Publishing Data to Azure IoT Central

Prerequisites

- Ignition with the Azure Injector Modules installed
  - Azure Injector Module 8.1.10 or later is required.
  - 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 Microsoft Azure account

Summary

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

- Configuring the Azure Injector Module to connect to Azure IoT Central
- Publishing live tag data and events to Azure IoT Central

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

Tutorial

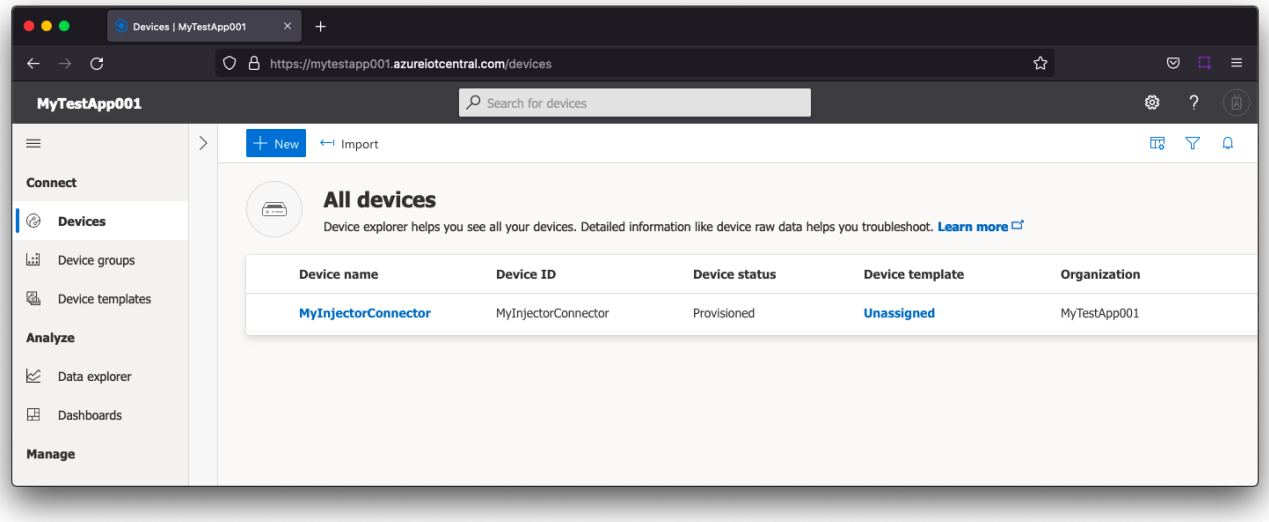
Step 1: Set up an IoT Central Application

Follow the instructions in the Microsoft IoT Central How-to Guide for creating an IoT Central Application

Step 2: Set up a Device

Follow the instructions in the Microsoft IoT Central How-to Guide for adding a device to your Azure IoT Central application

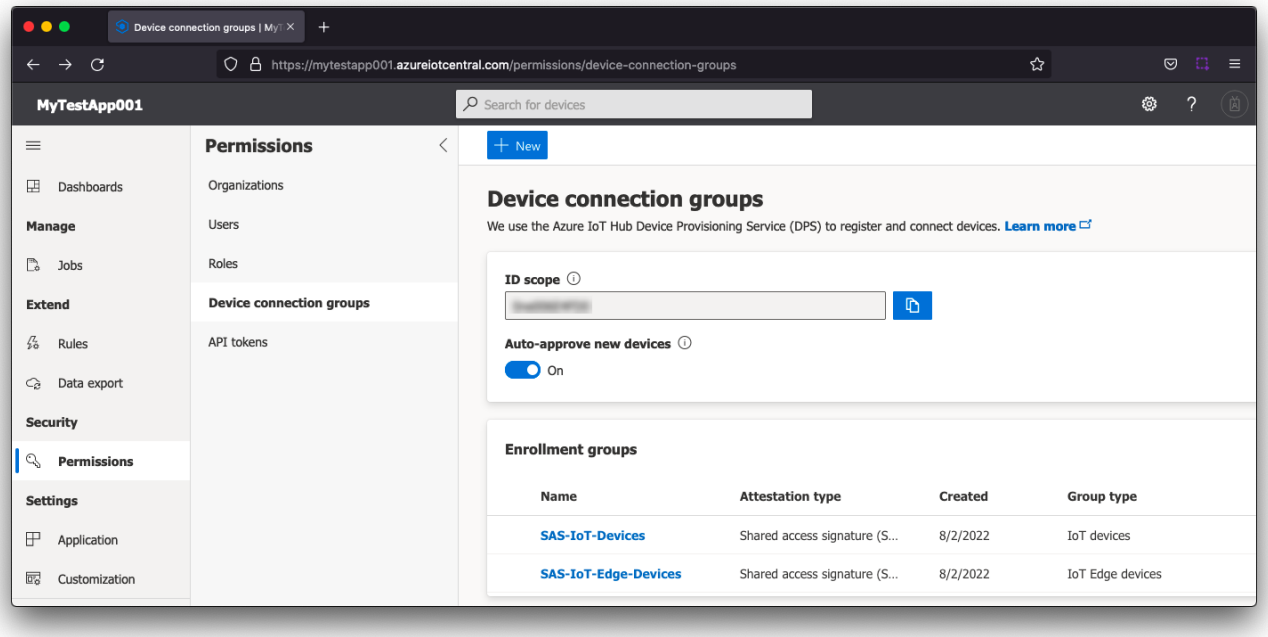
Once configured, the device should look similar to the following:



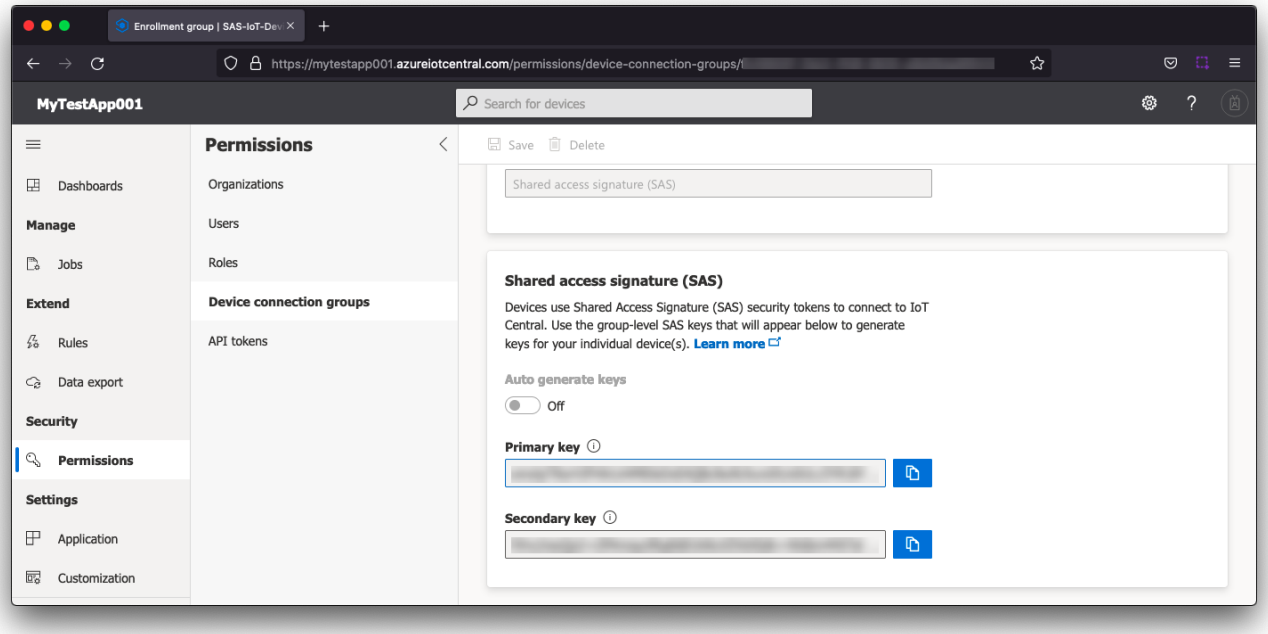
⚠ Make sure to note the 'Device ID' as this will be used later in the configuration of Azure Injector

Step 3: Record Security Permissions needed

From the left hand menu bar under Security > Permissions > Device connection groups, record the "ID scope" as this will be used later in the configuration of Azure Injector.



From the left hand menu bar under Security > Permissions > Device connection groups > SAS-IoT-Devices > Shared access signature (SAS), record either the "Primary key" or "Secondary key" as this will be used later in the configuration of Azure Injector.

