Signalling Application Networking Guide

Version 1.0 – February 2024

The DCC signalling system comes with application networking built in, making it ideal for control of larger layouts (where the layout gets broken down into multiple signalling areas) or splitting smaller layouts down to their individual block sections (with simulated block instruments) for real ‘true to prototype’ operation. Only one instance of the application needs to be running on the main Raspberry Pi (the instance providing the interface to the DCC bus and track sensor inputs from the GPIO pins). Other instances can be hosted on other platforms such as the Raspberry Pi Zero (which will provide additional sensor inputs) or Windows or Linux as required.

This guide provides instructions on how to configure and use the networking features.

If you have a purchased a DCC Signalling System from DCC Model Railway Signalling then the example layouts used in this guide can be found in the ‘user_guide’ folder. If you are just using the software, the example layouts can be found in the ‘user_guide’ folder of the Github repository (https://github.com/johnrm174/model-railway-signalling).

Table of Contents

Introduction to Networking

In a nutshell, the networking features of the DCC Signalling application enable different instances of the application to ‘talk to’ each other over your home WiFi network, exchanging information such as signal state, track occupancy, block instrument state, track sensors and DCC commands.

It is important to note that application instances will only ‘talk to’ other instances running on the same network and will therefore not adversely affect or interfere with any other devices connected to your home WiFi network (computers, smart phones, smart televisions etc).

If you only have a single system but you want to experiment with the networking features of the application then you can still do so – just run up two instances of the application on your system and configure them to ‘talk to’ each other over the network.

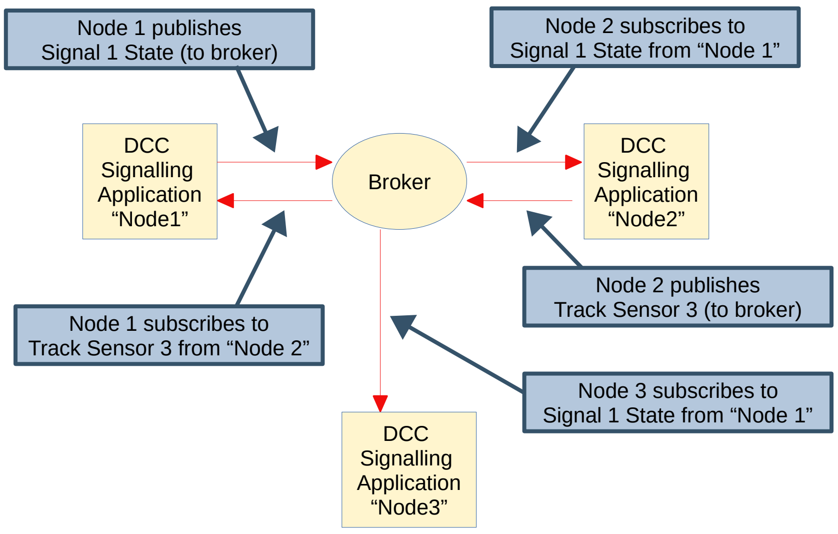
The Publish/Subscribe Model

The DCC Signalling application uses a ‘publish/subscribe’ model for networking. Each application instance (or network ‘node’) can be configured to ‘publish’ state information for other network nodes to receive. However, to receive the published state information, other nodes first need to ‘opt in’ by ‘subscribing’ to the appropriate items from the appropriate nodes.

At the heart of the ‘publish/subscribe’ model is a message ‘broker’. Think of this as a telephone exchange, routing messages from the ‘publishing’ node to all nodes that have expressed an interest in those messages by ‘subscribing’.

In the diagram below, Node 1 has been configured to ‘publish’ Signal 1 state updates to the broker and Nodes 2 and 3 have both ‘subscribed’ to Signal 1 state updates from Node 1. Now, whenever the displayed aspect of Signal 1 changes on Node 1, ‘signal state updated’ messages will be sent to the broker and then forwarded to both Nodes 2 and 3.

Node 2 has also been configured to ‘publish’ Sensor 3 state updates to the broker and Node 1 has ‘subscribed’ to Sensor 3 state updates from Node 2. Now, whenever track sensor 3 is triggered on Node 2, a ‘sensor updated’ message will be sent to the broker and then forwarded to Node 1.

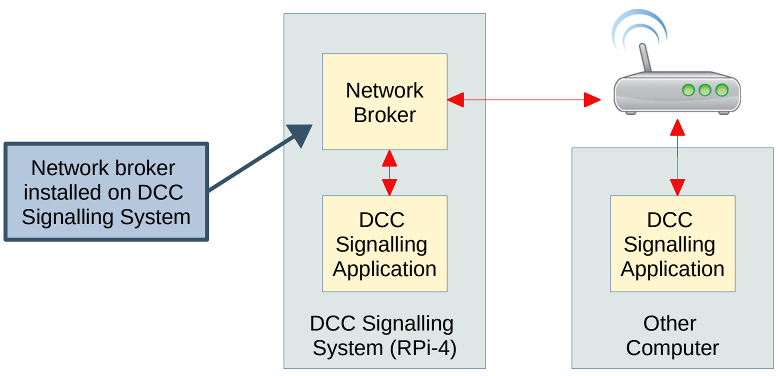

Node 1 publishes Signal 1 State (to broker)
Node 2 subscribes to Signal 1 State from “Node 1”
DCC Signalling Application “Node1”
Broker
DCC Signalling Application “Node2”
Node 2 publishes Track Sensor 3 (to broker)
Node 1 subscribes to Track Sensor 3 from “Node 2”
Node 3 subscribes to Signal 1 State from “Node 1”
DCC Signalling Application “Node3”

Configuring the Signalling Network

The remainder of this section assumes you have a purchased a DCC Signalling System from DCC Model Railway Signalling which comes pre-installed with a broker. If this is not the case you will need to either stand up your own broker on your local network (we'd recommend standing it up on the same device used to host the DCC Signalling application) or use a broker out on the internet (changing the IP addresses and ports as appropriate).

For this example, we're going to configure a network with two 'nodes':

- '*Node1*' – The application running on the DCC Signalling System itself (Raspberry Pi4)
- '*Node2*' – A second instance of the application running on another computer


Network broker installed on DCC Signalling System
Network Broker
DCC Signalling Application
DCC Signalling System (RPi-4)
DCC Signalling Application
Other Computer

Node1 (DCC Signalling system)

Open the networking configuration window from the main menubar (**Settings => MQTT**).

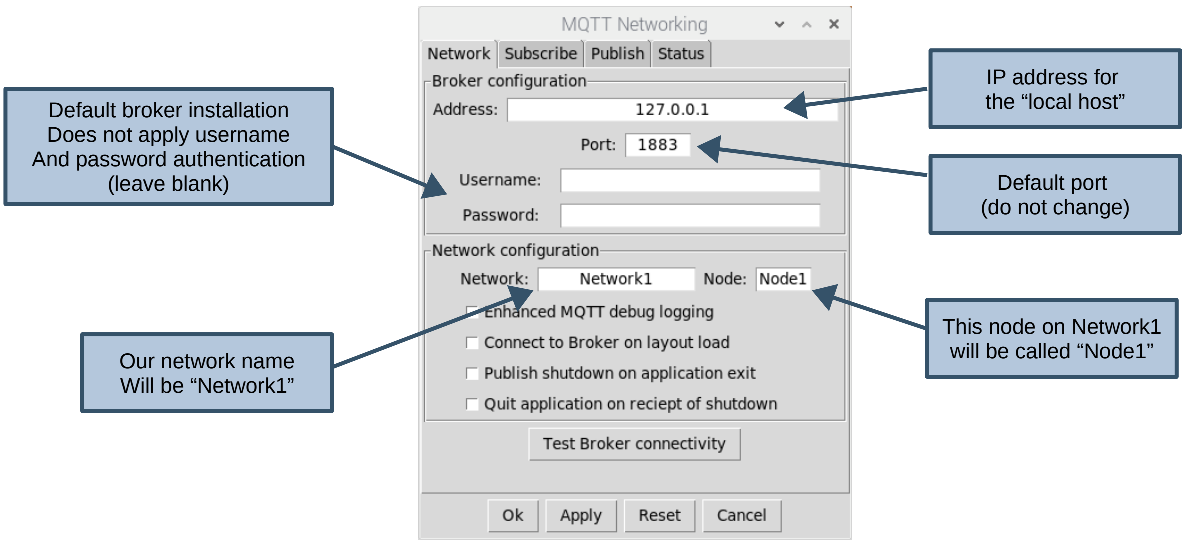
You will see that the broker IP (Internet Protocol) address and port fields are already populated. The special address of **127.0.0.1** refers to the "local host" (i.e. the machine the application is running on). As this is the DCC Signalling System node, the broker is pre-installed and therefore running on the same "local host" as the application. We therefore don't need to change this field for '*Node1*'. The default port number of **1883** should also be left unchanged.

