

Configure the connector to be compliant with UDMI

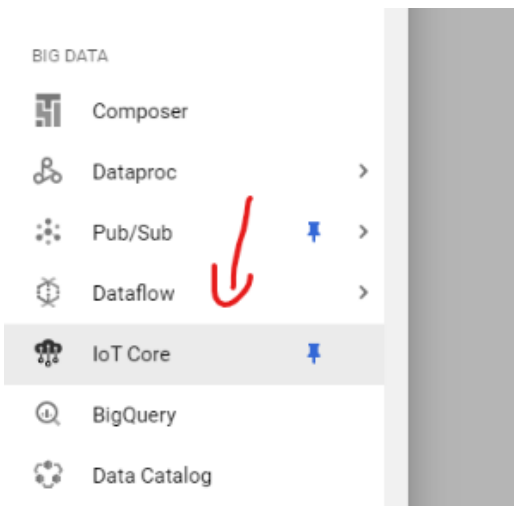
The Universal Device Management Interface (UDMI) provides high-level specifications for the management and operation of physical Internet of Things systems. This data is usually exchanged with cloud entities that can maintain "digital twins" or "shadow devices" in the cloud. It is nominally used with Google's Cloud IoT Core, and as an architecture, it can be applied to any data set or hosting setup. In addition, the architecture provides provisions for basic telemetry ingestion, such as the flow of data points from IoT devices.

In this tutorial we'll see step by step how to implement UDMI using the connectors Framework.

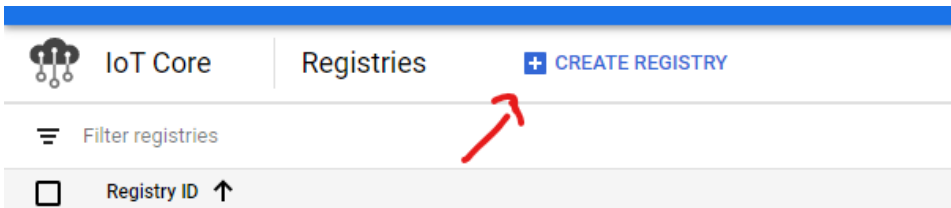
Cloud Setup

Create the registry

1. Go to the GCP console from the left side navigation pane find the **Big Data** section then choose **IoT Core** service.



2. Create a device registry.



3. Give your registry a name and select the most suited region for you.

IoT Core

← Create a registry

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
udmi-registry

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

Region
europe-west1

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

4. On the PubSub section create a new topic where events will be published by IoT Core devices.

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

Create a topic

Add a topic to Pub/Sub so that you can use it in your device registry.

Topic ID *
udmi-telemetry

Topic name: projects/btbiotcore/topics/udmi-telemetry

Encryption

☒ Google-managed key
No configuration required

☐ Customer-managed key
Manage via Google Cloud Key Management Service

CANCEL CREATE TOPIC

5. Add another topic for device state. Click on **show advanced config**

Device state topic (optional)

Device state data will be published to your selected topic on a best-effort basis, as well as to the default MQTT state topic (if your devices use MQTT protocol). [Learn more](#)

Select a Cloud Pub/Sub topic
projects/btbiotcore/topics/udmi-state ▼

6. Then hit **create**

You can add more certificates after you've created this registry.

