How to Connect your Industrial Assets to Google Cloud using N3uron's MQTT Module

Connecting Google Cloud and N3uron platforms via the MQTT module: Overview

MQTT has become the leading standard in IoT messaging, as it is ideal for connecting to remote devices and of course, it is supported by the Google Cloud Platform IoT. In one of our previous articles, *MQTT: The Universal Messaging Protocol for Cloud Providers and IIoT Systems*, we explain exactly what MQTT is, as well as everything you need to know about its functionalities, advantages, disadvantages, and how to use it.

OT infrastructures can be connected to GCP via MQTT, allowing access to the entire ecosystem of services and message exchanges between the N3uron application and Google IoT Core. Simply put, GCP IoT Core is the service that receives and routes MQTT messages from edge devices and applications such as N3uron.

This guide explains in detail how to bi-directionally communicate your industrial assets with the Google Cloud Platform's IoT Core in a secure way using N3uron's MQTT module and thus bridge the gap between OT and IT.

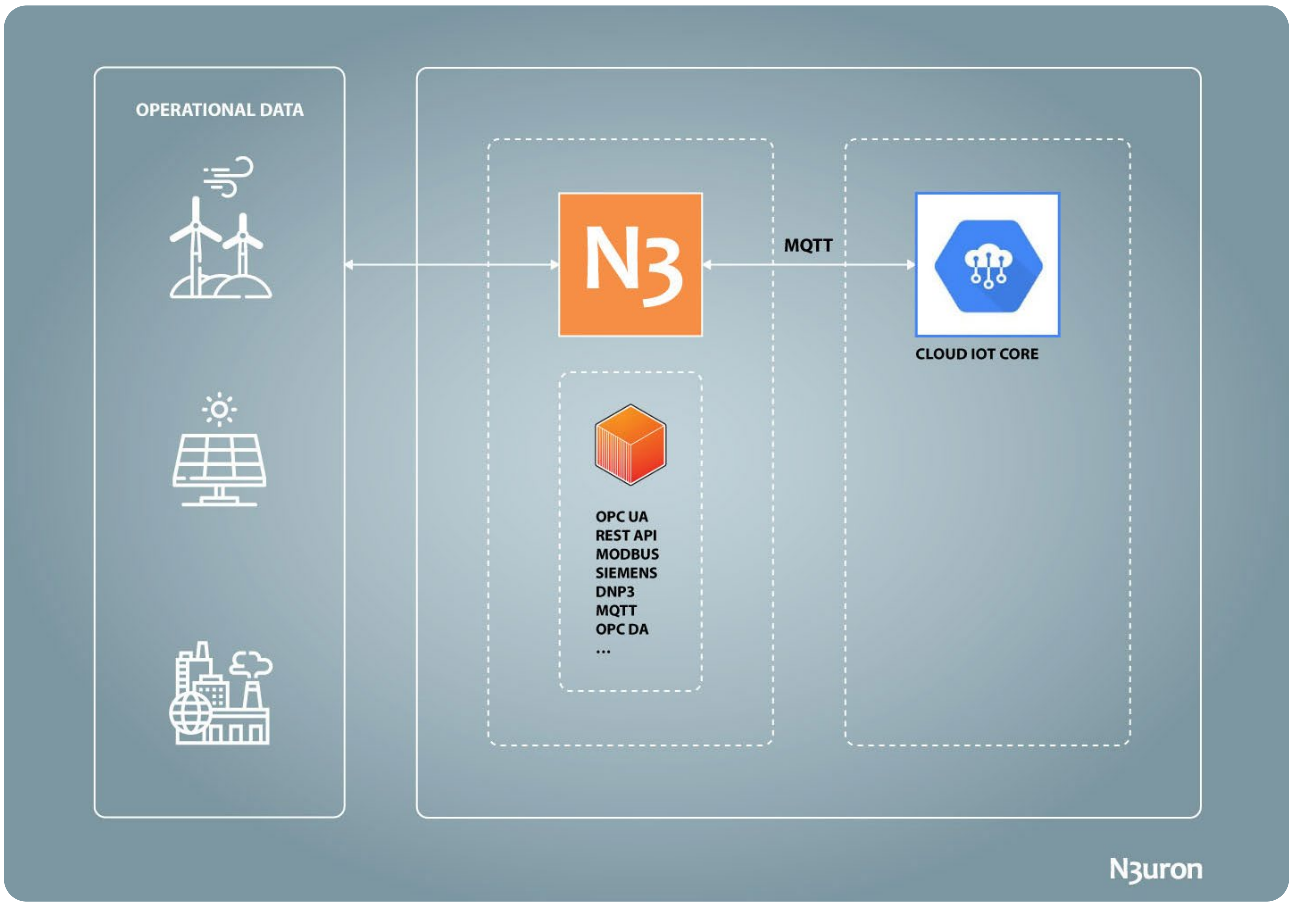


Diagram demonstrating operational data exchange between OT assets, N3uron's MQTT module, and the Google Cloud platform.

Google Cloud and N3uron Platform IoT Requirements

A Google Cloud Platform account is required. If you don't already have one, you can register and create one here: GCP Registry.

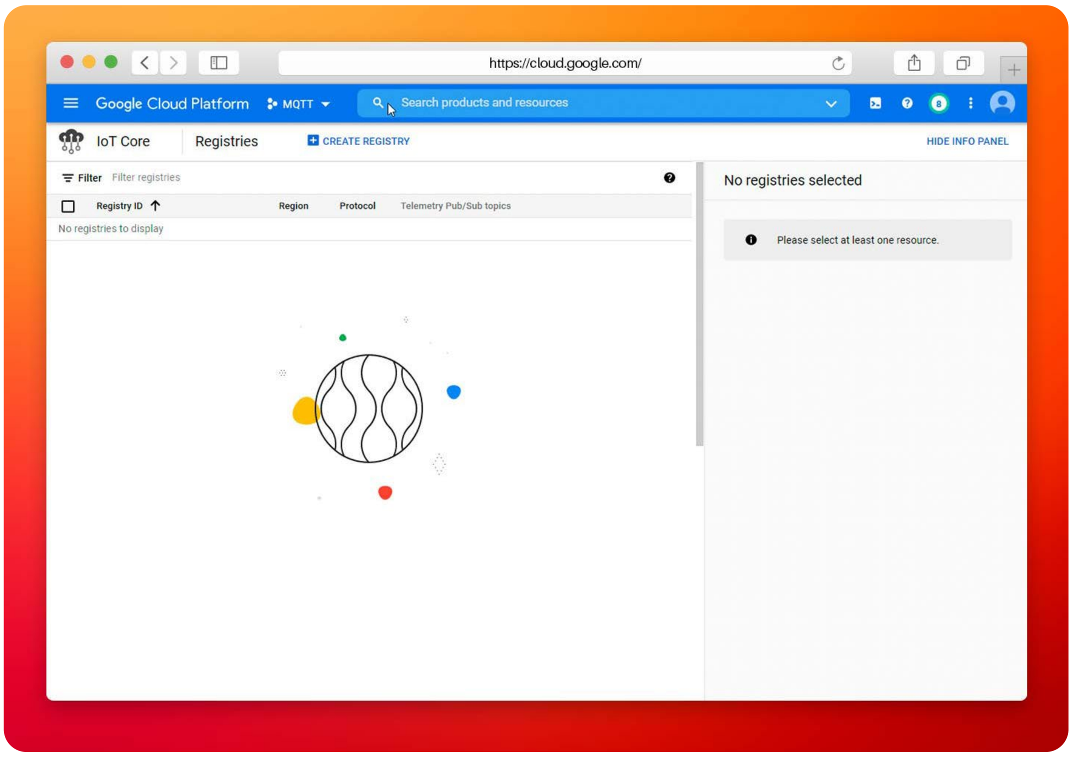
You should have also pre-installed N3uron on your system. If not, you can download it from https://n3uron.com/downloads/. If this is your first time using N3uron, you should take a look at our Quick User Guide. This guide will introduce you to a basic overview of N3uron, demonstrating its key functionalities and how to use the main modules and characteristics.

Google IoT Core Configuration: First Steps.

First, log in to Google IoT Core and open the Google Cloud Console.
Once you have logged in, you'll need to create all the the necessary resources required forto establishing athe connection to and exchanginge messages.

Log into Google IoT Core Platform and Create a Registry

– **Step 01:** First, create a new registry, as shown in the below image.



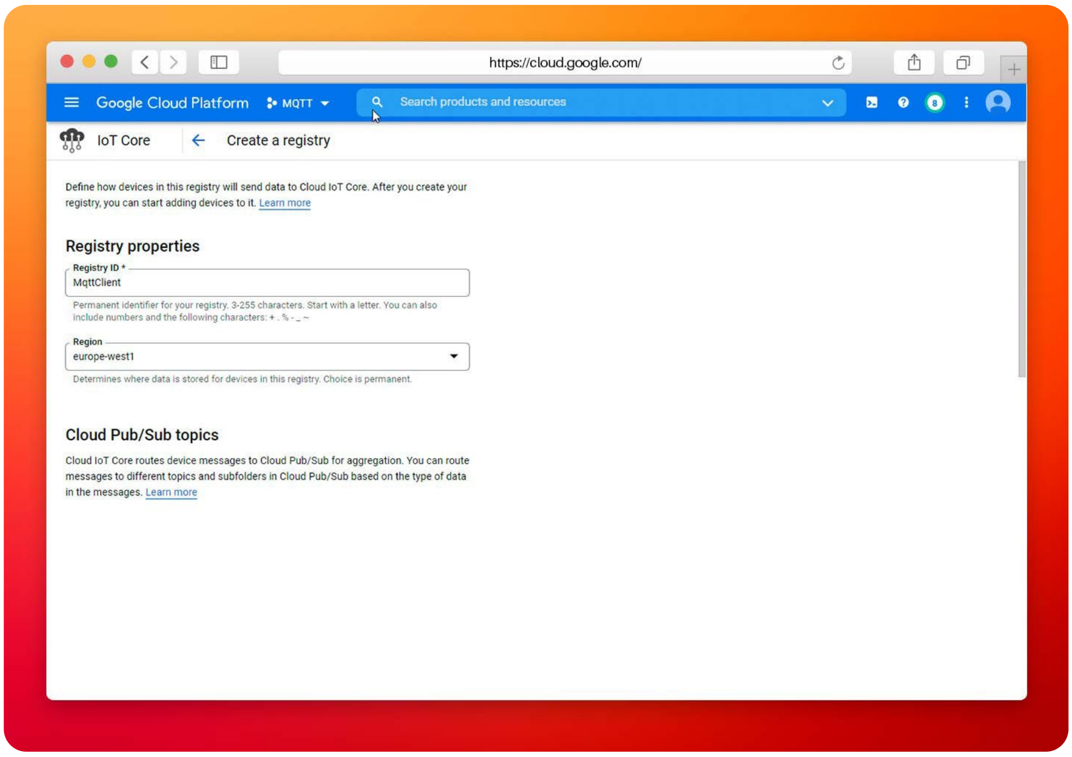
No registries to display

No registries selected

Please select at least one resource.

Screenshot displaying the Google Cloud platform IoT Core panel.

– **Step 02:** Next, fill in the required fields with the relevant characteristics fromof your registry:
Registry ID: name of the registry associated with your connection. In this case, MQTT Client.
Region: determines the location where data will be stored.This can't be modified once the registry has been created. In this example, the location is europe-west1.



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 *: MqttClient

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

Region: europe-west1

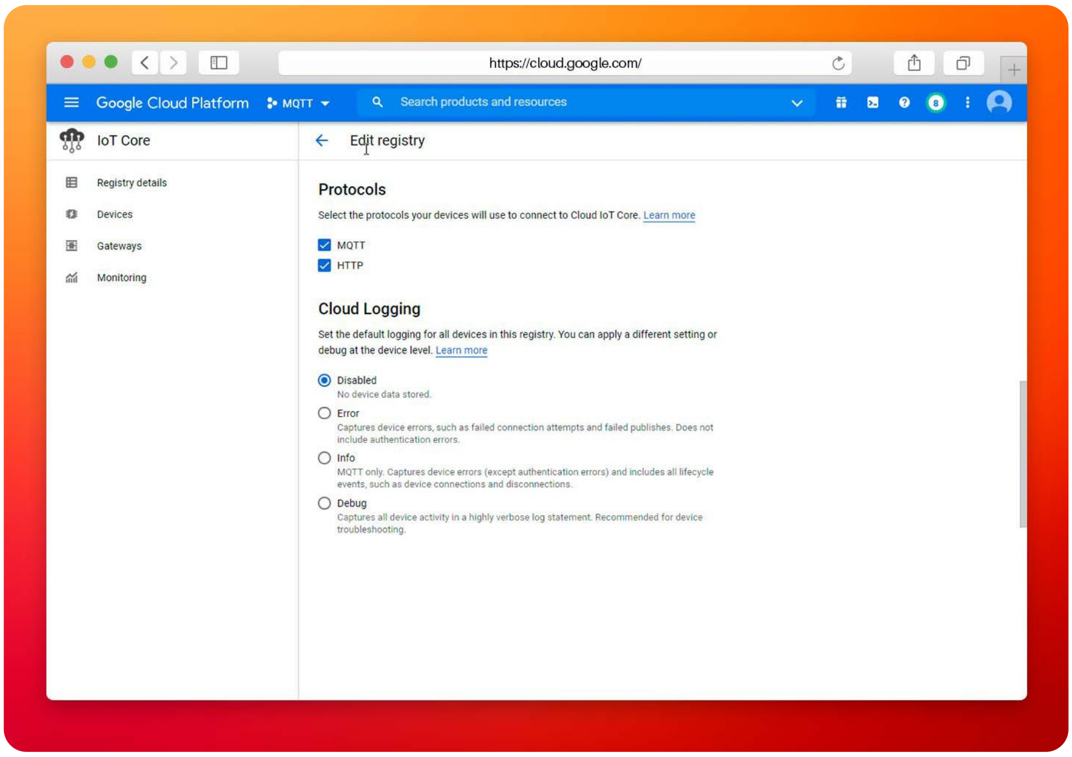
Determines where data is stored for devices in this registry. Choice is permanent.

