

I have data that is toggling between stale and healthy at my subscribing MQTT Client

There are two common causes for this issue – colliding MQTT Client IDs or colliding Sparkplug Edge Node Descriptors.

Colliding MQTT Client IDs occur when there are two or more MQTT clients connecting to an MQTT broker using the same Client ID. The broker uses the Client ID to identify the client and the current state of the client and therefore this ID must be unique per client and broker.

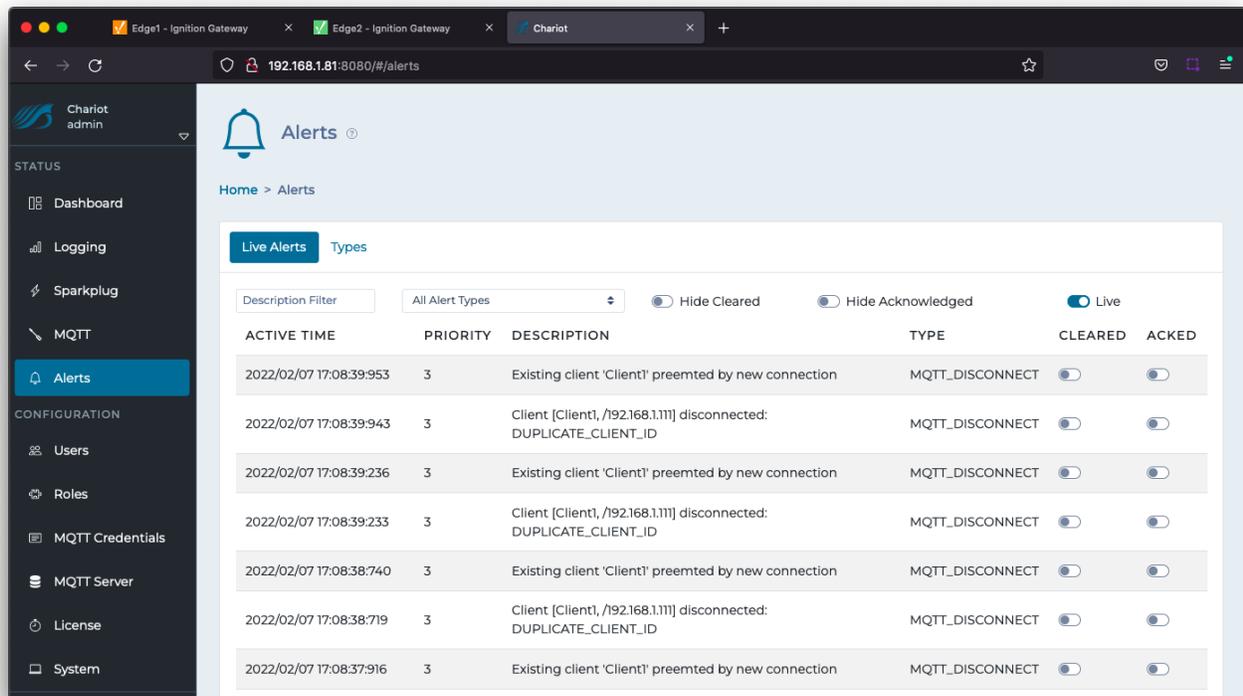
Colliding Edge Node Descriptors occur when there are two or more Edge Nodes publishing with a topic namespace that does not have a unique combination of Group ID and Edge Node ID. In a Sparkplug compliant system, it is this combination of IDs that identifies the Edge Node and is the 'Sparkplug Edge Node Descriptor'. Every Sparkplug Edge Node Descriptor within a Sparkplug environment must be unique.

Let's start by confirming the connection status of the Edge Nodes with your [Chariot](#) or [MQTT Distributor](#) server instance to identify which issue you have.

Chariot

From the Chariot UI navigate to Alerts in the left menu bar. Select Types and enable the alerts for MQTT_DISCONNECT

Under Live Alerts, if we can see in the logs that Chariot is logging the DUPLICATE_CLIENT_ID description, as shown below, you have [Colliding Client IDs](#). If not, we have a [Colliding Sparkplug Edge Node Descriptors](#) issue.



