

Managing records with MQTT Transmission and MQTT Recorder

Prerequisites

- Knowledge of Ignition and Module installation process: [Cirrus Link Module Installation](#)
- Installation of the following Cirrus Link MQTT modules selecting the [version compatible](#) with your Ignition installation. Module can be downloaded from the [Ignition Strategic Partner Modules](#) download page:
 - MQTT Distributor
 - MQTT Engine
 - MQTT Transmission
 - MQTT Recorder
- Installed SQL database such as [MySQL \(Ver 14.14 Distrib 5.7.12\)](#)

Overview

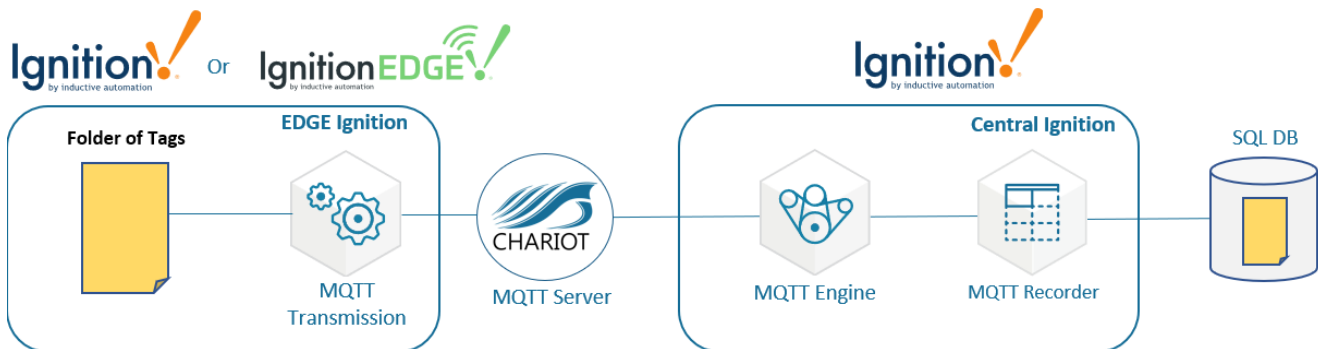
Within MQTT Transmission, a record is a collection of tags under an Ignition folder which are treated as a single entity and published on demand. They are used for sending a record of information such as a batch data entry form or flow computer event/alarm record.

Records are published via an MQTT client using a Sparkplug-like topic format with a protobuf payload.

Records received at MQTT Engine are passed to MQTT Recorder which automatically generates database tables and populates the rows within the tables with records as they are received.

This tutorial shows how to configure the MQTT modules to generate and publish records to be stored in an MySQL database including:

- [Step 1: Install the MQTT modules](#)
- [Step 2: Configure a database connection](#)
- [Step 3: Import tags to create Records in the Ignition tag provider](#)
- [Step 4: Configure the MQTT Transmission Records](#)
- [Step 5: Configure the MQTT Recorder module](#)
- [Step 6: Publish the records](#)
- [Step 7: Viewing the messages in the Ignition logs](#)
- [Video](#)



The topology of this example shows MQTT Distributor, MQTT Engine, MQTT Recorder Store, and MQTT Transmission all running in the same Ignition instance. This is done for simplicity of the tutorial, but this isn't required or even intended to be a real use case. In a more realistic scenario MQTT Transmission and MQTT Engine would be located on separate machines.

Step 1: Install the MQTT modules

Install the four MQTT modules listed in the pre-requisites onto your Ignition system following the [Cirrus Link Module Installation](#) guide.

By default, both MQTT Engine and MQTT Transmission are configured to connect to MQTT Distributor on [tcp://localhost:1883](#) and will show as Connected under their respective Servers configuration setting in the Ignition UI. MQTT Transmission will also have an Example Transmitter configured pointing to a set of tags that are configured in the Ignition "default" tag provider.



Review the [MQTT Transmission Transmitters and Tag Trees](#) tutorial for additional information on how Transmitter configurations interact with Ignition tag trees to create the Sparkplug IDs required.


This allows the three modules to automatically connect and provide a starting base for the tutorial.

Step 2: Configure a database connection

Follow the Ignition [Connecting to MySQL](#) guide to configure a database connection to your SQL database.


Step 3: Import tags to create Records in the Ignition tag provider

Within MQTT Transmission, a record is a collection of tags under an Ignition folder which are treated as a single entity and published on demand.