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

Screenshot displaying the Google Cloud platform's Registry Creation panel.

- **Protocol:** It allows usersyou to choose whichthe protocol to be used by the connection will use. To In order to create a **MQTT Client**, is necessary to chooseselect the MQTT option.
- **Stackdriver Logging:** It sets the default logging parameter for thee registry.



Protocols

Select the protocols your devices will use to connect to Cloud IoT Core. Learn more

☑ MQTT
☑ HTTP

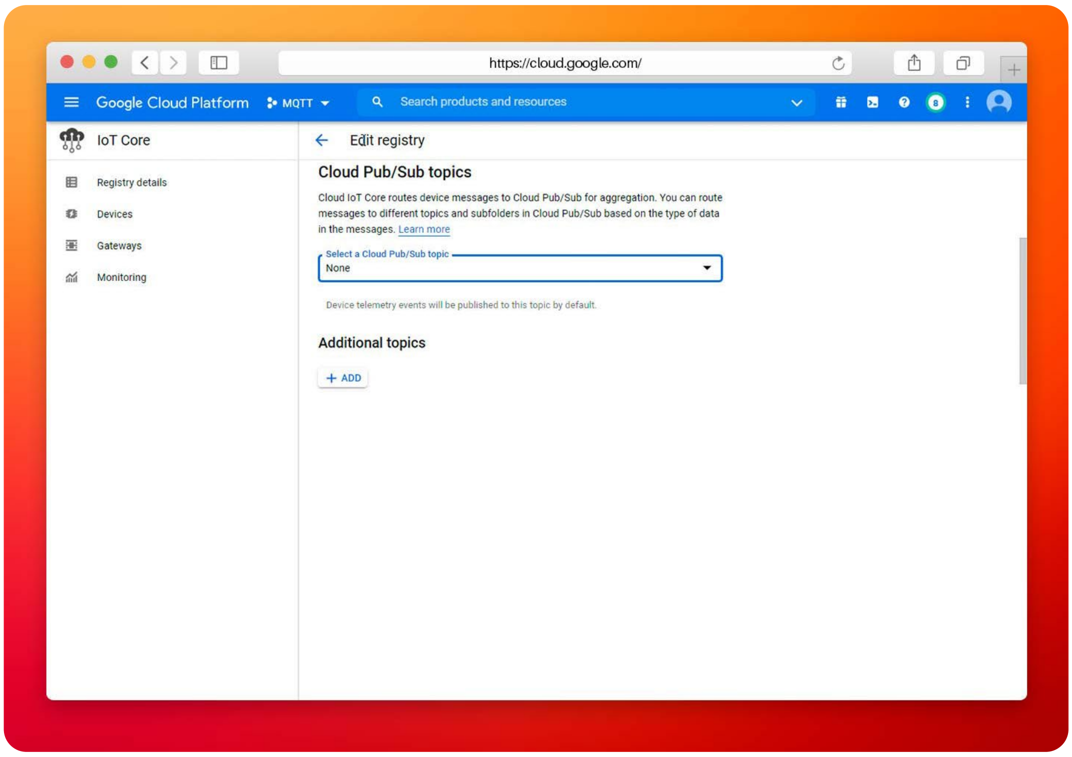
Cloud Logging

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.

Screenshot displaying the Google Cloud Platform's Registry configuration panel.

- **Client Pub/Sub Topics:** establishes the topic to publish/subscribe for telemetry and state events. This section can be updated and changed at any time after creation of the registry.



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

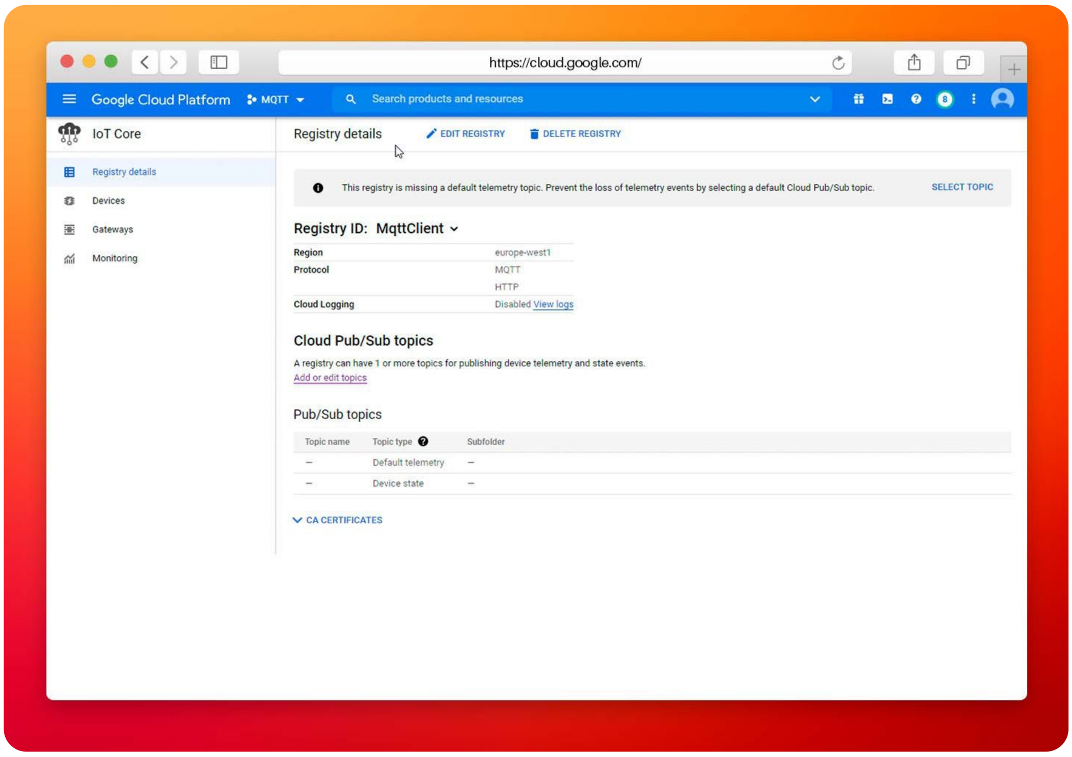
Device telemetry events will be published to this topic by default.

Additional topics

+ ADD

Screenshot displaying the "Edit Registry" panel in the Google Cloud Platform.

The following screenshot shows the defined characteristics:



Registry details

This registry is missing a default telemetry topic. Prevent the loss of telemetry events by selecting a default Cloud Pub/Sub topic.

Registry ID: MqttClient

Region	europe-west1
Protocol	MQTT HTTP
Cloud Logging	Disabled View logs

Cloud Pub/Sub topics

A registry can have 1 or more topics for publishing device telemetry and state events.

Pub/Sub topics

Topic name	Topic type	Subfolder
–	Default telemetry	–
–	Device state	–

Screenshot displaying the “Registry details” panel in the Google Cloud Platform.

At this point, the registry has already been created, so the next step is to establish one or more devices to connect to the N3uron module and exchange messages.

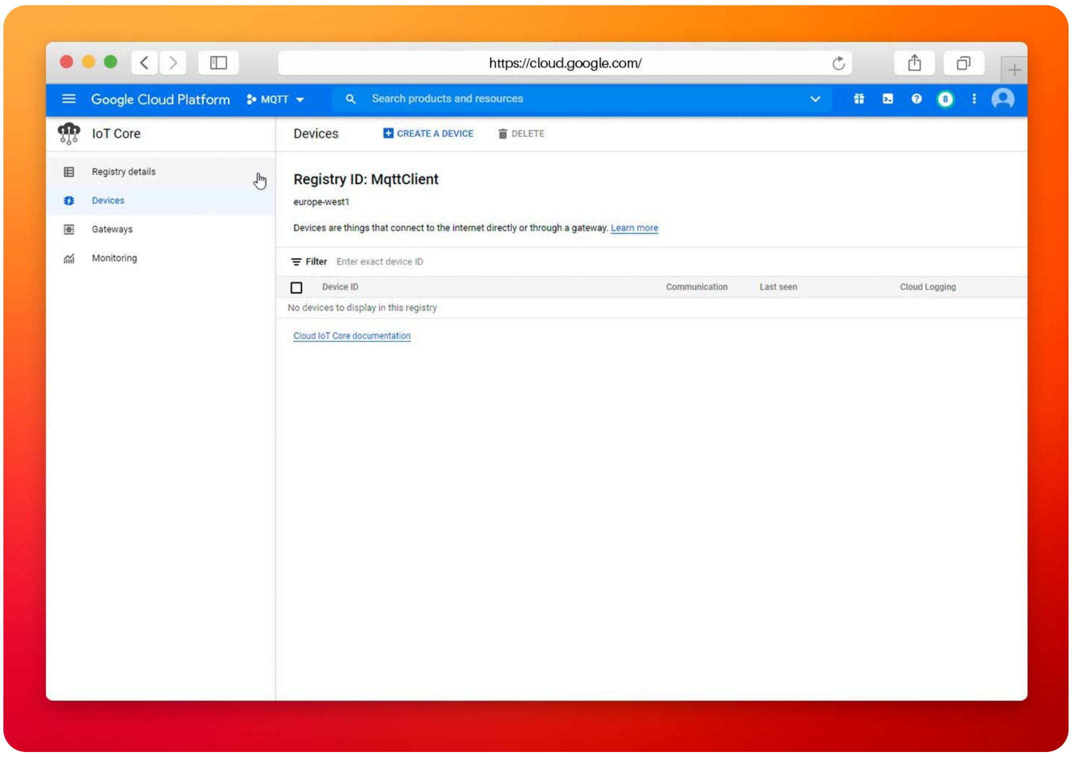
Creating a device in the Google Cloud Platform

Each device requires a private-public key to authenticate with IoT Core, which can be defined using the following OpenSSL commands:

```
Openssl genpkey -algorithm RSA -out rsa_private.pem -pkeyopt rsa_key-
gen_bits: 2048
Openssl rsa -in rsa_private.pem -pubout -out rsa_public.pem
```

This will create two files: rsa_private.pem, which contains the private key and rsa_public.pem, which contains the public key. These keys are very important when creating an **MQTT client** in N3uron, so make sure they are stored in a secured area.

The following screenshot demonstrates how to create a new device.



Devices

Registry ID: MqttClient

europe-west1

Devices are things that connect to the internet directly or through a gateway. Learn more

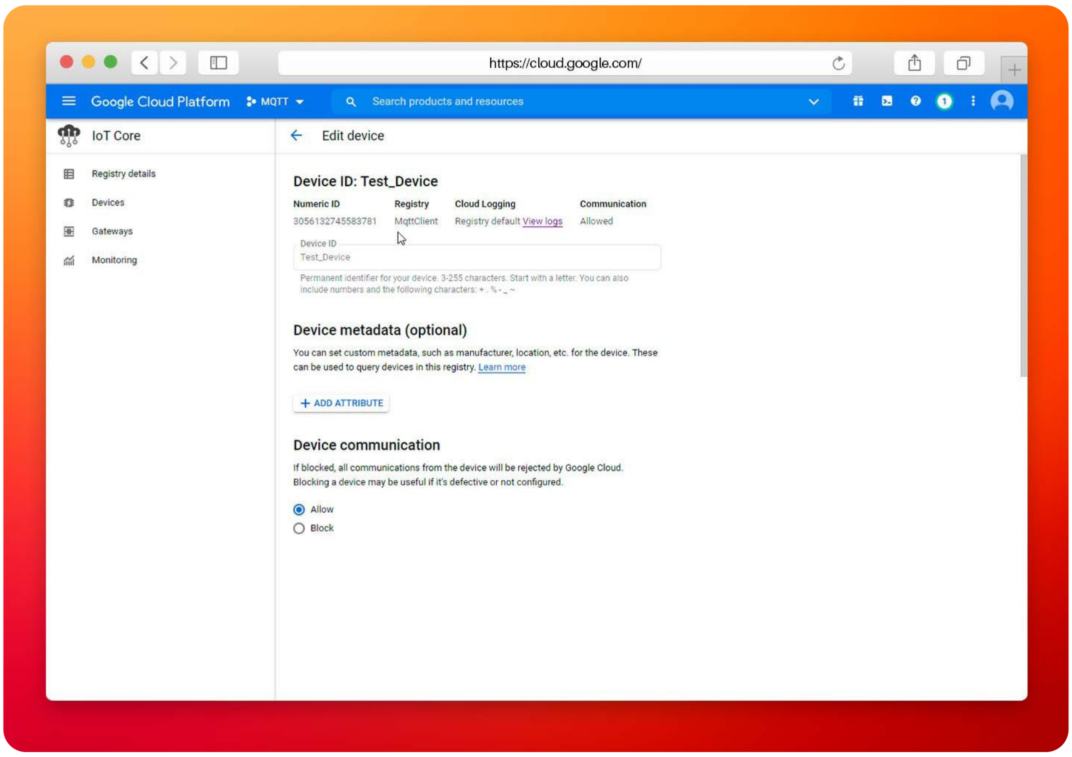
Filter Enter exact device ID

Device ID	Communication	Last seen	Cloud Logging
No devices to display in this registry			

Cloud IoT Core documentation

Screenshot displaying the “Create a Device” panel in the Google Cloud platform.