The screenshot shows the Chariot Alerts interface. The left sidebar contains navigation options: Dashboard, Logging, Sparkplug, MQTT, Alerts (selected), Users, Roles, MQTT Credentials, MQTT Server, License, and System. The main content area displays a table of alerts. The table has columns for Active Time, Priority, Description, Type, Cleared, and Acked. The alerts are filtered by 'All Alert Types' and 'MQTT_DISCONNECT'. The descriptions include 'Existing client 'Client' preempted by new connection' and 'Client [Client1, /192.168.1.111] disconnected: DUPLICATE_CLIENT_ID'. The 'Cleared' and 'Acked' columns have toggle switches.

ACTIVE TIME	PRIORITY	DESCRIPTION	TYPE	CLEARED	ACKED
2022/02/07 17:08:39:953	3	Existing client 'Client' preempted by new connection	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:39:943	3	Client [Client1, /192.168.1.111] disconnected: DUPLICATE_CLIENT_ID	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:39:236	3	Existing client 'Client' preempted by new connection	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:39:233	3	Client [Client1, /192.168.1.111] disconnected: DUPLICATE_CLIENT_ID	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:38:740	3	Existing client 'Client' preempted by new connection	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:38:719	3	Client [Client1, /192.168.1.111] disconnected: DUPLICATE_CLIENT_ID	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2022/02/07 17:08:37:916	3	Existing client 'Client' preempted by new connection	MQTT_DISCONNECT	<input type="checkbox"/>	<input checked="" type="checkbox"/>

MQTT Distributor

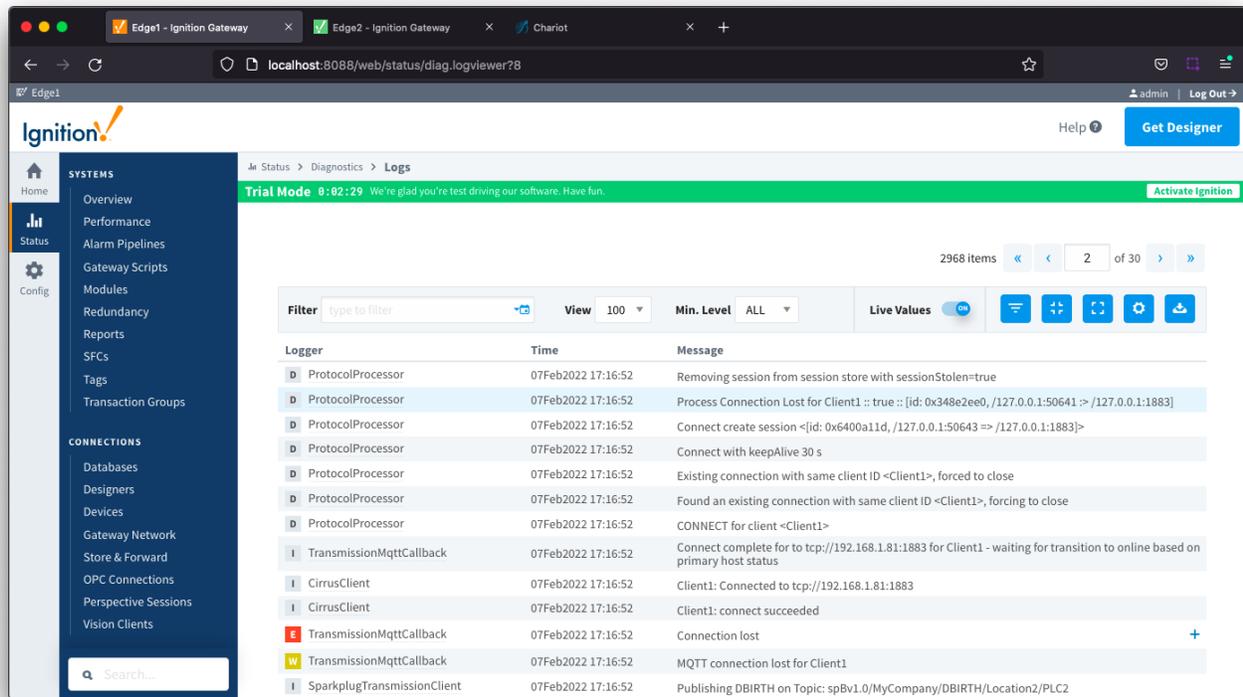
From the Ignition UI connected to your instance of MQTT Distributor, navigate to Status > Diagnostic > Logs.

