User Defined Types (UDTs) within Transmission

Summary

Ignition User Defined Types (UDTs) are a powerful tool for consistently modelling pieces of equipment that are replicated throughout a system. For example a pump may have multiple variables such as pressure, temperature and flow, and by creating a UDT definition for that pump we can then replicate every instance of that pump in the system.

A UDT definition also flows directly through our Cloud Injector Modules allowing for dynamic system modelling within your Bigdata platform.

In this tutorial we will explain how the data in a UDT is represented when using MQTT Transmission and MQTT Engine along with best practices for managing UDT definitions.

Prerequisites

- Have completed the Getting Started: Two Ignition Architecture tutorial
- Have installed and configured a second Edge device pointing to the Ignition Distributor gateway

Tutorials

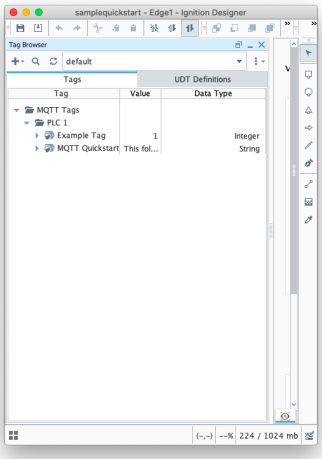
Creating and Publishing a UDT

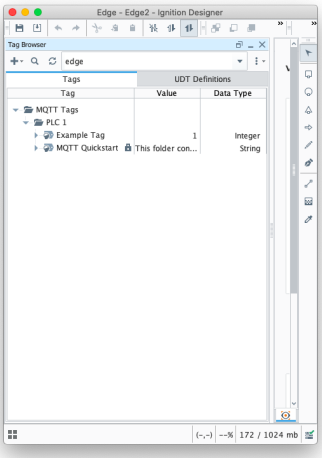
Step 1 - Open Designer sessions

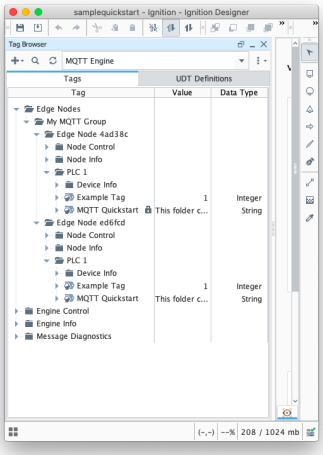
Open three separate Designer sessions and connect to each of the three instances of Ignition:

1. Edge1 running MQTT Transmission
2. Edge2 running MQTT Transmission
3. Ignition running MQTT Engine

In the Tag browser, set the Column selection to include Value and Data Type. Your designer views will look similar to the views below:







See the Ignition Creating a UDT Definition and Instance doc and video for instructions on how to create a definition and instance in Designer.

Step 2a - Create a UDT definition on Edge1

From the Designer pointing to your Edge1 instance create a new UDT with the following properties:

- Name: Pump

Add a Memory Tag member to the Pump UDT with the following properties:

- Name: Pressure
- Value: 123

Add a second Memory Tag to the Pump UDT with the following properties:

- Name: Flow

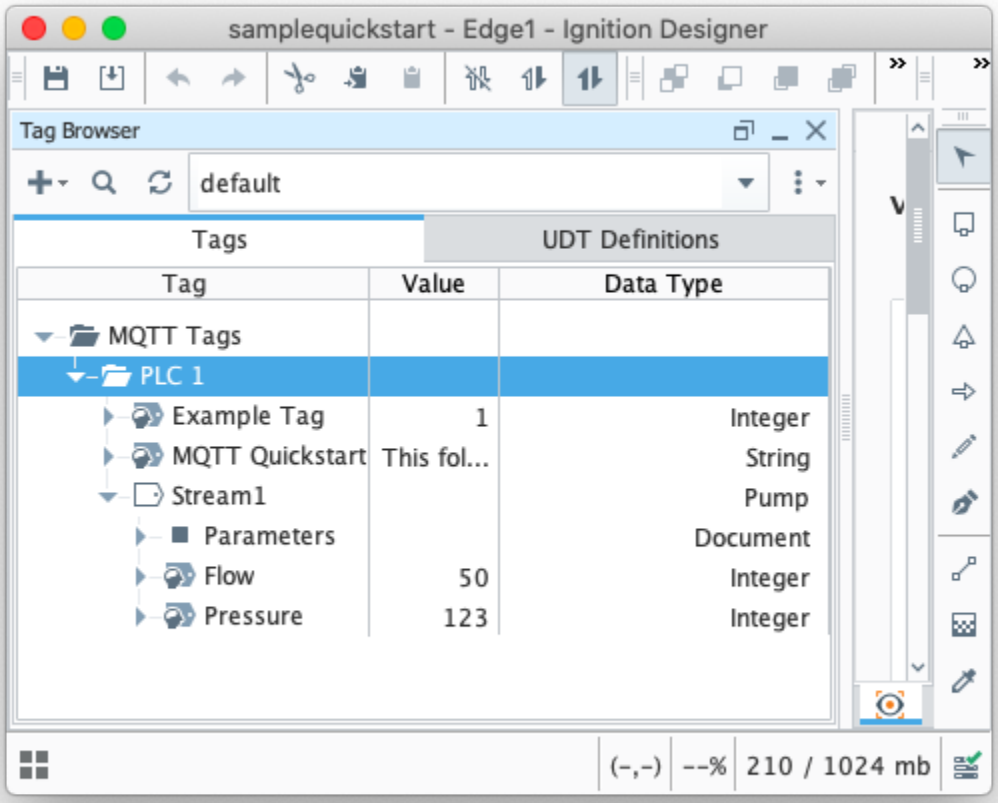
- Value: 50

Step 2b - Create an instance of the UDT on Edge1

Under your device PLC1, create an instance of the UDT "Pump" with the following properties:

- Name: Stream1

Your designer view connected to Edge1 will now look like this:

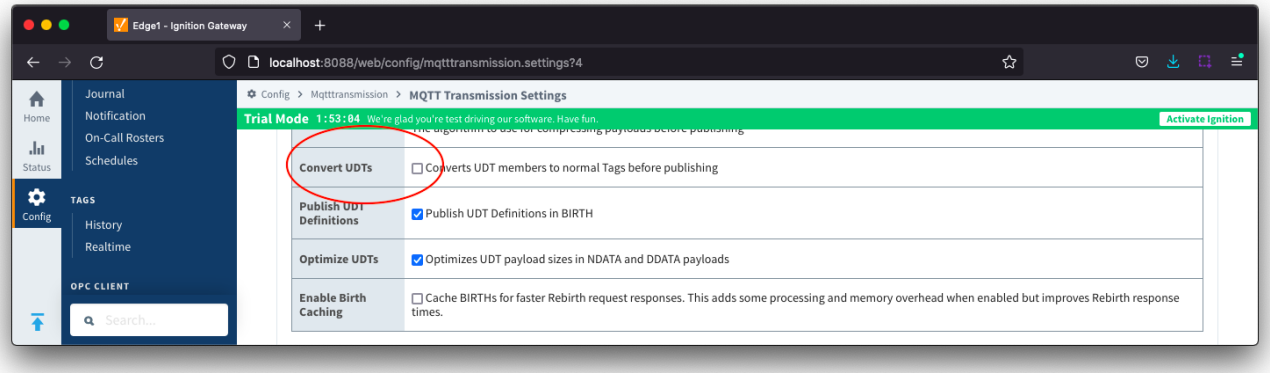


UDTs instances must be created at the Edge Node level or lower. Review MQTT Transmission Transmitters and Tag Trees to understand how the Sparkplug Identifiers are determined.