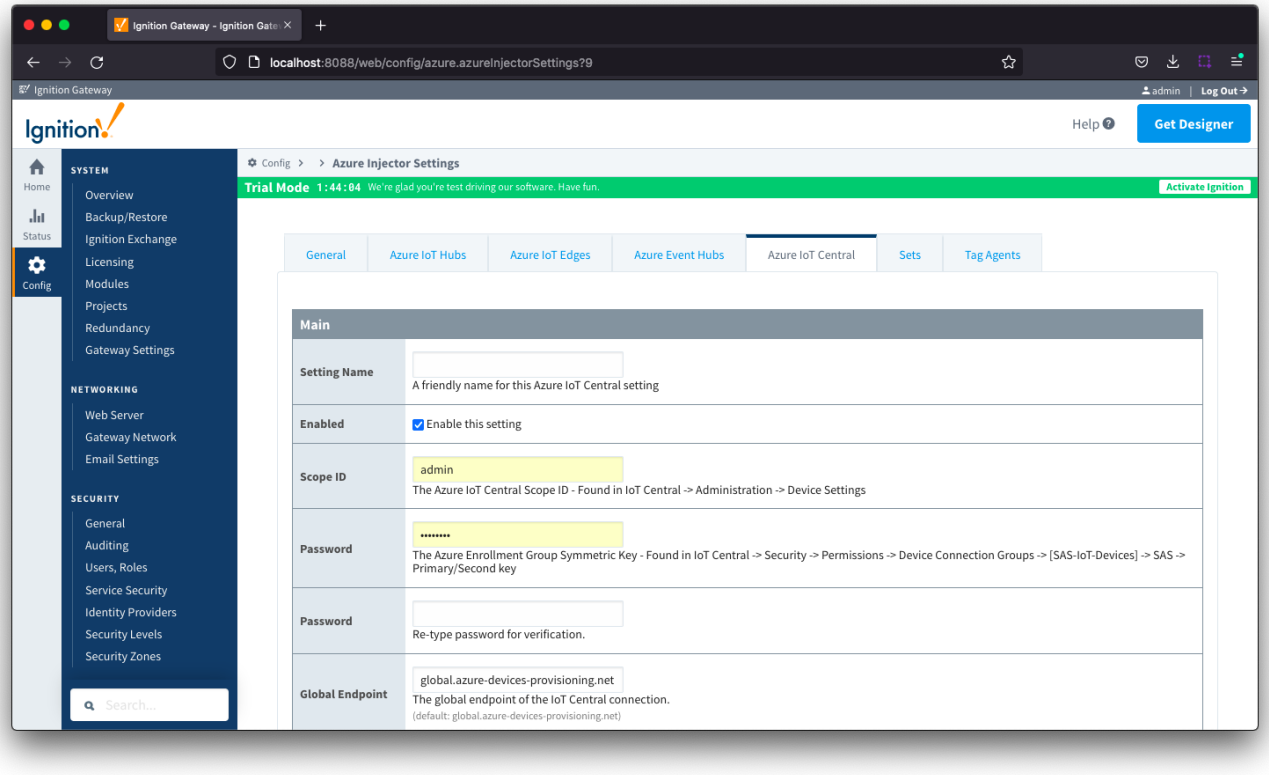


Step 4: Configure the Azure Injector Module

Once you have Ignition and the Azure Injector Module installed and running we can setup the configuration to connect to your existing Azure IoT Central endpoint.

Review the Azure Injector Module configuration guide for more details on each tab

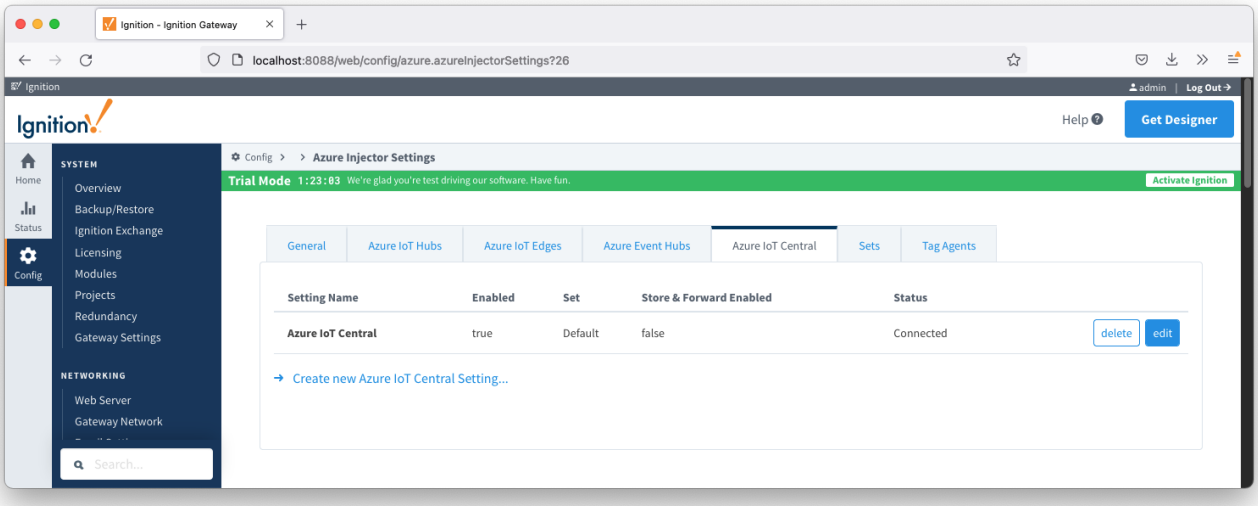
Navigate to the Azure Injector Modules configuration section from the left side bar in the Ignition Gateway and select the Azure IoT Central tab.



