Getting Started: Google Cloud Injector Quick Start

Google IoT Core was retired in summer or 2023. Clearblade IoT Core provides a drop-in replacement for the MQTT Server endpoint. As a result, Clearblade IoT Core works with Cirrus Link's Google Cloud Injector and the Google Cloud. More information is available here: https://iot.clearblade.com.

Prerequisites

- Ignition with Google Cloud Injector Module installed
  - Review the Cirrus Link Module Installation documentation for installation details.
- Ignition Designer installed
  - Review the Inductive Automation documentation for Launching Designer against the Ignition gateway
- An existing Clearblade IoT Core account with a Project, IoT Core registry, credentials, and device created.
  - Documentation on getting started with the Clearblade IoT Core can be found here

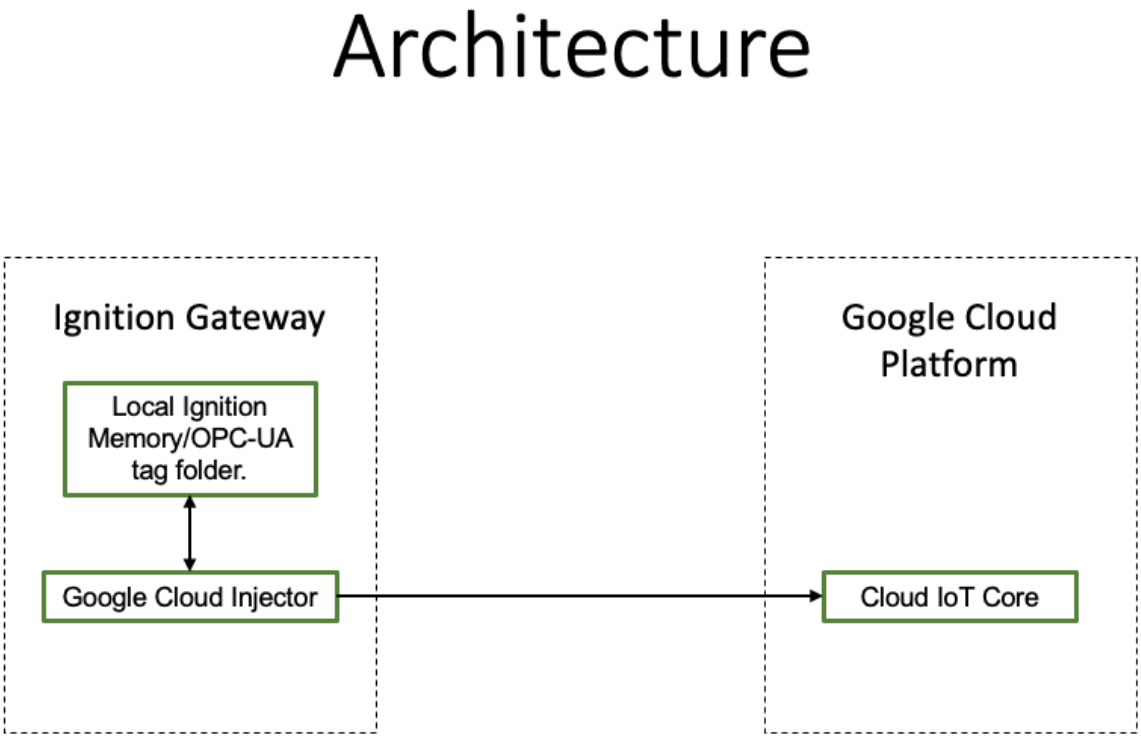
Summary

This tutorial will provide step-by-step instructions for the following:

- Configuring the Google Cloud Injector Module to connect to an existing Clearblade IoT Core
- Publishing live Tag data and events to the connected Clearblade IoT Core

Upon completion of this module you will have an Ignition Gateway connected and publishing live Tag data to a Clearblade IoT Core.