- **Step 01:** Click on **Create New Device**. This will take you to a configuration tab.
- **Step 02:** During device creation, the public key must be applied in the configuration. The parameters applied in this configuration can be observed below:
- **Device ID:** This is unique and must be equal to the **MQTT client ID** and the topic used for this device. Communication must also be enabled with the device, so there are some optional sections too. In this example, the name of the device is, Test_Device.



Edit device

Device ID: Test_Device

Numeric ID	Registry	Cloud Logging	Communication
3056132745583781	MqttClient	Registry default View logs	Allowed

Device ID: Test_Device

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

Device metadata (optional)

You can set custom metadata, such as manufacturer, location, etc. for the device. These can be used to query devices in this registry. Learn more

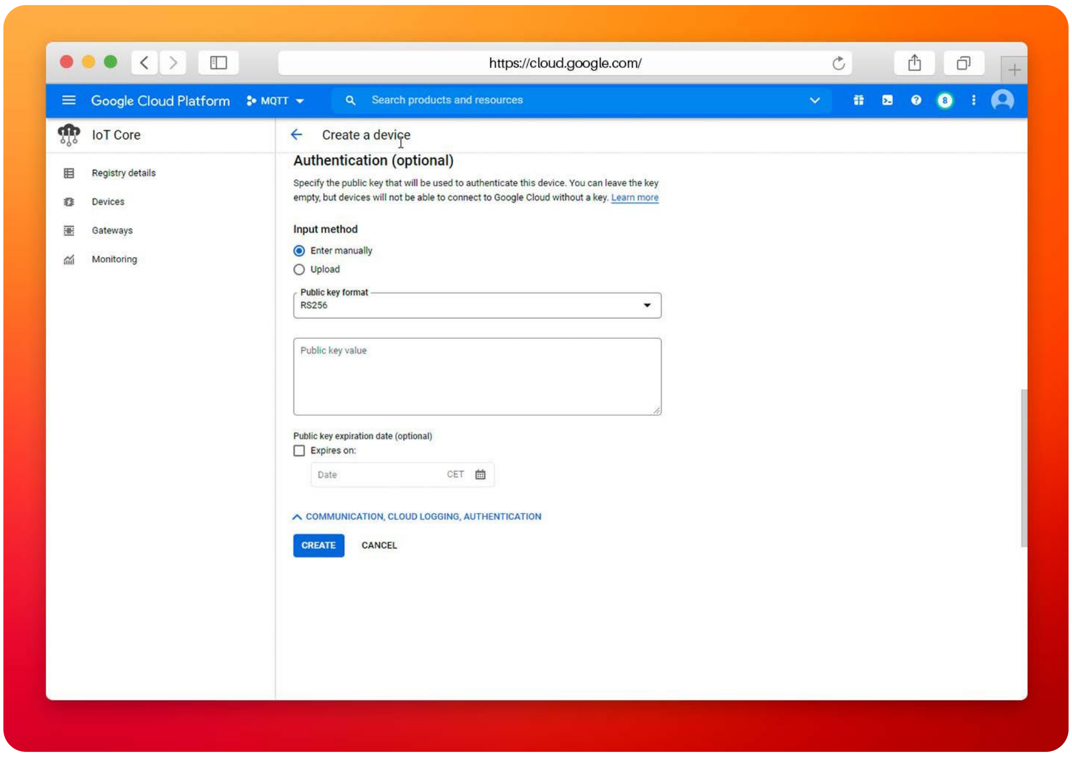
Device communication

If blocked, all communications from the device will be rejected by Google Cloud. Blocking a device may be useful if it's defective or not configured.

[✓ Allow] Block

Screenshot displaying the “Edit Device” panel in the Google Cloud platform.

– **Authentication:** determines how the key should be entered. There are 2 available options: manually enter the key name, or upload the file containing the public key. Here you should introduce the previ ously generated key. The easiest way to do this is to upload it.



Create a device

Authentication (optional)

Specify the public key that will be used to authenticate this device. You can leave the key empty, but devices will not be able to connect to Google Cloud without a key. Learn more

Input method

[✓ Enter manually]
Upload

Public key format: RS256

Public key value

Public key expiration date (optional)
Expires on:
Date CET

COMMUNICATION, CLOUD LOGGING, AUTHENTICATION

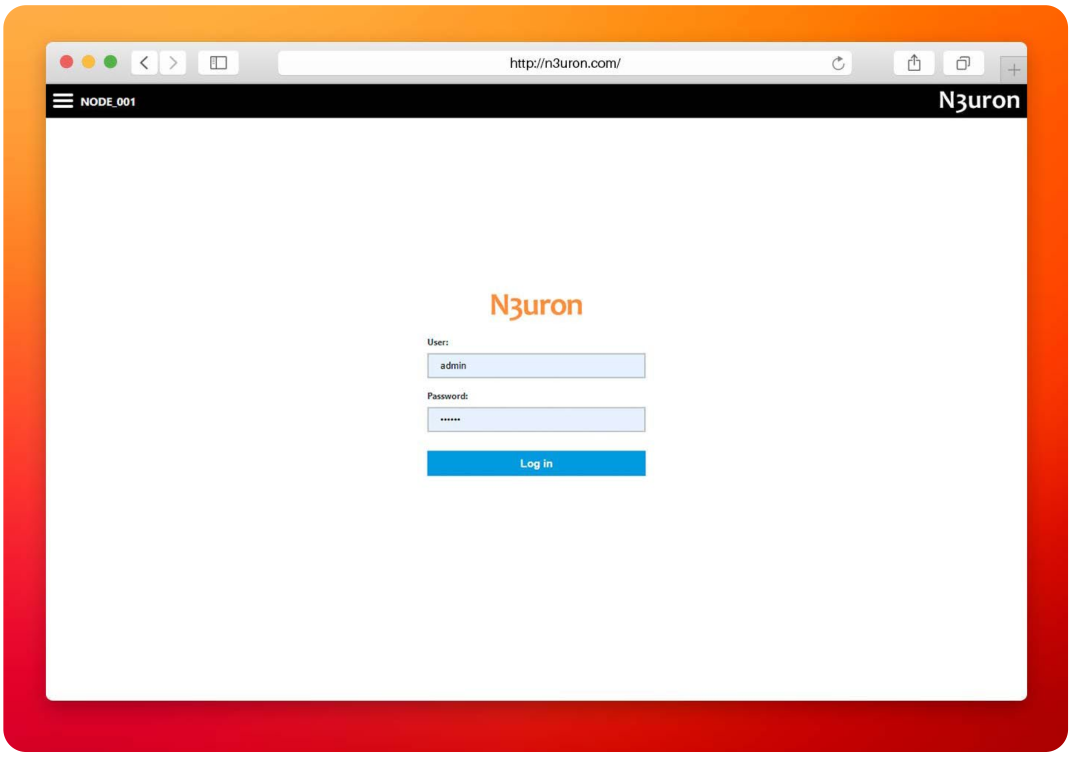
Screenshot displaying device configuration within the "Create a Device" panel in the Google Cloud Platform.

– **Step 03:** Once configuration is finished, click on **Create**, and the device will be created.

Configuring the N3uron IIoT Platform

Open your browser and log into N3uron's WebUI

Once N3uron is installed, users should open the user interface, which will take you to the below panel.



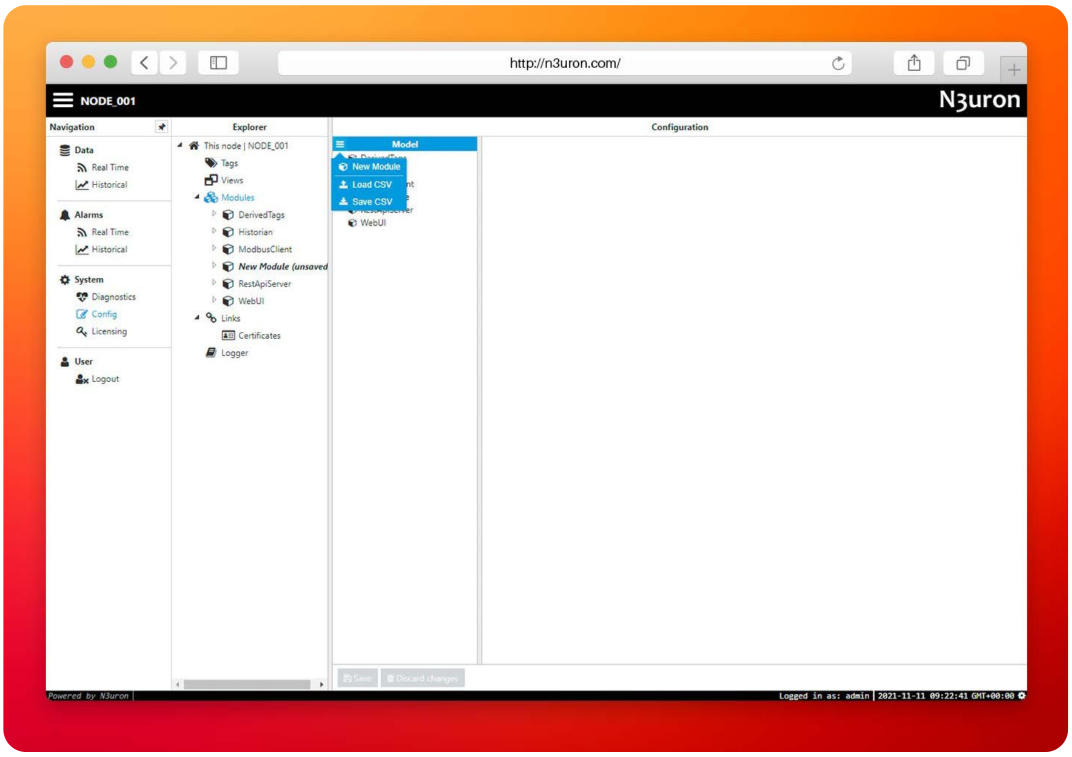
Screenshot displaying the log-in interface for N3uron's IIoT platform WebUI

If this is your first time accessing the platform, the following credentials should be entered: User=admin, Password=n3uron. You should now have gained access to the WebUI developed by N3uron and can start practicing with it.

Create an MQTT module instance

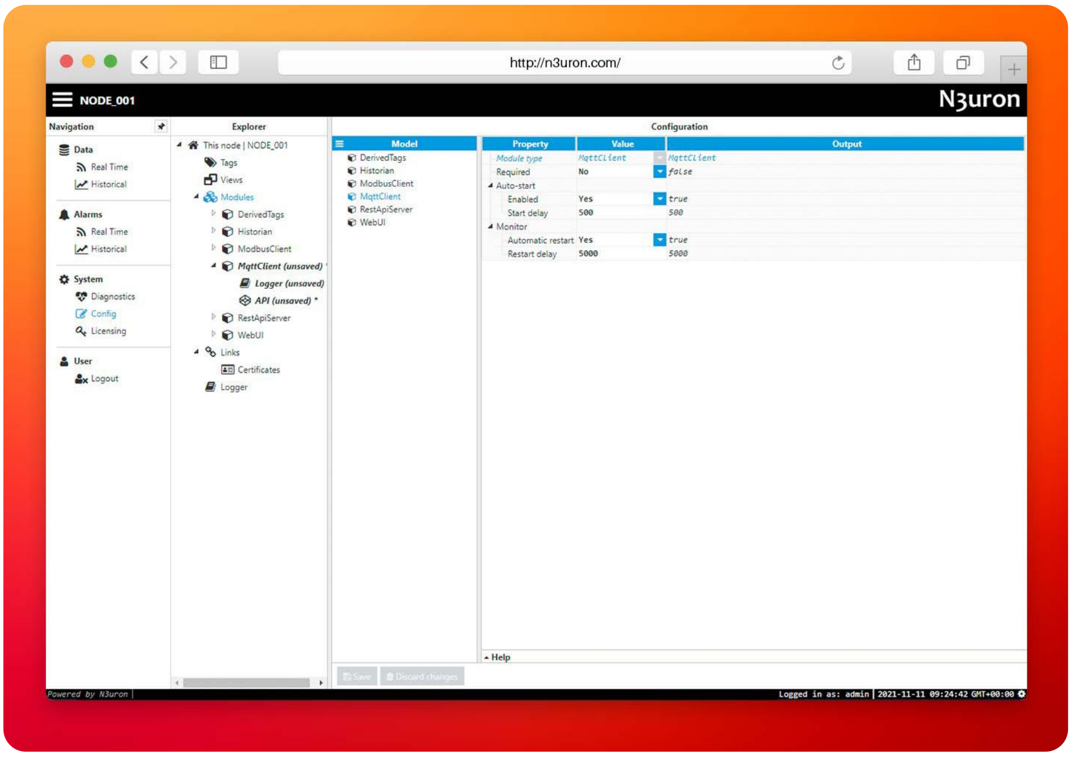
First, users must create a module for establishing the connection with Google Cloud. In this example, for the **Mqtt Client module**, the following steps should be taken:

– **Step 01:** Follow this route; System>Config > Modules > Model > New Module



Screenshot displaying how to create an instance using N3uron's MQTT Module panel

– **Step 02:** Configure the basic parameters of the module. Name=MqttClient, Module type= MqttClient.