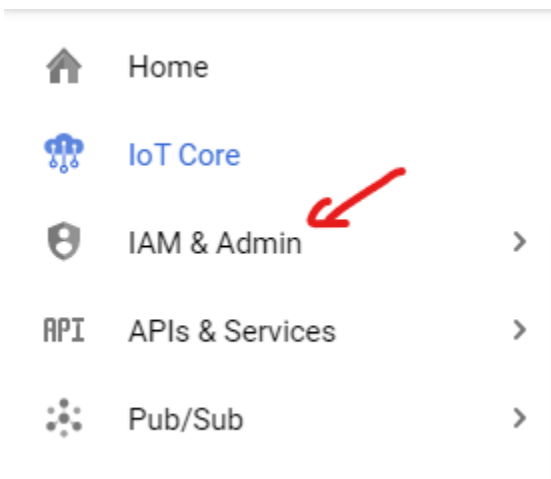
^ **HIDE ADVANCED OPTIONS**

CREATE

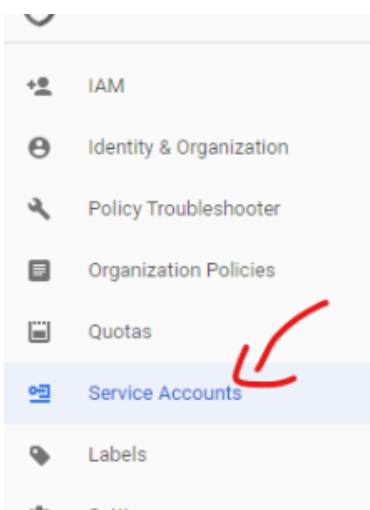
CANCEL

Create a service account

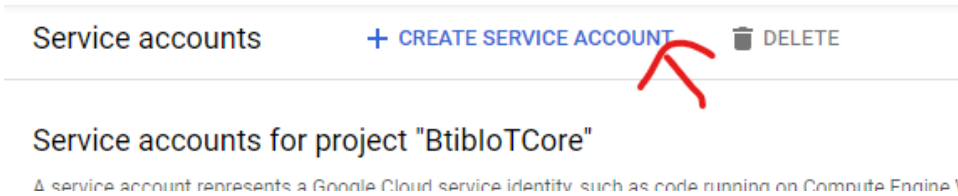
1. From the left navigation bar select **IAM & Admin**



2. Then **Service Accounts**



3. And hit create

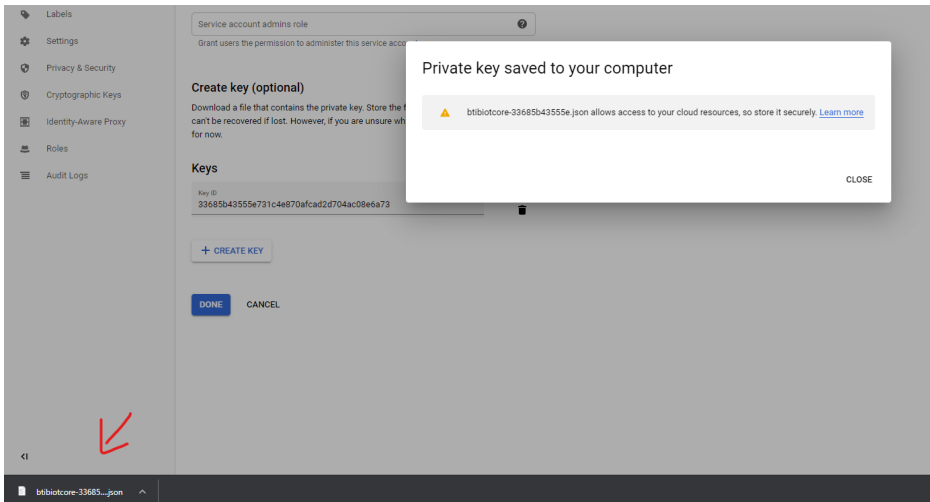


4. Then give your service account a name and a description.

5. Choose the Cloud IoT Admin (you can add conditions to give permission on a specific resources recommended). and click on Continue.