Architecture



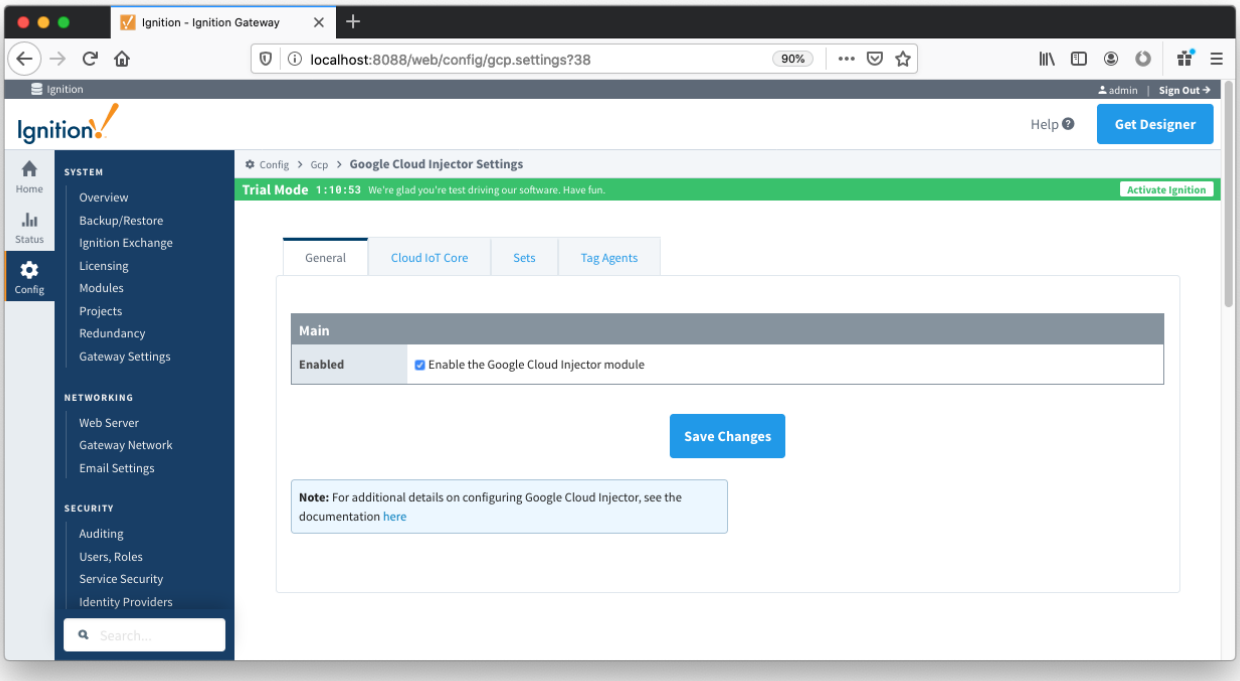
Tutorial

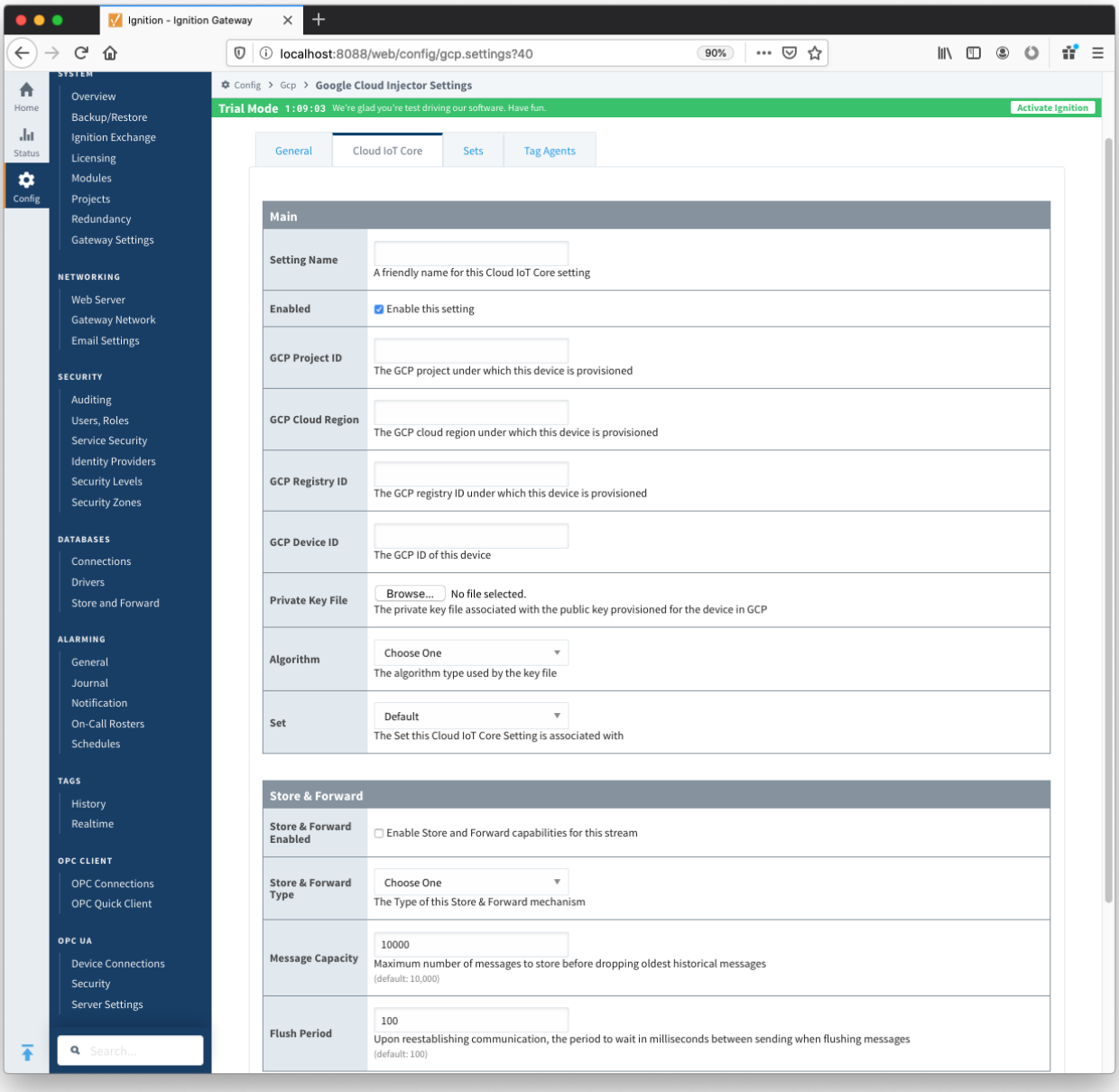
Step 1: Configure the Google Cloud Injector Module

Once you have Ignition and the Google Cloud Injector Module installed and running we can setup the configuration to connect to your existing Clearblade IoT Core.

> Review the Google Cloud Injector Module configuration guide for more details on each tab

Navigate to the Google Cloud Injector Modules configuration section from the left side bar in the Ignition Gateway and select the Clearblade IoT Core tab.





Set the following parameters:

- Setting Name
  - This can be any unique identifier. For this tutorial we will use "TestSetting".
- GCP Project ID
  - This is the ID of the project that was created with the Google Cloud Platform account.
- GCP Cloud Region
  - This should be obtained from the Cloud IoT Core that was created.
- GCP Registry ID
  - This should be obtained from the Cloud IoT Core that was created.
- GCP Device ID
  - This should be obtained from the Cloud IoT Core that was created.
- Private Key File
  - The Private Key File must be in PKCS8 DER format. See Google's documentation here on how to convert a PEM to PKCS8 DER format.
- Algorithm
  - Algorithm type used by the key file. Options are RS256 and ES256

Click on "Create New Cloud IoT Core Setting" to save create the new configuration setting.

