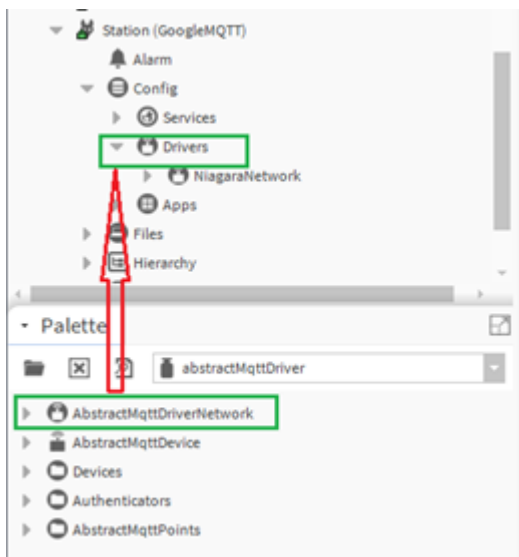
Topics covered in this chapter

- ◆ About the Google Mqtt Gateway component mixins
- ◆ GCP IoT registry messages
- ◆ GCP IoT device messages

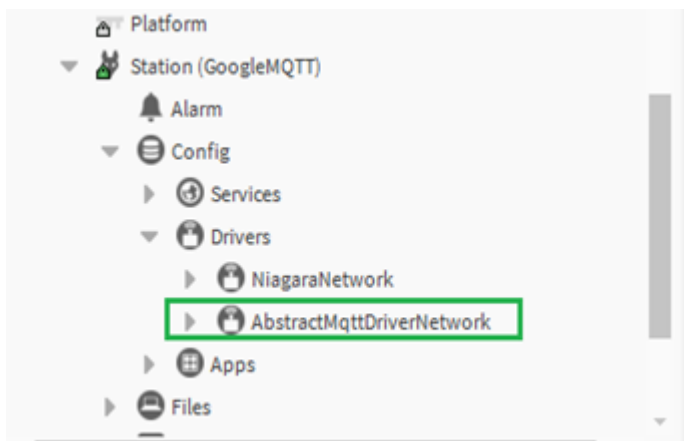
This topic describes how you can set up a Google gateway in Niagara.

Prerequisites: You have created and started a new station.

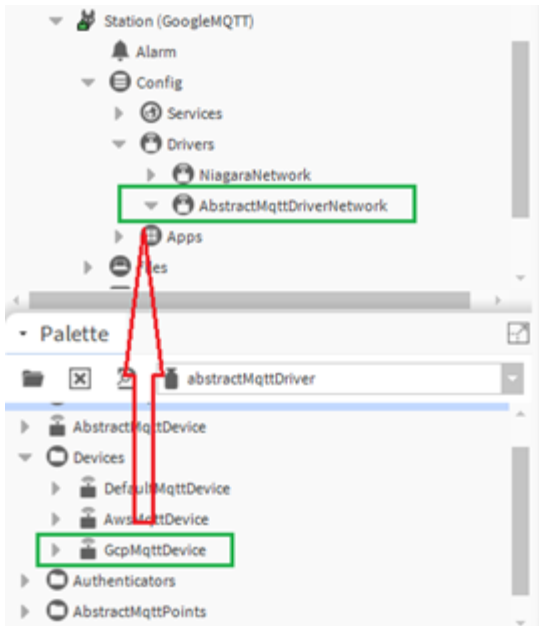
Step 1 From the **abstractMqttDriver** palette, add the **AbstractMqttDriverNetwork** component to the **Drivers** container in the Nav tree.



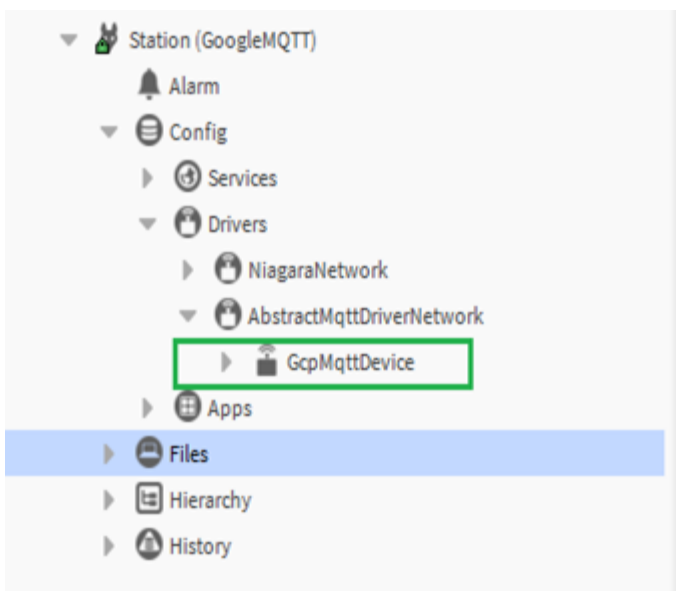
The **AbstractMqttDriverNetwork** is added to the **Drivers** container.