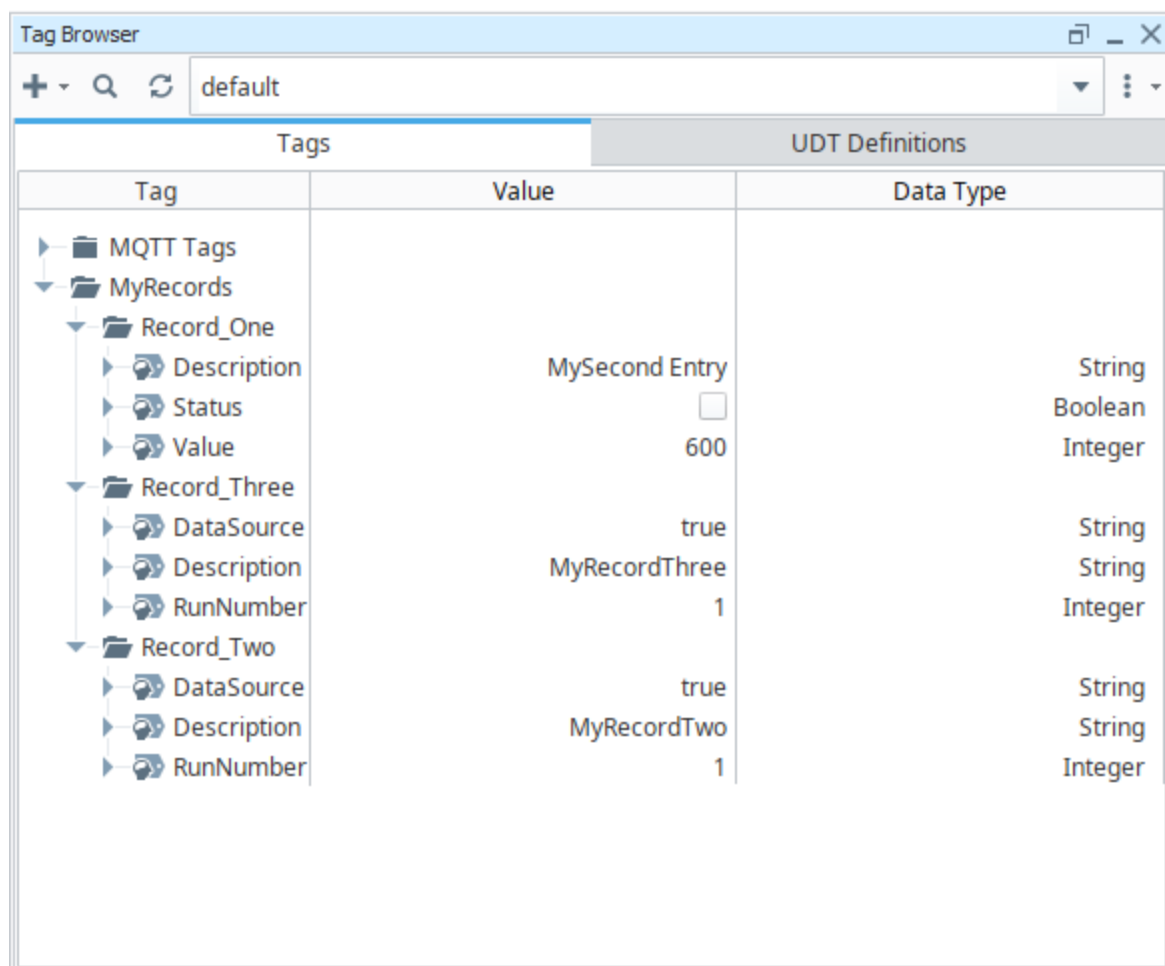
 UDT instances are not currently supported as a record tag

The next step is to create the Ignition folders representing the Records within Ignition.

Use the Design Launcher to launch a Designer connection to the Ignition Gateway and create a new project. In the 'default' tag provider, [import the recorder_tutorial_tags.json](#) to build a set of Ignition folders to represent the records.