 Read the user manual [Diagnostics - Logs](#) explaining how to use the Logs console in Ignition

If we can see in the logs that the MQTT broker is continually forcefully disconnecting an existing connection to allow another client with the same Client ID to connect, as shown below, you have [Colliding Client IDs](#). If not, we have a [Colliding Edge Node Descriptors](#) issue.

The logging shows both the Client Id and the associated IP addresses.

If running MQTT Distributor 4.0.13 or earlier, set the debug level for the io.moquette.spi.impl.ProtocolProcessor logger to TRACE and set the filter of the Logs view to ProtocolProcessor.



If running MQTT Distributor 4.0.14 or later, logging will come out as warnings for the com.cirruslink.chariot.server.core.PacketHandler logger.

Filter	View	Min. Level	Live Values
packethandler	100	ALL	OFF
Logger	Time	Message	
I PacketHandler	03May2023 17:42:25	SUBSCRIBE - [f5eab3f8-3a91-46ec-9fc4-90dc49e0db43, MT-1714a23f-36f8-4d72, /127.0.0.1] on topic(s) [[STATE/!amHost][1]]	
I PacketHandler	03May2023 17:42:25	SUBSCRIBE - [f5eab3f8-3a91-46ec-9fc4-90dc49e0db43, MT-1714a23f-36f8-4d72, /127.0.0.1] on topic(s) [[spBv1.0/STATE/!amHost][1]]	
I PacketHandler	03May2023 17:42:25	SUBSCRIBE - [f5eab3f8-3a91-46ec-9fc4-90dc49e0db43, MT-1714a23f-36f8-4d72, /127.0.0.1] on topic(s) [[spBv1.0/G1/NCMD/E2][0], [spBv1.0/G1/DCMD/E2/#][0], [spBv1.0/G1/NDEATH/E2][0]]	
W PacketHandler	03May2023 17:42:25	CONNECT - Active client session with ID: MT-1714a23f-36f8-4d72, address: /192.168.1.106 already exists, ending it	
W PacketHandler	03May2023 17:42:25	CONNECT - [d1920936-a91e-4b7e-9236-9975372c360d, MT-1714a23f-36f8-4d72, /127.0.0.1] Known Client Session	
I PacketHandler	03May2023 17:42:25	SUBSCRIBE - [d1920936-a91e-4b7e-9236-9975372c360d, MT-1714a23f-36f8-4d72, /192.168.1.106] on topic(s) [[spBv1.0/SasolATP_TagProvider/NCMD/E1][0], [spBv1.0/SasolATP_TagProvider/DCMD/E1/#][0], [spBv1.0/SasolATP_TagProvider/NDEATH/E1][0]]	
W PacketHandler	03May2023 17:42:24	CONNECT - Active client session with ID: MT-1714a23f-36f8-4d72, address: /127.0.0.1 already exists, ending it	
W PacketHandler	03May2023 17:42:24	CONNECT - [f830d8ec-6bed-4a77-808c-28e5499e17ca, MT-1714a23f-36f8-4d72, /192.168.1.106] Known Client Session	
I PacketHandler	03May2023 17:42:23	SUBSCRIBE - [f830d8ec-6bed-4a77-808c-28e5499e17ca, MT-1714a23f-36f8-4d72, /127.0.0.1] on topic(s) [[STATE/!amHost][1]]	
I PacketHandler	03May2023 17:42:23	SUBSCRIBE - [f830d8ec-6bed-4a77-808c-28e5499e17ca, MT-1714a23f-36f8-4d72, /127.0.0.1] on topic(s) [[spBv1.0/STATE/!amHost][1]]	

Resolving Colliding Client ID

To resolve the colliding Client IDs you will need to review your system configurations on the physical Edge Nodes identified and remove the conflicts.

In the logs if you see different IP addresses for the Edge Nodes attempting to connect with the same Client ID, then the same MQTT Client ID has been set on different physical Edge Nodes. Review the configuration for physical Edge Nodes with these IP addresses.

If using MQTT Transmission, there are two additional scenarios to consider if the logs show the same IP address for the Edge Nodes attempting to connect with the same Client ID.