Now the Google Cloud Injector module is connected to the Cloud IoT Core, we have to determine if there are are changes needed to the Tag Agent tab to be able to push data.

If you already have Ignition tags defined, for example from the Ignition OPC UA Server, then depending on the depth of your tag tree you may need to configure the Sparkplug Settings.

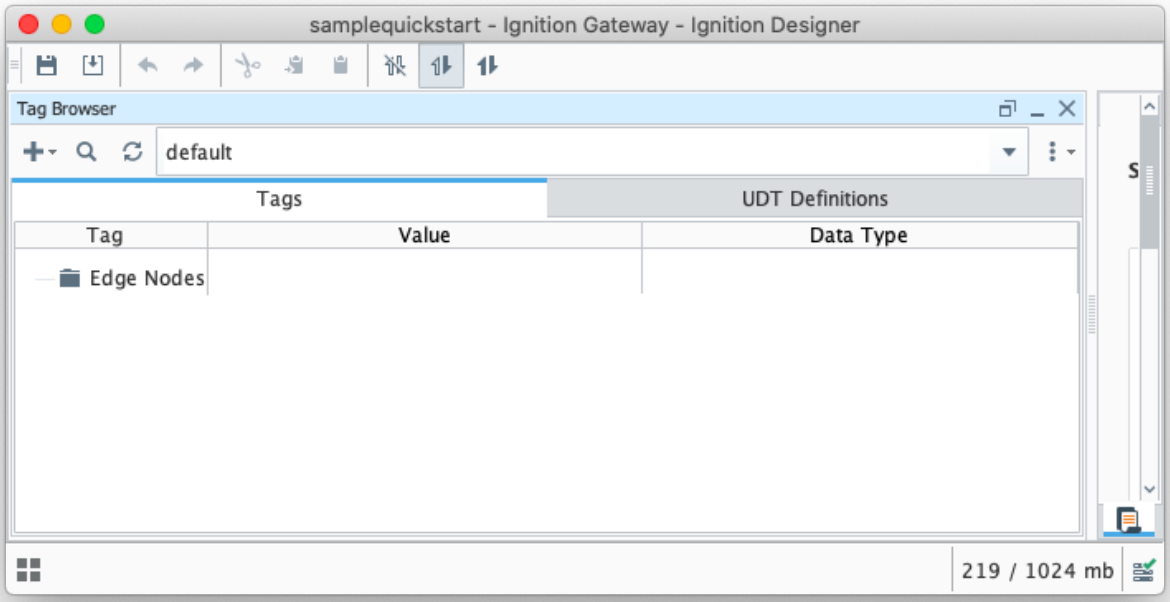
> Review the Cloud Injector Tag Agents and Tag Trees document which describes how Cloud Injector Agent configurations interact with Ignition tag trees to push messages and tag change events to the cloud service. It explains how tags get identified to be pushed as well as what specific 'topics' will be included with the messages. It also goes over some example configurations to show how the system will behave in different scenarios.

Once the Tag Agent is setup as needed, you can jump to Step 3: Publishing data.

If you do not have Ignition tags defined, that will be done in the next step with a tag tree depth that requires no additional Sparkplug settings.