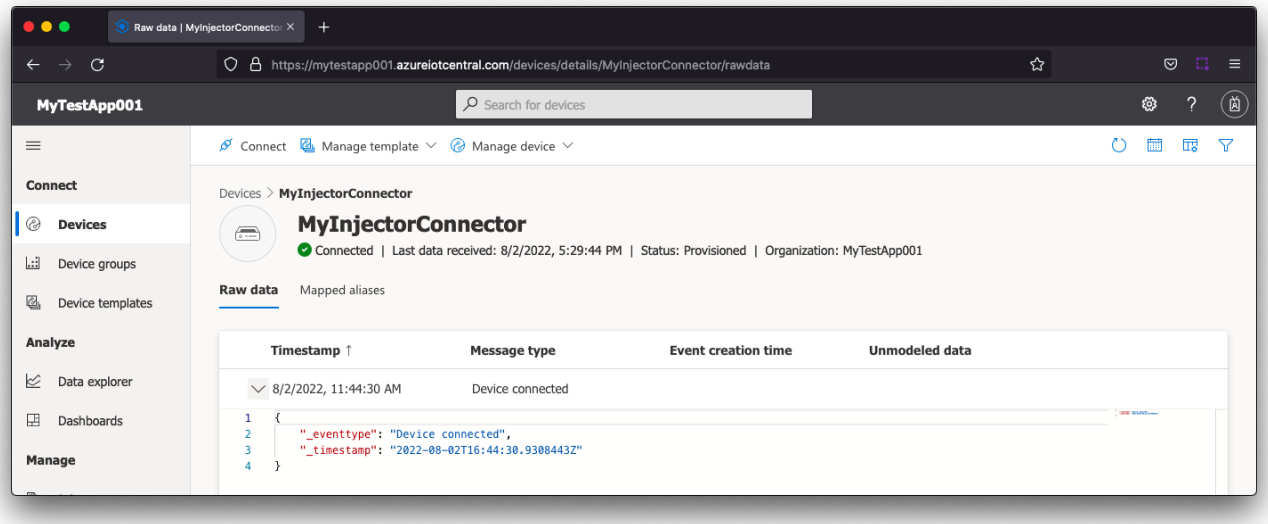
Set the following fields.

- Setting Name
  - This can be any string that makes sense that represents this connection.
- Enabled
  - Leave checked
- Scope ID
  - This is the ID scope recorded in Step 3
  - Found under Security -> Permissions -> Device connection groups and labeled 'ID scope'
- Password (Azure Enrollment Group Symmetric Key)
  - This is the Primary key or Secondary key recorded in Step 3
  - Found under Security -> Permissions -> Device connection groups -> [SAS-IoT-Devices] -> SAS -> Primary key or Secondary key. Either key can be used for the connection.
- Global Endpoint
  - Leave default
- Provisioned Device ID
  - This is the Device ID that was provisioned in Step 2 of this tutorial

All other fields can remain default. Finally, click 'Save Changes' at the bottom of the configuration page. After a bit of time (about 30s or so) you should see the status go from 'Disconnected' to 'Connected' as shown below.



Also, in the IoT Central Application portal under devices you should see the device is connected.



Now the Azure Injector module is connected to the MQTT server in Azure IoT Central, 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.