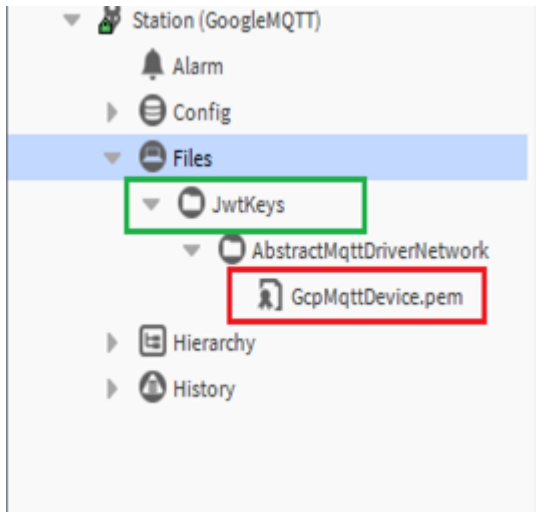
Step 2 From the **abstractMqttDriver** palette, drag the **GcpMqttDevice** component from the **Devices** folder to the **AbstractMqttDriverNetwork** folder.



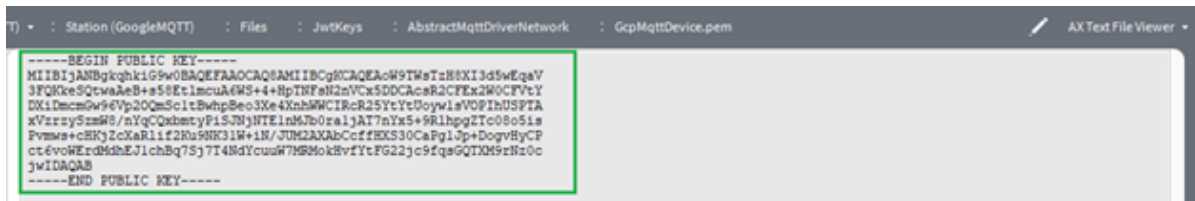
The **GcpMqttDevice** is added to **AbstractMqttDriverNetwork**.



After the **GcpMqttDevice** is added to **AbstractMqttDriverNetwork**, a **JwtKeys** folder containing a "pem" file is automatically added in the **Files** container of the station.

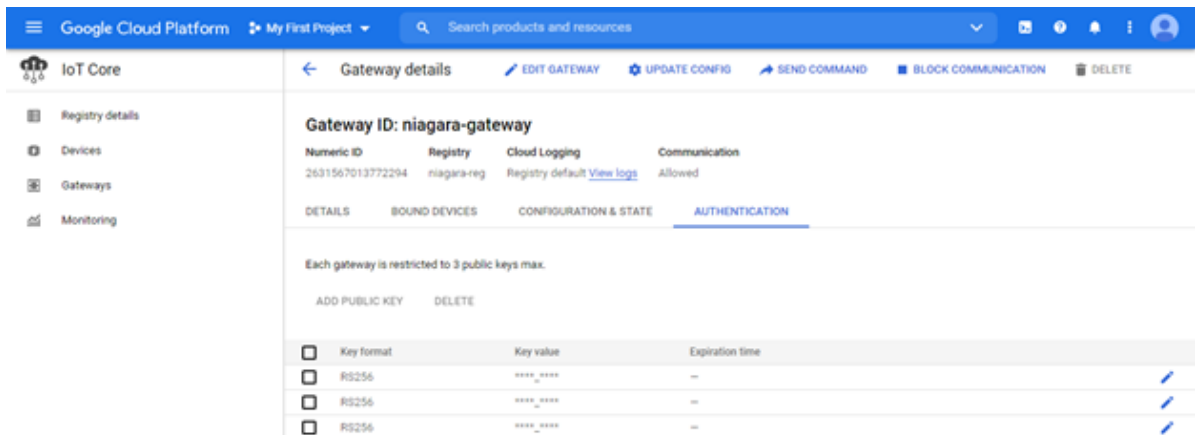


Step 3 Open the pem (located in the **AbstractMqttDriverNetwork** folder) in the **AX Text File Viewer**, copy the public key, as shown below, and add it to the GCP gateway authentication field, as described in the following steps.