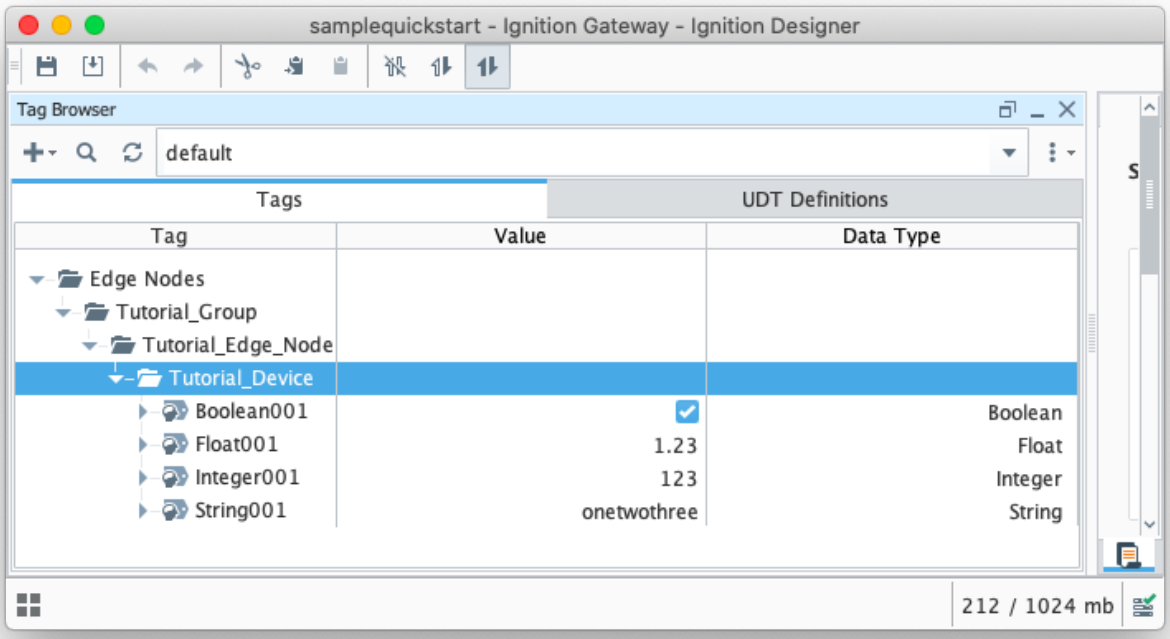
Step 2: Create tags to be published in Designer

When the Google Cloud Injector module is installed in Ignition, an Edge Node folder is automatically created in the 'default' Ignition tag provider.



Create a tree structure under this folder as shown below with some memory tags - this folder structure creates the same hierarchy that is described in the Sparkplug B specification of Group ID, Edge ID, and Device ID.

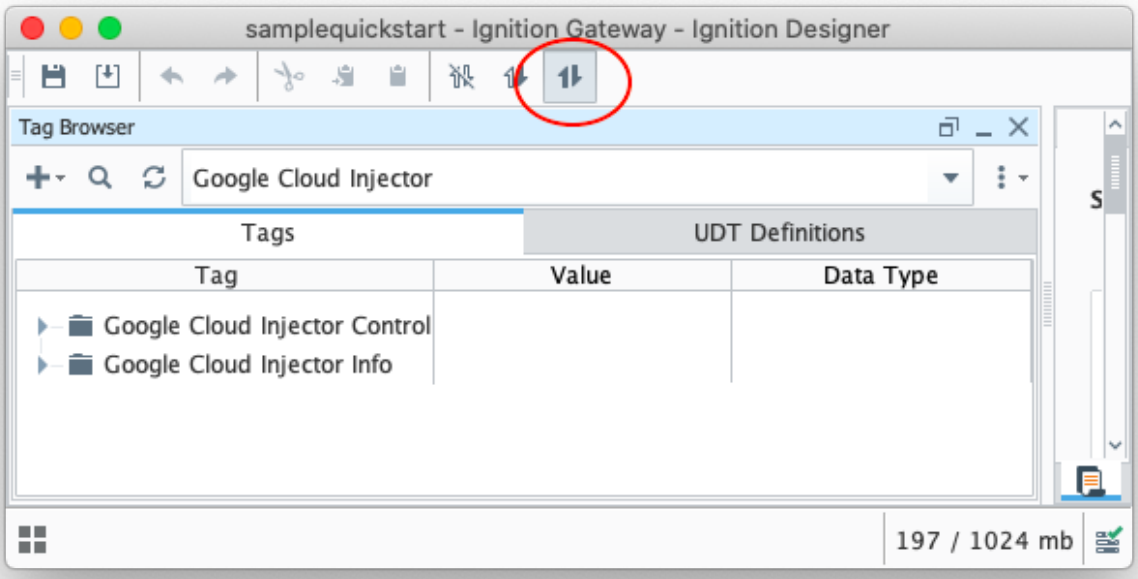
> Refer to the Ignition Tag Browser and Creating Tags documentation for assistance in configuring Ignition tags