 Note the set of tags under the MQTT Tags folder that were automatically configured when the MQTT Transmission module was installed

The imported tags will create three records as shown below:

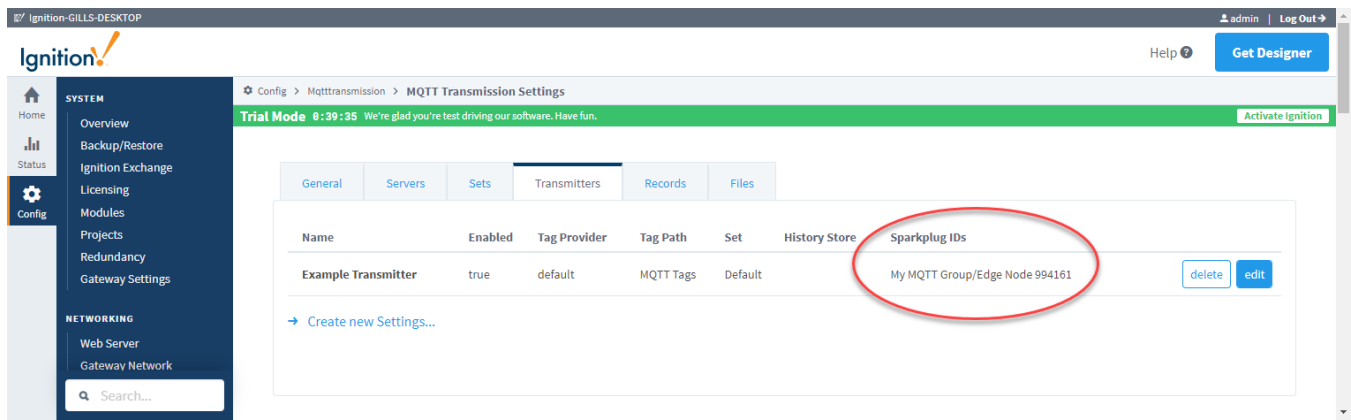


Tags		UDT Definitions	
Tag	Value	Data Type	
MQTT Tags			
MyRecords			
Record_One			
Description	MySecond Entry	String	
Status	<input type="checkbox"/>	Boolean	
Value	600	Integer	
Record_Three			
DataSource	true	String	
Description	MyRecordThree	String	
RunNumber	1	Integer	
Record_Two			
DataSource	true	String	
Description	MyRecordTwo	String	
RunNumber	1	Integer	

Step 4: Configure the MQTT Transmission Records

Now we can configure the MQTT Transmission module to publish records using the existing MQTT client created by the Example Transmitter. Navigate to the MQTT Transmission > Settings in the left side bar of the Ignition Gateway UI and select the Transmitters tab. Make note of the Sparkplug IDs configured for your transmitter.

For our example we have the Group ID as *My MQTT Group* and the Edge Node ID as *Edge Node 994161*



Now we can setup three configurations each of which points to a record folder and also configs the boolean tag which controls the on demand publish of the record.



For this tutorial we will manually trigger the record publishes however these tag changes can easily be scripted at the Edge or managed through control writes from MQTT Engine

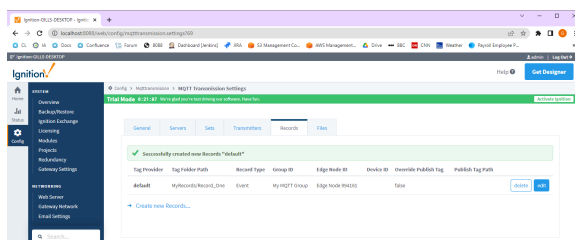


Review the MQTT Transmission Configuration guide for more details on the [MQTT Transmission Records](#) module

Select the Records tab and then click the Create new Records.. link. Use the following configuration settings:

- Tag Settings
 - Tag Provider
 - default
 - Tag Folder Path
 - MyRecords/Record_One
 - Record Type
 - Event
- Sparkplug Settings
 - Group ID
 - My MQTT Group
 - Edge Node ID
 - Use the Edge Node ID from your Transmitters Sparkplug IDs. For my Transmitter, I will use Edge Node 994161
 - Device ID
 - Leave blank
- Advanced Settings
 - Override Publish Tag
 - Leave as unchecked. This will create a boolean tag named *Publish* in the Tag Folder Path *MyRecords/Record_One* which will be used to trigger the record publish
 - Publish Tag Path
 - Leave blank