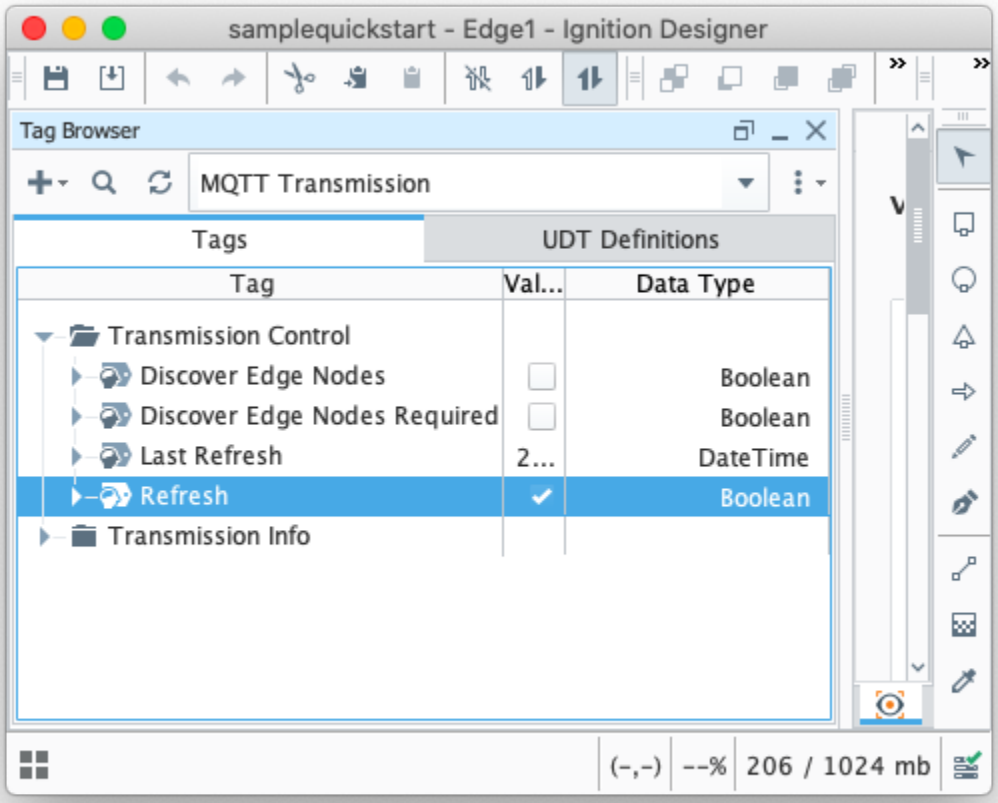
Step 3 - Publish the newly added UDT instance

On the Edge1 Ignition instance UI, navigate to the MQTT Transmission Transmitters settings by selecting Config > MQTT TRANSMISSION > Settings and selecting the Transmitters tab.

