

# MQTT Engine Tag Latching

## Prerequisites

Familiarity with creating event scripts and tag trees in Designer

Installation of Cirrus Link Modules at v4.0.10 or greater and familiarity with configuring the modules

- Distributor
- Transmission
- Engine

## Abstract

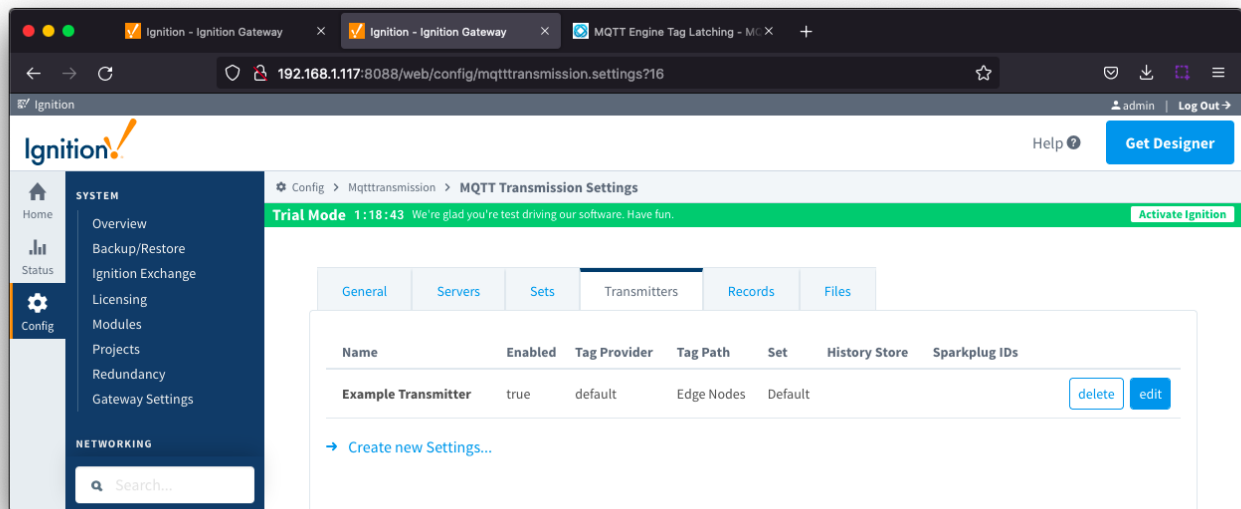
Tag Latching is used for synchronizing events at an Ignition Gateway running MQTT Engine when using tag change scripts.

If tag change events are occurring very quickly (many times per second) this can lead to synchronization problems in tag change scripts and the use of trigger and latch tags can be used resolve this. Another scenario for rapidly changing tags is when the Edge Node is flushing large volumes of historical data and MQTT Engine is configured to write these historical events directly to the tag.

In this tutorial we will show how to use trigger and tag latches to synchronize events.

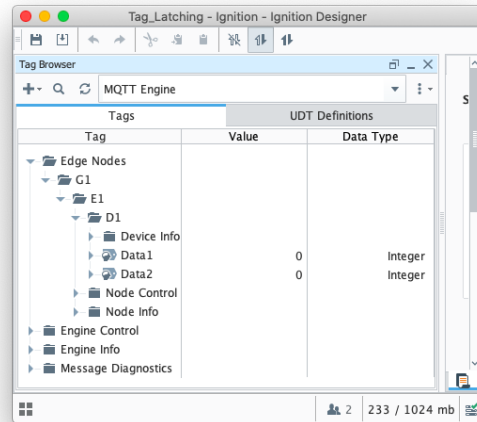
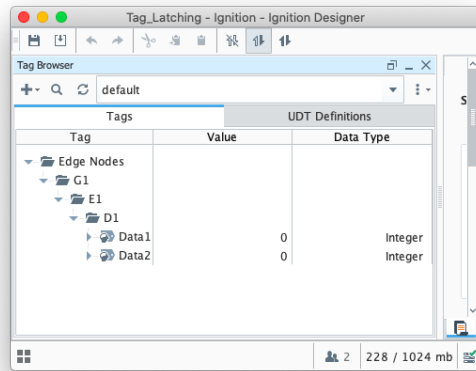
## System Setup - Transmission