You should note that there is no username/password security enabled as part of the default broker installation so these fields should be left blank. This should be absolutely fine for normal use on your local network as its 'behind' the firewall/router provided by your Internet Service Provider (ISP) and the information we are passing over the network is limited to the internal state of the signalling system (i.e. its not exactly the critical national infrastructure of the real railway). That said, if you do want to enable username/password security for the broker you will be able to find the appropriate instructions on the internet (search Raspberry Pi and mosquitto MQTT broker).

We first need to choose a name for our signalling network. This is user-configurable to enable multiple signalling networks to operate independently over the same local WiFi network without interfering with each other (e.g. to support club nights or model railway exhibitions).

We also need to choose a unique name for this particular node on the network.

In this case we'll use '***Network1***' for the network name and '***Node1***' for the node name.


MQTT Networking
Network
Subscribe
Publish
Status
Default broker installation Does not apply username And password authentication (leave blank)
IP address for the “local host”
Default port (do not change)
Our network name Will be “Network1”
This node on Network1 will be called “Node1”
Test Broker connectivity
Ok
Apply
Reset
Cancel

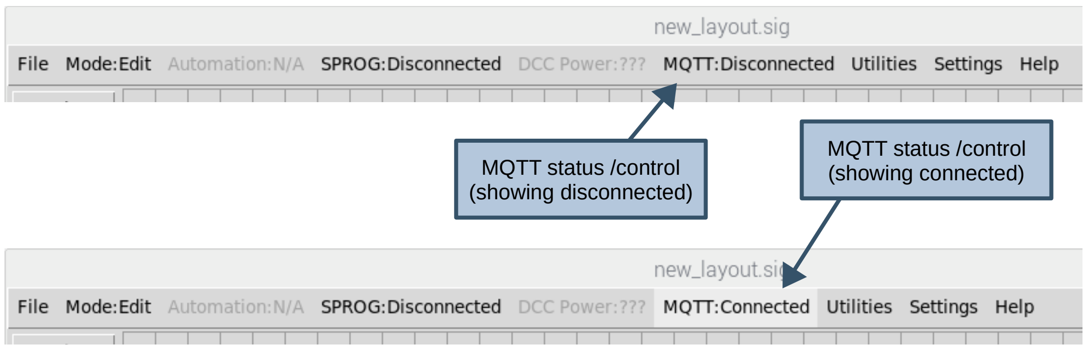
Broker configuration
Address: 127.0.0.1
Port: 1883
Username:
Password:
Network configuration
Network: Network1 Node: Node1
☐ Enhanced MQTT debug logging
☐ Connect to Broker on layout load
☐ Publish shutdown on application exit
☐ Quit application on reciept of shutdown

Other options you can select (as appropriate) are:

- Enhanced MQTT debug loggging – Realistically you would only ever need to select this if you are having real problems with your configuration and need to send the application logs through to us at DCC Model Railway Signalling for further analysis/assistance.
- Connect to Broker on Layout load – The MQTT broker is always disconnected when loading a new layout file. Select this option (and save the layout file) if you want to automatically connect to the broker on layout load.
- Publish shutdown on Application exit – Primarily intended for networks that include remote sensor nodes (i.e. nodes providing additional track sensor inputs to the signalling system and running without a keyboard, mouse or monitor connected). If this option is selected then a shutdown message will be published to all other network nodes on application exit.
- Quit application on reciept of shutdown – Primarily intended for remote sensor nodes (see above) to trigger a graceful system shutdown on reciept of the shutdown message.

You now need to **Apply** the settings before we attempt connecting to the broker (don’t click OK as we want to keed the networking configuration window open until we have completed the setup).

To connect to the broker, either click on the **Test Broker Connectivity** button in the networking configuration window or click on the **MQTT** control on the Main Menubar and select **Connect** from the dropdown menu. If successful, the Main Menubar MQTT status will show ‘Connected’.



Before we configure the other network node, we first need to find the IP address of the the main DCC signalling System on the local WiFi network as this will be the broker address we will have to specify when configuring the other signalling network node. Note that when we talk about IP addresses on the local network, these are the <u>local</u> network address assigned by your WiFi router and not the internet-facing IP address (assigned by the Internet Service Provider).

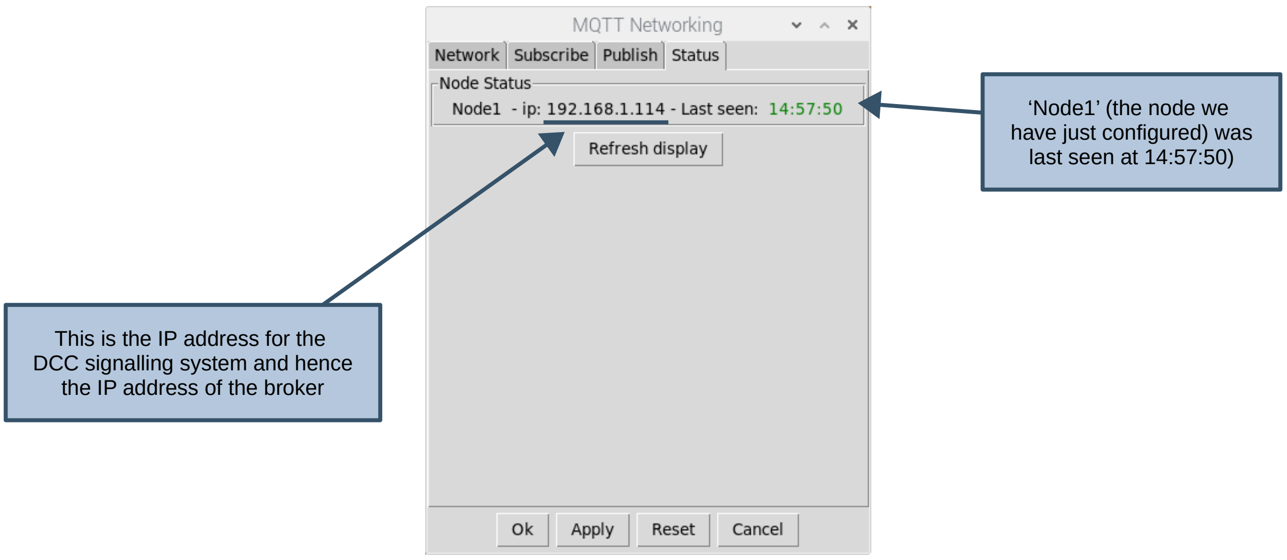
Note that most WiFi routers will assign devices a "fixed" local IP address when they are first seen on the network. This means that when devices are shut down and subsequently powered up, the router will recognise them and assign the same IP address as before. Once you have configured your signalling network you should therefore never need to update the configuration unless you change your router or perform a factory reset. Even routers that do not assign "fixed" local IP addresses by default can usually be configured to do so. If in doubt check the documentation for your home router or contact the supplier (normally your Internet Service Provider).

To find the local network address, select the **Status** tab of the networking configuration window.

This provides a list of all nodes on the signalling network that have been 'seen', the time they were last seen and an indication of status (the time is coloured green if they have been seen within the last 10 seconds, otherwise red). The local IP address of each node is also shown in the list.

Assuming we have successfully connected to the broker, our node should be the only item in the list (as this is the only network node we have configured). If the broker is connected but nothing is showing in the list then try clicking **Refresh Display**.