6. Create a key and choose json format then hit Create.

7. A json file will be downloaded containing the access key for this service account save it we will need it later.



8. Type this commands to generate a public/private keys.

```
openssl req -x509 -newkey rsa:2048 -keyout rsa_private.pem -nodes -days 36500 -out rsa_cert.pem -
subj "/CN=unused"
```

And

```
openssl pkcs8 -topk8 -inform PEM -outform DER -in rsa_private.pem -nocrypt > rsa_private_pkcs8
```

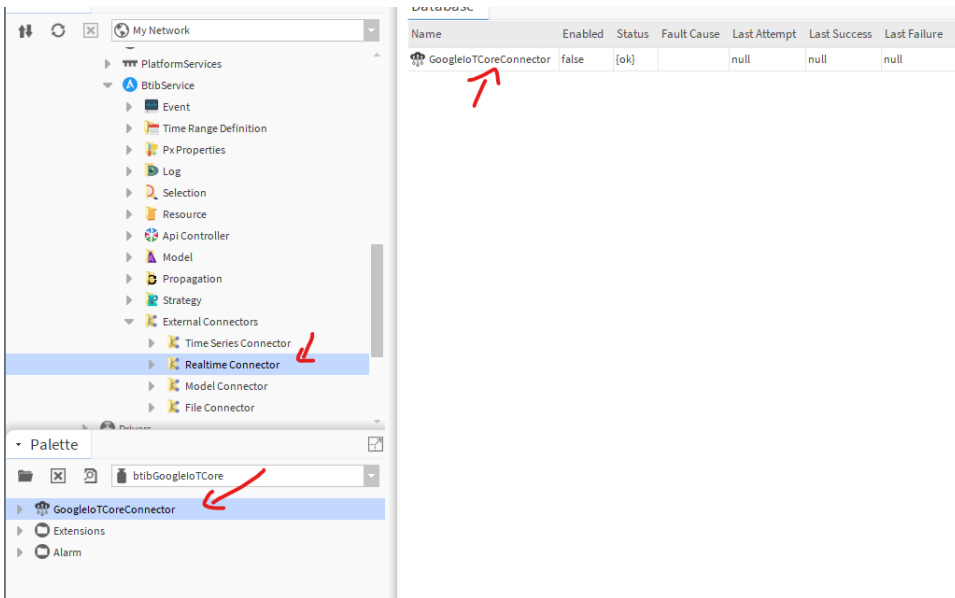
These commands will generate 3 files we will need 2 of them:

- **rsa_cert.pem.**
- **rsa_private_pkcs8.**

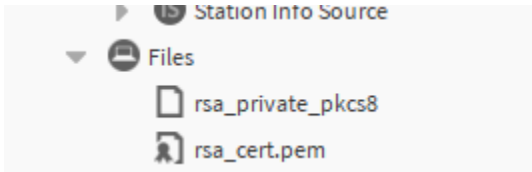
Connector Setup

Connection setup

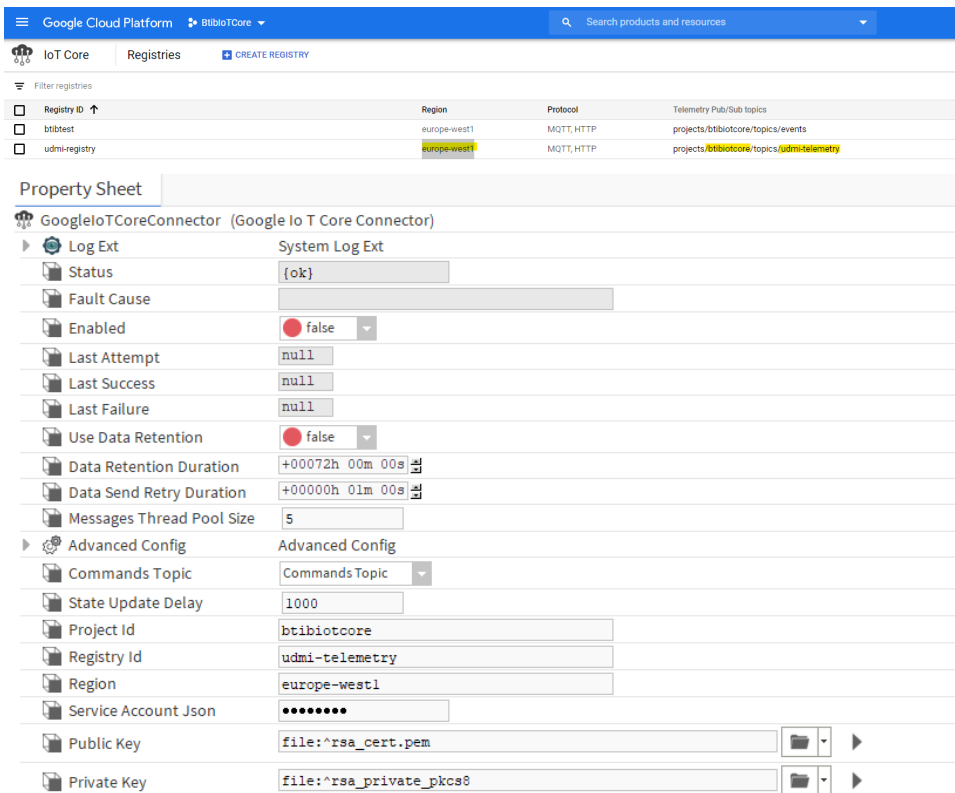
1. Go to the GoogleIotCore palette and drag and drop the connector to **BtibService** **ExterbalConnectors** **RealtimeConnector**