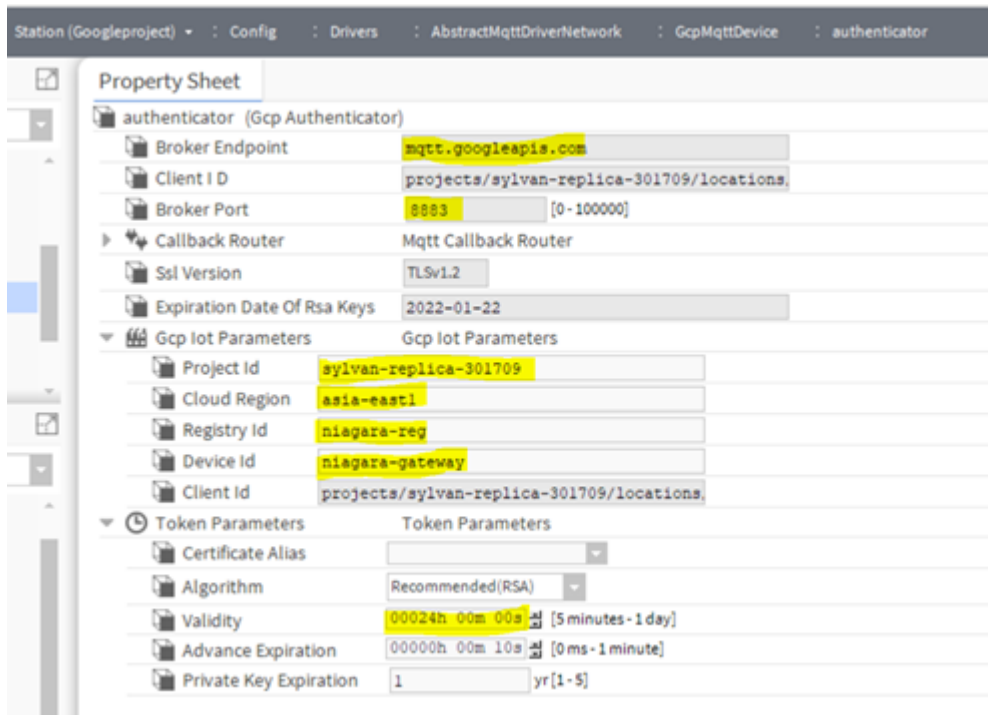
Step 4 Open the [Google Cloud Platform](#), page in the web browser and log in with valid credentials.

Step 5 In GCP, navigate to **Gateway Details** and select the **Authentication** tab.



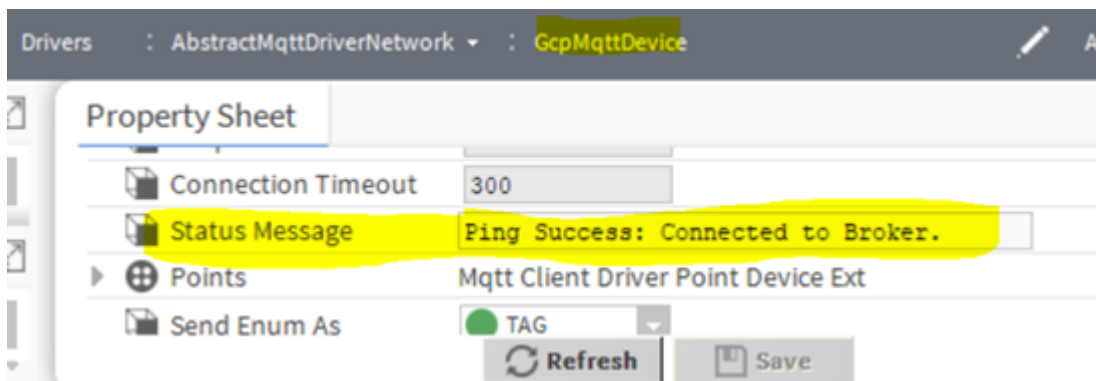
Step 6 To add the public key, click **Add Public Key** or update any individual existing key (if not in use) with the copied public key. Select RS256 as the key format.

Step 7 In Workbench, under the **GcpMqttDevice**, open the **authenticator** property sheet to configure the **Authenticator**, **GCP Iot Parameter**, and **Token Parameters** properties.



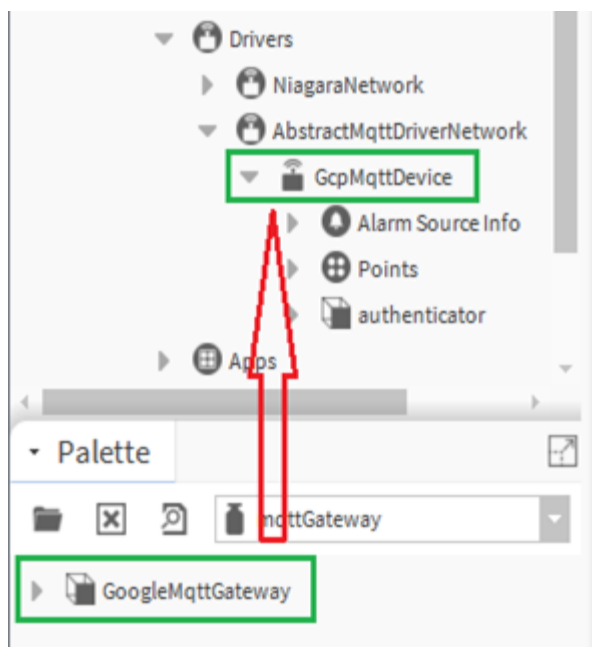
Refer to the *Google_IOT.pdf* document for more details about the properties in this view.

- Step 8 To connect the device, right-click **GcpMqttDevice**→**Actions**→**Ping**. Once the ping operation is successful, the **Status Message** property displays a successful status message, for example: Ping Success: Connected to Broker.