In the MQTT Transmission module, create a Transmitter pointing to the default Tag Provider with Edge Nodes as the Tag Path:

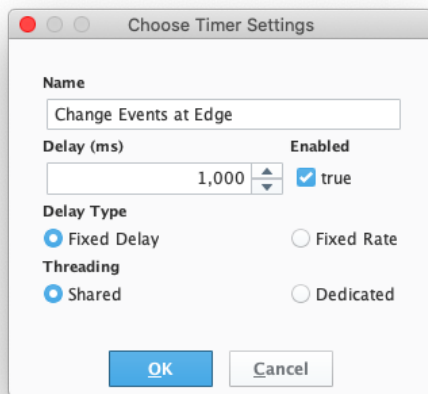


## System Setup - Designer

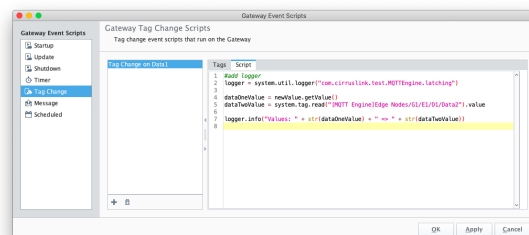
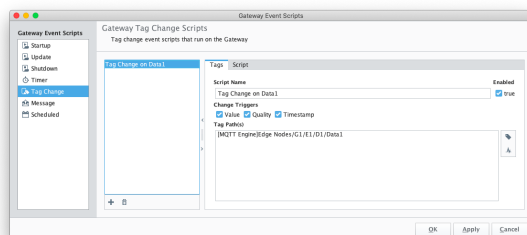
In Designer, create a file structure under the default tag provider as shown below and confirm that the data tags are reported in MQTT engine:



Now to simulate change events at the Edge (MQTT Transmission), we will add the following Gateway Event Timer script with a delay of 1,000ms which will write the same value to both Data1 and Data2 tags once per second. (Scripts to use/copy can be found [here](#))



We will also add a Gateway Tag Change script for that compares the Data1 and Data2 tags at MQTT Engine to verify they are the same using [MQTT Engine]Edge Nodes/G1/E1/D1/Data1 as our trigger tag. (Scripts to use/copy can be found [here](#))



Make sure to save your project in Designer for these changes to take effect.

Now when we look the the Logs in Ignition, we can see the following output as expected showing Data1 and Data2 have the same value.

The screenshot shows the Ignition Gateway web interface. The left sidebar contains navigation links for Home, Status, and Config. The main content area displays the Logs page, showing a list of log entries. The log entries are filtered by 'latching' and show values for Data1 and Data2. The messages are: 'Values: 92 => 92', 'Values: 91 => 91', 'Values: 90 => 90', 'Values: 89 => 89', 'Values: 88 => 88', 'Values: 87 => 87', 'Values: 86 => 86', 'Values: 85 => 85', 'Values: 84 => 84', 'Values: 83 => 83', and 'Values: 82 => 82'. A green banner at the top indicates 'Trial Mode 0:41:08'.

The problem arises when MQTT Engine is calling 'update tag' on both Data1 and Data2 very quickly (many times a second). This can lead to synchronization problems in our tag change script because by the time the script is reading Data2, MQTT Engine has already written a new value to it and we end up with an erroneous output.

To simulate this scenario, let's edit the Gateway Timer Script and set the delay to only 5ms

The 'Choose Timer Settings' dialog box is shown. It has a 'Name' field with the text 'Change Events at Edge'. Below it is a 'Delay (ms)' field with a value of 5. To the right of the delay field is an 'Enabled' checkbox which is checked. Below the delay field are two radio buttons for 'Delay Type': 'Fixed Delay' (selected) and 'Fixed Rate'. Below the delay type are two radio buttons for 'Threading': 'Shared' (selected) and 'Dedicated'. At the bottom are 'OK' and 'Cancel' buttons.