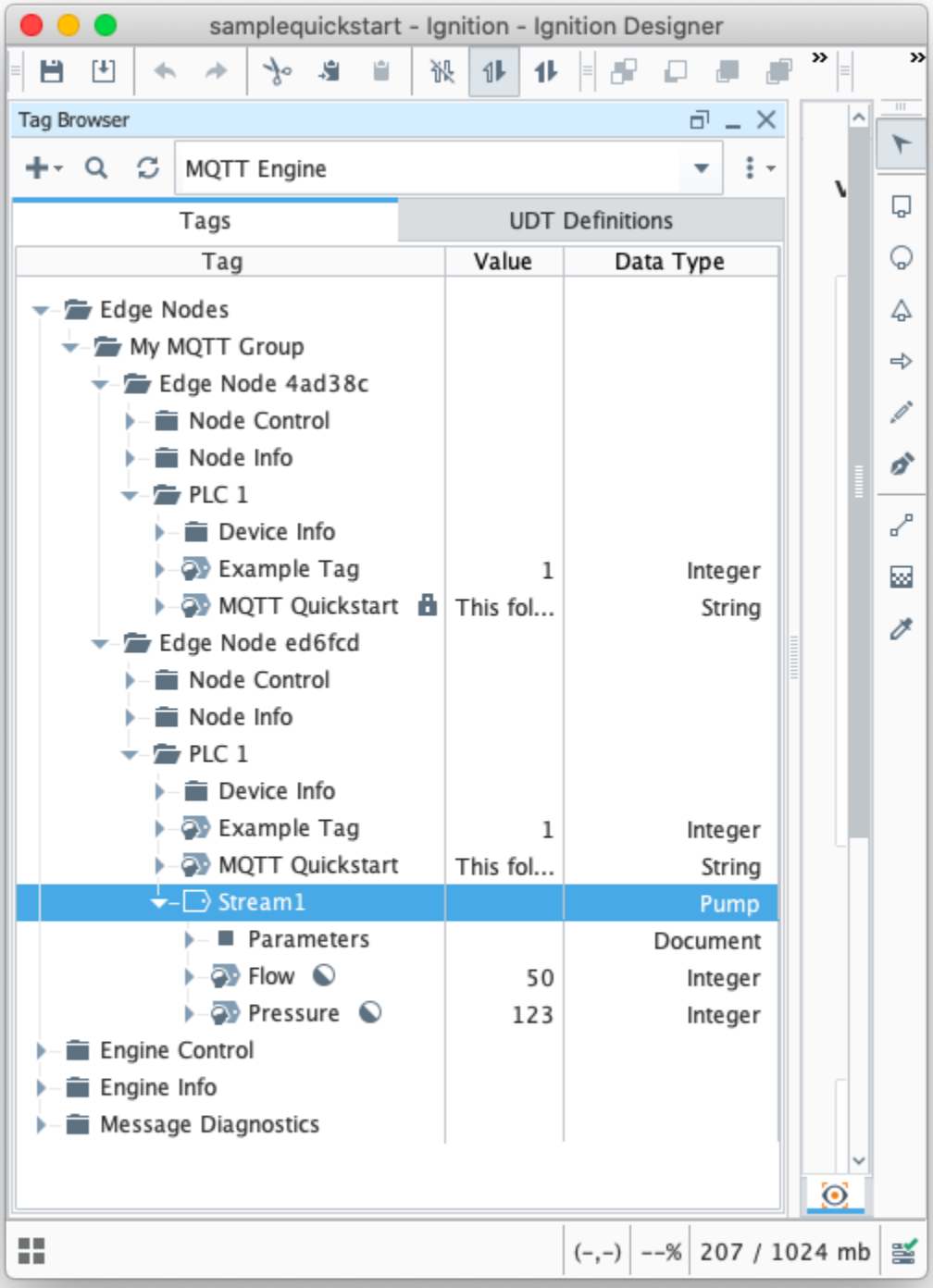
Edit the Example Transmitter and ensure that the Convert UDTs parameter is deselected. This will enable the UDT Definition to be included as part of the BIRTH and disable the UDT member tags from being converted to normal tags before publishing.



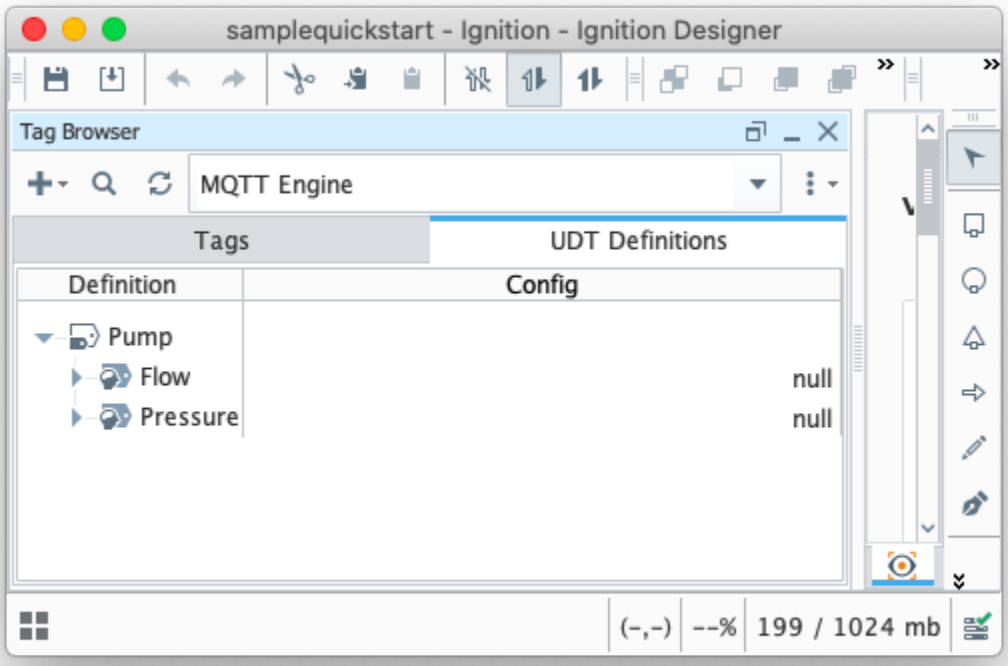
From the Designer pointing to your Edge1 instance force a refresh by switching the provider to MQTT Transmission and selecting the Refresh click box under Transmission Control. This Boolean control will automatically deselect once the BIRTH has been published.