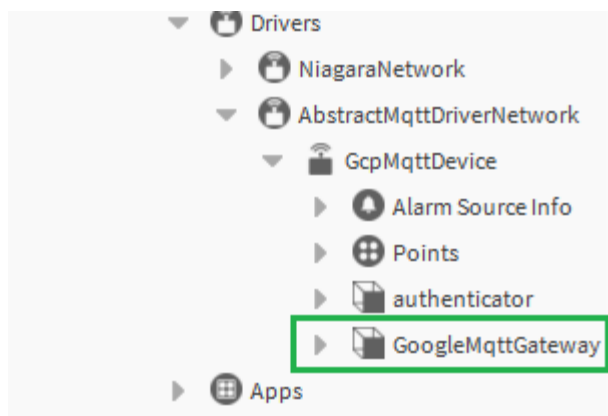


NOTE: If the connection is unsuccessful, restart the station and try to connect again.

- Step 9 From the **mqttGateway** palette, add the **GoogleMqttGateway** component to the **GcpMqttDevice** folder in the Nav tree.



The **GoogleMqttGateway** is added to the device.



Step 10 Configure the properties of **GoogleMqttGateway**.

Property Sheet	
GoogleMqttGateway (Google Mqtt Gateway)	
Status	{ok}
Fault Cause	
Enabled	<input checked="" type="radio"/> true
Udmi Version	1.3.14
Make	
Model	1.4
Firmware Version	4.11.1.16
Operational	<input checked="" type="radio"/> true
Last Config	23-Nov-2018 11:29 AM EST
Message	Google MQTT Integration
Category	device.state.com
Level	600
▶ Alarm Source Info	Alarm Source Info
Device Alarm Enabled	<input checked="" type="radio"/> true
Store Payload In History	<input type="radio"/> false
Serial Number	
▶ {A} gatewayStateSchema	Relative Json Schema

When the **GoogleMqttGateway** is added, the following actions subsequently occur:

- a. Two subscribe points are added to the **Points** container.
 - GatewayErrorSubscribePoint: subscribed to `"/devices/{Gateway-ID}/errors"` topic.
 - GatewayConfigSubscribePoint: subscribed to `"/devices/{Gateway-ID}/config"` topic.

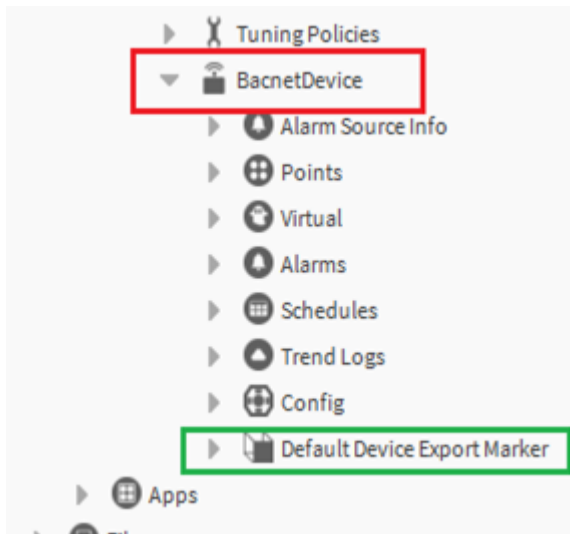
These points are updated if the gateway receives messages published to these topics from the GCP.

The gateway publishes the gateway state message when the gateway receives a Config message from the gateway Config topic.

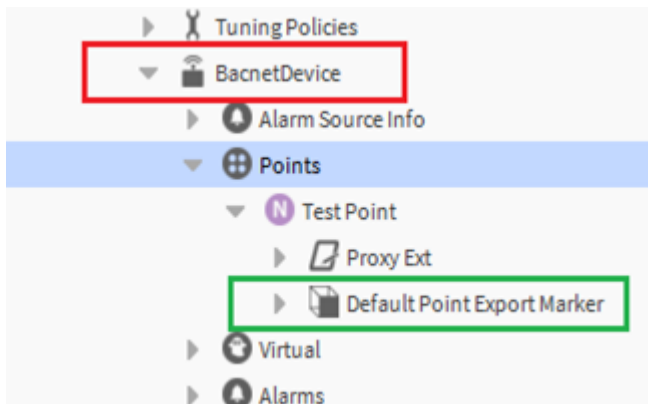
For devices with the following error messages, the gateway sets the device's attach status to false.

- GATEWAY_ATTACHMENT_ERROR
- GATEWAY_DEVICE_NOT_FOUND
- GATEWAY_DETACHMENT_DEVICE_ERROR

Step 11 A **Default Device Export Marker** mixin is added for any existing devices or any new device. This mixin is automatically added when the **GoogleMqttGateway** is added to the station.



Step 12 A **Default Point Export Marker** mixin is added for any existing points or any new point. This mixin is automatically added when **GoogleMqttGateway** is added in the station.



The **Points** folder can have subfolders, and control points are available in these subfolders. This feature is optional and can be helpful if the device has many points. Control point names and subfolder names should be unique. A station restart is required whenever a new subfolder is created.