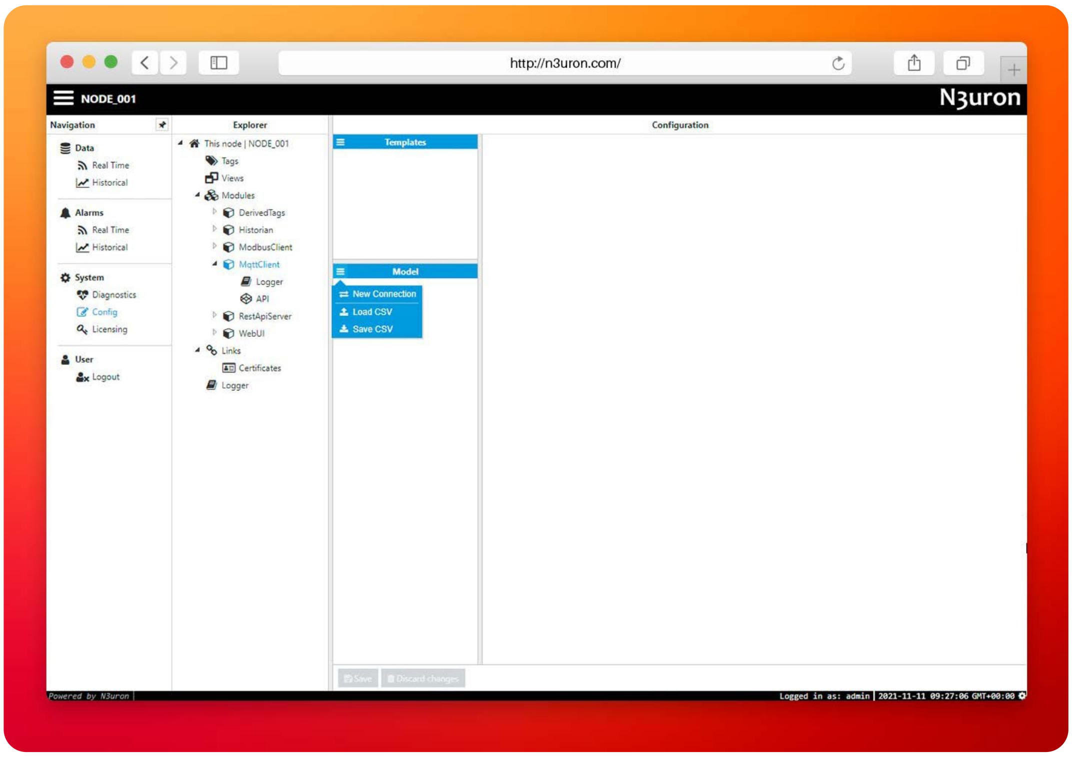


Property	Value	Output
Module type	MqttClient	MqttClient
Required	No	false
Auto-start		
Enabled	Yes	true
Start delay	500	500
Monitor		
Automatic restart	Yes	true
Restart delay	5000	5000

Screenshot displaying module configuration within the N3uron MQTT module panel.

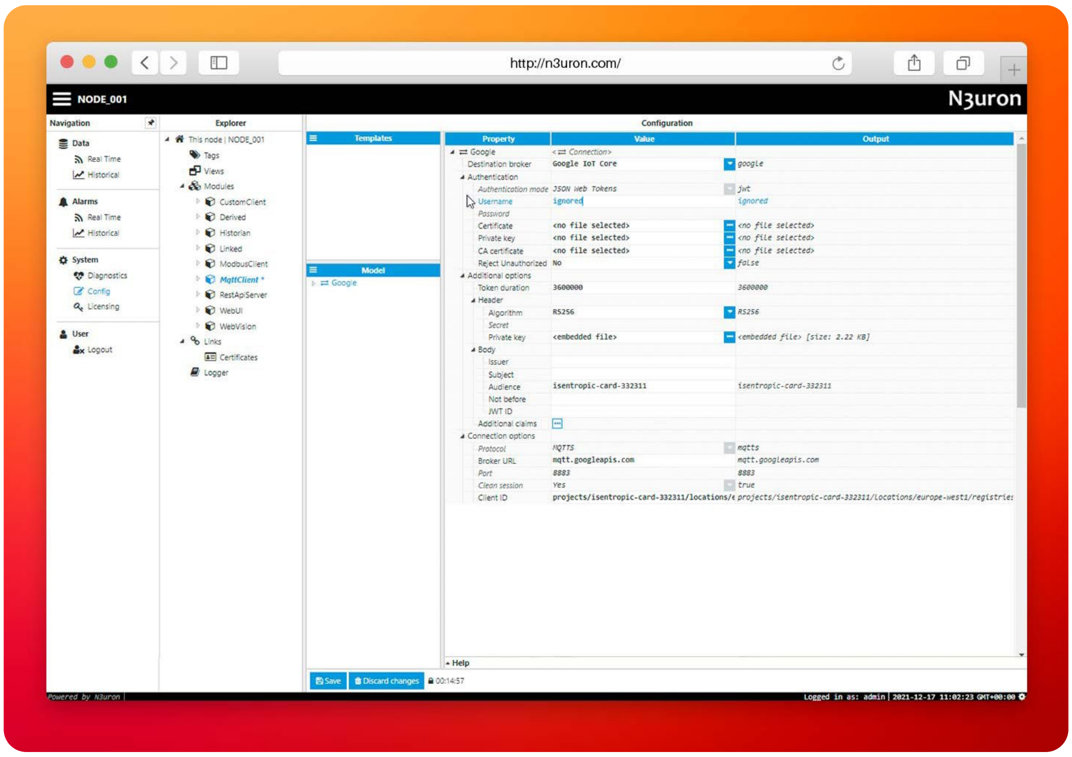
Before moving onto the next step, it is mandatory to save the default **Logger** and **API** settings. At this point, all other parameters on the main screen will be set to default.

– **Step 03:** Click on the MQTT Module and create a new connection(client) by following **Module name > Model > New connection.**



Screenshot demonstrating new connection creation via N3uron's MQTT module panel.

– **Step 04:** Establish a name for the connection and continue configuring the basic parameters for the connection. In this case, the name of the connection is Google.



Property	Value	Output
Google	<Connection>	
Destination broker	Google IoT Core	google
Authentication		
Authentication mode	JSON Web Tokens	jwt
Username	ignored	ignored
Password		
Certificate	<no file selected>	<no file selected>
Private key	<no file selected>	<no file selected>
CA certificate	<no file selected>	<no file selected>
Reject Unauthorized	No	false
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<embedded file>	<embedded file> [size: 2.22 KB]
Body		
Issuer		
Subject		
Audience	isentropic-card-332311	isentropic-card-332311
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/e	projects/isentropic-card-332311/locations/europe-west1/registrie:

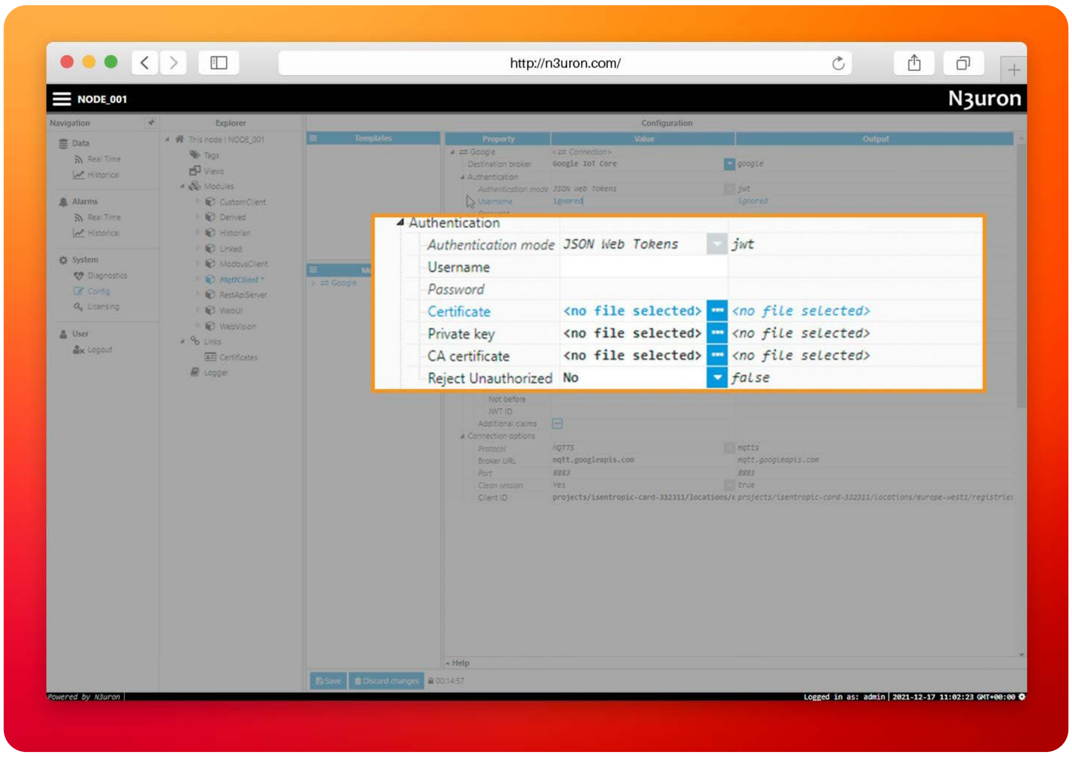
Screenshot demonstrating how to configure connections within N3uron's MQTT module panel.

You should define each parameter of the client step by step, which includes:

– **Destination broker:** this setting specifies the type of connection to be established. There are 4 possible options; Microsoft Azure, Amazon Web Services, Google IoT Core, and a custom one. In this case, we should select Google IoT Core.

– **Authentication:** there are five different types of authentication available. The first option is no authentication, which is useful when there is no private or important data. The password option authenticates with the broker using a password. The certificate option is the same as Password, but uses a certificate instead. The Password + Certificate option is more secure and requires a password and a certificate to authenticate. Finally, the last option is JSON Web Tokens, which authenticates using automatically generated Web Tokens.

In this case, Google IoT Core only uses the JSON Web Tokens option to authenticate.


Authentication

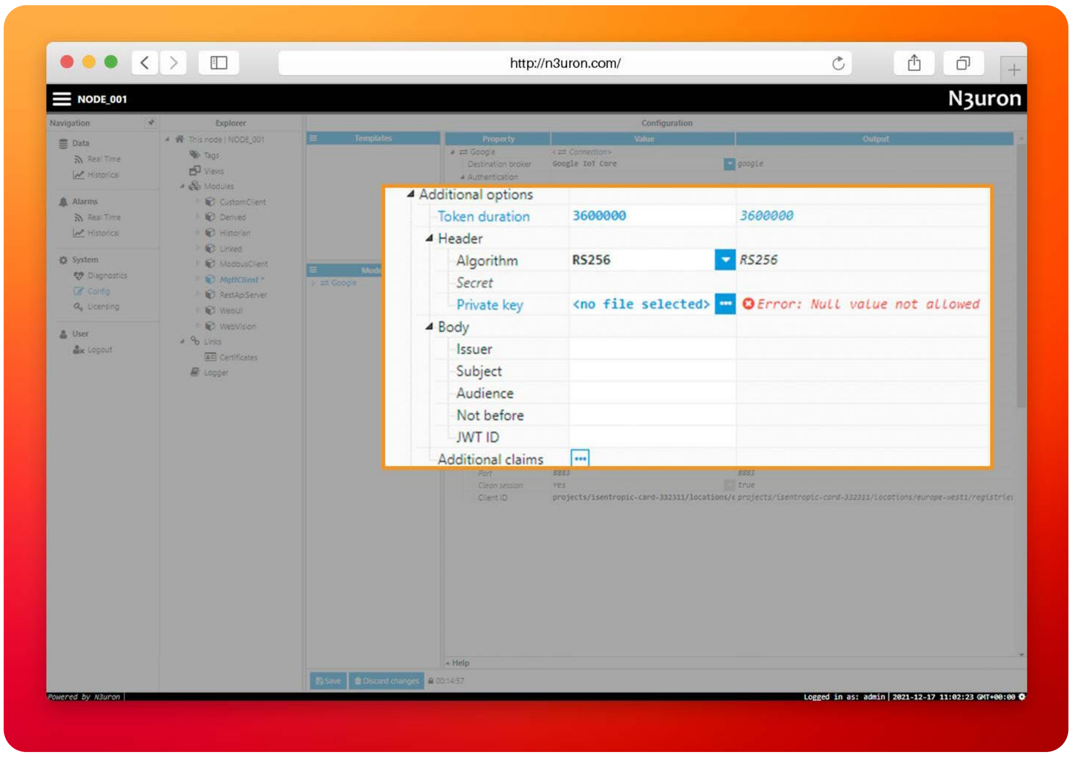
Authentication mode	JSON Web Tokens	jwt
Username		
Password		
Certificate	<no file selected>	<no file selected>
Private key	<no file selected>	<no file selected>
CA certificate	<no file selected>	<no file selected>
Reject Unauthorized	No	false

Property	Value	Output
Google	<Connection>	
Destination broker	Google IoT Core	google
Authentication		
Authentication mode	JSON Web Tokens	jwt
Username	ignored	ignored
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/	projects/isentropic-card-332311/locations/europe-west1/registrie

Screenshot displaying the “Authentication” section within the N3uron MQTT module panel.