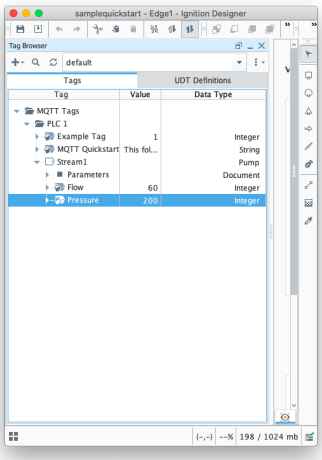
From the Designer pointing to your MQTT Engine instance, you will now see the tag **Stream1** with a data type of **Pump** published from the Edge1 device:

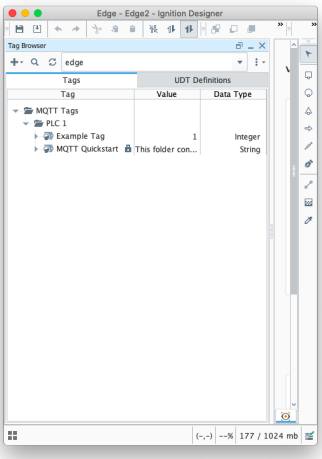


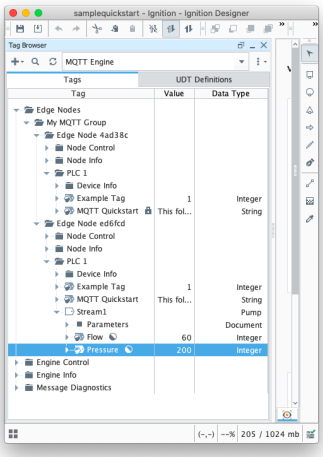
and can see the Pump UDT Definition in the UDT Definitions tab:



From the Designer connected to the Edge1 instance, make changes to the Stream1 tag Flow and Pressure member tags and see the data published and displayed on the MQTT engine instance of Designer.







Now we have our UDT defined we can replicate this across our other edge devices.

Replicating a UDT across other Edge devices

Step 1 - Export/Import the Pump UDT definition

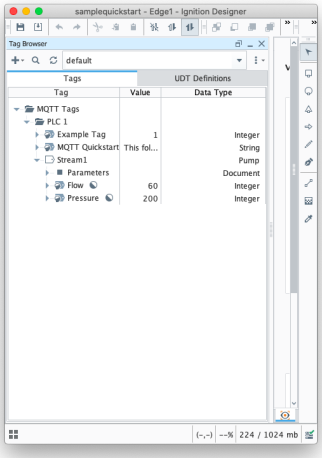
Following the instructions Ignition Importing and Exporting Tags, select the UDT Definition to export from the Edge1 device and import to the Edge2 device

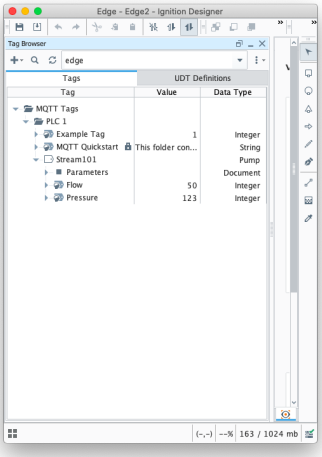
Step 2 - Create a new instance of the Pump UDT on Edge2 and publish