Select Create New Record and you will see the Publish boolean in the tag browser

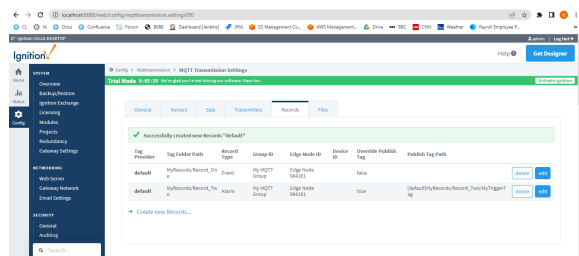


Tags		UDT Definitions	
Tag	Value		Data Type
MQTT Tags			
MyRecords			
Record_One			
Description	MySecond Entry		String
Publish	<input type="checkbox"/>		Boolean
Status	<input type="checkbox"/>		Boolean
Value	600		Integer
Record_Three			
Record_Two			

Repeat the steps to add a second record configuration using the following configuration settings:

- Tag Settings
 - Tag Provider
 - default
 - Tag Folder Path
 - MyRecords/Record_Two
 - Record Type
 - Alarm
- Sparkplug Settings
 - Group ID
 - My MQTT Group
 - Edge Node ID
 - Use the Edge Node ID from your Transmitters Sparkplug IDs. For my Transmitter, I will use Edge Node 994161
 - Device ID
 - Leave blank
- Advanced Settings
 - Override Publish Tag
 - Check this to allow use to define how to create the boolean tag which will be used to trigger the record publish
 - Publish Tag Path
 - [default]MyRecords/Record_Two/MyTriggerTag

Select Create New Record and you will see the MyTriggerTag boolean in the MyRecords/Record_Two folder



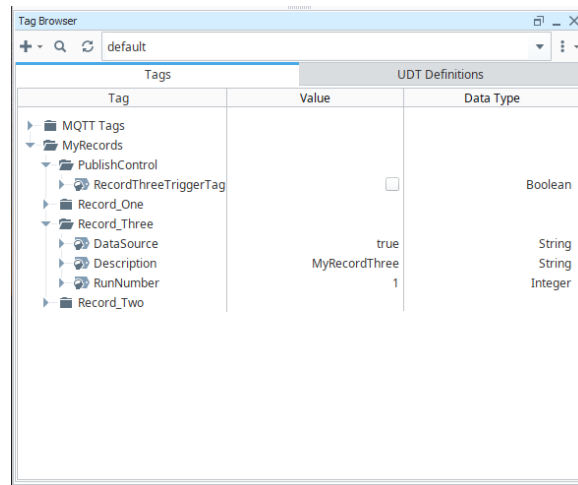
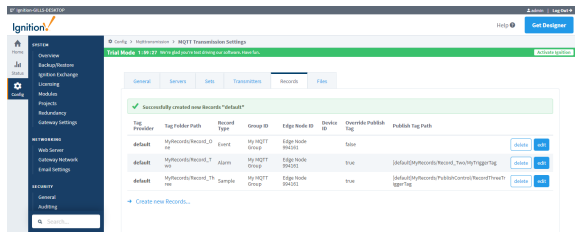
Tags		UDT Definitions	
Tag	Value		Data Type
MQTT Tags			
MyRecords			
Record_One			
Record_Three			
Record_Two			
DataSource	true		String
Description	MyRecordTwo		String
MyTriggerTag	<input type="checkbox"/>		Boolean
RunNumber	1		Integer

Repeat the steps to add a third record configuration using the following configuration settings:

- Tag Settings
 - Tag Provider
 - default
 - Tag Folder Path
 - MyRecords/Record_Three

- Record Type
 - Sample
- Sparkplug Settings
 - Group ID
 - My MQTT Group
 - Edge Node ID
 - Use the Edge Node ID from your Transmitters Sparkplug IDs. For my Transmitter, I will use Edge Node 994161
 - Device ID
 - Leave blank
- Advanced Settings
 - Override Publish Tag
 - Check this to allow use to define how to create the boolean tag which will be used to trigger the record publish
 - Publish Tag Path
 - [default]MyRecords/PublishControl/RecordThreeTriggerTag

Select Create New Record and you will see the RecordThreeTriggerTag created in a new PublishControl folder