1. The MQTT Client ID is set on a single physical Edge Node device where a single Transmitter is dynamically picking up multiple virtual Edge Nodes.
2. The MQTT Client ID is set on a single physical Edge Node where multiple transmitters are configured for one or more virtual Edge Nodes.

In either of these two setups, the MQTT connection for each virtual Edge Node requires a unique Client ID. The Client ID in the the MQTT Transmission Configuration should be left blank allowing MQTT Transmission to auto-generate unique Client IDs for each Edge Node connection.



Refer to the [MQTT Transmission Transmitters and Tag Trees](#) Tutorial/HowTo for detail on how a virtual Edge Node is dynamically created.

Colliding Edge Node Descriptors

MQTT Transmission uses the Sparkplug B specification which defines the topic namespace to publish data as spBv1.0/group_ID/message_type/edge_ID/[device_ID]

In a Sparkplug compliant system, it is the combination of Group ID and Edge Node ID that identifies the Edge Node and is the 'Sparkplug Edge Node Descriptor'.

Every Sparkplug Edge Node Descriptor within a Sparkplug environment must be unique because these are used as 'addresses' in the system to identify the edge node. It is a bit like having two houses with the same postal address. It isn't possible for other MQTT clients in the system to tell where messages are coming from and when sending messages to them, they will both receive the messages.

The topic used for tags published to the MQTT Server is a combination of the MQTT Transmission 'Transmitter' configuration as well as the arrangement of tags in the Ignition tag tree. In the Transmitter configuration, the Tag Path points to the folder where the tag tree will start and the next three folders will be picked up as the group_ID, edge_ID and device_ID.

If you have not carefully managed your tag tree structure, you can create duplicate Sparkplug Edge Node Descriptors.



You can override the functionality of pulling the namespace directly from the tag path by setting the Sparkplug IDs directly for the Group ID, Edge Node ID and, optionally, Device ID. If configured, these elements will be used in the topic namespace and the payload will be the folders pointed to by the Tag Path.

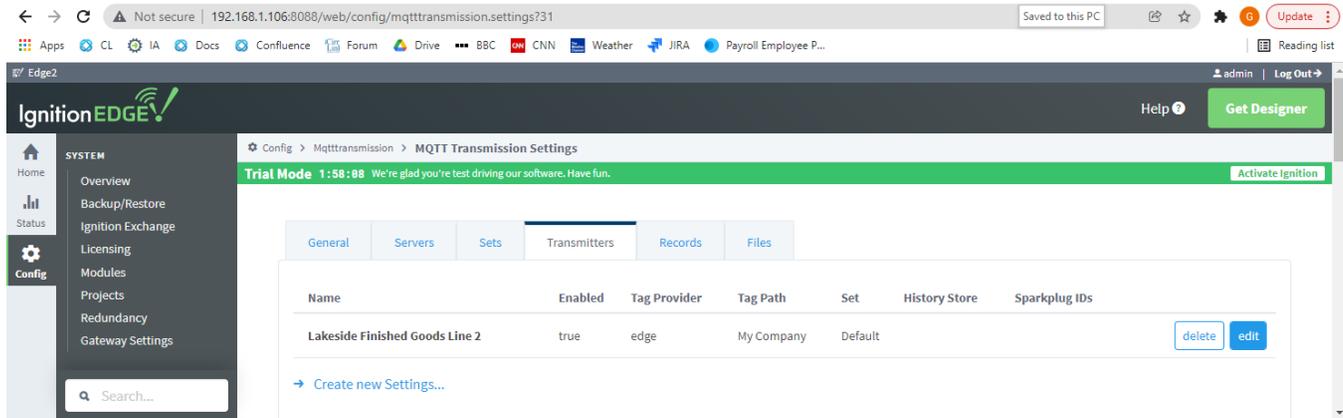
As an example we have two physical Edge Nodes setup with a single Transmitter configured on each.



Whilst the worked example below uses two physical Edge Nodes setup with a single Transmitter configured on each, this can also occur on a single physical Edge Node and the same trouble shooting steps will apply.