Step 3: Publishing data

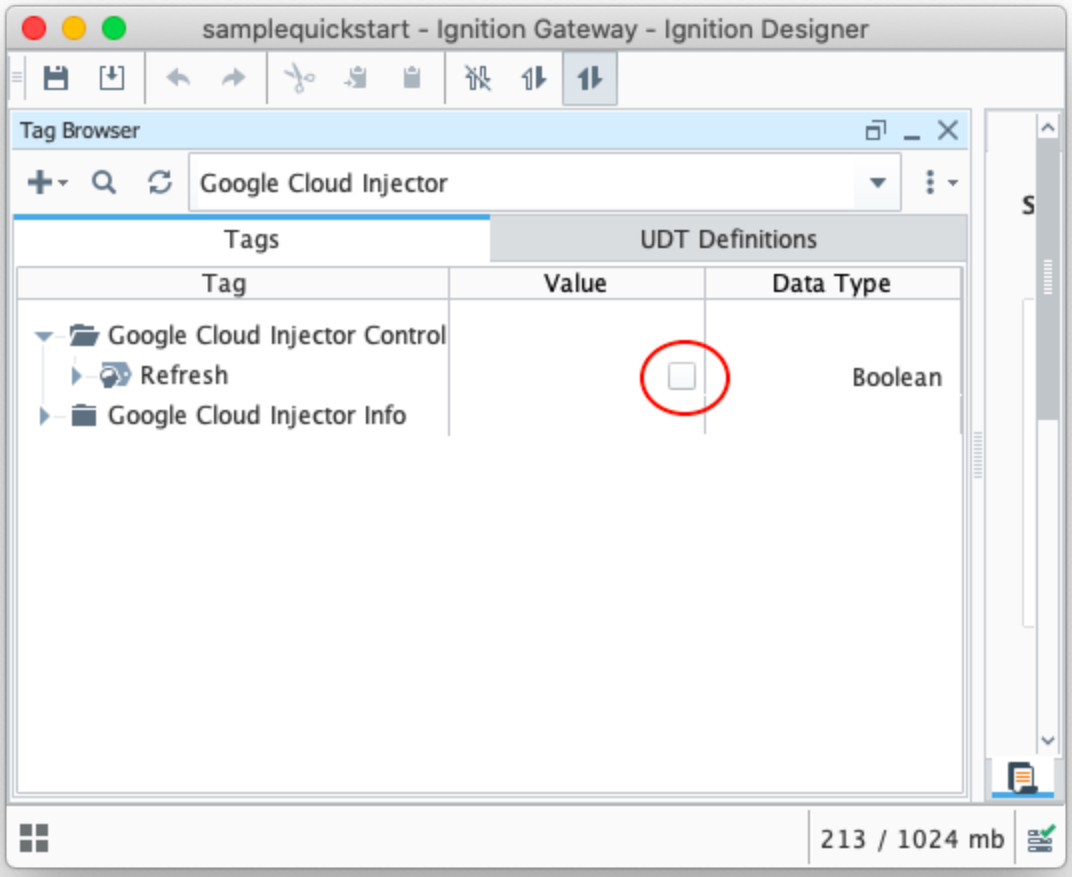
When the Google Cloud Injector module is installed in Ignition, a Google Cloud Injector tag provider is automatically created. This folder will contain both information tags about the module's version and state, as well as control tags for refreshing the module and Tag Agents.

Make sure that the Ignition Designer has read/write communications turned on by selecting the Project/Comm Read/Write button highlighted in the image below.



Review the Inductive Designer Interface documentation for additional assistance on setting the project communication mode

To refresh the default Tag Agent, open the folder "Google Cloud Injector Control" and click on the Refresh Boolean. When this happens, the Tag Agent will scan the "Edge Nodes" folder and find the new Memory Tags that we have created, construct JSON payloads representing those tags with their current values and publish the payload to the Cloud IoT Core that we have configured.



