MRG-W01 Commissioning & Feature Guide

The MRG module is intended for connection to the main IDRANet network. It receives its power from that network just like any other wired IDRANet module.

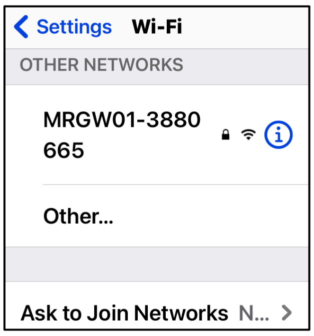
Unlike other modules, the MRG does not presently issue a startup notification after a reset or power up, so Cortex will not detect the power up. However for some types of error the module WILL issue packets to Cortex so a representative object is required to avoid ‘unknown node’ reports. The GPC network node object will be used for this purpose.

In Cortex introduce a GPC object to represent the MRG module (Tools | Design network | Add Idratek Network Object) and manually assign this object with the unique ID of 2FFB (via its object properties menu). This is the default NID of an MRG module. Do not Network enable this object

Once the module has powered up, the red LED will be on, the green LED, after a brief flash, will be held ON until the blue LED comes on (if default settings) to indicate some form of WiFi connection is operational after which the green LED will turn off.

After any reset the module will always enable it’s ‘Access Point’ mode (AP mode). This means it behaves like a mini router to which you can connect at a fixed known address. Module settings are accessed via a web browser. If a WiFi connection has been configured, AP mode will remain enabled for around 10 minutes, otherwise it will be left on indefinitely until a WiFi connection is configured (see below). AP mode will also automatically be enabled if an established WiFi connection becomes broken. This provides a way to access the module directly if not acessible via the main WiFi link. Cortex also provides a utility to manage and access multiple WiFi modules once they have been connected to a WiFi router.

Step 1 - Connect to the module Access Point:



With the AP mode operational you can access the module directly via your smart phone or WiFi enabled PC. Visit your phone/PC WiFi settings and look for the SSID (network name) of the module. Normally this will start with the module type name followed by a number. Now connect to (join) this network. Some devices may report this connection to have ‘weak security but don’t be too concerned as the connection is just temporarily being used for configuration purposes. Now enter the default AP password, which is either **IdratekWFM01** or as labelled on the module if different.

Step 2 - Connect to and set password for the on board web server:

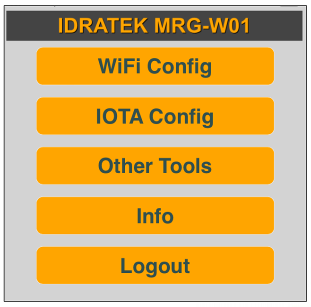
Assuming you have successfully joined the module’s AP you will be able to access the on board configuration web server. To do so you should use a browser on your device and browse to **http://20.0.0.1** or as labelled on the module if different.

The initial page will force you to set a password for accessing the module’s web server itself, before allowing you to proceed any further. Once you have entered such a password you will be asked to enter it again in order to proceed to the next step.

Step 3 - Change password for the Access Point function:

The next step asks you to change the module’s AP network password. In other words to change it away from the default value described in step 1. Once you submit the new password the module will reset and you will have to repeat the joining network process with the new AP network password. When you have rejoined you can browse again to **http://20.0.0.1** and log in.

Home Page:



When you log in you will be presented with the module’s home page. This allows you to access various module settings and information directly.

WiFi Config
Allows setting up of connections to a router and changing parameters relating to station or access point modes
IOTA Config
Allows setting up parameters for various communication protocols
Other Tools

Other module settings/functions such as enabling/disabling WiFi/activity indicator, setting module name, firmware updates, and performing soft or factory resets

Info

Display various technical items of information such as module name, firmware versions, RSSI, assigned IP address etc

Logout

Forces a logout – note that logout will automatically be performed typically after 5 minutes of inactivity

Step 4 - Connect the module to a WiFi router:

Click on **WiFi Config**. On the next page click on **Scan For Network**. After a brief pause a list of networks visible to the module will be presented. Select your router from this list and click **Submit.** You will then be asked for the router WiFi password. After this is entered the module will attempt to connect. A message will appear on your browser asking you to wait for 30s, after which the module will attempt to refresh the page with information about the success of the connection and the IP address assigned to the module by that router. At this point the module will be operating in BOTH AP and Station (STN) modes and your browser is still connected via the AP. If the process breaks the connection to the AP for some reason, then you will still be able to connect back to it using the WiFi settings on your browsing device as you did at the start. You may then check if the WiFi connection was in fact successful by observing that the WiFi Config button has turned blue and/or via the Info page. If not, you can go back to the home page and try again (e.g perhaps misspelled password).

Once the module has been connected to the main router for the first time it is advisable to perform a soft reset for good measure - either via the reset button (click once only) or via the Other Tools options. After such a reset the module usually takes a few seconds whilst connecting to the router after which the blue indicator LED will come on and the green LED will turn off.

Other Notes

Note: your smart phone may still be connected to the module's AP even after a reset so you should remember to disconnect it and rejoin your main router once you have completed such an exercise.

Accessing the module via the main router

You can access the module's on board web server at any time via the local IP address which was assigned by the router. If you have forgotten this or it has changed for some reason then, since the AP will be enabled for a few minutes after a reset, you will be able to access the module via the AP route after a reset and visit the Info page in order to find this. Alternatively it may be more convenient to utilise the Cortex WiFi module management utility (Set-up | WiFi Network) which has a process for discovering modules on the main route.