The screenshot shows the Ignition web interface for configuring MQTT Transmission. The browser address bar shows the URL `192.168.1.111:8088/web/config/mqtttransmission.settings?25`. The interface includes a sidebar with navigation options like Home, Status, and Config. The main content area is titled 'MQTT Transmission Settings' and features a 'Transmitters' tab. A table lists the configuration for a transmitter named 'Lakeside Finished Goods Line 1'. The table has columns for Name, Enabled, Tag Provider, Tag Path, Set, History Store, and Sparkplug IDs. The 'Lakeside Finished Goods Line 1' entry is shown with 'Enabled' set to true, 'Tag Provider' as default, 'Tag Path' as My Company, and 'Set' as Default. There are 'delete' and 'edit' buttons for this entry. A 'Create new Settings...' link is also visible.

Name	Enabled	Tag Provider	Tag Path	Set	History Store	Sparkplug IDs
Lakeside Finished Goods Line 1	true	default	My Company	Default		delete edit



The Ignition tag tree structure is *Company/Location/Process Area/Line/PLC* where Line is the physical Edge Node device connected to PLCs. With the tag path for both transmitters set to My Company, we can see that both Edge Nodes will publish on the following namespaces:

Transmitter on Edge 1 named Lakeside Finished Goods Line 1 `spV1.0/Lakeside/message_type/Finished Goods/Line1`

Transmitter on Edge 2 named Lakeside Finished Goods Line 2 `spV1.0/Lakeside/message_type/Finished Goods/Line2`

Since the Sparkplug Edge Node Descriptor (group_id = Lakeside and edge_id = Finished Goods) does not uniquely define the edge node, data from these two transmitters will be sent with the same topic resulting in the next message sequence number expected by the MQTT client being incorrect.

As a result, the MQTT Client will mark the data as stale and request a rebirth from the transmitter. Depending on the frequency of the published data this manifests as the data from different edge nodes toggling between stale and healthy. If you have multiple MQTT Clients subscribing to the namespace, this will also likely create a firestorm of rebirth requests across the system.

Your browser does not support the HTML5 video element

Now we can use the logging associated with your [Chariot](#) or [MQTT Engine/Distributor](#) instance to determine the physical or virtual Edge Nodes with duplicate Sparkplug Edge Node Descriptors.

Chariot

From the Chariot UI navigate to Alerts in the left menu bar. Select Types and enable the alerts for SPARKPLUG_GROUP_EDGE_COLLISION

Under Live Alerts, we can see in the logs the Edge Node ID along with the Client IDs causing the collisions.

The next step is to review and update as necessary the configuration for each of the listed Client IDs. If using MQTT Transmission, see [here](#) for how to identify the [Client IDs from Ignition Designer](#).

Alerts

Home > Alerts

Live Alerts Types

ACTIVE TIME	PRIORITY	DESCRIPTION	TYPE	CLEARED	ACKED
2022/01/31 17:37:20:273	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-565b02f7-8195-4d13' and 'MT-f5ad438c-cd3a-4ab3'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>
2022/01/31 17:37:12:434	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-f5ad438c-cd3a-4ab3' and 'MT-565b02f7-8195-4d13'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>
2022/01/31 17:37:09:210	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-565b02f7-8195-4d13' and 'MT-f5ad438c-cd3a-4ab3'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>
2022/01/31 17:37:07:412	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-f5ad438c-cd3a-4ab3' and 'MT-565b02f7-8195-4d13'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>
2022/01/31 17:36:59:198	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-565b02f7-8195-4d13' and 'MT-f5ad438c-cd3a-4ab3'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>
2022/01/31 17:36:57:242	3	Collision detected for Edge Node 'Lakeside/Finished Goods' and clients 'MT-f5ad438c-cd3a-4ab3' and 'MT-565b02f7-8195-4d13'	SPARKPLUG_GROUP_EDGE_COLLISION	<input type="checkbox"/>	<input type="checkbox"/>

MQTT Engine

From the Ignition UI connected to your instance of MQTT Engine, navigate to Status > Diagnostic > Logs.

 Read the user manual [Diagnostics - Logs](#) explaining how to use the Logs console in Ignition

Set the debug level for the `com.cirruslink.mqtt.engine.gateway.sparkplug.SparkplugBPayloadHandler` logger to TRACE and set the filter of the Logs view to `SparkplugBPayloadHandler`.