NOTE:

In order to meet requirements of the UDMI specification, the total number of subfolders should be two less than the telemetry trigger interval (converted to seconds) divided by 10.

For example, if the telemetry trigger is set to 1 minute (60 seconds), then the maximum number of allowed subfolders is 4, as shown in the following equation:

$$(60/10) - 2 = 4$$

If you need to exceed the maximum number of allowed subfolders, you can set the "store payload in history" flag in the **GoogleMqttGateway** component to `true` and restart the station. The point telemetry is now stored in histories and sent accordingly.

About the Google Mqtt Gateway component mixins

When you add the **GoogleMqttGateway** component to **GcpMqttDevice**, two types of mixins are added to the devices and points. This topic provides details about the added mixins.

DefaultDeviceExportMarker

This mixin is added to all the devices in the station.

Property Sheet

Default Device Export Marker (Gcp Device Export Marker)

Enabled	<input checked="" type="checkbox"/> true
Attached	<input checked="" type="checkbox"/> true
Attached Status	Attached to the Gateway.
Subscribed Topics	Gcp Subscribed Device Topics
Device Name	AHU_1
Device State Schemas Folder	Device State Schemas Folder
Point Telemetry Schemas Folder	Point Schemas Folder
Make	
Model	
Firmware Version	
Serial Number	
Last Config	11-Mar-2021 05:35 AM EST
Category	device.state.com
Level	400
Telemetry Trigger	4 minutes {Sun Mon Tue Wed Thu Fri Sat}

Property	Value	Description
Enabled	true (default) or false	Indicates if the device is enabled for publishing data or receiving messages from GCP or not.
Attached	true (default) or false	Indicates if the device is attached to the gateway or not.
Attached Status	read-only	Indicates the status when the device is attached to the gateway.
Subscribed Topics	additional properties	See "SubscribedTopics" section below.
Device Name	default (default) editable text field	Provides a custom device name for MQTT.
Make	text	Configurable property.
Model	Numeric value	Configurable property.
Firmware Version	text	Configurable property.
Last Config	timestamp	Reports the time when the last config message was received from GCP.
Category	text	Configurable property.
Level	numeric value	Is updated by the Config message response. Configurable property.

Property	Value	Description
PointSchemasFolder	container	Contains point telemetry schemas for the points that are available under each device. Folders are organized as device subfolders for each individual point telemetry schema, using the naming pattern "point-TelemetrySchema_{subfolder name}", where the default subfolder name is point-TelemetrySchema. These schemas are generated once the station is started and the device is enabled.
DeviceStateSchemasFolder	container	Contains device state schemas for the points that are available under subfolders of the points folder of a device. Each subfolder has a device state schema with the naming pattern "deviceStateSchema_{subfolder name}". These schemas have all the same properties as the deviceStateSchema. The default one is deviceStateSchema. These schemas are generated once the station gets started and device is enabled.


Subscribed Topics


The following contains the GCP topic details that the gateway uses to subscribe when the device is attached to the gateway. The following topics are already created:

Property	Description
Errors	<p>For the device, the errors topic for each device is subscribed by this component. It contains the following properties:</p> <p>Enabled: The topic is subscribed only if enabled.</p> <p>Topic: Indicates the topic name subscribed by the gateway when the device is attached.</p> <p>Received Message: Specifies the error message received from GCP.</p> <p>Last Received Time: Indicates the time when the error message is received.</p>
Commands	<p>For the device, the commands topic for each device is subscribed by this component. It contains the following properties:</p> <p>Enabled: The topic is subscribed only if enabled.</p> <p>Topic: Indicates the topic name subscribed by the gateway when the device is attached.</p> <p>Received Message: Indicates the message that is received while sending a command from GCP.</p> <p>Last Received Time: Indicates the time when the command message is received.</p>
Config	<p>For the device, the config topic for each device is subscribed by this element. It contains the following properties:</p> <p>Enabled: The topic will be subscribed only if enabled.</p> <p>Topic: Indicates the topic name subscribed by the gateway when the device is attached.</p> <p>Received Message: Indicates the message that is received as a config message from GCP.</p> <p>Last Received Time: Indicates the time when the config message is received.</p> <p>When the gateway receives a device config message, the gateway sets the point mixin's Enabled to <code>true</code> only if the point is part of the config message from GCP in "pointset." If not, the point mixin's Enabled is set to <code>false</code> and the point telemetry is not sent for the point in the subsequent telemetry message.</p>


DefaultPointExportMarker


This mixin is added to all the points in the station.



Default Point Export Marker (Gcp Point Export Marker)


Enabled

☒ true


Source


Category


Level

Property	Value	Description
Enabled	true (default) or false	Specifies if the point is enabled for publishing data or updating values from GCP or not.
Source	read-only	Configurable property
Category	read-only	Configurable property
Level	numeric value	Configurable property

GCP IoT registry messages

There are two types of messages that can be sent from the GCP IoT registry.