Make a note of the displayed IP address for '*Node1*' (the DCC Signalling system) – this is the broker address we shall need to specify when configuring other network nodes.

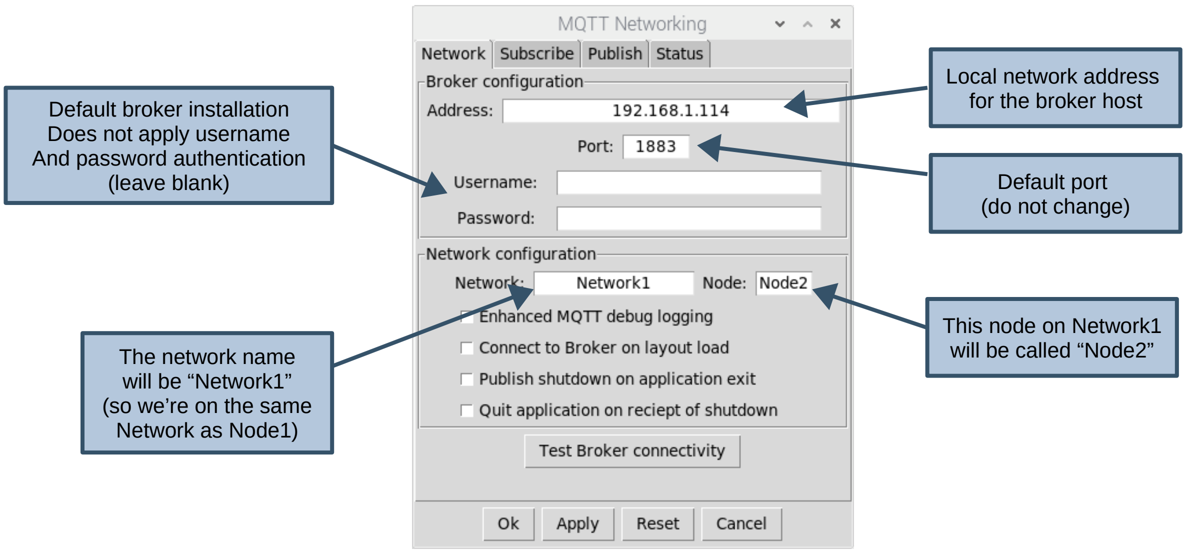

Node1 - ip: 192.168.1.114 - Last seen: 14:57:50

MQTT Networking
Network
Subscribe
Publish
Status
Node Status
Refresh display
Ok
Apply
Reset
Cancel
'Node1' (the node we have just configured) was last seen at 14:57:50)
This is the IP address for the DCC signalling system and hence the IP address of the broker

Node2 (Secondary Node)

On the second computer, start up the Signalling application and open the networking configuration window from the main menubar (**Settings => MQTT**).

Configuration is similar to '*Node1*', but in this case we need to specify the local network address of the main DCC Signalling node for the broker address (this is the IP address we made a note of in the previous section) and set a unique name for the node (this will be '***Node2***').


MQTT Networking
Network
Subscribe
Publish
Status
Default broker installation Does not apply username And password authentication (leave blank)
Local network address for the broker host
Default port (do not change)
The network name will be "Network1" (so we're on the same Network as Node1)
This node on Network1 will be called "Node2"
Test Broker connectivity
Ok
Apply
Reset
Cancel

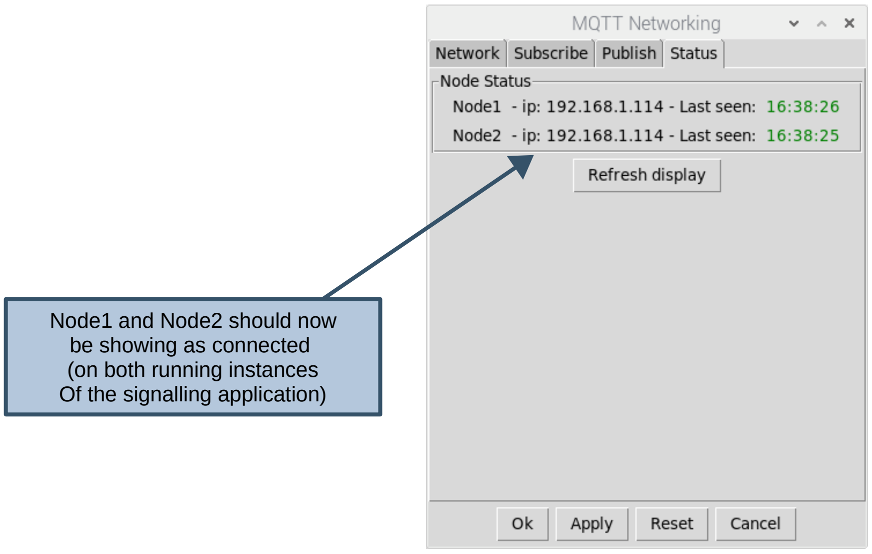
Broker configuration
Address: 192.168.1.114
Port: 1883
Username:
Password:

Network configuration
Network: Network1 Node: Node2
☐ Enhanced MQTT debug logging
☐ Connect to Broker on layout load
☐ Publish shutdown on application exit
☐ Quit application on reciept of shutdown

You now need to **Apply** the settings before we attempt connecting to the broker (don't click OK as we want to keed the networking configuration window open until we have completed the setup).

To connect to the broker, either click on the **Test Broker Connectivity** button in the networking configuration window or click on the **MQTT** control on the Main Menubar and select **Connect** from the dropdown menu. If successful, the Main Menubar MQTT status will show 'Connected'.

If you now select the **Status** tab of the networking configuration window (on either node), both nodes should now appear in the list.


MQTT Networking
Network
Subscribe
Publish
Status
Node1 and Node2 should now be showing as connected (on both running instances Of the signalling application)
Refresh display
Ok
Apply
Reset
Cancel

Node Status
Node1 - ip: 192.168.1.114 - Last seen: 16:38:26
Node2 - ip: 192.168.1.114 - Last seen: 16:38:25

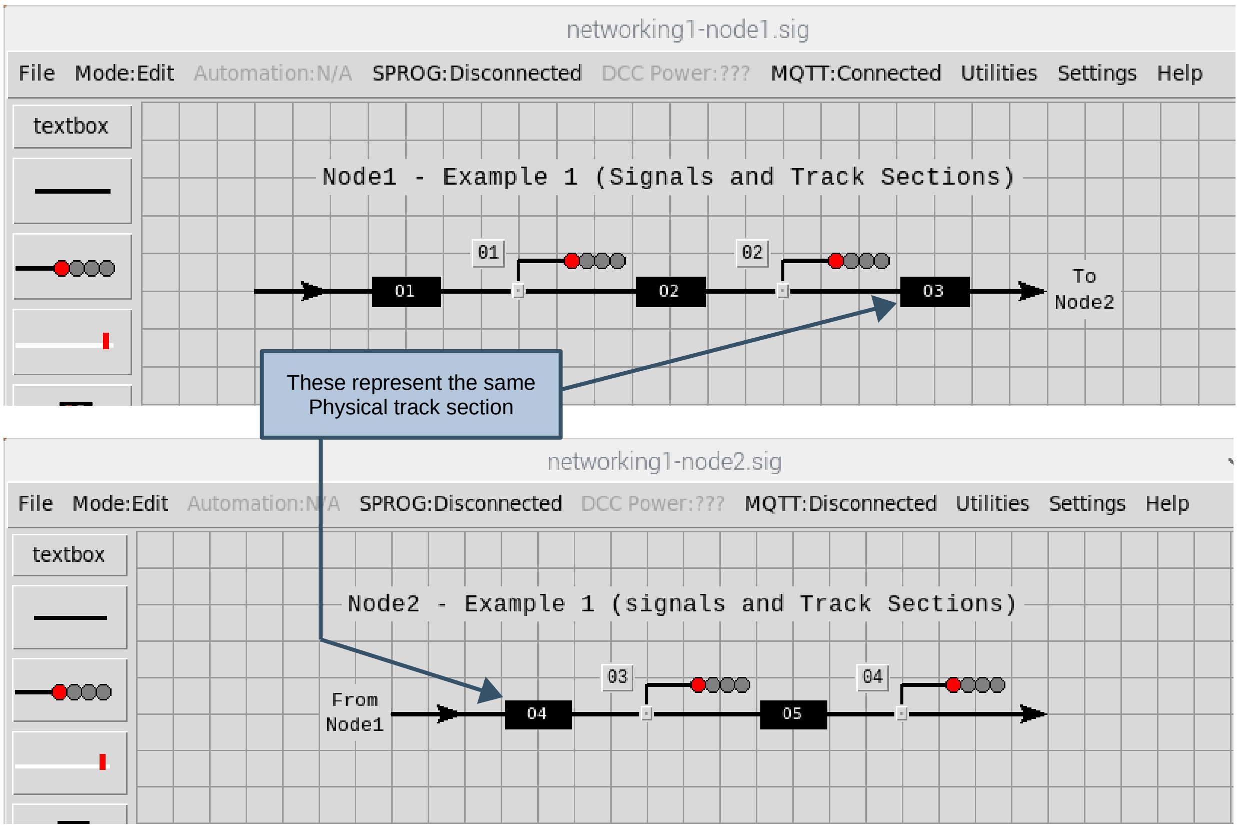
And its as simple as that – the basic network is now configured.