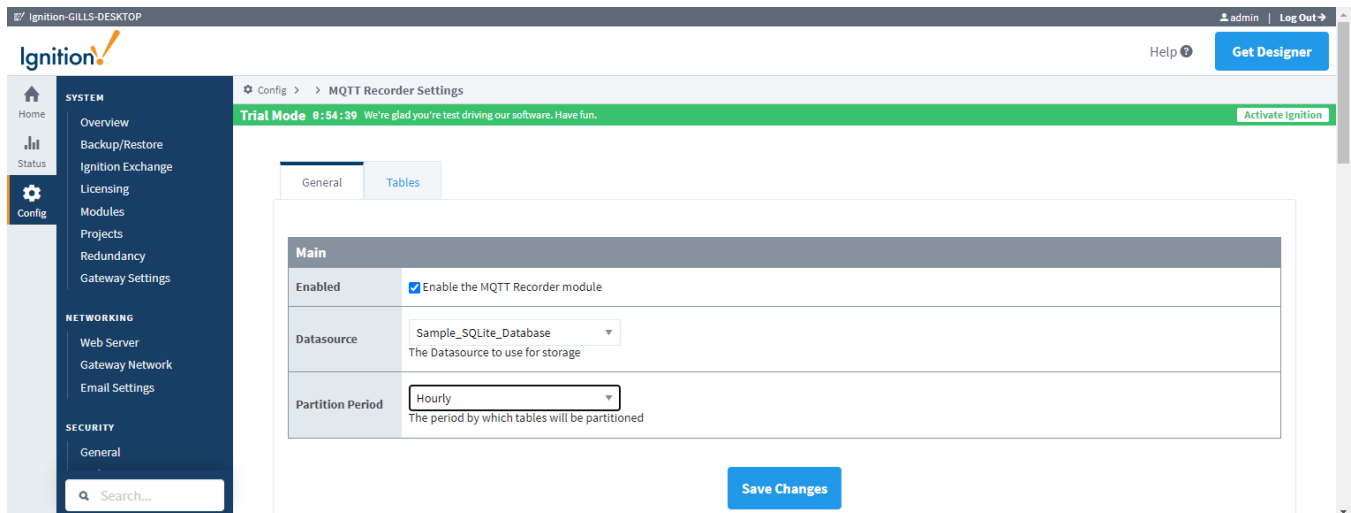


Step 5: Configure the MQTT Recorder module

Now we can configure the MQTT Recorder module to receive the published records and store in a database. Navigate to the MQTT Recorder > Settings in the left side bar of the Ignition Gateway UI and select the General tab.

Select the Datasource as the database connection configured in Step 2.

Select a Partition Period - in this tutorial we will use "Hourly"



Select the Tables tab. By default, a single table will be defined for MQTT Recorder with a table name of rs_\$(type). With this single table definition, a separate table will be created for each unique record type.



Review the [MQTT Recorder Configuration guide](#) for more details on the MQTT Recorder module including the filters available for configuring dynamic custom table names using tokens

Step 6: Publish the records

Now we are ready to publish records where the following process will occur:

- MQTT Transmission constructs a Record containing Fields representing the Tags in the Record Tag Folder Path folder along with their current values.
- MQTT Transmission publishes the Record
- MQTT Engine receives the Record and delivers it to the MQTT Recorder module
- MQTT Recorder stores the Record in the database

Let's start by publishing Record_One. To do this, in Designer click the [default]MyRecords/Record_One/Publish tag - it will automatically become unchecked once the publish has occurred.

Browse to your database configured in Step 2. You will see a new table created with the table name of rs_event_YYYY_MM_DD_hh - this is because we configured the Record Type as Event for the Record_One folder path.

Query the database table and you will see the DB schema created as follows:

- rs_id
 - Record id
- rs_type
 - Record type
- rs_group
 - Sparkplug group ID
- rs_edge_node
 - Sparkplug edge node ID
- rs_device
 - Sparkplug device ID
- rs_record_time
 - Timestamp for record creation at Edge
- rs_recoder_time
 - Timestamp for record insert into database
- rs_fields
 - Pipe delimited description for record field and data type
 - Note that this is for internal Cirrus Link module use and should not be edited
- Fields for each of the tags contained in the folder
 - Status
 - Value
 - Description