You will see errors logged indicating that data messages (type of DDATA) are not being handled correctly along with rebirth requests to the duplicate Sparkplug Edge Node Descriptors.

W	SparkplugBPayloadHandler	31Jan2022 16:18:25	Received DDATA for OFFLINE edge node that didn't send a birth: spBv1.0/Lakeside/DDATA/Finished Goods/Line1
W	SparkplugBPayloadHandler	31Jan2022 16:18:23	Received DDATA for OFFLINE edge node that didn't send a birth: spBv1.0/Lakeside/DDATA/Finished Goods/Line2
E	SparkplugBPayloadHandler	31Jan2022 16:18:22	Failed to handle the DDATA message for spBv1.0/Lakeside/DDATA/Finished Goods/Line1
E	SparkplugBPayloadHandler	31Jan2022 16:18:20	Failed to handle the DDATA message for spBv1.0/Lakeside/DDATA/Finished Goods/Line1
W	SparkplugBPayloadHandler	31Jan2022 16:18:20	Received DDATA for OFFLINE edge node that didn't send a birth: spBv1.0/Lakeside/DDATA/Finished Goods/Line2
I	SparkplugBPayloadHandler	31Jan2022 16:18:20	Requesting Rebirth from Lakeside/Finished Goods
W	SparkplugBPayloadHandler	31Jan2022 16:18:19	Received DDATA for OFFLINE edge node that didn't send a birth: spBv1.0/Lakeside/DDATA/Finished Goods/Line1
W	SparkplugBPayloadHandler	31Jan2022 16:18:19	Received DDATA for OFFLINE edge node that didn't send a birth: spBv1.0/Lakeside/DDATA/Finished Goods/Line2

Expanding the *Failed to handle DDATA message* exposes the sequence number error we would expect in this scenario.

E	SparkplugBPayloadHandler	31Jan2022 16:18:22	Failed to handle the DDATA message for spBv1.0/Lakeside/DDATA/Finished Goods/Line1
<pre>sparkplug.util.exception.SequenceNumberException: For Group=Lakeside and Edge Node=Finished Goods - Message Sequence number ERROR: expected=3 but received=2 engine.gateway.sparkplug.SparkplugPayloadHandler.handleSeqNumberCheck(SparkplugPayloadHandler.java:403) engine.gateway.sparkplug.SparkplugBPayloadHandler.onDeviceMessage(SparkplugBPayloadHandler.java:1540) engine.gateway.sparkplug.SparkplugBPayloadHandler.handleDeviceData(SparkplugBPayloadHandler.java:1108) engine.gateway.sparkplug.SparkplugPayloadHandler.handlePayload(SparkplugPayloadHandler.java:137) engine.gateway.EngineCallback.lambda\$messageArrived\$1(EngineCallback.java:229) concurrent.ThreadPoolExecutor.runWorker(Unknown Source) concurrent.ThreadPoolExecutor\$Worker.run(Unknown Source) thread.run(Unknown Source)</pre>			
E	SparkplugBPayloadHandler	31Jan2022 16:18:20	Failed to handle the DDATA message for spBv1.0/Lakeside/DDATA/Finished Goods/Line1

From the Ignition UI connected to your instance of MQTT Distributor, navigate to Status > Diagnostic > Logs.

Set the debug level for the io.moquette.spi.impl.ProtocolProcessor logger to TRACE and in the filter search for the NBIRTH messages for the duplicate Edge Node ID. In this example the filter will be for Lakeside/NBIRTH/Finished Goods. Now we can see which Client IDs are responding to the rebirth requests from MQTT Engine.

The next step is to review and update as necessary the configuration for each of the listed Client IDs. If using MQTT Transmission, see here for how to identify the [Client IDs from Ignition Designer](#).