Figure 1 GCP IoT registry messages



1. **Config:** On the GCP IoT page, select **UPDATE CONFIG** to open a window that is used to send gateway config data to the gateway over the `"/devices/{GATEWAY-ID}/config"` topic. **Last Config** property of the gateway component in Niagara is set to `current time` if the config message is empty and set to `time sent` in **timestamp** property of the config message for the valid schema.

NOTE: Config messages follow the strict schema.

2. **Command:** You can send a command by clicking **SEND COMMAND**.

GCP IoT device messages

There are two types of messages that can be sent from the GCP IoT devices.

Figure 2 GCP IoT device messages



1. **Config:** On the GCP IoT page, select **UPDATE CONFIG** to open a window that is used to send device config data to the gateway over the `"/devices/{DEVICE-ID}/config"` topic. Every time a config message is received, the `publishDevcieState` action is called.
 - **Last Config** property is present in **DeviceExportMarker** of the device in Niagara. It is set to `current time` if the config message is empty and set to `time sent` in the **timestamp** property of the config message for the valid schema.
 - **Level** property of **DeviceExportMarker** for the device is set to `min_loglevel` value.
 - **DeviceExportMarker→Subscribed Topics→Config→Last Received Messages** for the device is updated with the current time when config message is received.
 - **DeviceExportMarker→Subscribed Topics→Config→Received Messages** for the device is updated with the the message sent.
 - **Network (containing the device)→Tuning Policies→Default Policy→Min Write Time** is set to `sample_limit_sec` value.
 - **GoogleMqttGateway→Telemetry Trigger→Trigger Mode→Interval** is set to `sample_rate_sec` value.
 - The points are set to their corresponding `fix_value` value if its value is valid for the point type.

For example:

```
{
  "version": 1,
  "timestamp": "2018-02-17T16:24:28.364Z",
  "system": {
    "min_loglevel": 500
  },
  "pointset": {
    "sample_limit_sec": 60,
    "sample_rate_sec": 70,
    "points": {
      "NumericWritable": {
```

```
"fix_value": 23
}
}
}
```

NOTE: Config messages follow the strict schema.

2. **Command:** You can send a command by clicking **SEND COMMAND**.

- **DeviceExportMarker→Subscribed Topics→Config→Last Received Time** for the device is updated with the current time when the command message is received.
- **DeviceExportMarker→Subscribed Topics→Config→Received Messages** for the device is updated with the message sent.

Chapter 3 About gateway actions

Topics covered in this chapter

◆ Gateway actions

This section describes the gateway actions on the **GoogleMqttGateway** component.

Gateway actions for the MQTT integration with GCP include:

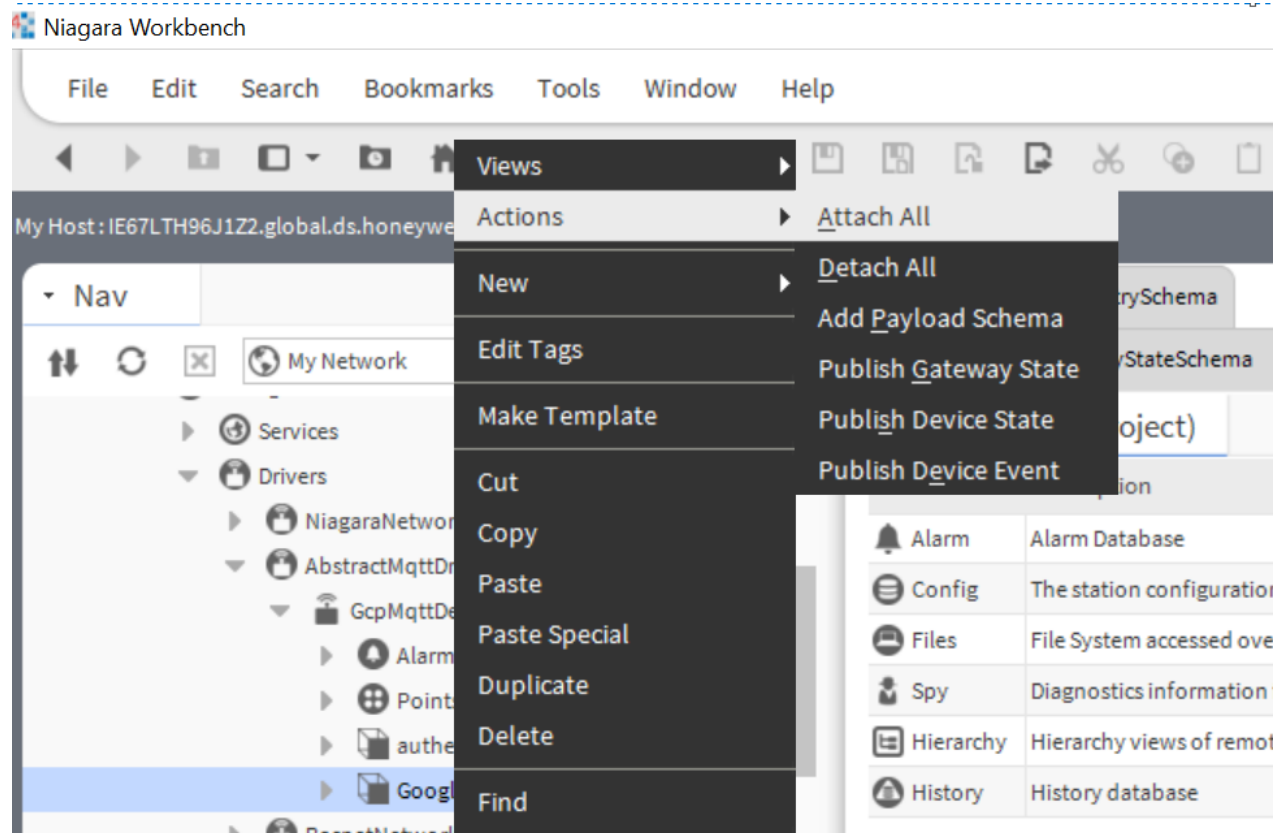
- Add Payload Schema
- Attach All
- Detach All
- Publish Gateway State
- Publish Device State
- Publish Device Event