The screenshot displays the Ignition Designer interface. The 'Database Query Browser' window shows a query: `SELECT * FROM rs_event_2022_12_20_11`. Below the query, a 'Resultset 1' table is visible with columns: rs_id, rs_type, rs_group, rs_edge_node, rs_device, rs_record_time, rs_recoder_time, rs_fields, Status, Value, and Description. The first row contains the following data: 1, Event, My MQTT Gro..., Edge Node 994161, 1671557077718, 1671557078310, Status|11,Description|12,Val..., 0, 600, MySecond Entry. The 'Tag Browser' window on the right shows a tree structure under 'default' with folders for 'MQTT Tags', 'MyRecords', 'PublishControl', 'RecordThreeTriggerTag', 'Record_One', 'Record_Two', and 'Record_Three'. The 'Publish' tag under 'Record_One' is selected, showing its value as 'MySecond...' and data type as 'Boolean'. The 'Schema History' window at the bottom right shows the schema for the selected tag, listing fields like rs_id (INTEGER), rs_device (TEXT), rs_edge_node (TEXT), rs_fields (TEXT), rs_group (TEXT), rs_record_time (INTEGER), rs_recoder_time (INTEGER), rs_type (TEXT), Status (INTEGER), and Value (INTEGER).

Change the [default]MyRecords/Record_One/Value tag to a new value and re-trigger the Publish tag. Query the database table to see the second record in the database.

Database Query Browser

```
SELECT * FROM rs_event_2022_12_20_11
```

Limit SELECT to: 1000 rows

rs_id	rs_type	rs_group	rs_edge_node	rs_device	rs_record_time	rs_recorder...	rs_fields	Status	Value	Description
1	Event	My MQTT Group	Edge Node 994161	167155707718	1671557078...	1671557875...	Status 11, Descripti...	0	600	MySecond Entry
2	Event	My MQTT Group	Edge Node 994161	1671557874538	1671557875...	1671557875...	Status 11, Descripti...	0	900	MySecond Entry

2 rows fetched in 0.003s

Tag Browser

Tag	Value	Data Type
MyRecords/RecordThreeTriggerTag/Record_One/Value	900	Integer

Repeat the process for the other two records by selecting the `[default]MyRecords/Record_Two/MyTriggerTag` and `[default]MyRecords/PublishControl/RecordThreeTriggerTag` tags.

In the database you will see two additional tables named `rs_alarm_YYYY_MM_DD_hh` and `rs_sample_YYYY_MM_DD_hh`

Querying the DB will show the two records:

Database Query Browser

```
SELECT * FROM rs_alarm_2022_12_20_11
```

Limit SELECT to: 1000 rows

rs_id	rs_type	rs_group	rs_edge_node	rs_device	rs_record_ti...	rs_recorder...	rs_fields	Description	RunNumber	DataSource
1	Alarm	My MQTT Group	Edge Node 994161	1671558108...	1671558109...	1671558109...	Description 12, Ru...	MyRecordTwo	1	true

1 row fetched in 0.002s

Tag Browser

Tag	Value	Data Type
MyRecords/RecordThreeTriggerTag/Record_Two/Value	true	String

Database Query Browser

```
SELECT * FROM rs_sample_2022_12_20_11
```

Limit SELECT to: 1000 rows

rs_id	rs_type	rs_group	rs_edge_node	rs_device	rs_record_ti...	rs_recorder...	rs_fields	Description	RunNumber	DataSource
1	Sample	My MQTT Group	Edge Node 994161	1671558111...	1671558111...	1671558111...	Description 12, Ru...	MyRecordThree	1	true

1 row fetched in 0.001s

Tag Browser

Tag	Value	Data Type
MyRecords/RecordThreeTriggerTag/Record_Two/Value	true	String

Step 7: Viewing the messages in the Ignition logs

The records published from MQTT Transmission can be viewed in the Ignition Gateway logs by setting the `com.cirruslink.mqtt.engine.gateway.sparkplug.SparkplugPayloadHandler` logger to TRACE.



Review the [Gateway Loggers](#) from Ignition for details on how to do this.

Video

[Managing records with MQTT Transmission and MQTT Recorder.mp4](#)

Extra Activities

At this point you have a fully functional system which can be expanded or modified as required. Below are some activities that you may want to try on your own:

- Add a script to that writes to the record publish trigger tag on a fixed interval
- Setup the Records Signature and trigger a record. After verifying the Record was received by MQTT Recorder, use the digital signature verification Tags in MQTT Recorder to verify the signature on the new Record row.
- Create a new record with a publish trigger pointing to tag accessible by MQTT Engine for example `[default]MQTT Tags/PLC1/MyEngineTriggerTag`
 - Refresh Transmission to update MQTT Engine
 - Configure MQTT Engine to allow outbound edge node tag writes and write to the trigger tag at MQTT Engine.

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/>