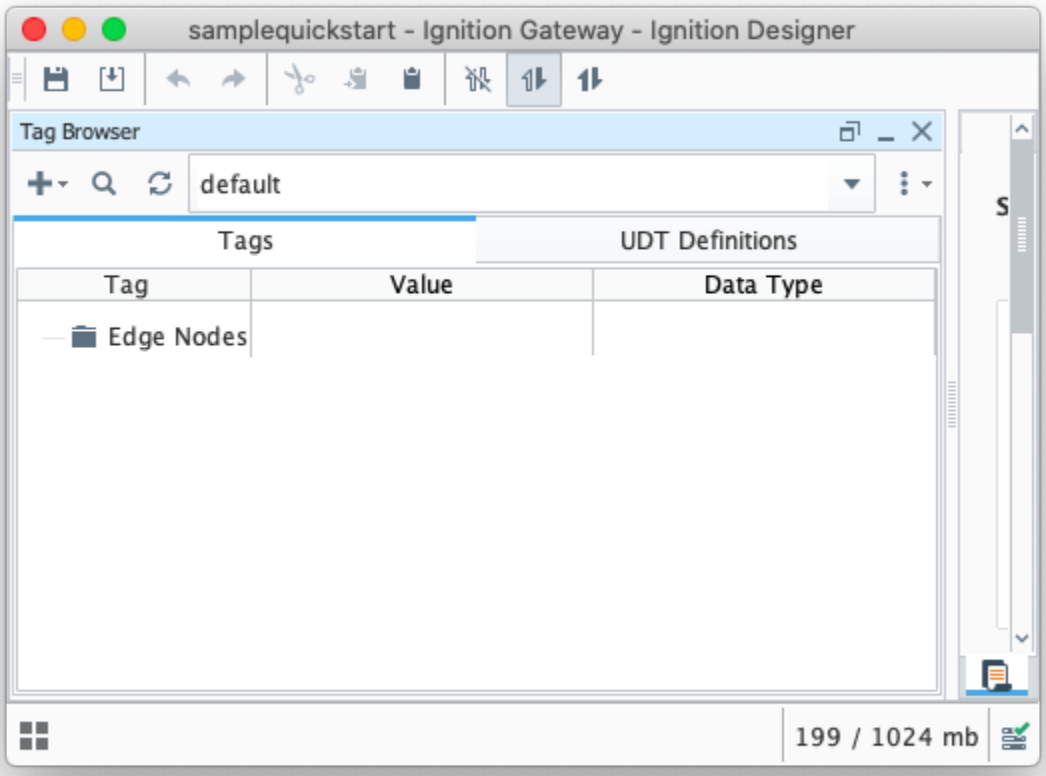
> Review the Cloud Injector Tag Agents and Tag Trees document which describes how Cloud Injector Agent configurations interact with Ignition tag trees to push messages and tag change events to the cloud service. It explains how tags get identified to be pushed as well as what specific 'topics' will be 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 6: Publishing data.

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

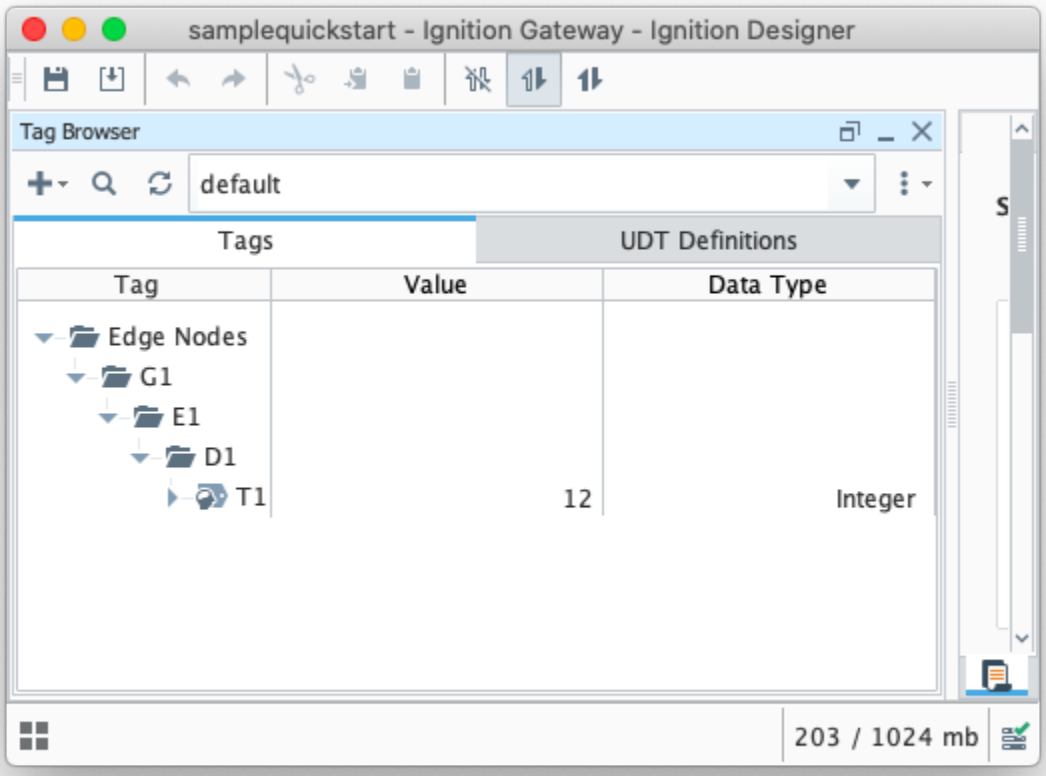
Step 5: Create tags to be published in Designer