Now when we look the the Logs in Ignition, we can see the following output showing that the values are out of synchronization.

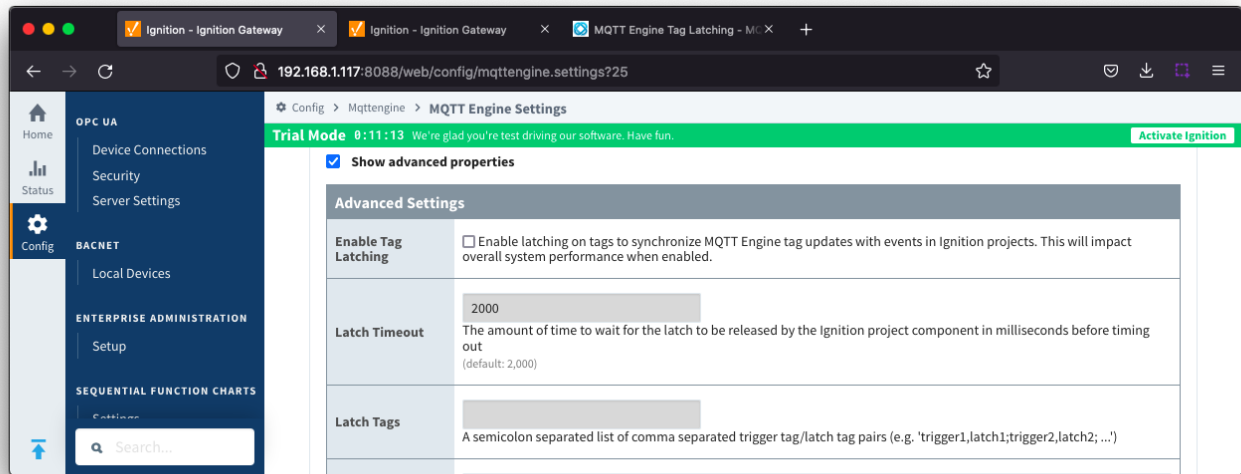
Logger	Time	Message
I latching	05Aug2022 14:04:21	Values: 159 => 159
I latching	05Aug2022 14:04:21	Values: 158 => 159
I latching	05Aug2022 14:04:21	Values: 157 => 158
I latching	05Aug2022 14:04:21	Values: 156 => 157
I latching	05Aug2022 14:04:21	Values: 155 => 156
I latching	05Aug2022 14:04:21	Values: 154 => 155
I latching	05Aug2022 14:04:21	Values: 153 => 154
I latching	05Aug2022 14:04:21	Values: 152 => 153
I latching	05Aug2022 14:04:21	Values: 151 => 152
I latching	05Aug2022 14:04:21	Values: 150 => 151
I latching	05Aug2022 14:04:21	Values: 149 => 150
I latching	05Aug2022 14:04:21	Values: 148 => 149
I latching	05Aug2022 14:04:21	Values: 147 => 148
I latching	05Aug2022 14:04:21	Values: 146 => 147
I latching	05Aug2022 14:04:21	Values: 145 => 146
I latching	05Aug2022 14:04:21	Values: 144 => 145
I latching	05Aug2022 14:04:21	Values: 143 => 144
I latching	05Aug2022 14:04:21	Values: 142 => 143
I latching	05Aug2022 14:04:21	Values: 141 => 142
I latching	05Aug2022 14:04:21	Values: 140 => 141
I latching	05Aug2022 14:04:21	Values: 139 => 140
I latching	05Aug2022 14:04:21	Values: 138 => 139
I latching	05Aug2022 14:04:21	Values: 137 => 138
I latching	05Aug2022 14:04:21	Values: 136 => 137

## Configuring Tag Latching

When this happens, we can use tag latching to synchronize MQTT Engine with the tag change script.