Signal State and Track Occupancy

The remainder of this guide will assume you are already familiar with basic layout configuration after having worked through the application quickstart guide.

For this example, we need to create two seperate layout files, one to run on '*Node1*' (the main DCC Signalling System) and one to run on '*Node2*' (the second computer on our signalling network).

Note that the Track Section which 'joins' the two layouts needs to be represented on both layouts as each Node needs to be able to change the state ('*Node1*' will be passing trains into this section when Signal 2 is passed and '*Node2*' will be clearing this section when Signal 3 is passed).


networking1-node1.sig
File Mode:Edit Automation:N/A SPROG:Disconnected DCC Power:??? MQTT:Connected Utilities Settings Help
textbox
Node1 - Example 1 (Signals and Track Sections)
01
02
01
02
03
To Node2
These represent the same Physical track section
networking1-node2.sig
File Mode:Edit Automation:N/A SPROG:Disconnected DCC Power:??? MQTT:Disconnected Utilities Settings Help
textbox
Node2 - Example 1 (signals and Track Sections)
03
04
From Node1
04
05

Configuring Publish and Subscribe

We first need to configure these layouts so that all signals (when cleared) will display the correct aspect, taking into account the displayed aspect of the signal ahead. A*s a reminder - this is achieved by specifying the 'signal ahead' on the* ***Interlocking*** *tab of the signal configuration dialog*.

For Signal 2 (on '*Node1*') the 'signal ahead' will be Signal 3 (on '*Node2*'). This means that:

- '*Node2*' needs to 'publish' Signal 3 to the broker
- '*Node1*' needs to 'subscribe' to Signal 3 from '*Node2*'

We also want to seamlessly pass trains from track section to track section as each signal is passed. *As a reminder - this is achieved by specifying the 'section behind' and ''section ahead' on the* ***Automation*** *tab of the signal configuration dialog.*

As Section 3 (on '*Node1*') and Section 4 (on '*Node2*') represent the same physical Track Section, we need to configure these to 'mirror' each other – so when the state of Section 3 is changed (by '*Node1*' the changes are also reflected in Section 3 (on '*Node2*') and vice versa.

This means that:

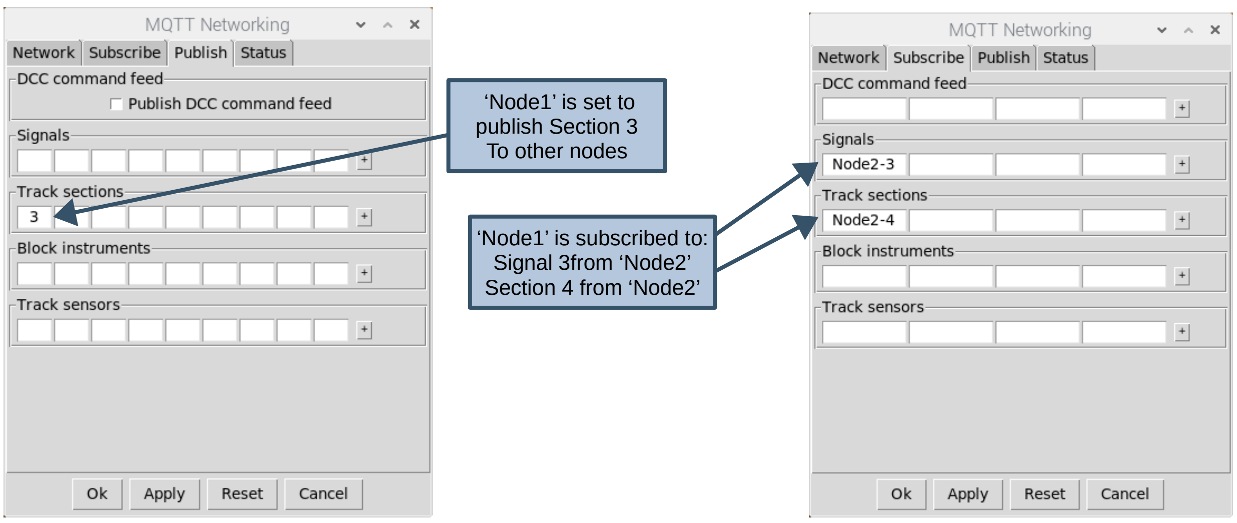
- ‘*Node1*’ needs to ‘publish’ Section 3 to the broker
- ‘*Node2*’ needs to ‘publish’ Section 4 to the broker
- ‘*Node1*’ needs to ‘subscribe’ to Section 4 from ‘*Node2*’
- ‘*Node2*’ needs to ‘subscribe’ to Section 3 from ‘*Node1*’

The ‘publishing’ and ‘subscribing’ of layout items is configured via the ‘**publish**’ and ‘**subscribe**’ tabs of the networking configuration window (**Settings => MQTT**) .

Publishing is straightforward in that we just need to specify the IDs of the items we want to make available to other nodes. However, subscribing is slightly different as we also also need to specify the source node for each item.

Remote item IDs are therefore specified as ‘**nodeID-itemID**’ (note the hyphen used as a seperator), so for example, a subscription to Signal 1 from ‘*Node2*’ would be specified as ‘**Node2-1**”

Node1 (DCC Signalling system) publish and subscribe



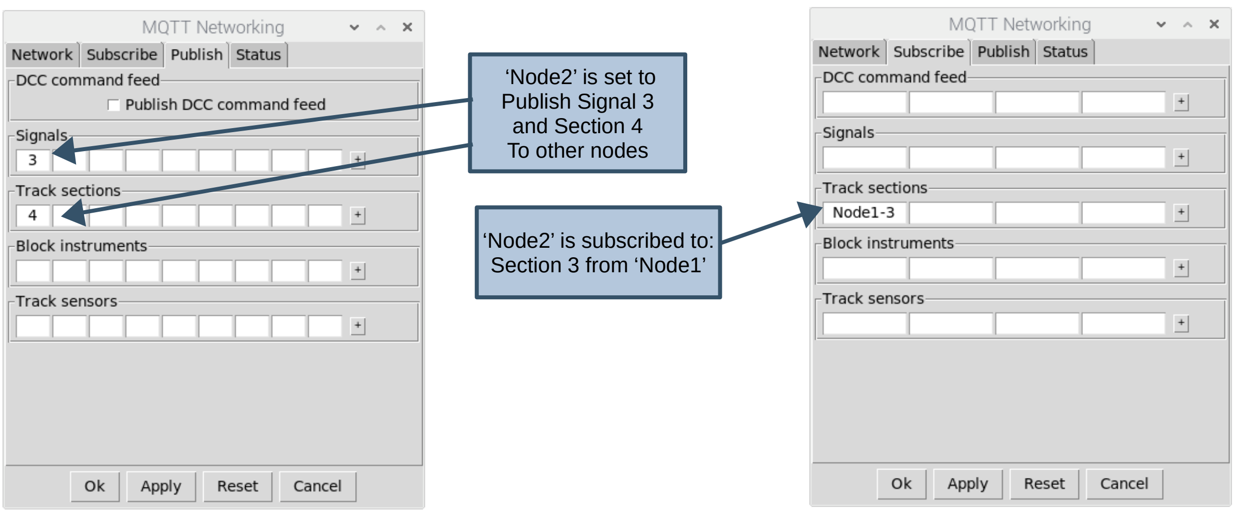
DCC command feed
Publish DCC command feed
Signals
Track sections: 3
Block instruments
Track sensors