- In **connection** options, users are required to define the type of broker, which can be either MQTT or MQTTS, the **URL** generated by our broker, and the port that will be used to connect to the broker. Finally, a unique ClientID must be provided for each connection defined in this section.
- **Last will and testament:** allows users to set a last will and testament for each connection, which will be sent to the broker when the session is disconnected and will be the last message sent.
- **Agents:** users can define an agent, which can be a publisher, writer, or subscriber, depending on whether you want to export or import data.

The Google Case offers one further option: **additional options**.



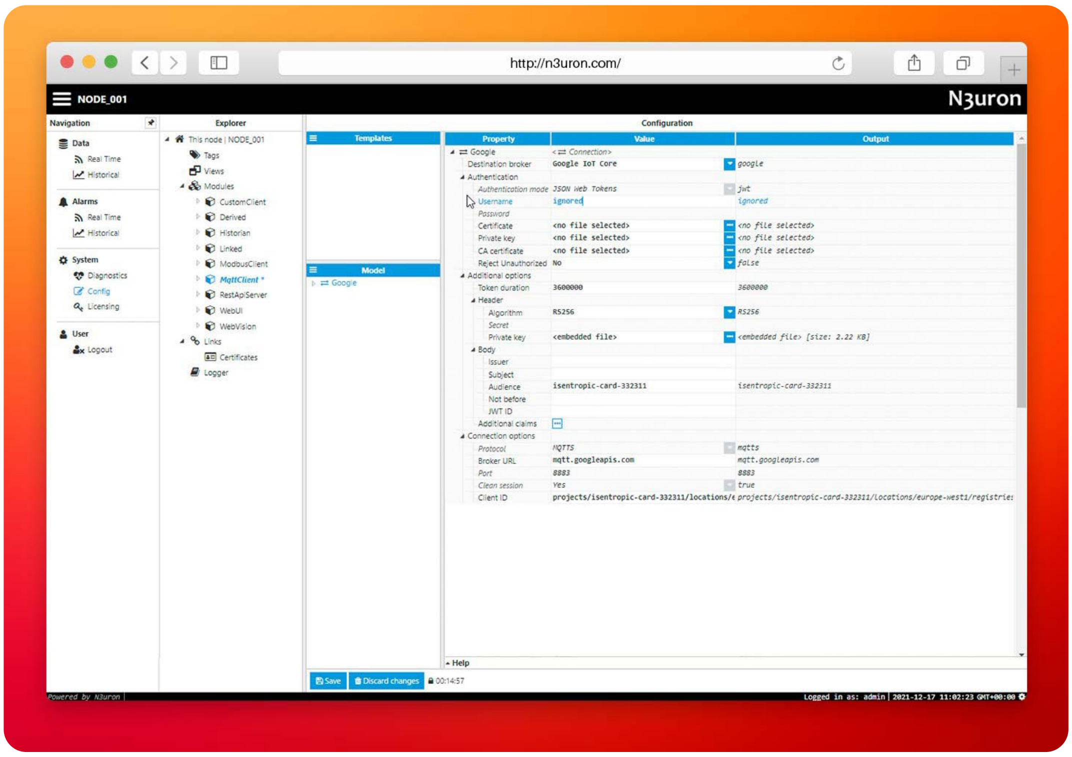
Property	Value	Output
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<no file selected>	Error: Null value not allowed
Body		
Issuer		
Subject		
Audience		
Not before		
JWT ID		
Additional claims		

Screenshot displaying the “Additional Options” section within N3uron’s MQTT module panel.

In summary, the **MqttClient module** defined in N3uron can connect with Google Cloud using these configuration settings:

- **Destination broker:** Google IoT Core
- **Authentication mode:** JSON Web Tokens
- **Token duration:** the maximum value is a full day, so any value equal to or less than this is valid.
- **Algorithm:** RS256 to sign the JWT (there are a wide variety of options to choose from).
- **Private key:** the generated key in .pem format stored in the device.
- **Audience:** sets the project_id generated in GCP. In this case, isentropic-card-332311.
- **Protocol:** MQTTS (MQTT can also be used).
- **Host:** specifies the URL for the MQTT broker, in this case, mqtt.googleapis.com.
- **Port:** the port is 8883.
- **ClientID:** specifies the MQTT client for this connection. This is unique for each connection. The Client ID must follow this pattern: projects/PROJECT ID/locations/CLOUD REGION/registries/REGISTRY ID/devices/DEVICE ID.
- **QoS:** with the exception of QoS 2, users can select 0 or 1 for a stable connection.

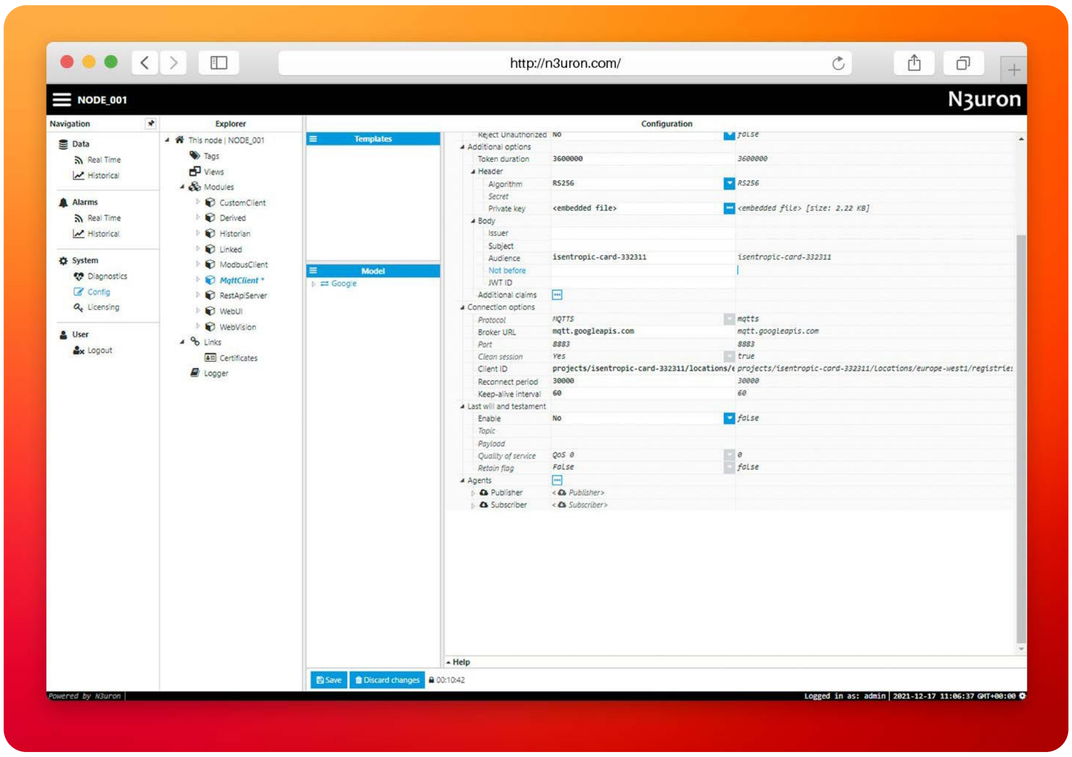
The below screenshot demonstrates this example:



Property	Value	Output
Google	<Connection>	
Destination broker	Google IoT Core	google
Authentication		
Authentication mode	JSON Web Tokens	jwt
Username	ignored	ignored
Password		
Certificate	<no file selected>	<no file selected>
Private key	<no file selected>	<no file selected>
CA certificate	<no file selected>	<no file selected>
Reject Unauthorized	No	false
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<embedded file>	<embedded file> [size: 2.22 KB]
Body		
Issuer		
Subject		
Audience	isentropic-card-332311	isentropic-card-332311
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/e	projects/isentropic-card-332311/locations/europe-west1/registrie:

Logged in as: admin | 2021-12-17 11:02:23 GMT+00:00

Screenshot displaying an example of the MQTT Client panel.



Property	Value	Output
Reject Unauthorized	No	false
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<embedded file>	<embedded file> [size: 2.22 KB]
Body		
Issuer		
Subject		
Audience	isentropic-card-332311	isentropic-card-332311
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/e	projects/isentropic-card-332311/locations/europe-west1/registrie:
Reconnect period	30000	30000
Keep-alive interval	60	60
Last will and testament		
Enable	No	false
Topic		
Payload		
Quality of service	QoS 0	0
Retain flag	False	false
Agents		
Publisher	<Publisher>	
Subscriber	<Subscriber>	

Logged in as: admin | 2021-12-17 11:06:37 GMT+00:00

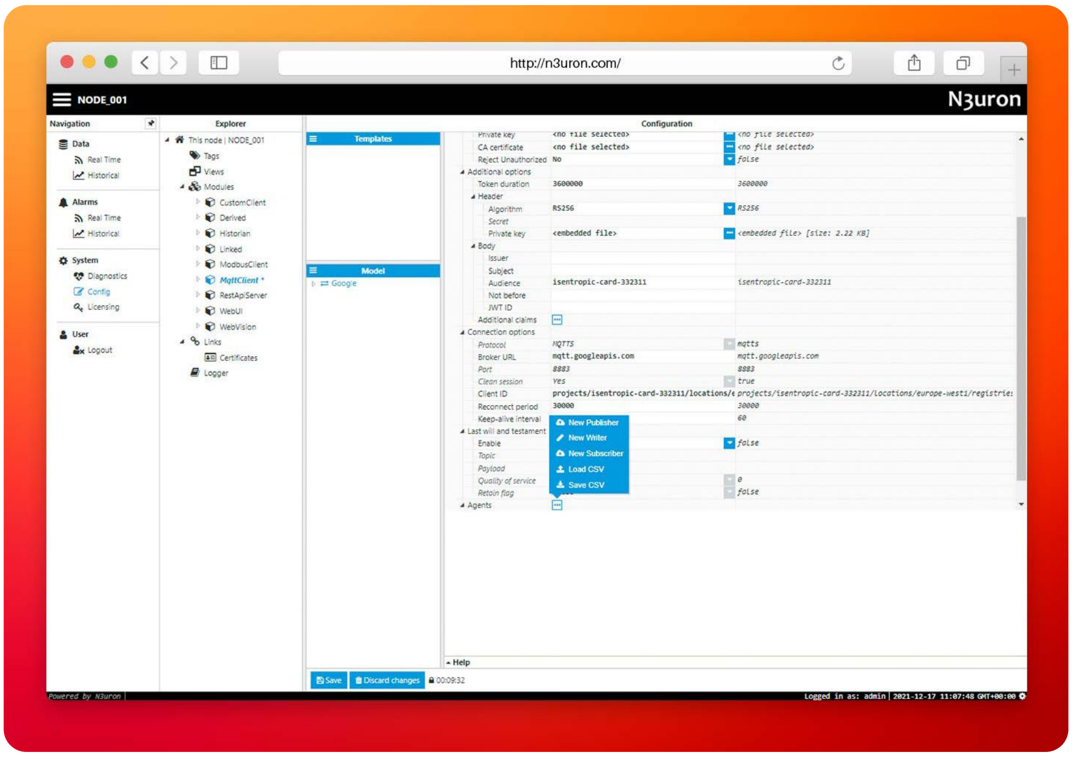
Screenshot displaying a different example of the MQTT Client panel.

The **MQTT module** has now been created and is ready to use. Therefore, it is now time to generate a publication and subscription.