The configuration for Tag Latching is available under MQTT Engine General tab > Advanced Settings with the following properties:

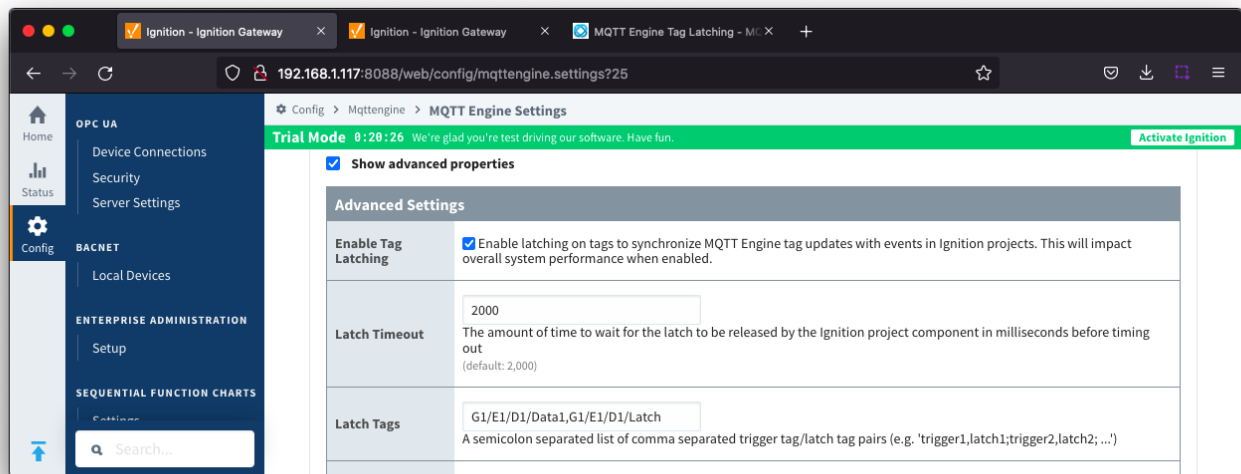
- **Enable Tag Latching**
  - Whether or not to enable Tag latching to synchronize MQTT Tag updates with events in Ignition.
- **Latch Timeout**
  - The amount of time in milliseconds MQTT Engine will wait for a tag latch to be released before it does it on its own.
- **Latch Tags**
  - A semicolon separated list comma separated list of Trigger Tag and Latch Tag pairs.
  - A 'Trigger Tag' is a 'real tag' that you would normally use in a standard Transaction Group or Tag Change Script as a trigger.
  - The 'Latch Tag' is a tag that will be created in the [MQTT Engine]Engine Info/Latches folder. Each latch tag will be set to true when MQTT Engine calls updateTag for that given trigger tag. Then MQTT Engine waits until the timeout or until the latch tag gets set back to false by a script, transaction group, etc - whichever comes first. The latch tag is what should be used by scripts or transaction groups and must be set back to false at the end of the operation to allow MQTT Engine to continue processing incoming Sparkplug messages.
  - The 'Latch Tags' must be of the form:
    - Group ID/Edge Node ID/Device ID/Trigger Tag,Group ID/Edge Node ID/Device ID/Latch Tag for example:
    - G1/E1/D1/Trigger Tag 1,G1/E1/D1/Latch Tag 1
  - You can also have longer folder paths on any trigger tag or latch tag. For example:
    - G1/E1/D1/my/longer/tag/path/Trigger Tag 1,G1/E1/D1/my/longer/path/Latch Tag 1
  - If you want to specify two or more latches, separate them with a semicolon:
    - G1/E1/D1/Trigger Tag 1,G1/E1/D1/Latch Tag 1;G1/E1/D1/Trigger Tag 2,G1/E1/D1/Latch Tag 2
  - You can also specify multiple trigger tags for any given latch. Just create two or more entries that each point to the same latch tag:
    - G1/E1/D1/Trigger Tag 1,**G1/E1/D1/Latch Tag 1**;G1/E1/D1/Trigger Tag 2,**G1/E1/D1/Latch Tag 1**
  - You can specify as many as trigger and latch tags as you want.