‘Node1’ is set to publish Section 3 To other nodes

‘Node1’ is subscribed to: Signal 3from ‘Node2’ Section 4 from ‘Node2’

DCC command feed
Signals: Node2-3
Track sections: Node2-4
Block instruments
Track sensors

Node2 (Secondary Node) publish and subscribe



DCC command feed
Publish DCC command feed
Signals: 3
Track sections: 4
Block instruments
Track sensors

‘Node2’ is set to Publish Signal 3 and Section 4 To other nodes

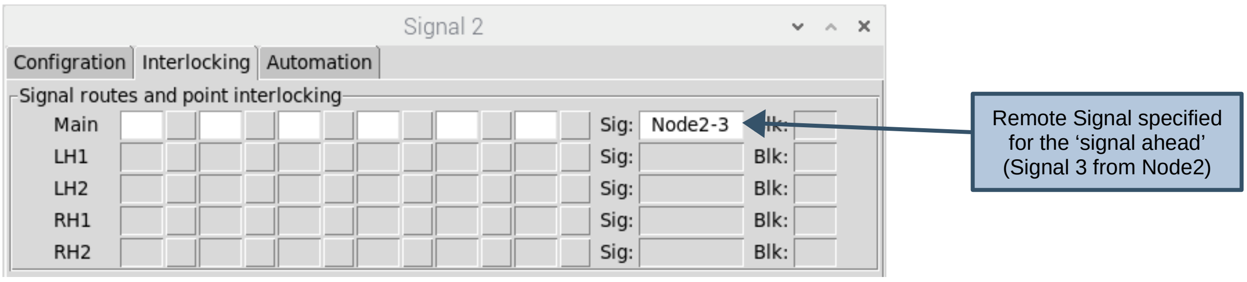
‘Node2’ is subscribed to: Section 3 from ‘Node1’

DCC command feed
Signals
Track sections: Node1-3
Block instruments
Track sensors

Configuring the layouts

Once the publish and subscribe configurations have been applied to both nodes, the ‘subscribed’ Signals and Track Sections can then be used as part of the normal layout configuration.

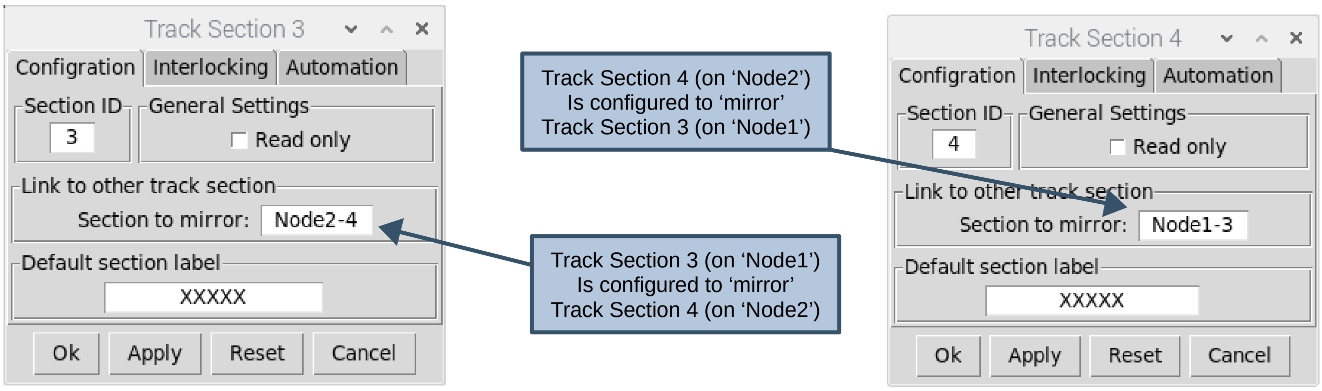
For ‘*Node1*’ Signal 2, the ‘signal ahead’ can now be specified as ‘**Node2-3**’:



Main Sig: Node2-3

Remote Signal specified for the ‘signal ahead’ (Signal 3 from Node2)

Section 3 (on ‘*Node1*’) can now be configured to mirror ‘**Node2-4**’ and similarly Section 4 (on ‘*Node2*’) can be configured to mirror ‘**Node1-3**’:



Track Section 3: Section ID: 3; Section to mirror: Node2-4; Default section label: XXXXX

Track Section 4: Section ID: 4; Section to mirror: Node1-3; Default section label: XXXXX

Track Section 4 (on ‘Node2’) Is configured to ‘mirror’ Track Section 3 (on ‘Node1’)

Track Section 3 (on ‘Node1’) Is configured to ‘mirror’ Track Section 4 (on ‘Node2’)

To complete the networked layout configuration:

- Configure all other signals with details of the ‘local’ signal ahead.
- Configure Signal 4 as a Timed Signal (this will be the ‘exit’ signal from our layout).
- Configure all signals (except Signal 4) to be Overridden if the section ahead is occupied.
- Configure all signals with details of the ‘local’ section ahead and section behind – we don’t need to worry about ‘remote’ track sections for the signals’ track occupancy configuration as this will be handled by our ‘mirrored sections’ configuration.

Testing the layouts

To test the configuration, ensure both nodes are connected to the broker and all signals are cleared.

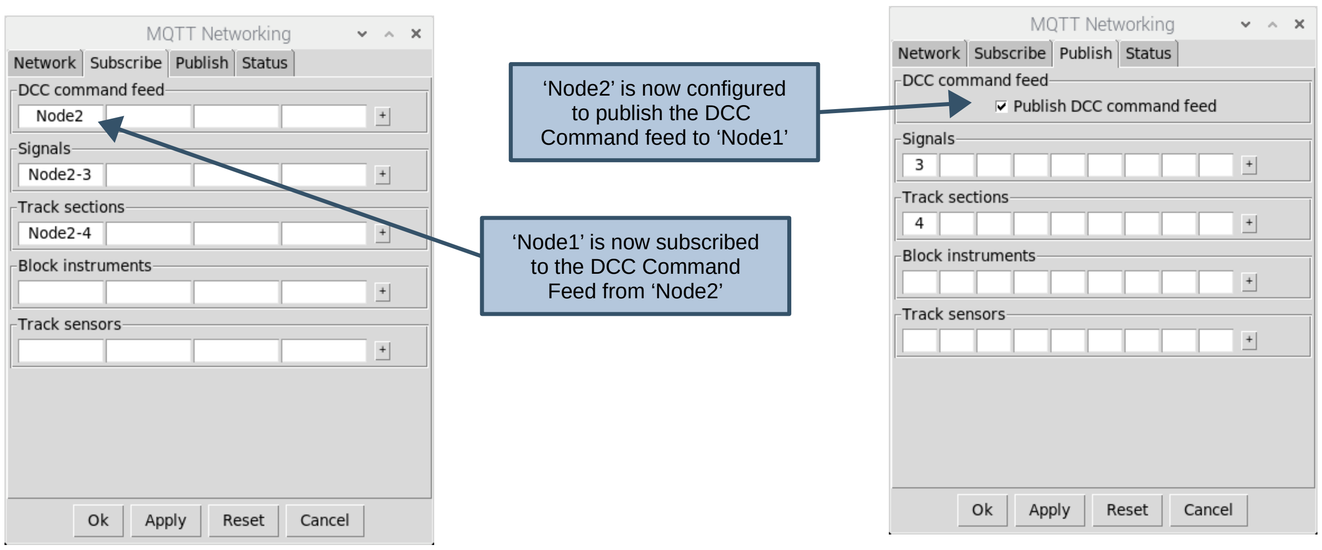
If you click on Section 3 (on ‘*Node1*’) to change it to OCCUPIED, Section 4 (on ‘*Node2*’) should also change to OCCUPIED. Similarly, if you click on Section 4 (on ‘*Node2*’) to change it to CLEAR, Section 3 (on ‘*Node1*’) should also change to CLEAR.