Filter	View	Min. Level	Live Values
Lakeside/NBIRTH/Finished	100	ALL	<input type="checkbox"/>

Logger	Time	Message
ProtocolProcessor	31Jan2022 18:56:04	send publish message to <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods>
ProtocolProcessor	31Jan2022 18:56:04	directSend invoked clientid <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> QoS MOST_ON E retained false messageId null
ProtocolProcessor	31Jan2022 18:56:04	route2Subscribers republishing to existing subscribers that matches the topic spBv1.0/Lakeside/NBIRTH/Finished Goods
ProtocolProcessor	31Jan2022 18:56:04	PUBLISH from clientID <MT-e8f884a6-65c6-48ca> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> with QoS MOST_ONE
ProtocolProcessor	31Jan2022 18:56:04	send publish message to <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods>
ProtocolProcessor	31Jan2022 18:56:04	directSend invoked clientid <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> QoS MOST_ON E retained false messageId null
ProtocolProcessor	31Jan2022 18:56:04	route2Subscribers republishing to existing subscribers that matches the topic spBv1.0/Lakeside/NBIRTH/Finished Goods
ProtocolProcessor	31Jan2022 18:56:04	PUBLISH from clientID <MT-2ab0d011-08a9-4d29> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> with QoS MOST_ONE
ProtocolProcessor	31Jan2022 18:55:59	send publish message to <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods>
ProtocolProcessor	31Jan2022 18:55:59	directSend invoked clientid <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> QoS MOST_ON E retained false messageId null
ProtocolProcessor	31Jan2022 18:55:59	route2Subscribers republishing to existing subscribers that matches the topic spBv1.0/Lakeside/NBIRTH/Finished Goods
ProtocolProcessor	31Jan2022 18:55:59	PUBLISH from clientID <MT-2ab0d011-08a9-4d29> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> with QoS MOST_ONE
ProtocolProcessor	31Jan2022 18:55:59	send publish message to <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods>
ProtocolProcessor	31Jan2022 18:55:59	directSend invoked clientid <ME-537ee249-439f-49b5> on topic <spBv1.0/Lakeside/NBIRTH/Finished Goods> QoS MOST_ON E retained false messageId null

Finding Client IDs using Ignition Designer

To identify the MQTT Client ID for each Edge Node in Designer, select the Tag Browser *MQTT Transmission* and expand the *Transmission Info* folder tree for each of your transmitters to expose the MQTT Client. ID.

samplequickstart - Edge1 - Ignition Designer

File Edit View Project Component Tools Help

Tag Browser

MQTT Transmission

Tags	UDT Definitions	
Tag	Value	Data Type
Transmission Control		
Transmission Info		
History Store		
Transmitters		
Lakeside Finished Goods Line 1		
Edge Nodes		
Lakeside		
Finished Goods		
MQTT Client		
Command Latency	-1	Long
Enable Latency Check	<input type="checkbox"/>	Boolean
MQTT Client ID	MT-2ab0d011-08a9-4d29	String
Offline DateTime	null	DateTime
Online	<input checked="" type="checkbox"/>	Boolean
Online DateTime	2022-01-31 6:52:05 PM	DateTime
Primary Host ID	null	String
Target Server URL	tcp://192.168.1.81:1883	String
Largest Mesg Xmit	185	Long
Largest Mesg Xmit Timestamp	2022-01-31 6:52:05 PM	DateTime
Refresh Edge Node	<input type="checkbox"/>	Boolean
Refresh Required	<input type="checkbox"/>	Boolean
Reset Metrics	<input type="checkbox"/>	Boolean
Total Bytes Xmit	95,309	Long
Total Mesg Xmit	1,256	Long
Redundancy Role	Independent	String
Redundancy State	Active	String
Refresh Required	<input type="checkbox"/>	Boolean
Version	4.0.10-SNAPSHOT (b20220...	String

Resolving Colliding Edge Nodes Descriptors

To resolve the colliding Edge Node Descriptors you will need to review your system configurations which generated each of the conflicting Edge Nodes Descriptors and remove the conflicts.



Refer to the [MQTT Configuration](#) guide and the [MQTT Transmission Transmitters and Tag Trees](#) Tutorial/HowTo for configuration help.



When troubleshooting remember that an update to any MQTT Transmission configuration parameter or a forced refresh of a transmitter, will mean that any dynamic Client IDs will be recreated. New logs will have reviewed after the update or refresh in order to correctly identify the problematic edge Nodes.

Unable to Resolve?

If the troubleshooting tips did not help you resolve your issues, please open a ticket with [Support](#) making sure to include the MQTT Engine or MQTT Distributor logs as appropriate.

From the Ignition Logs view, select the Download icon to download a copy of the system-name.idb file to your local file system. You will need to compress (zip, 7z or rar) this file before sending to Support.

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/>