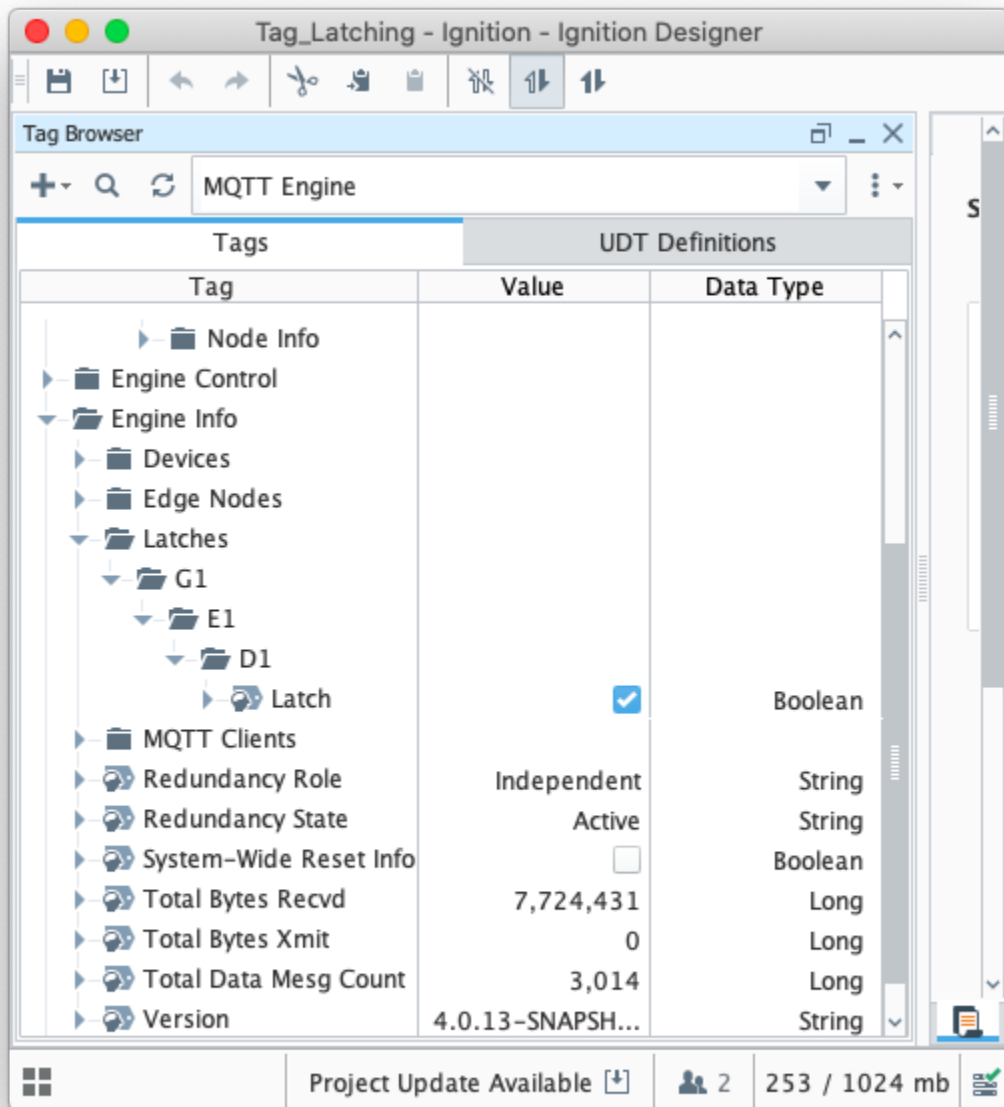
Edit the configuration as shown below to set the Latch Tags field as 'G1/E1/D1/Data1,G1/E1/D1/Latch'.



Note we are using Data1 as the trigger tag since that is the 'trigger' for our existing tag change script.