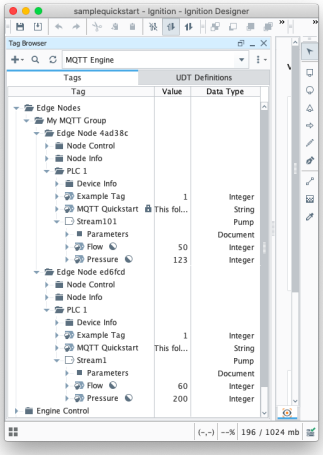
Following the instructions above in Step 2a and Step 3 from Creating and Publishing a UDT create a new instance of the Pump UDT on the Edge2 device and publish.

The name of the instance does not have to be different as the tag path of group/edge/device will fully define the data.

Your Designer views will look similar to the ones below:







Managing UDTs

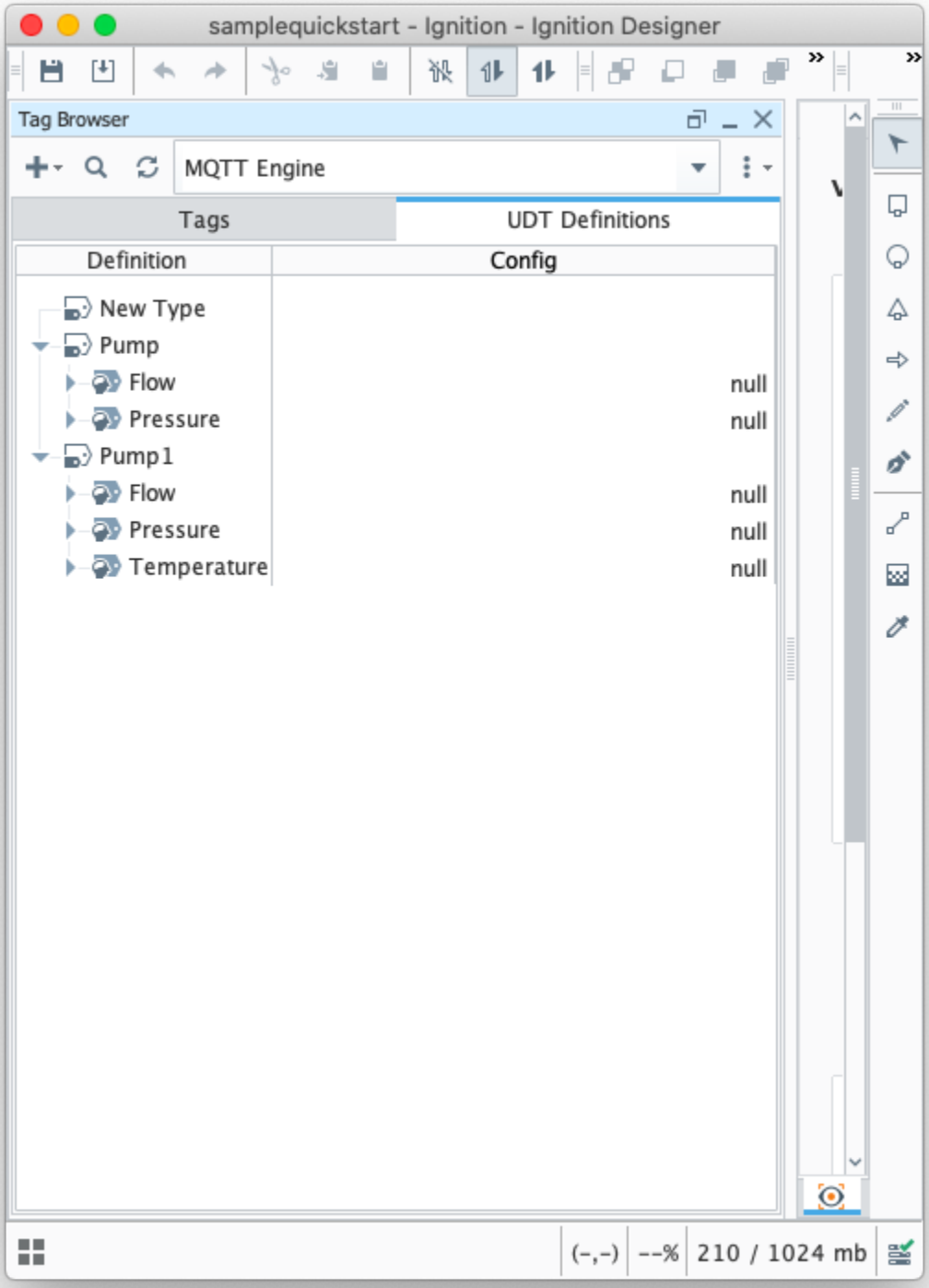
> You cannot have UDTs definitions with the same name and different tag members within the system. The MQTT Engine will ignore any new UDT definitions received that share the same name as the first recorded UDT definition.