2. Copy the keys to your station









3. Fill the connection information



4. Enable the connector you should see the last attempt and last success values populated.

Property Sheet

 GoogleIoTCoreConnector (Google IoT Core Connector)

▶  Log Ext	System Log Ext
 Status	{ok}
 Fault Cause	
 Enabled	● true
 Last Attempt	18-May-2020 11:00 AM CEST
 Last Success	18-May-2020 11:00 AM CEST

UDMI Setup

UDMI specification uses custom format for messages exchanged between the cloud and the device

Device Telemetry schema:

- Schema <https://github.com/faucetsdn/daq/blob/master/schemas/udmi/pointset.json>
- Example:

```
{
  "version": 1,
  "timestamp": "2018-08-26T21:39:29.364Z",
  "points": {
    "reading_value": {
      "present_value": 21.30108642578125
    }
  }
}
```

Device State:

- Schema <https://github.com/faucetsdn/daq/blob/master/schemas/udmi/state.json>
- Example:

```
{
  "version": 1,
  "timestamp": "2018-08-26T21:39:29.364Z",
  "system": {
    "make_model": "ACME Bird Trap",
    "firmware": {
      "version": "3.2a"
    },
    "last_config": "2018-08-26T21:49:29.364Z",
    "operational": true,
    "statuses": {
      "base_system": {
        "message": "Tickity Boo",
        "category": "device.state.com",
        "timestamp": "2018-08-26T21:39:30.364Z",
        "level": 600
      }
    }
  },
  "pointset": {
    "points": {
      "return_air_temperature_sensor": {
        "units": "Celsius",
        "status": {
          "message": "Invalid sample time",
          "category": "device.config.validate",
          "timestamp": "2018-08-26T21:39:28.364Z",
          "level": 800
        }
      }
    }
  }
}
```

```

    }
  }
}

```

Device Commands

- Schema <https://github.com/faucetsdn/daq/blob/master/schemas/udmi/config.json>
- Example:

```

{
  "version": 1,
  "timestamp": "2018-08-26T21:39:29.364Z",
  "system": {
    "min_loglevel": 500
  },
  "gateway": {
    "devices": {
      "AHU-123": {
        "protocol": "bacnet",
        "local_id": "327412"
      }
    }
  },
  "pointset": {
    "points": {
      "nexus_sensor": {
        "fix_value": 21.1
      }
    }
  }
}

```

1. Go to the advanced config slot on the connector.

The screenshot shows the 'Advanced Config' interface for a connector. The left sidebar contains a tree view with the following structure:

- My Network
 - Propagation
 - Strategy
 - External Connectors
 - Time Series Connector
 - Realtime Connector
 - GoogleTCoreConnector
 - Log Ext
 - Advanced Config** (selected)
 - Model Connector
 - File Connector
- Drivers
- Upps
- Station Info Source
- rchy
- ry
- fficeBtib)
- btibGoogleTCore

The main area displays the 'Property Sheet' for the 'Advanced Config' (Advanced Config) section. The properties are as follows:

Property	Value
Device Tags Destination	Device Metadata
Point Tags Destination	Not Monitored
Point Status Destination	State Topic
Point Value Destination	Events Topic
Alarm Destination	Events Topic
Default Variables	\$(timestamp) \$(pointName) \$(pointId) \$(pointTags) \$(pointStatus) \$(pointValue) \$(deviceId) \$(deviceName) \$(deviceStatus)
Custom Variables	\$()
Point Status Message Template	<pre>{ "pointId": "\${pointId}", "timestamp": "\${timestamp}", "value": \${pointValue}, "status": "\${pointStatus}" }</pre>
Point Value Message Template	<pre>{ "pointId": "\${pointId}", "timestamp": "\${timestamp}", "value": \${pointValue}, "status": "\${pointStatus}" }</pre>
Command Device Id	{json('deviceId')}
Command Point Id	{json('pointId')}
Command Action	{json('payload.action')}
Command Value	{json('payload.value')}
Command Duration	{json('payload.duration')}
Invalid Value Policy	Ignore Value

2. On the message value template put the udmi template below


```
{
  "version": 1,
  "timestamp": "${timestamp}",
  "points": {
    "${pointId)": {
      "present_value": $(pointValue)
    }
  }
}
```

 Point Value Message Template

```
{
  "version": 1,
  "timestamp": "${timestamp}",
  "points": {
    "${pointId)": {
      "present_value": $(pointValue)
    }
  }
}
```

3. On the state template use.

```
{
  "version": 1,
  "timestamp": "${timestamp}",
  "system": {
    "make_model": "${deviceName}",
    "firmware": {
      "version": "1"
    },
  },
  "statuses": {
    "base_system": {
      "message": "${deviceStatus}",
      "category": "device.state",
      "timestamp": "${timestamp}"
    }
  },
  "pointset": {
    "points": {
      "${pointId)": {
        "status": {
          "message": "${pointStatus}",
          "category": "point.state",
          "timestamp": "${timestamp}"
        }
      }
    }
  }
}
```

 Point Status Message Template

```
{
  "version": 1,
  "timestamp": "${timestamp}",
  "system": {
    "make_model": "${deviceName}",
    "firmware": {
      "version": "1"
    },
  },
  "statuses": {
```

4. For commands we use this template and [SFormat](#) to extract the json information needed.

```
{
  "version": 1,
  "pointset": {
    "points": {
      "nexus_sensor": {
        "value": 21.1
      }
    }
  }
}
```

Command Device Id	<input type="text"/>	?
Command Point Id	<code>{json('pointset.points.firstKey')}</code>	?
Command Action	<code>OVERRIDE</code>	?
Command Value	<code>{json('pointset.points.firstKeyValue.value')}</code>	?
Command Duration	<code>0</code>	?

Testing configuration

1. Create a new device and add a device ext.

The screenshot shows the Google Cloud IoT Core console. On the left, the 'Nav' pane is open, showing a tree view of the device configuration. Under 'TestDevice', 'GoogleIoTCoreDeviceExt' is selected. On the right, the 'Property Sheet' for 'GoogleIoTCoreDeviceExt' is displayed. It includes fields for 'Status' (set to '{ok}'), 'Fault Cause' (empty), 'Enabled' (set to 'true'), and 'Connector' (set to 'GoogleIoTCoreConnector').