Trains can also be ‘passed through’ the layout by setting Section 1 on ‘*Node1*’ to OCCUPIED and then simulating ‘signal passed’ events for each signal along the route in turn. The train should be seamlessly passed from one node to the other (in terms of track occupancy), with all signals (on both nodes) displaying the correct aspects in relation to the signal ahead.

DCC Command Feeds

If you only have a single DCC Signalling system providing the DCC bus interface to the layout, then we also need to configure the network such that all other nodes send their DCC commands to this node, and then from this node out to the layout (to change the signals and points).

This configuration is straightforward, in that we just need to configure '*Node2*' to publish it's DCC command feed to the broker and configure '*Node1'* to subscribe to this command feed:



DCC command feed: Node2
Signals: Node2-3
Track sections: Node2-4
Block instruments
Track sensors

'Node2' is now configured to publish the DCC Command feed to 'Node1'

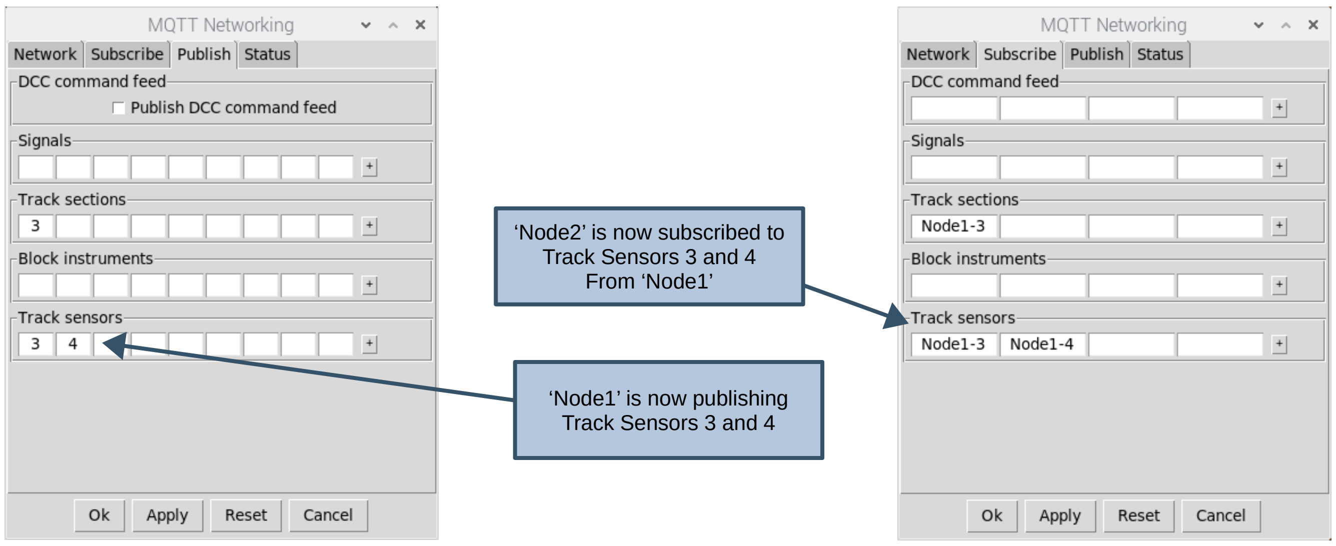
'Node1' is now subscribed to the DCC Command Feed from 'Node2'

DCC command feed: ☑ Publish DCC command feed
Signals: 3
Track sections: 4
Block instruments
Track sensors

Now the Signals on '*Node2*' can be configured with the the appropriate DCC commands to change the aspects. These commands will now be sent via the broker to '*Node1'* which will then forward them to the SPROG DCC programmer/controller and then out to the layout via the DCC bus.

Track Sensors

We might also want to use the main DCC Signalling system node to provide the interface for all layout track sensors. To achieve this, the configuration follows the same pattern as for signals and track sections in that '*Node1*' will need to publish the appropriate track sensors to the broker and '*Node2*' will need to subscribe to those sensors from '*Node1*':



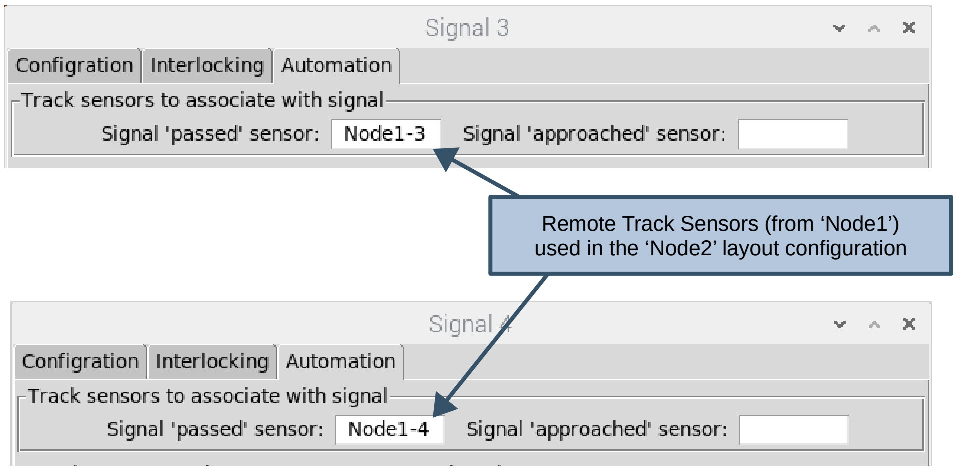
DCC command feed: ☐ Publish DCC command feed
Signals
Track sections: 3
Block instruments
Track sensors: 3 4

'Node2' is now subscribed to Track Sensors 3 and 4 From 'Node1'

'Node1' is now publishing Track Sensors 3 and 4

DCC command feed
Signals
Track sections: Node1-3
Block instruments
Track sensors: Node1-3 Node1-4

The subscribed track sensors (fron '*Node1*') can now be used in the configuration of the '*Node2*' layout to generate 'signal passed' events:


Signal 3
Configration
Interlocking
Automation
Track sensors to associate with signal
Signal 'passed' sensor: Node1-3
Signal 'approached' sensor:
Remote Track Sensors (from 'Node1') used in the 'Node2' layout configuration
Signal 4
Configration
Interlocking
Automation
Track sensors to associate with signal
Signal 'passed' sensor: Node1-4
Signal 'approached' sensor:

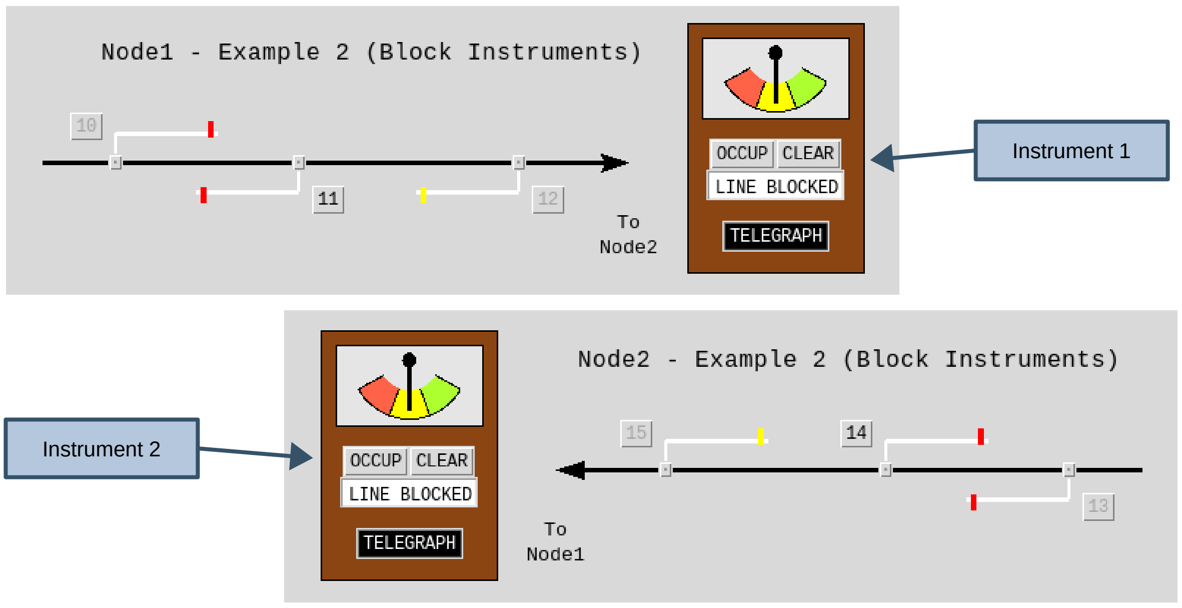
Now, whenever Track Sensor 3 is triggered on '*Node1*' (by the mapped external GPIO track sensor), this event will be sent to '*Node2*' and will trigger a 'signal passed' event on '*Node1*' for Signal 3. Similarly, Track Sensor 4 (on '*Node1*') will trigger a 'signal passed' event on '*Node2*' for Signal 4.