> From MQTT Engine release 4.0.16 onward, UDT collision detection is logged.

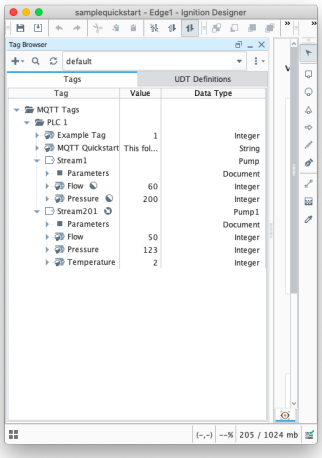
This means that if you are updating a UDT, you **cannot** simply add or remove a tag member from the UDT and force the refresh to publish a birth certificate as the MQTT Engine will ignore the revised definition.

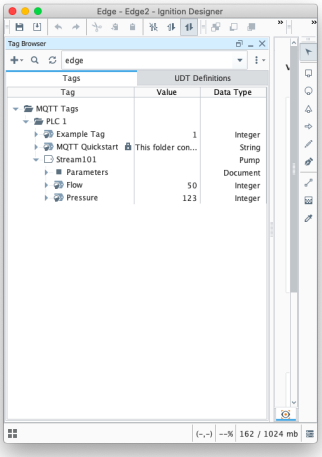
The recommended method is to create a **new** UDT definition on the Edge device that inherits from the existing UDT definition. This is done by setting the property *Parent Data Type* of the new UDT definition to the existing UDT definition. This will allow you to manage versioning for your UDT definitions and prevent published data being ignored by MQTT engine.

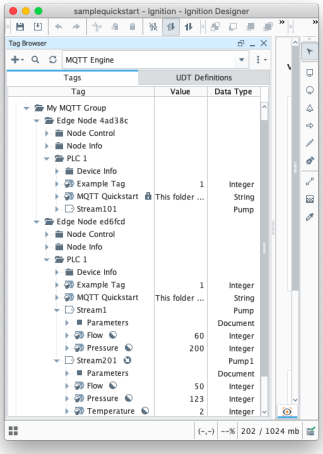
In the example below, we have created a new UDT named Pump1 with the inherited tags from Pump and added a new memory tag of *Temperature*.



Once the new definition is created, you can create an instance on the edge device and force the refresh. The new UDT definition will be recorded by MQTT engine and the data processed correctly without impacting the data published from other edge devices. The system can run like this indefinitely with the updated definition imported into any other locations that need to use that new definition.







Changes made at an edge device to UDT definition member tags are not propagated through the system. You will see that any tag instances of that UDT at the edge device will reflect the changes, but the recorded definition for the UDT at engine will not change nor will tag instances for other edge devices using that definition.

If you make any change to an existing UDT definition you will need to also:

- Delete the UDT definition at the MQTT Engine
- Delete all instances of tags using the UDT definition at MQTT Engine
    - Note: If using MQTT Engine 4.0.16 and newer, this step is no longer required
- Delete the stale variant of the UDT definition in all edge devices through out the system
- Export the updated UDT definition from the edge device and import into all other edge devices
- Force a refresh at each edge device

UDT Collision Detection

UDT collision detection is now available in MQTT Engine module 4.0.16 and newer and uses the MD5 sum of the UDT to detect differences.