Gateway actions

This section gives you detailed information about the gateway actions on the **GoogleMqttGateway** component.

Actions

To invoke the actions, right-click **GoogleMqttGateway**, select **Actions** and click the appropriate action.



Add Payload Schema

This actions adds required payload schemas to the mixins, which adds JSON schemas to the gateway state, device state and point telemetry. This action is automated at the start of station and required before attaching the device or publishing a message to GCP.

Attach All

The GCP gateway attaches BMS devices to the gateway in GCP as non-gateway devices. If a device fails to attach to the gateway, the information in the **Status** property on the **Device Mixin** component is updated.

Detach All

The GCP gateway detaches BMS devices from the gateway in GCP. If a device fails to detach from the gateway, the information in the **Status** property on the **Device Mixin** component is updated.

Publish Gateway State

The GCP gateway publishes the gateway state from Niagara using the Gateway State UDMI schema.

The gateway state metadata consist of the following information:

- **Version:** Use the latest UDMI Version.
- **Timestamp:** Current timestamp.
- **Make_Model:** Configurable property on the gateway.
- **Last Config:** Timestamp of the last received gateway Config message.
- **Operational:** Configurable property on the gateway.
- **Message:** Configurable property on the gateway for sending message to the cloud.
- **Category:** Defaults to `device.state.com`. Configurable property on the gateway.
- **Level:** Defaults to 600. Configurable property on the gateway.
- **Error_ids:** Devices with attach error.

You can retrieve gateway state messages in GCP console by using the subscriptions and pull command:

For example: `gcloud pubsub subscriptions pull --auto-ack projects/sylvan-replica-301709/subscriptions/state-sub --limit 20`

Gateway Data:

```
{
  "system": {
    "software": {
      "firmware": ""
    },
    "operational": true,
    "serial_no": "",
    "hardware": {
      "MQTT": {
        "model": "",
        "make": ""
      },
      "last_config": "2018-11-23T11:29:00.004-0500",
      "status": {
        "level": 600,
        "message": "",
        "category": "device.state.com",
        "timestamp": "2022-08-01T16:33:41.324-0400"
      },
      "version": "1.3.14",
      "gateway": {
        "devices": [
          {
            "AHU-4": {
              "status": {
                "level": 600,
                "message": "Error attaching device",
                "category": "device.state.com",
                "timestamp": "2022-08-01T16:33:41.834-0400"
              }
            }
          }
        ]
      },
      "timestamp": "2022-08-01T16:33:41.324-0400"
    }
  }
}
```

Publish Device State

GCP gateway publishes the device state for all the BMS devices from Niagara using the Device State UDMI schema.

Device state metadata consist of the following information:

- **Version:** Use the latest UDMI Version.
- **Timestamp:** Current timestamp.
- **Make_Model:** Configurable property on the device added through mixin.
- **Last Config:** Timestamp of the last received Config message.

- **Operational:** This field in JSON represents the current state of the device in Niagara. The Operational value is `true` if the device is with `ok` status. If the device is in fault/down state, the operational value is set to `false` and "statuses" will be updated with the fault cause..
- **Message:** fault cause value in case device is in fault/error status.
- **Category:** category of the status. Default is "device.state.com". Configurable property..

You can retrieve device state messages in GCP console by using the subscriptions and pull command :

For example: `gcloud pubsub subscriptions pull -auto-ack projects/sylvan-replica-301709/subscriptions/state-sub -limit 20`