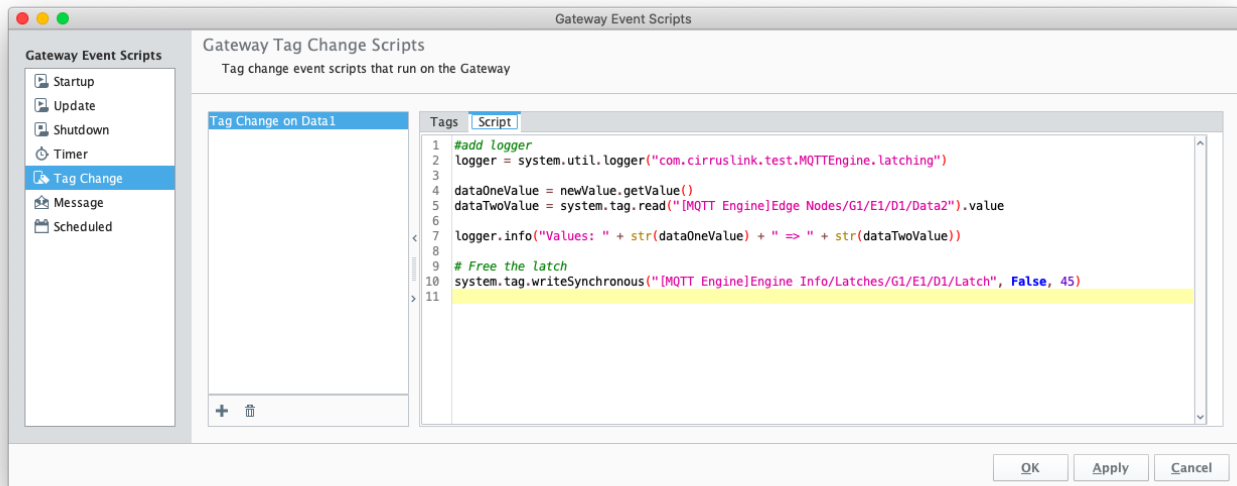
With this set, MQTT Engine will now set a new tag '[MQTT Engine]Engine Info/Latches/G1/E1/D1/Latch' to true every time the trigger tag changes.



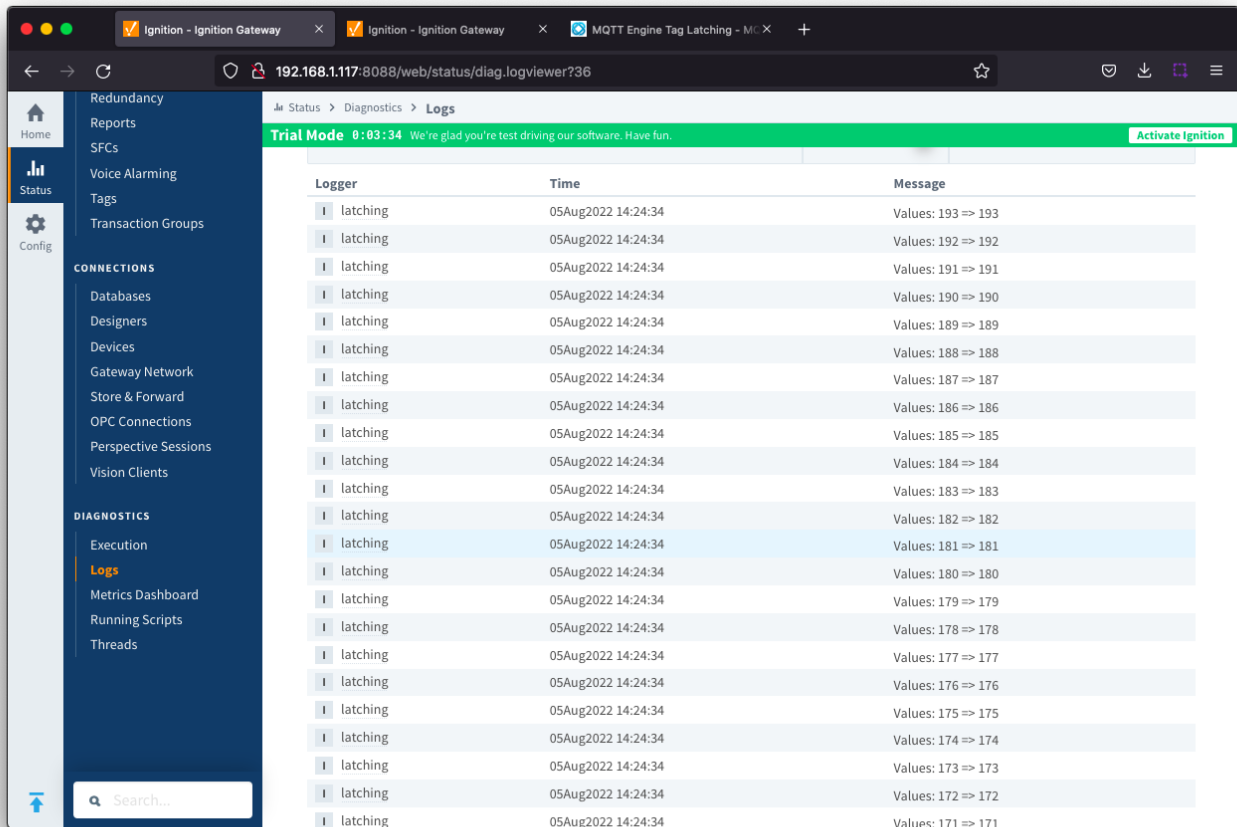
It is the responsibility of the script, transaction group, or external application to reset the latch tag to allow MQTT Engine to continue processing data normally. If this is not done MQTT Engine will stop processing incoming change events until the 'Latch Timeout' elapses.