If a collision is detected, a warning will be logged and you can see the detail by setting the com.cirruslink.mqtt.engine.gateway.sparkplug.SparkplugBPayloadHandler to TRACE.

This detail will show the UDT definition at MQTT Engine and also the UDT definition published in the NBIRTH message. The offending edge node can be identified from the NBIRTH message.

T	SparkplugBPayloadHandler	18Apr2023 10:40:00	Metric received on topic spBv1.0/My MQTT Group/NBIRTH/Edge Node e64a6d: Metric [name=MyUDT, alias=null, timestamp=null, dataType=Template, isHistorical=null, isTransient=null, metaData=null, properties=null, value=Template [version=null, templateRef=null, isDefinition=true, metrics=[Metric [name=Tag1, alias=null, timestamp=null, dataType=Int32, isHistorical=null, isTransient=null, metaData=null, properties=null, value=null, isNull=null], Metric [name=Tag2, alias=null, timestamp=null, dataType=Int32, isHistorical=null, isTransient=null, metaData=null, properties=null, value=null, isNull=null]], parameters=null], isNull=false]
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	Metric on MQTT Engine side: Metric [name=MyUDT, alias=null, timestamp=null, dataType=Template, isHistorical=null, isTransient=null, metaData=null, properties=null, value=Template [version=null, templateRef=null, isDefinition=true, metrics=[Metric [name=Tag1, alias=null, timestamp=null, dataType=Int32, isHistorical=null, isTransient=null, metaData=null, properties=null, value=null, isNull=null], Metric [name=Tag2, alias=null, timestamp=null, dataType=Int32, isHistorical=null, isTransient=null, metaData=null, properties=null, value=null, isNull=null], Metric [name=Tag3, alias=null, timestamp=null, dataType=Int32, isHistorical=null, isTransient=null, metaData=null, properties=null, value=null, isNull=null]], parameters=null], isNull=false]
W	SparkplugBPayloadHandler	18Apr2023 10:40:00	UDT definition collision detected for MyUDT. Set log level to 'TRACE' for details.
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	MD5 of the MyUDT metric (received on topic spBv1.0/My MQTT Group/NBIRTH/Edge Node e64a6d): 3ce5645bc9e8cdd2a2f0c08e48d56be4
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	MD5 of the MyUDT metric (calculated) on the MQTT Engine side: 2a0b05c48a2bfb9d206ebe257facf11a
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	MD5 of the MyUDT metric (from metadata) on the MQTT Engine side: 2a0b05c48a2bfb9d206ebe257facf11a
D	SparkplugBPayloadHandler	18Apr2023 10:40:00	UDT Definition MyUDT already exists, ignoring
D	SparkplugBPayloadHandler	18Apr2023 10:40:00	New UDT Definition MyUDT
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	UDT definition _types_/MyUDT already exists
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	Determining if UDT definition _types_/MyUDT already exists
D	SparkplugBPayloadHandler	18Apr2023 10:40:00	Sorting 1 Template definitions
T	SparkplugBPayloadHandler	18Apr2023 10:40:00	payload is not null: 5
I	SparkplugTransmissionClient	18Apr2023 10:40:00	MT-7ef6775d-31eb-4a5f: Publishing DBIRTH on Topic: spBv1.0/My MQTT Group/DBIRTH/Edge Node e64a6d/G1
D	SparkplugBPayloadHandler	18Apr2023 10:40:00	Processing NBIRTH from Edge Node My MQTT Group/Edge Node e64a6d with Seq# 0
I	SparkplugTransmissionClient	18Apr2023 10:40:00	MT-7ef6775d-31eb-4a5f: Publishing NBIRTH on Topic: spBv1.0/My MQTT Group/NBIRTH/Edge Node e64a6d
I	TransmissionClient	18Apr2023 10:40:00	[MAIN THREAD] Handling transition to online

Additional Activities

1. Add additional Edge devices to the system using the same UDT.
2. Create nested UDTs to build local systems.