Publishing & Subscribing Data via N3uron and Google Cloud Platforms

Publishing Data Using N3uron's MQTT Module Within the WebUI Explorer Panel

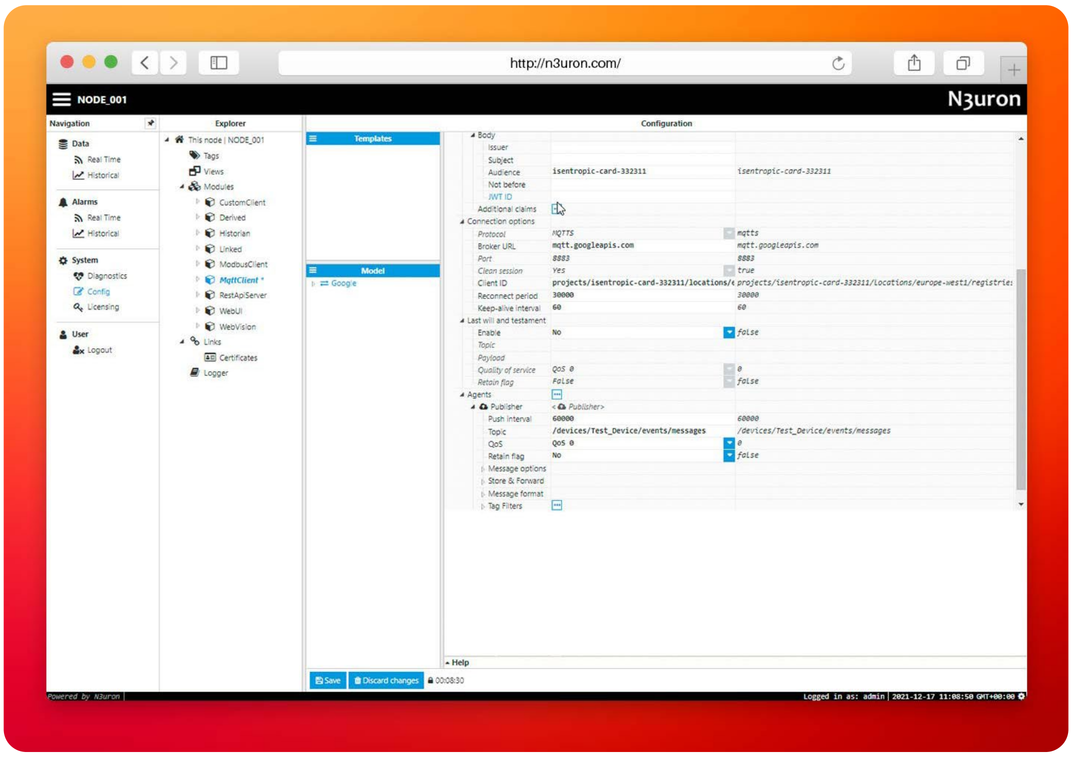
– **Step 01:** Generating a publication is very easy. Simply click on **new publisher** in the **agents section** of the MQTT module, as shown in the screenshot.



Private key	<no file selected>	<no file selected>
CA certificate	<no file selected>	<no file selected>
Reject Unauthorized	No	false
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<embedded file>	<embedded file> [size: 2.22 KB]
Body		
Issuer		
Subject		
Audience	isentropic-card-332311	isentropic-card-332311
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/e	projects/isentropic-card-332311/locations/europe-west1/registrie:
Reconnect period	30000	30000
Keep-alive interval		60
Last will and testament		
Enable		false
Topic		
Payload		
Quality of service		0
Retain flag		false
Agents		

Screenshot displaying the "New Publisher" option within the N3uron MQTT module panel.

– **Step 02:** Provide a name. In this case, a very simple name has been used: Publisher.



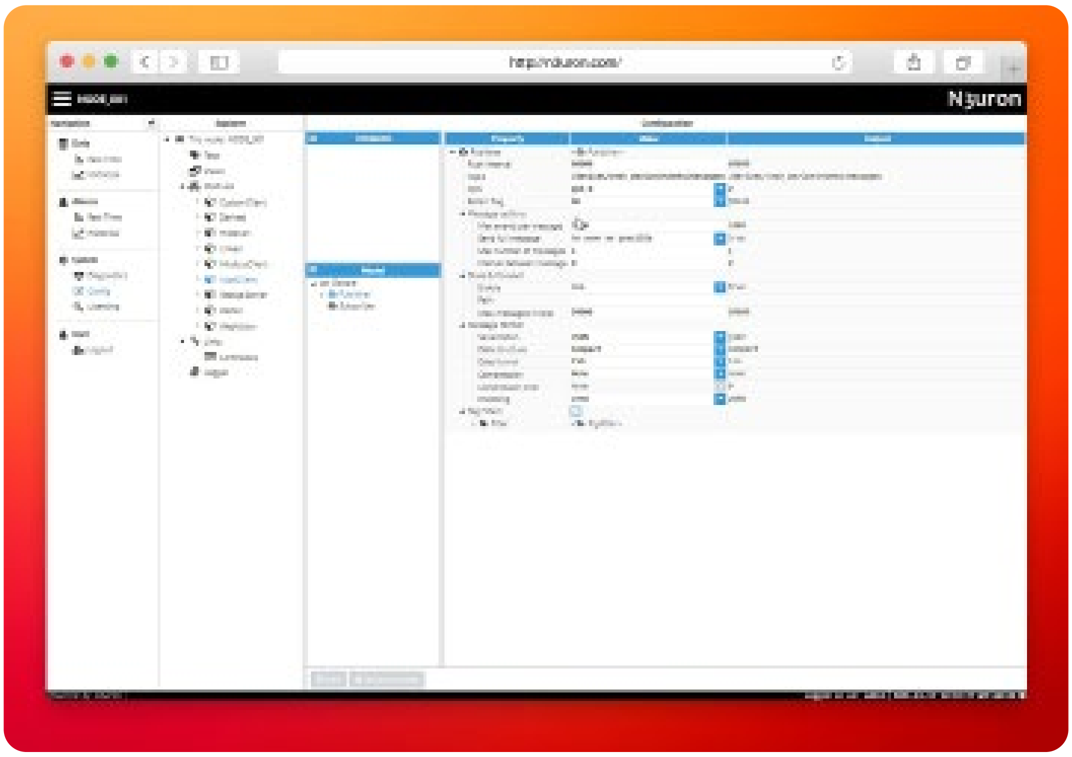
```
Body
    Issuer
    Subject
    Audience            isentropic-card-332311                 isentropic-card-332311
    Not before
    JWT ID
  Additional claims
Connection options
  Protocol            MQTTS                                  mqtts
  Broker URL          mqtt.googleapis.com                    mqtt.googleapis.com
  Port                8883                                   8883
  Clean session       Yes                                    true
  Client ID           projects/isentropic-card-332311/locations/e  projects/isentropic-card-332311/locations/europe-west1/registrie
  Reconnect period    30000                                  30000
  Keep-alive interval 60                                     60
Last will and testament
  Enable              No                                     false
  Topic
  Payload
  Quality of service  QoS 0                                  0
  Retain flag         False                                  false
Agents
  Publisher           < Publisher>
    Push interval     60000                                  60000
    Topic             /devices/Test_Device/events/messages   /devices/Test_Device/events/messages
    QoS               QoS 0                                  0
    Retain flag       No                                     false
    Message options
    Store & Forward
    Message format
    Tag Filters
```

Screenshot displaying the publisher configuration settings within the N3uron MQTT module panel.

The device is created with an MQTT client and is connected to the MQTT bridge. You can now publish telemetry events to a topic with the following format: **/devices/Device_ID/events**

Messages sent to this topic are redirected to the topic that has been set for telemetry by default in the registry. The default telemetry topic is the Cloud Pub/Sub topic defined in the eventNotificationConfigs[i].pubsubTopicName field of the registry. If no default topic is defined, the data will be lost. To publish messages to other Cloud Pub/Sub topics, see Publishing telemetry events to additional Cloud Pub/Sub topics.

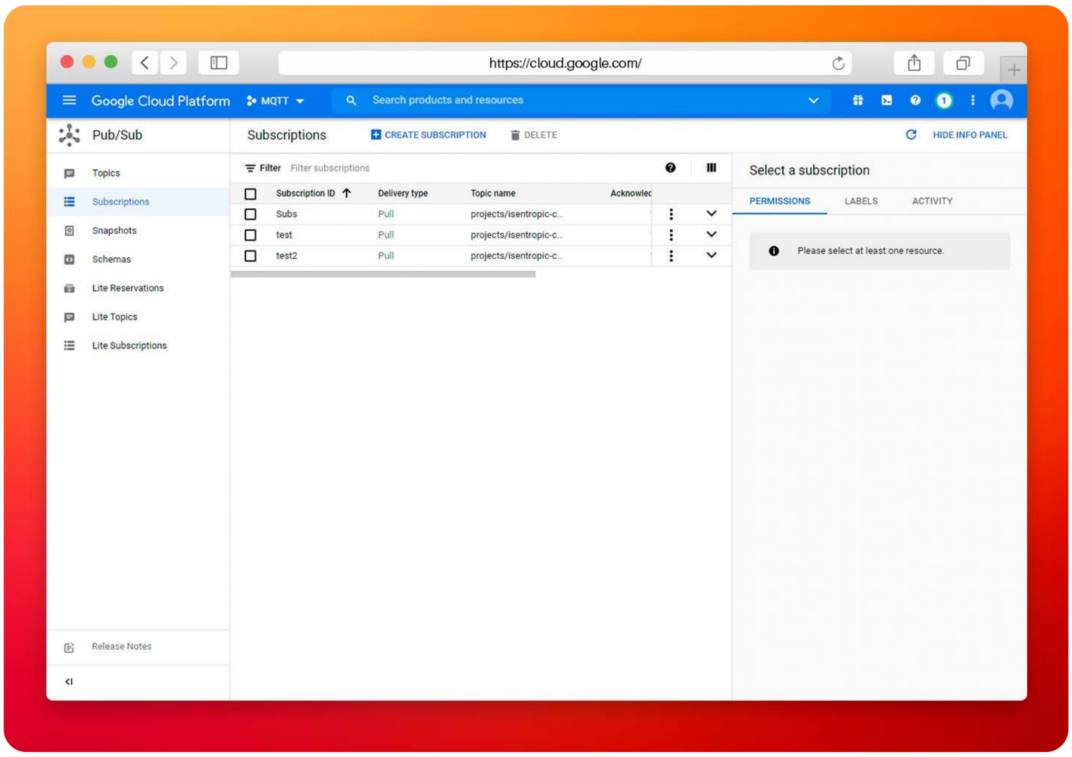
– **Step 03:** To finish configuration, users should define the name of the **topic** within which they want to publish and create a **tag filter** without constraints. This allows all tags to publish their data in the topic. In this case, the topic used is /devices/Test_Device/events/messages.



Screenshot displaying the publisher configuration icon within the Model area of the MQTT module panel.

From now on, all data will be continuously published to the cloud. In order to display this data, users must enter the GCP console, go to the subscriptions section, and create a subscription for the previously generated default topic. Failure to complete this step will result in published data being lost. To do so, users should follow the below steps:

– **Step 01:** Click on the **Subscription section** and select **Create Subscription**.

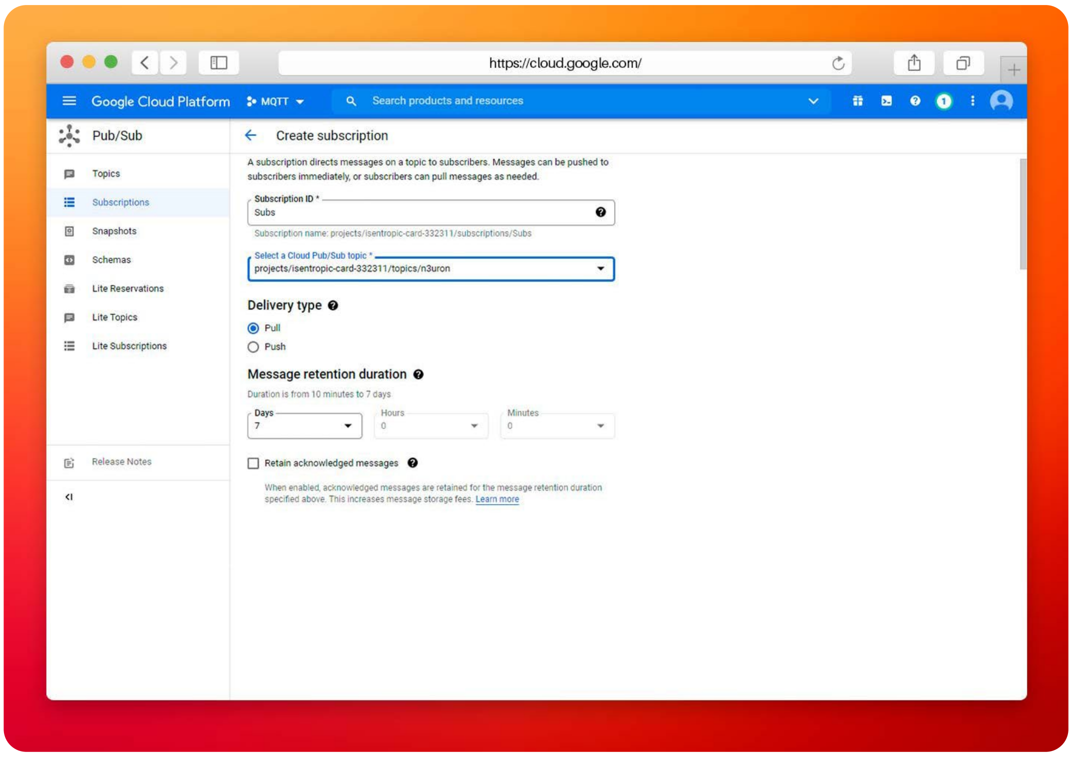


Subscription ID	Delivery type	Topic name	Acknowlec
Subs	Pull	projects/isentropic-c...	
test	Pull	projects/isentropic-c...	
test2	Pull	projects/isentropic-c...	

Please select at least one resource.

Screenshot displaying the “Subscription” panel in the Google Cloud Platform.

– **Step 02:** Give a name to the subscription and select the topic you want to subscribe to. In this case, the name of the subscription is subs (although this is not relevant) and the name of the topic subscribed is projects/isentropic-card-332311/topics/n3uron.