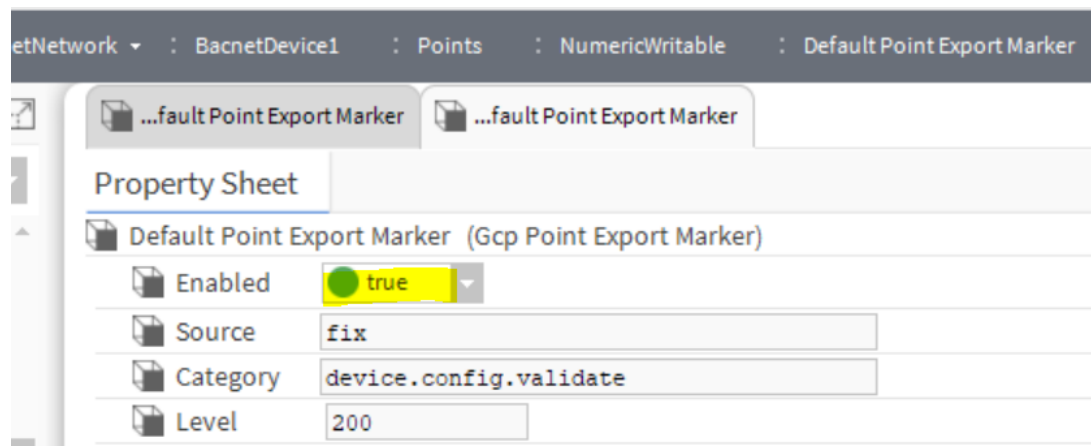
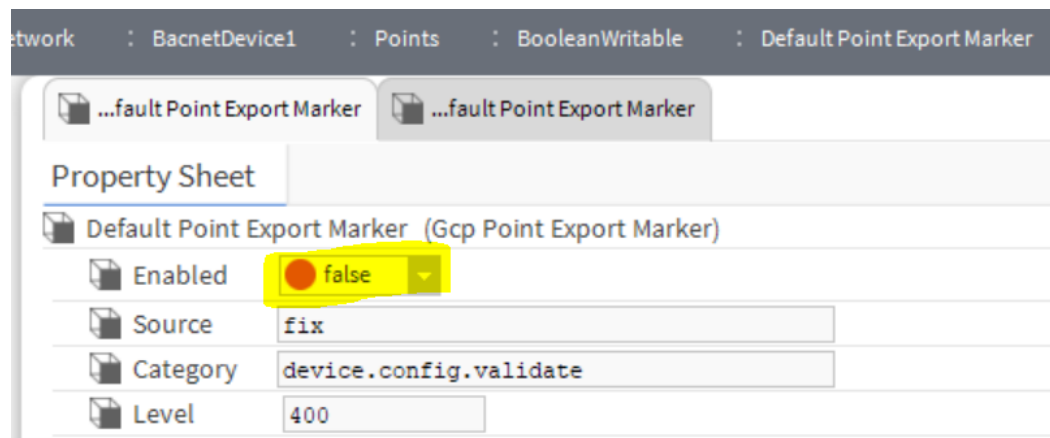
BacnetDevice1 Data:

```
{
  "system": {
    "software": {
      "firmware": "",
      "operational": false,
      "serial_no": ""
    },
    "hardware": {
      "model": "",
      "make": ""
    },
    "last_config": "2018-02-15T11:35:00.004-0500",
    "status": {
      "level": 600,
      "message": "{fault}",
      "category": "device.state.com",
      "timestamp": "1969-12-31T19:00:00.000-05:00"
    },
    "pointset": {
      "points": {
        "point_41": {
          "units": "NULL",
          "status": {
            "level": 800,
            "message": "{fault, stale}@def",
            "category": "device.config.validate",
            "timestamp": "2022-08-01T16:34:20.080-04:00"
          }
        }
      }
    },
    "version": "1.3.14",
    "timestamp": "2022-08-01T16:34:19.571-0400"
  }
}
```

Publish Telemetry State

The GCP gateway publishes point data for all the BMS devices from Niagara using the Pointset UDMI schema.

There is a mixin for points with a boolean property to indicate if the point should be included in the telemetry data. This Boolean property is configurable.



You can retrieve point telemetry messages in GCP console by using the subscriptions and pull command :

For example: `gcloud pubsub subscriptions pull -auto-ack projects/sylvan-replica-301709/subscriptions/events-sub -limit 20`

Telemetry Data

```
{"version":1,"timestamp":"2021-01-28  
18:05:45.664+0530","points":{"NumericWritable":{"present_value":66}}} | 2004867416053967 | |  
deviceId=BacnetDevice1
```

Glossary

Google Cloud Platform	<p>A suite of products and services that include:</p> <ul style="list-style-type: none">• Google Compute Engine• Google Container Engine• Google BigQuery• Google Cloud Datastore• Google Cloud DNS <p>These are the technologies that Google uses in their own products and services.</p>
Device	<p>A physical box running Niagara and CloudLink for connectivity to Niagara Cloud Suite services. For example, this could be a JACE, a Niagara Supervisor, or a virtual machine running Niagara.</p>
JWKS	<p>JSON Web Key Set</p> <p>It is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server.</p>
JWT	<p>JSON Web Token</p> <p>It is an open standard that provides secure authentication between the client and server.</p>
JSON	<p>JavaScript Object Notation</p>

Index

A

About gateway actions27

B

Binding and unbinding a device..... 11

C

Creating a device.....9

Creating a gateway..... 10

Creating a topic for publish and subscribe 14

D

document change log5

G

GCP IoT device messages25

Google Gateway 15

GoogleMqttGateway
Gateway actions.....27

I

IoT registry messages25

M

mixins21