When the Azure 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.

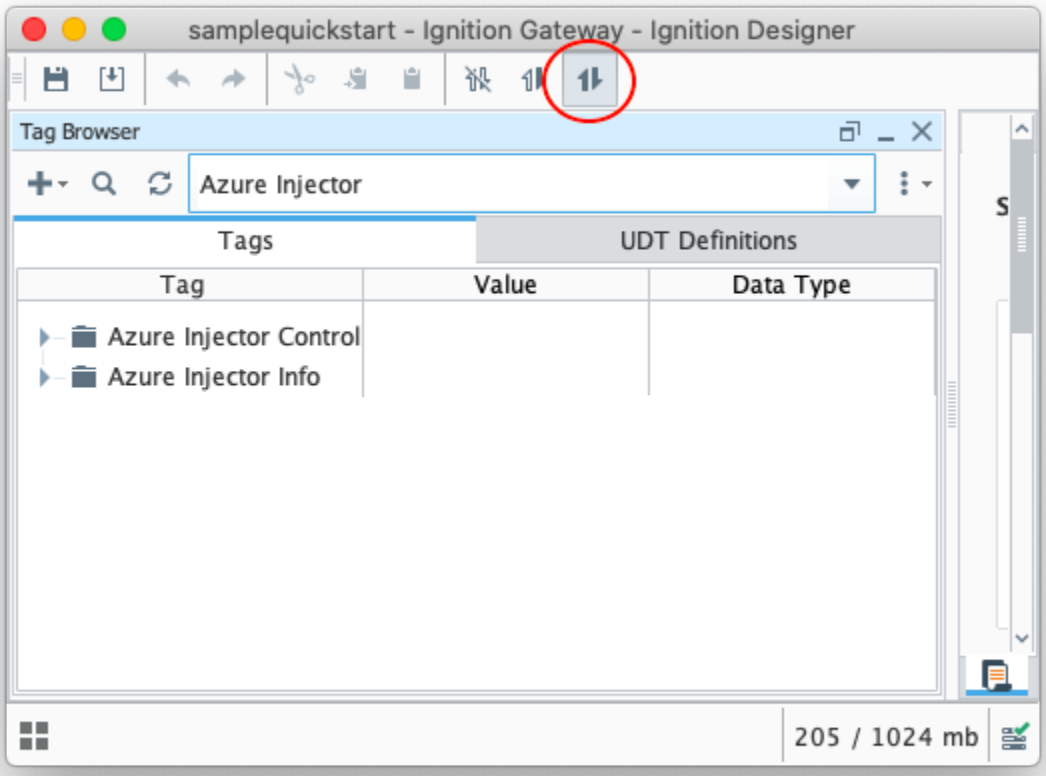
Refer to the Ignition Tag Browser and Creating Tags documentation for assistance in configuring Ignition tags