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 *: Subs

Subscription name: projects/isentropic-card-332311/subscriptions/Subs

Select a Cloud Pub/Sub topic *: projects/isentropic-card-332311/topics/n3uron

Delivery type

[✓ Pull] Push

Message retention duration

Duration is from 10 minutes to 7 days

Days: 7 Hours: 0 Minutes: 0

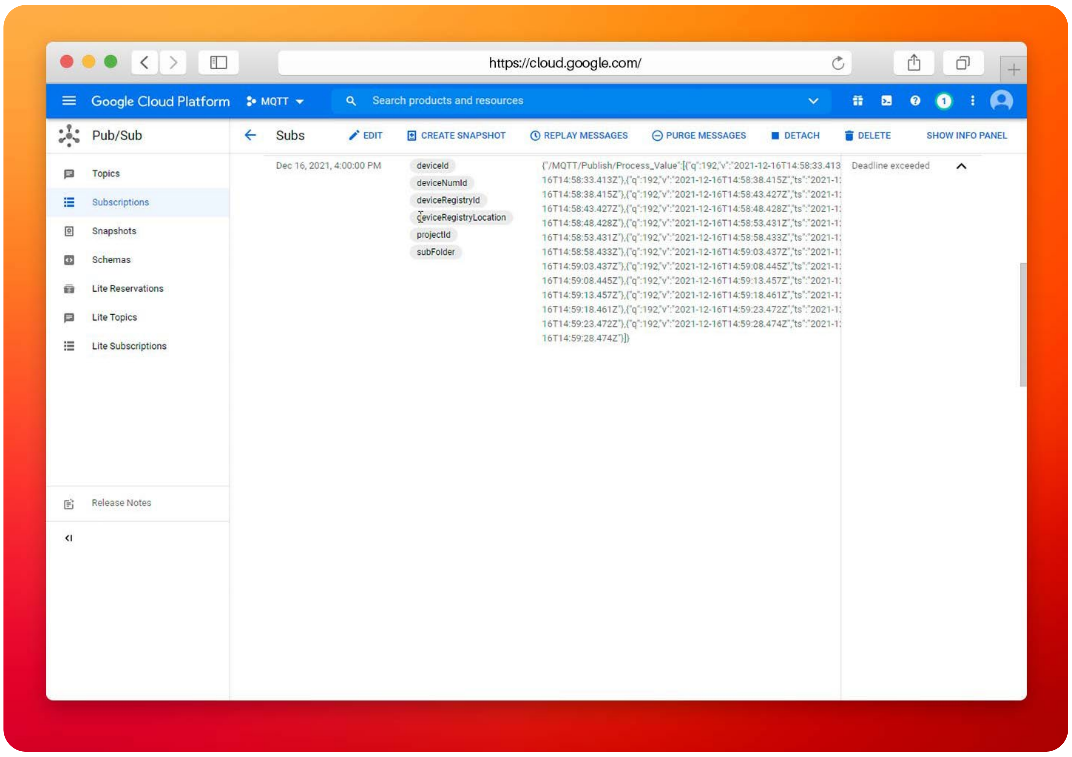
☐ Retain acknowledged messages

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

Screenshot displaying the “Create Subscription” panel in the Google Cloud Platform.

– **Step 03:** Click on **create** to finish configuration.

You should now enter the created subscription and select pull in the messages section to obtain the published messages. The result should be similar to the screenshot below:



Dec 16, 2021, 4:00:00 PM

deviceId
deviceNumId
deviceRegistryId
deviceRegistryLocation
projectId
subFolder

```
{"/MQTT/Publish/Process_Value":[{"q":192,"v":"2021-12-16T14:58:33.413
16T14:58:33.413Z"},{"q":192,"v":"2021-12-16T14:58:38.415Z","ts":"2021-1
16T14:58:38.415Z"},{"q":192,"v":"2021-12-16T14:58:43.427Z","ts":"2021-1
16T14:58:43.427Z"},{"q":192,"v":"2021-12-16T14:58:48.428Z","ts":"2021-1
16T14:58:48.428Z"},{"q":192,"v":"2021-12-16T14:58:53.431Z","ts":"2021-1
16T14:58:53.431Z"},{"q":192,"v":"2021-12-16T14:58:58.433Z","ts":"2021-1
16T14:58:58.433Z"},{"q":192,"v":"2021-12-16T14:59:03.437Z","ts":"2021-1
16T14:59:03.437Z"},{"q":192,"v":"2021-12-16T14:59:08.445Z","ts":"2021-1
16T14:59:08.445Z"},{"q":192,"v":"2021-12-16T14:59:13.457Z","ts":"2021-1
16T14:59:13.457Z"},{"q":192,"v":"2021-12-16T14:59:18.461Z","ts":"2021-1
16T14:59:18.461Z"},{"q":192,"v":"2021-12-16T14:59:23.472Z","ts":"2021-1
16T14:59:23.472Z"},{"q":192,"v":"2021-12-16T14:59:28.474Z","ts":"2021-1
16T14:59:28.474Z"}]}
```

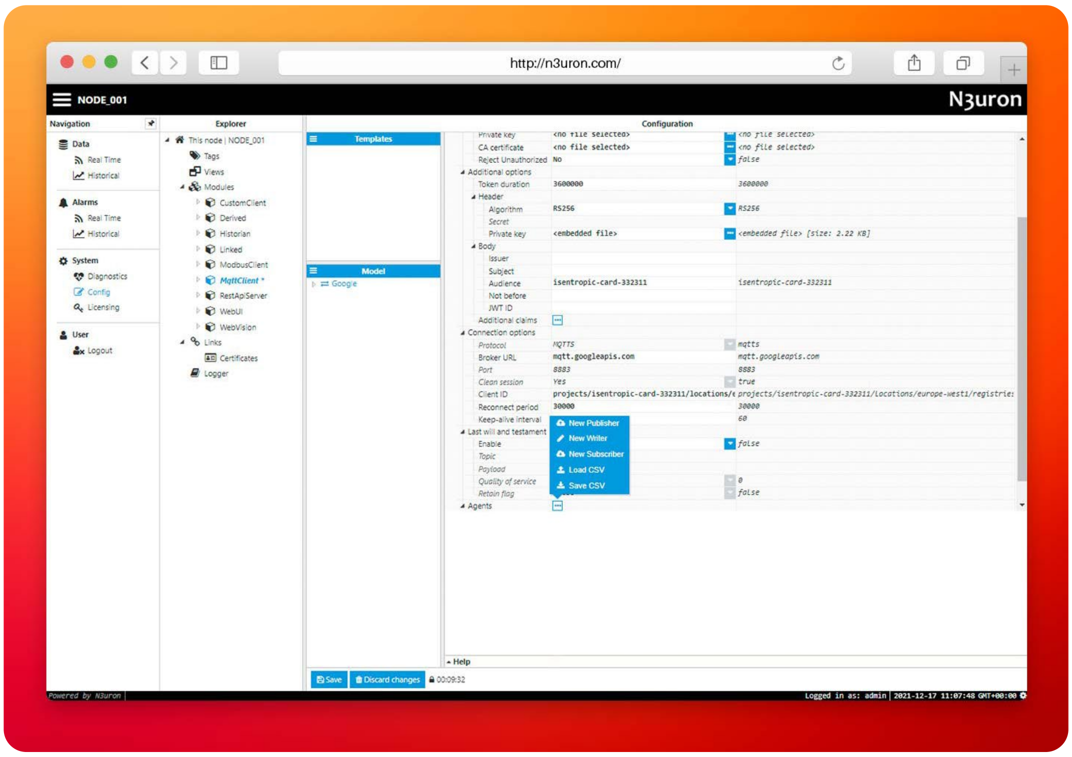
Deadline exceeded

Screenshot displaying published messages once subscribed to the topic in the Google Cloud Platform.

Subscribing Data within the Google Cloud Platform

The process for generating subscriptions is similar to the publication process. The objective of the subscription is to receive tag events from remote devices via MQTT.

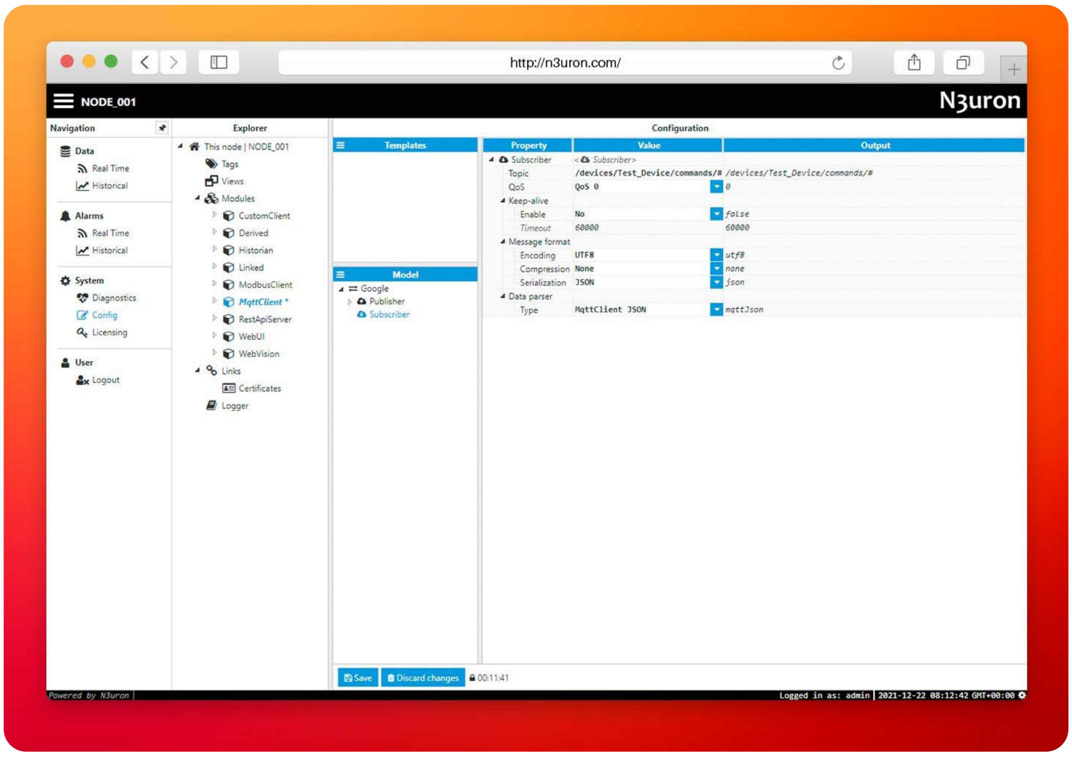
– **Step 01:** First, create a **new subscriber**, as previously done for the publisher.



Private key	<no file selected>	<no file selected>
CA certificate	<no file selected>	<no file selected>
Reject Unauthorized	No	false
Additional options		
Token duration	3600000	3600000
Header		
Algorithm	RS256	RS256
Secret		
Private key	<embedded file>	<embedded file> [size: 2.22 KB]
Body		
Issuer		
Subject		
Audience	isentropic-card-332311	isentropic-card-332311
Not before		
JWT ID		
Additional claims		
Connection options		
Protocol	MQTTS	mqtts
Broker URL	mqtt.googleapis.com	mqtt.googleapis.com
Port	8883	8883
Clean session	Yes	true
Client ID	projects/isentropic-card-332311/locations/e	projects/isentropic-card-332311/locations/europe-west1/registries
Reconnect period	30000	30000
Keep-alive interval		60
Last will and testament		
Enable		false
Topic		
Payload		
Quality of service		0
Retain flag		false
Agents		

Screenshot displaying the “New Publisher” option within N3uron’s MQTT module panel.

– **Step 02:** Name it as Subscriber in the subscriber name section.



Property	Value	Output
Subscriber	<Subscriber>	
Topic	/devices/Test_Device/commands/#	/devices/Test_Device/commands/#
QoS	QoS 0	0
Keep-alive		
Enable	No	false
Timeout	60000	60000
Message format		
Encoding	UTF8	utf8
Compression	None	none
Serialization	JSON	json
Data parser		
Type	MqttClient JSON	mqttJson

Screenshot displaying subscriber configuration settings in the N3uron Tags panel.

In order to receive a command, the device must:

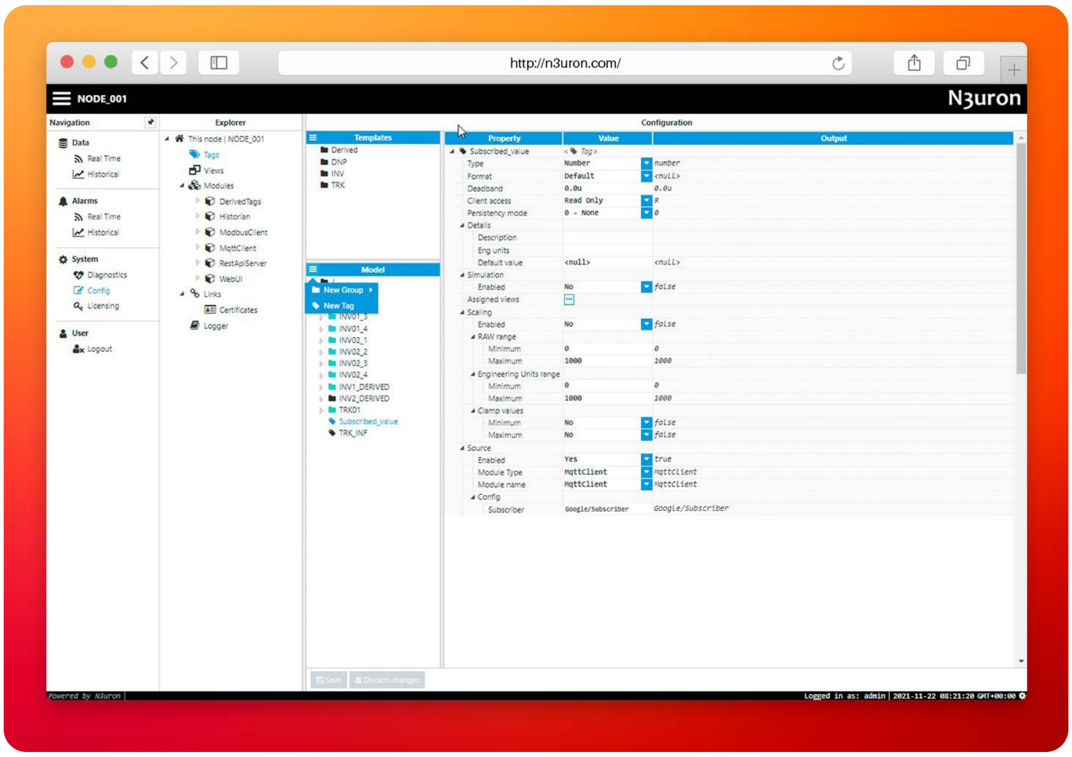
1. Be connected to Google IoT Core using the MQTT protocol
2. Subscribe to the topic/devices/Device_ID/commands/# (the # wildcard is necessary)

By subscribing to this topic, the device will be able to receive messages sent to **/devices/Device_ID/commands/#**, as well as those sent to subfolders, **/devices/Device_ID/commands/#/{subfolder}.** Subscriptions to a specific subfolder are not possible.