2. On the device registry verify that the device provisioned successfully.

Registry ID: udmi-registry

europa-west1

Devices are things that connect to the internet directly or through a gateway. [Learn](#)

Enter exact device ID

☐ Device ID

☐ TestDevice_p0RzKdjc3YrALoHVaiQqF

[Cloud IoT Core documentation](#)

3. Create a point and add a point ext.

The screenshot shows the Google IoT Core console interface. On the left, the 'Nav' pane displays a tree view of resources. Under 'My Network', the 'Points' folder is expanded, and 'GoogleIoTCorePointExt' is selected. A red arrow points to this item. Below the 'Nav' pane is the 'Palette' pane, which also shows 'GoogleIoTCorePointExt' selected, with another red arrow pointing to it. On the right, the 'Property Sheet' for 'GoogleIoTCorePointExt' is displayed. It shows various configuration options: 'Status' is '[ok]', 'Fault Cause' is empty, 'Enabled' is 'true', 'Device Query' is 'slot:...!neql:traverse n:parentDevice->', 'Trigger On Value Change Only' is 'true', and 'Can Write' is 'true'.

4. To view messages we should create a subscription to the telemetry and state topics. On the Google console go to the **pubsub service subscription create**.

The screenshot shows the Google Cloud Platform console. The top navigation bar includes the 'Google Cloud Platform' logo and the 'BtlibIoTCore' project name. The left sidebar shows the 'Pub/Sub' section expanded, with 'Subscriptions' selected. A red arrow points to the 'Subscriptions' link. The main content area displays the 'Subscriptions' page. At the top, there is a 'CREATE SUBSCRIPTION' button with a red arrow pointing to it, and a 'DELETE' button. Below this is a table with the following data:

Subscription ID	Delivery type	Topic name
events-sub	Pull	projects/btlibiotcore/topics/events
state-sub	Pull	projects/btlibiotcore/topics/state

5. Give the subscription a name then choose the telemetry topic and click on Create.

Pub/Sub

Topics

Subscriptions

Snapshots

Create subscription

A subscription directs messages on a topic to subscribers. Messages can be pushed to subscribers immediately, or subscribers can pull messages as needed.

Subscription ID *

udmi-telemetry-sub

Subscription name: projects/btbiotcore/subscriptions/udmi-telemetry-sub

Select a Cloud Pub/Sub topic *

projects/btbiotcore/topics/udmi-telemetry

Delivery type

☒ Pull
 ☐ Push

Subscription expiration

☐ Expire after this many days of inactivity (up to 365)

A subscription is inactive if there is no subscriber activity such as open connections, active pulls, or successful pushes.

☒ Never expire

The subscription will never expire no matter the activity.

Acknowledgement deadline

Deadline time is from 10 seconds to 600 seconds

10

Seconds

Message retention duration

Duration is from 10 minutes to 7 days

Days

7

Hours

0

Minutes

0

☐ Retain acknowledged messages

When enabled, acknowledged messages are retained for the message retention duration specified above. This increases message storage fees. [Learn more](#)

Dead lettering

☐ Enable dead lettering

Subscriptions may configure a maximum number of delivery attempts. When a message cannot be delivered, it is republished to the specified dead letter topic.

CREATE

6. Do the same for the state topic subscription

Pub/Sub

Topics

Subscriptions

Snapshots

Create subscription

A subscription directs messages on a topic to subscribers. Messages can be pushed to subscribers immediately, or subscribers can pull messages as needed.

Subscription ID *

udmi-state-sub

Subscription name: projects/btbiotcore/subscriptions/udmi-state-sub

Select a Cloud Pub/Sub topic *

projects/btbiotcore/topics/udmi-state

Delivery type

☒ Pull
 ☐ Push

Subscription expiration

☐ Expire after this many days of inactivity (up to 365)

A subscription is inactive if there is no subscriber activity such as open connections, active pulls, or successful pushes.

☒ Never expire

The subscription will never expire no matter the activity.