Step 6: Publishing data

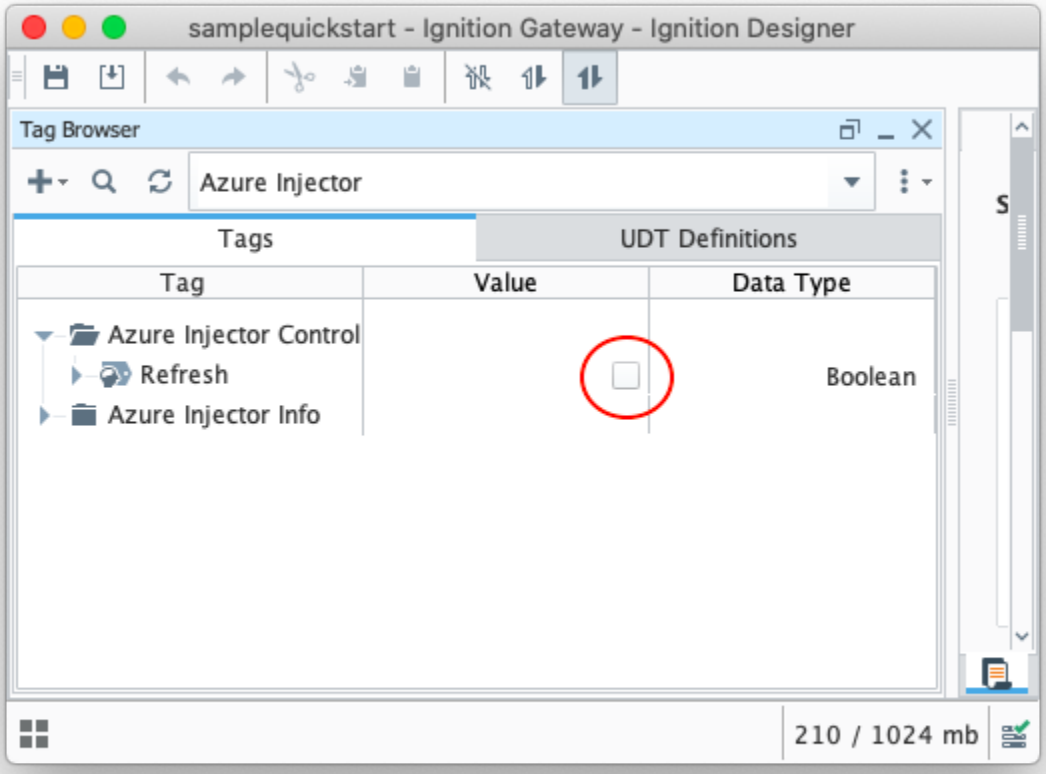
When the Azure Injector module is installed in Ignition, an Azure Injector tag provider is automatically created. This folder will contain both information tags about the module's version and state, as well as control tags for refreshing the module and Tag Agents.

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 "Azure 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 Azure IoT Central endpoint that we have configured.