– **Step 03:** In this case, the unique compulsory configuration requires users to introduce the name of the topic from which the data will be obtained. In this case, the topic used is, /devices/Test_Device/commands/#.

A tag must now be created in N3uron, which will be updated according to the events received from the Google Core IoT. Follow the below steps to create this tag:

– **Step 01:** First click on **Config > Tags > Model > New tag** and label it with a specific name. In this example, Subscribed_Value is used.
– **Step 02:** Define the Source section and complete it using the previously defined **MQTT Client module**.



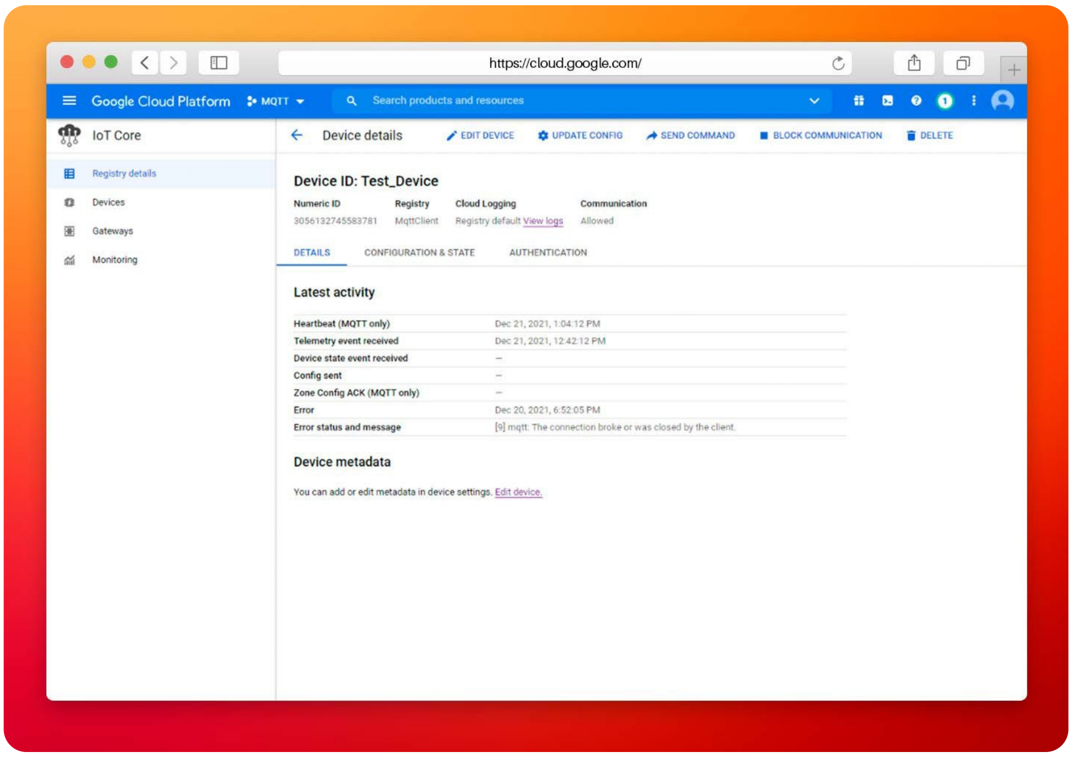
Property	Value	Output
Subscribed_value	< Tag>	
Type	Number	number
Format	Default	<null>
Deadband	0.0u	0.0u
Client access	Read Only	R
Persistency mode	0 - None	0
Details		
Description		
Eng units		
Default value	<null>	<null>
Simulation		
Enabled	No	false
Assigned views		
Scaling		
Enabled	No	false
RAW range		
Minimum	0	0
Maximum	1000	1000
Engineering Units range		
Minimum	0	0
Maximum	1000	1000
Clamp values		
Minimum	No	false
Maximum	No	false
Source		
Enabled	Yes	true
Module Type	MqttClient	MqttClient
Module name	MqttClient	MqttClient
Config		
Subscriber	Google/Subscriber	Google/Subscriber

Screenshot displaying tag configuration within the N3uron Tags panel.

– In **Module name**, the name used must be exactly the same as the module name.
– In the **subscriber** section, users must introduce the established connection (in this case Google)/the defined subscriber (in this case Subscriber). In this case, it would be Google/Subscriber.

Once this has been configured, the defined tag will be able to receive events from the Google Cloud.

– **Step 01:** Go to the GCP console, click on devices, select the device you created and select the send command.



Device details

Device ID: Test_Device

Numeric ID	Registry	Cloud Logging	Communication
3056132745583781	MqttClient	Registry default View logs	Allowed

Latest activity

Heartbeat (MQTT only)	Dec 21, 2021, 1:04:12 PM
Telemetry event received	Dec 21, 2021, 12:42:12 PM
Device state event received	–
Config sent	–
Zone Config ACK (MQTT only)	–
Error	Dec 20, 2021, 6:52:05 PM
Error status and message	[9] mqtt: The connection broke or was closed by the client.

Device metadata

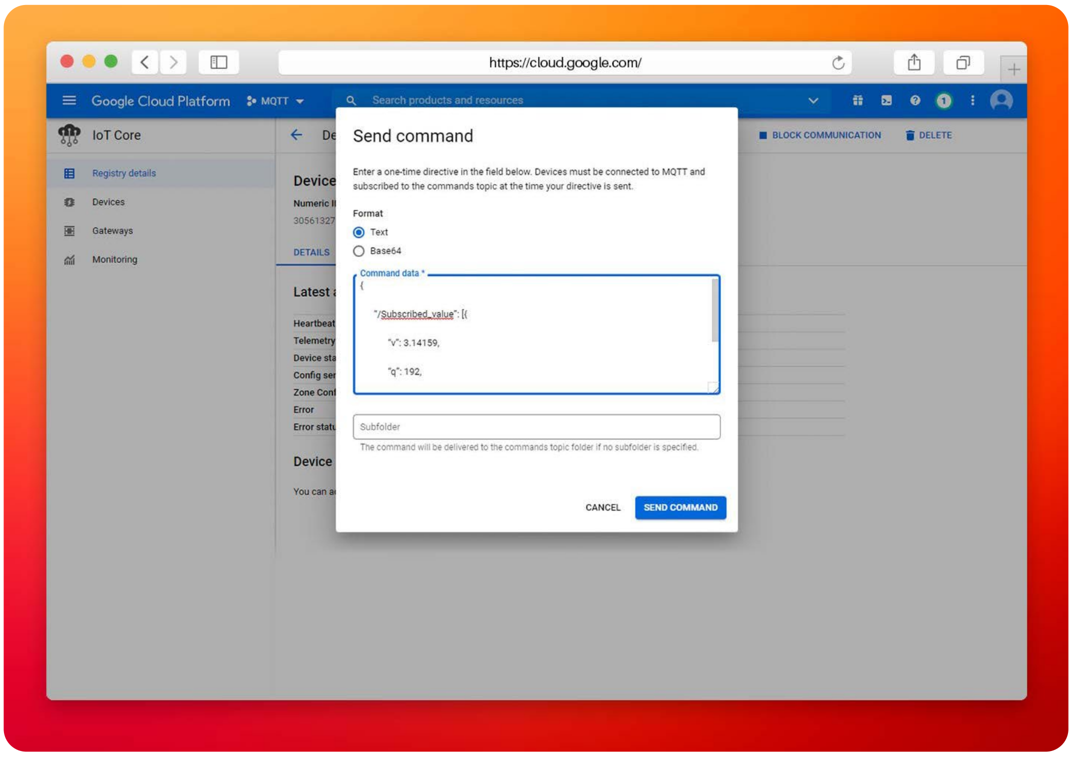
You can add or edit metadata in device settings. Edit device.

Screenshot displaying the publication configuration in the Google Cloud Platform.

– **Step 02:** Within the Message, Payload will introduce the following:

```
{

    "/Subscribed_value": [{

        "v": 3.14159,

        "q": 192,

        "ts": 1630668488618

    }]

}
```

– **Step 03:** Click on the Send Command button.



Send command

Enter a one-time directive in the field below. Devices must be connected to MQTT and subscribed to the commands topic at the time your directive is sent.

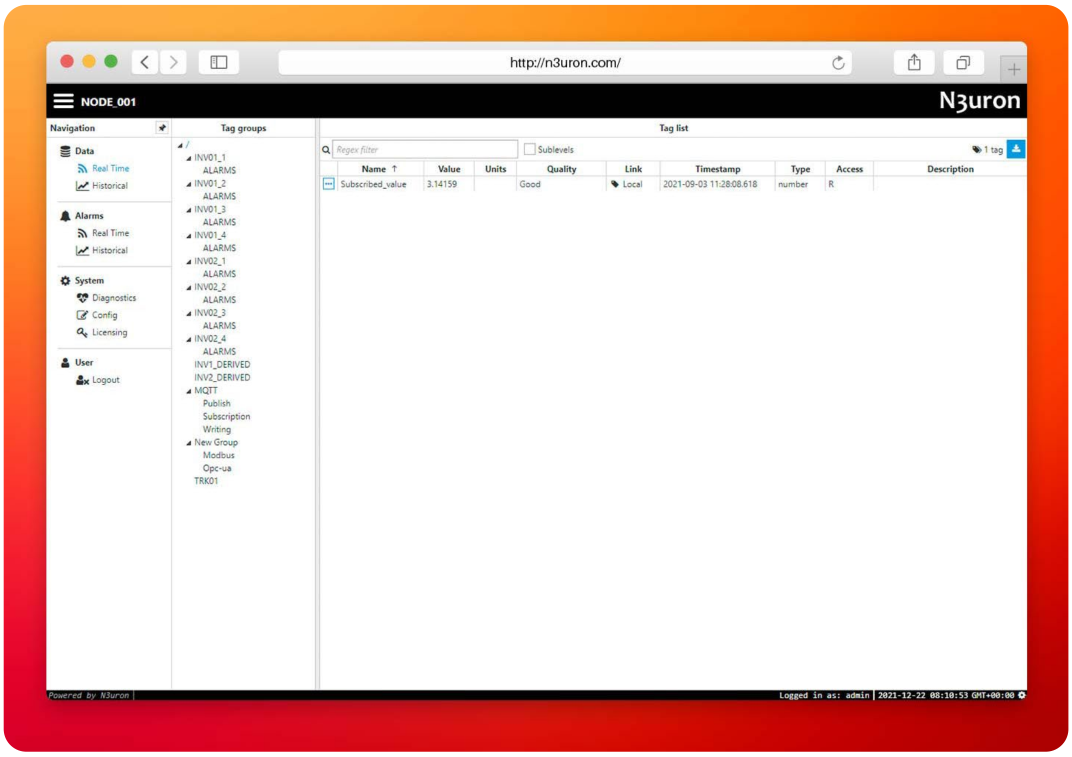
Format
[✓ Text] Base64

Command data *
```
{

    "/Subscribed_value": [{

        "v": 3.14159,

        "q": 192,
```

Subfolder
The command will be delivered to the commands topic folder if no subfolder is specified.

Screenshot displaying the "Send Command" panel in the Google Cloud Platform.

– **Step04:** Go back to the N3uron WebUI interface and select **Data/Real-Time** in the left-hand side panel. You should now see the Subscribed_Value tag you previously created with a value of 3.14159.



Name ↑	Value	Units	Quality	Link	Timestamp	Type	Access	Description
Subscribed_value	3.14159		Good	Local	2021-09-03 11:28:08.618	number	R	

Screenshot displaying real-time values in the Real-Time section of the N3uron Navigation column panel.

Conclusion: How to connect industrial Assets to Google Cloud using N3uron's MQTT Module

Connecting your assets to Google Cloud is extremely easy using **N3uron's MQTT module**. If you are ready to start using the MQTT module, download the N3uron free trial version and read our MQTT Manual on how to implement and use N3uron's MQTT module on our communication platform.