The Boolean tag will not change to true. This is really a one-shot and as a result, the tag will not change to true.

The Google Cloud Injector Tag Agent will publish two JSON payloads to the Cloud IoT Core. The format of these messages closely follows the Sparkplug B Specification's payload structure.

The first payload represents the Edge Node and will contain the following:

- The Sparkplug elements: Namespace, Group ID, Edge Node ID. They will be grouped under "topic".
- A "timestamp" for when the payload was constructed.
- A "bdSeq" sequence number to track the "session" of the Tag Agent.
- Any Edge Node tags defined in the "Tutorial_Edge_Node" folder (in our example we have none).

It will look something like this:

First Payload


```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial_Group",
    "edgeNodeId": "Tutorial_Edge_Node"
  },
  "payload": {
    "timestamp": 1504739061495,
    "metrics": [
      {
        "name": "bdSeq",
        "timestamp": 1504739061495,
        "dataType": "Int64",
        "value": 0
      }
    ],
    "seq": 0
  }
}
```


The second payload represents the Device and will contain the following:

- The Sparkplug elements: Namespace, Group ID, Edge Node ID, Device ID. They will be grouped under "topic".
- A "timestamp" for when the payload was constructed.
- Any Device tags defined in the "Tutorial Device" folder.

It will look something like this:

Second Payload


```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial_Group",
    "edgeNodeId": "Tutorial_Edge_Node",
    "deviceId": "Tutorial Device"
  },
  "payload": {
    "timestamp": 1504739061501,
    "metrics": [
      {
        "name": "Boolean001",
        "timestamp": 1504739061546,
        "dataType": "Boolean",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": true
      },
      {
        "name": "String001",
        "timestamp": 1504739061546,
        "dataType": "String",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": "onetwothree"
      },
      {
        "name": "Integer001",
        "timestamp": 1504739061546,
        "dataType": "Int32",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": 123
      },
      {
        "name": "Float001",
        "timestamp": 1504739061546,
        "dataType": "Float",
        "properties": {
          "Quality": {
            "type": "Int32",
            "value": 192
          }
        },
        "value": 1.23
      }
    ],
    "seq": 1
  }
}
```


Step 4: Use Ignition Designer to Publish Tag Data (Live Tag Changes)

Now we can change the values of the new Memory Tags and generate payloads that contain the Tag change events. Click on the value of the "Boolean001" Memory Tag to change the value.

This will result in the following payload to be constructed to represent this Tag change event and pushed to the Cloud IoT Core:

Change event Payload


```
{
  "topic": {
    "namespace": "spBv1.0",
    "groupId": "Tutorial_Group",
    "edgeNodeId": "Tutorial_Edge_Node",
    "deviceId": "Tutorial Device"
  },
  "payload": {
    "timestamp": 1504740884529,
    "metrics": [
      {
        "name": "Boolean001",
        "timestamp": 1504740883526,
        "dataType": "Boolean",
        "value": true
      }
    ],
    "seq": 2
  }
}
```


Step 5: Google Cloud Platform Applications

It is beyond the scope of this tutorial to show how to design an application in Google Cloud Platform to handle the payloads as they are pushed in to the Cloud IoT Core. For additional information on developing applications to consume this data see https://cloud.google.com/iot-core/.

Additional Resources

- Inductive Automation's Ignition download with free trial
  - Current Ignition Release
- Cirrus Link Solutions Modules for Ignition
  - Ignition Strategic Partner Modules
- Support questions
  - Check out the Cirrus Link Forum: https://forum.cirrus-link.com/
  - Contact support: support@cirrus-link.com
- Sales questions
  - Email: sales@cirrus-link.com
  - Phone: +1 (844) 924-7787
- About Cirrus Link
  - https://www.cirrus-link.com/about-us/