Acknowledgement deadline

Deadline time is from 10 seconds to 600 seconds

10

Seconds

Message retention duration

Duration is from 10 minutes to 7 days

Days

7

Hours

0

Minutes

0

☐ Retain acknowledged messages

When enabled, acknowledged messages are retained for the message retention duration specified above. This increases message storage fees. [Learn more](#)

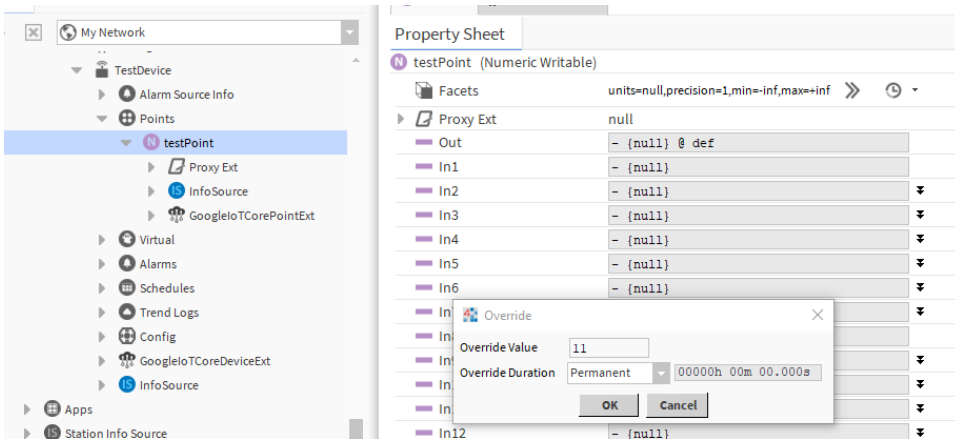
Dead lettering

☐ Enable dead lettering

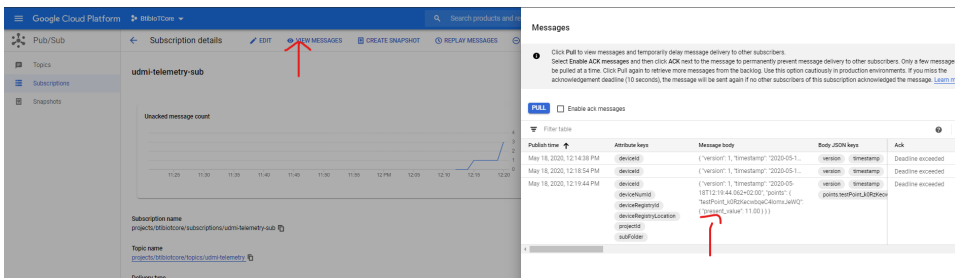
Subscriptions may configure a maximum number of delivery attempts. When a message cannot be delivered, it is republished to the specified dead letter topic.

CREATE

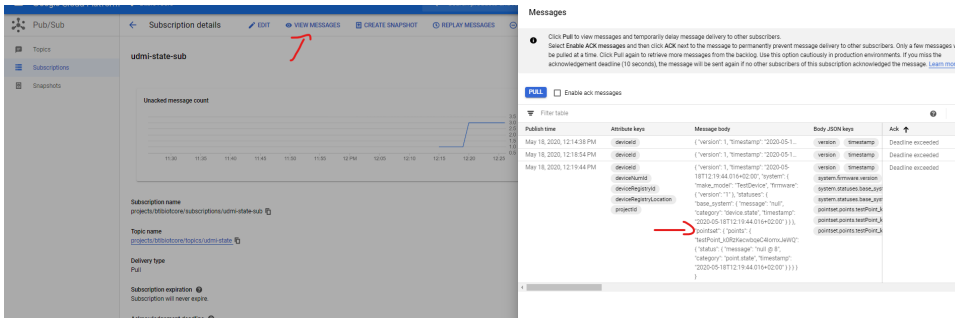
7. Now go to the Niagara point and change the value and the state.



8. On the telemetry subscription you should see your message.



9. The same for the state subscription



Congratulations you finished the tutorial for more information check the connector documentation.