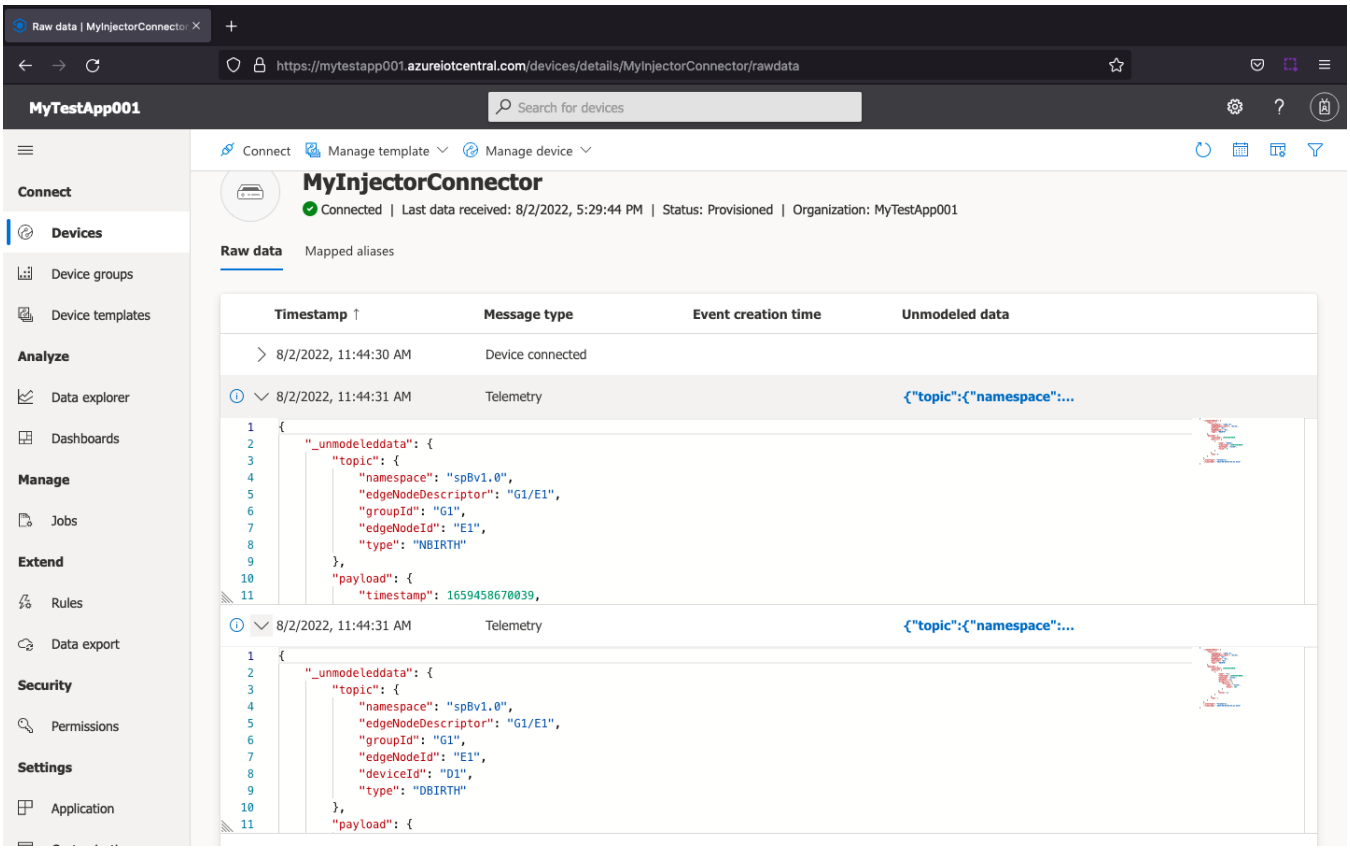


> ⚠ The Boolean tag will not change to true. This is really a one-shot and as a result, the tag will not change to true.

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

The first message shows the 'NBIRTH' message which is an indication that the Sparkplug Edge Node has come online.

The second message is a Sparkplug DBIRTH message denoting that a Sparkplug Device has come online along with its 'metrics' or tags, tag metadata, and values. In this case only a single tag is included in the payload.



This includes the following data messages.

```
{
    "_unmodeleddata": {
        "topic": {
            "namespace": "spBv1.0",
            "edgeNodeDescriptor": "G1/E1",
            "groupId": "G1",
            "edgeNodeId": "E1",
            "type": "NBIRTH"
        },
        "payload": {
            "timestamp": 1659458670039,
            "metrics": [
                {
                    "name": "bdSeq",
                    "timestamp": 1659458670039,
                    "dataType": "Int64",
                    "value": 0
                }
            ],
            "seq": 0
        }
    },
    "_eventtype": "Telemetry",
    "_timestamp": "2022-08-02T16:44:31.511Z"
}
```

```
{
    "_unmodeleddata": {
        "topic": {
            "namespace": "spBv1.0",
            "edgeNodeDescriptor": "G1/E1",
            "groupId": "G1",
            "edgeNodeId": "E1",
            "deviceId": "D1",
            "type": "DBIRTH"
        },
        "payload": {
            "timestamp": 1659458670049,
            "metrics": [
                {
                    "name": "T1",
                    "timestamp": 1659458670049,
                    "dataType": "Int32",
                    "metaData": {},
                    "properties": {
                        "Quality": {
                            "type": "Int32",
                            "value": 192
                        }
                    },
                    "value": 12
                }
            ],
            "seq": 1
        }
    },
    "_eventtype": "Telemetry",
    "_timestamp": "2022-08-02T16:44:31.527Z"
}
```


Step 7: Force a data change

Because Azure Injector is driven by tag change events, try writing a '10' to the T1 tag. Do this by double clicking the T1 tag in Designer and updating the Value parameter.

This will result in a new DDATA message as shown below.


```
{
    "_unmodeleddata": {
        "topic": {
            "namespace": "spBv1.0",
            "edgeNodeDescriptor": "G1/E1",
            "groupId": "G1",
            "edgeNodeId": "E1",
            "deviceId": "D1",
            "type": "DDATA"
        },
        "payload": {
            "timestamp": 1659462601542,
            "metrics": [
                {
                    "name": "T1",
                    "timestamp": 1659462601542,
                    "dataType": "Int32",
                    "value": 10
                }
            ],
            "seq": 2
        }
    },
    "_eventtype": "Telemetry",
    "_timestamp": "2022-08-02T17:50:02.771Z"
}
```

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/