It should also be noted that this does have a performance impact on MQTT Engine. Because we must synchronize the processing of incoming events MQTT Engine will not process these events as quickly. From the time MQTT Engine sets the latch and the script releases it, MQTT Engine pauses all processing of tag change events.

So for this example, the tag change script is modified as follows to release the latch at the end of the script. (Scripts to use/copy can be found [here](#))



Now with this setup, we can see the output is correct as shown below. Note all of this output came from within the same second.



## Gateway Event Scripts

### Gateway Timer Script

```
value = system.tag.read("[default]Edge Nodes/G1/E1/D1/Data1").value
value += 1
if value > 500:
    value = 0

system.tag.writeSynchronous("[default]Edge Nodes/G1/E1/D1/Data2", value, 30)
system.tag.writeSynchronous("[default]Edge Nodes/G1/E1/D1/Data1", value, 30)
```

### Gateway Tag Change Script

```
#add logger
logger = system.util.logger("com.cirruslink.test.MQTTEngine.latching")

dataOneValue = newValue.getValue()
dataTwoValue = system.tag.read("[MQTT Engine]Edge Nodes/G1/E1/D1/Data2").value
logger.info("Values: " + str(dataOneValue) + " => " + str(dataTwoValue))
```

### Gateway Tag Change Script with Latching

```
#add logger
logger = system.util.logger("com.cirruslink.test.MQTTEngine.latching")

dataOneValue = newValue.getValue()
dataTwoValue = system.tag.read("[MQTT Engine]Edge Nodes/G1/E1/D1/Data2").value
logger.info("Values: " + str(dataOneValue) + " => " + str(dataTwoValue))

# Free the latch
system.tag.writeSynchronous("[MQTT Engine]Engine Info/Latches/G1/E1/D1/Latch", False, 45)
```

## Additional Resources

- Inductive Automation's Ignition download with free trial
  - <https://inductiveautomation.com/downloads/>
- Azure Injector download with free trial
  - <https://inductiveautomation.com/downloads/third-party-modules>
- Questions about this tutorial?
  - Check out the Cirrus Link Forum: <https://forum.cirrus-link.com/>
  - Contact support: [support@cirrus-link.com](mailto:support@cirrus-link.com)
- Sales questions
  - Email: [sales@cirrus-link.com](mailto:sales@cirrus-link.com)
  - Phone: +1 (844) 924-7787
- About Cirrus Link
  - <https://www.cirrus-link.com/about-us/>