The ability to use networked Track Sensors is particularly useful as this enables additional remote sensor nodes (running without monitor, keyboard or mouse) to be installed around the layout, providing additional inputs and simplifying the wiring to the Main DCC Signalling system.

Block Instruments

The DCC Signalling application also allows Block Instruments to be networked together. This enables layouts to be split into individual 'block sections' each being controlled by its own 'signal box'. Trains can then be 'offered' and 'accepted' between signal boxes, with 'bell codes' being used for communication between them.

For this example, we're going to add a second layout to the example used in the sections above, comprising two block sections (one hosted on each node), with a single line joining the two:

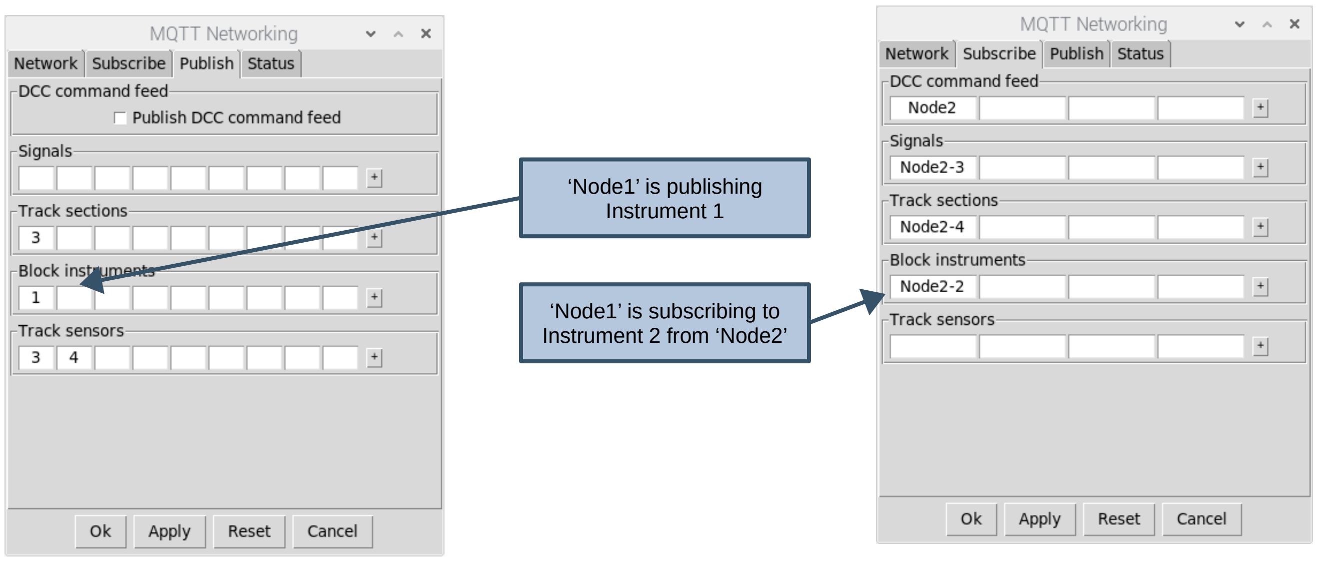


Configuring Publish and Subscribe

The network configuration is straightforward in that each ‘block section’ is self-contained so we don’t need to publish and/or subsribe to any signals – just the block instruments. Before we can link the block instruments together, we therefore need to configure:

- ‘*Node1*’ publishes Instrument 1 to the broker
- ‘*Node1*’ subscribes to Instrument 2 from ‘*Node2*’
- ‘*Node2*’ publishes Instrument 2 to the broker
- ‘*Node2*’ subscribes to Instrument 1 from ‘*Node1*’

Node1 (DCC Signalling system) publish and subscribe



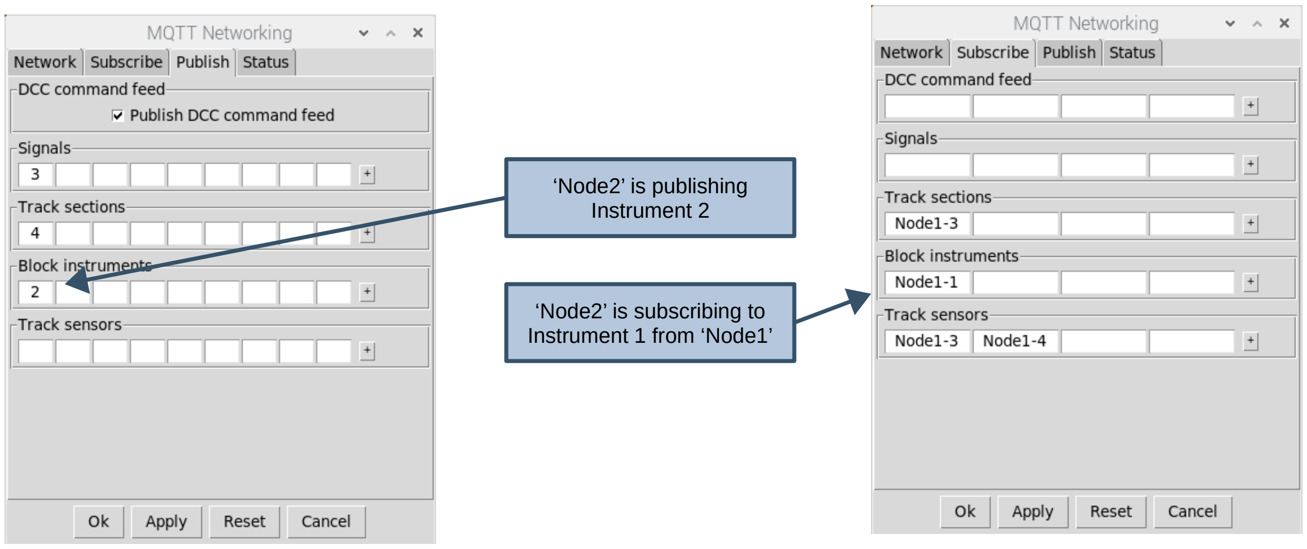
‘Node1’ is publishing Instrument 1

‘Node1’ is subscribing to Instrument 2 from ‘Node2’

Publish tab:
DCC command feed
☐ Publish DCC command feed
Signals
Track sections: 3
Block instruments: 1
Track sensors: 3, 4

Subscribe tab:
DCC command feed: Node2
Signals: Node2-3
Track sections: Node2-4
Block instruments: Node2-2
Track sensors

Node2 (Secondary Node) publish and subscribe



‘Node2’ is publishing Instrument 2

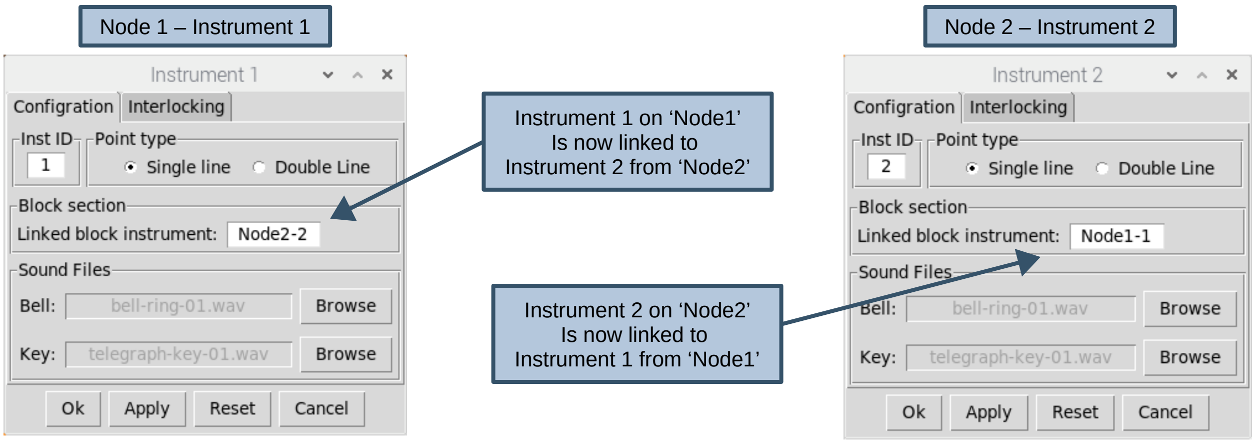
‘Node2’ is subscribing to Instrument 1 from ‘Node1’

Publish tab:
DCC command feed
☑ Publish DCC command feed
Signals: 3
Track sections: 4
Block instruments: 2
Track sensors

Subscribe tab:
DCC command feed
Signals
Track sections: Node1-3
Block instruments: Node1-1
Track sensors: Node1-3, Node1-4

Configuring the layouts

The block instruments on their respective nodes can now be linked together. On each Node, edit the block instrument configuration and specify the linked block instrument to be the ‘subscribed’ block instrument from the other networked node:


Node 1 – Instrument 1

Instrument 1

Inst ID: 1
Point type: [✓ Single line] Double Line
Block section
Linked block instrument: Node2-2
Sound Files
Bell: bell-ring-01.wav
Key: telegraph-key-01.wav

Instrument 1 on ‘Node1’
Is now linked to
Instrument 2 from ‘Node2’

Node 2 – Instrument 2

Instrument 2

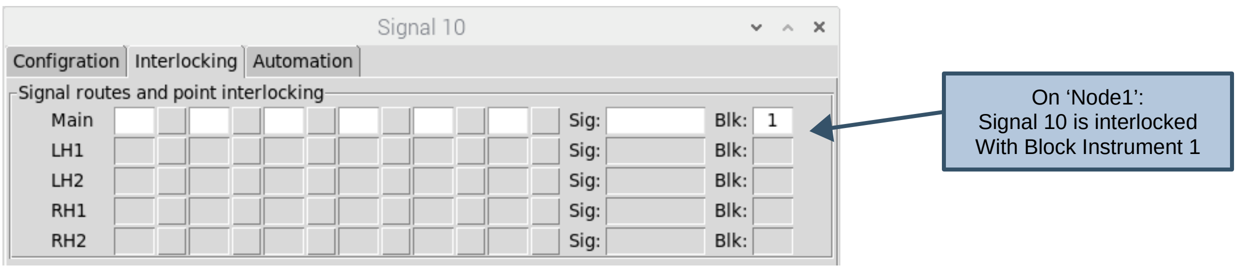
Inst ID: 2
Point type: [✓ Single line] Double Line
Block section
Linked block instrument: Node1-1
Sound Files
Bell: bell-ring-01.wav
Key: telegraph-key-01.wav

Instrument 2 on ‘Node2’
Is now linked to
Instrument 1 from ‘Node1’

To complete the configuration, the home signal that controls access into the next block section should then be interlocked with the ‘local’ block instrument protecting that section.

- On ‘Node1’ - Signal 10 needs interlocking with Instrument 1.
- On ‘Node2’ - Signal 13 needs interlocking with Instrument 2.

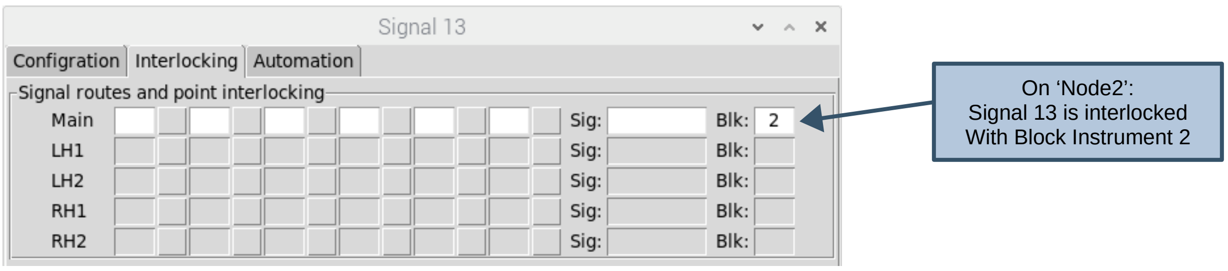
This is achieved by specifying the local block instrument on the **Interlocking** tab of the Signal Configuration Dialog:


Signal 10

Signal routes and point interlocking

	Sig:	Blk:
Main		1
LH1		
LH2		
RH1		
RH2		

On ‘Node1’:
Signal 10 is interlocked
With Block Instrument 1


Signal 13

Signal routes and point interlocking

	Sig:	Blk:
Main		2
LH1		
LH2		
RH1		
RH2		

On ‘Node2’:
Signal 13 is interlocked
With Block Instrument 2

Testing the layouts

To test the linking of the block instruments, **left click** on the **Telegraph** key of one of the Block Instruments. The Telegraph key on the other Block Instrument should flash and you should also hear the telegraph bell (assuming you have sound enabled).

To test communication the other way, **left click** on the **Telegraph** key of the other Instrument.

*Note that if you **right click** on the Telegraph key, this will bring up list of common bell codes in another window that you can use for reference for the remainder of this section.*

In their default states, both Instruments are showing 'LINE BLOCKED', which means that Signal 10 (on '*Node1*') and signal 13 (on '*Node2*') should both be locked (at DANGER).

A typical sequence of events might be:

- Signalman at '*Node1*' calls attention to the signalman at 'Node2' via the Telegraph.
- Signalman at '*Node2*' acknowledges (by repeating the same code back).
- Signalman at '*Node1*' asks if the line is clear for an express passenger train.
- Signalman at '*Node2*' acknowledges (by repeating the same code back)
- Signalman at '*Node2*' sets the Block Instrument to '**CLEAR**' (Line Clear).
    - *Note that the 'Node1' Block Instrument will also show 'Line clear' (repeating the 'Node2' Instrument) with greyed out controls as 'Node2' now has control.*
- Signalman at '*Node1*' can now set Signal 10 to CLEAR (as it is now unlocked) to allow the train stopped at Signal 10 to proceed into the next block section.
- As the train passes the signal, the '*Node1*' Signalman sends "Train Entering Section" and returns Signal 10 to DANGER.
- Signalman at '*Node2*' acknowledges (by repeating the same code back) and sets the Block Instrument to '**OCCUP**' (Train on Line).
    - *Note that the 'Node1' Block Instrument will also show 'Train on Line' (repeating the 'Node2' Instrument) and the controls will remain greyed out as 'Node2' still has control.*
    - *Signal 10 will also now be locked at DANGER, preventing the 'Node1' signalman sending another train into the same block section.*
- Signalman at 'Node2' sends "Train Arrived" as soon as the train passes the 'Node2' distant signal and sets the Block Instrument back to '**LINE BLOCKED**'.
    - *Note that the 'Node1' Block Instrument will also show 'Line Blocked' (repeating the 'Node2' Instrument) . In this state the controls are re-enabled (allowing either signalman to accept a train offered by the signalman at the other signal box.*
    - *Signal 10 will also remain locked at DANGER.*