Note that if you are using this or any other IDRATEK WiFi module as part of a Cortex managed network you do not need to concern yourself with IP addresses unless you wish to access the module web server for some specific purpose (e.g to perform a firmware update)

Factory Reset

There are two methods to accomplish a factory reset. If the module is accessible via a browser then you can reach this option via the Other Tools menu. Otherwise the other option is via the physical reset button. If this is pressed **exactly 8** times in quick succession (<1.5s between successive presses) then the module will perform a memory wipe to factory state. This will be indicated shortly afterwards by the blue LED flashing slowly until the process is complete (a few 10s of seconds). This will then be followed by a module self reset.

LED indications

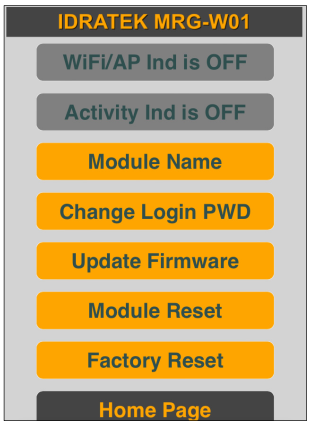
Red Led: IDRANet power state

Green LED: Similar to standard IDRATEK modules but if ON implies awaiting the radio section (master processor) to complete initialisation and establishing a connection to a router or AP mode if none found

Blue LED:

- Steady ON means either the AP is ON or that the module has connected to a router (AP mode will automatically be switched off after a few minutes).
- If WiFi activity indication enabled (via Module options) then will flash briefly when transmitting data.
- If router credentials have been set but the router cannot be reached then the AP mode will be turned on in a temporary mode – meaning that the module will continuously try to re-establish a connection to the router at regular intervals. In this case the blue LED will briefly flash every couple of seconds
- Slow flashing: indicates the initial retry phase after a connection loss to the router i.e before turning the AP mode back on.
- Slow flashing: After a factory reset request the blue LED will flash slowly for some seconds until the data wipe is complete

Other Tools Menu



WiFi/AP Ind
This button allows you to set whether the blue LED will be illuminated to indicate status of the WiFi connection.

Activity Ind
This button allows you to set whether the blue LED flashes upon IOTA activity to and from the module

Module Name
Allows you to set the name of the module for easier referencing. Note that this does not have any connection with the name that might be given to its representative object in Cortex

Change Login PWD
Allows you to change the login password (i.e access to these settings pages)

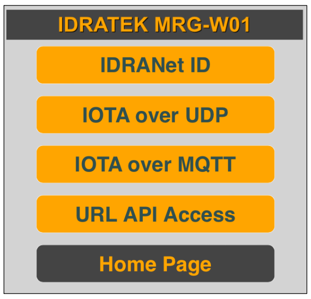
Update Firmware
Firmware updates are presently conducted manually. There is no update version check feature. You will simply be informed from time to time if there is an update and if so will be informed of a file name to enter in the relevant field. The updates themselves will normally be transmitted from the IDRATEK server so you will need to have a route to the internet (e.g you may already be connected via the home router) to access these when required.

Module Reset
This simply performs a soft reset - as if you pressed the physical reset button

Factory Reset
This clears the entire non volatile memory thus returning the module to its factory state. After such a reset the module will therefore restart with the default Access Point connectivity only – See Commissioning instructions at the beginning of this guide. Note: A factory reset may take a minute or so to complete so be patient.

Advanced Topics

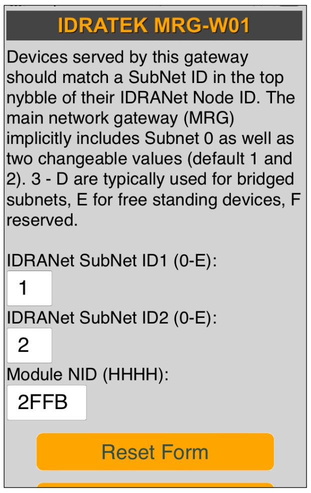
IOTA Config



IOTA (IDRANet Over The Air) is the term used to collectively describe both the communication medium (e.g. WiFi or other non wired channel) and the protocol that is used to convey information over this. In as much as is viable, the protocol element is consistent with that used over the wired network, such that communications between Cortex, wired, and wireless modules can be consistent and make use of existing integration features.

IDRANet & SubNet IDs

The IDRANet Node ID (NID) is fundamental to distinguishing different modules on the wired IDRANet. It can be thought of like an IP address and is used to enable the routing of packets to different modules. Although the WiFi module also possesses an IP address, communications between modules and even Cortex are still designed to use the IDRANet protocol so the NID is still an important parameter even for a WiFi module. The NID is a 16 bit number represented in hexadecimal format (4 characters). The MRG module is a 'special' module in the sense that it is acting as a gateway between Cortex, the wired network, and the WiFi domain. As such the module has to decide which packets which it see's on the wired network it should forward to the WiFi domain and which packets to simply ignore. It makes this decision based on the nature of the packets. Firstly whether point to point or broadcast. If point to point then also depending on the target device Node ID – or more specifically the SubNet ID value (which is the top nibble of the Node ID). By default the MRG considers SubNet IDs 0-2 as allocated to its local wired network so it will not forward any point to point packets targeted at such IDs to the WiFi domain. Conversely, if the MRG see's a WiFi packet targeted at a device with an ID in its local wired subnet then it will pass down that WiFi packet onto the wired network. Whilst these filtering parameters can be altered, it is best to try and stick with the default convention even if it means changing existing wired module IDs from values above 2FFF


IDRATEK MRG-W01

Devices served by this gateway should match a SubNet ID in the top nybble of their IDRANet Node ID. The main network gateway (MRG) implicitly includes Subnet 0 as well as two changeable values (default 1 and 2). 3 - D are typically used for bridged subnets, E for free standing devices, F reserved.

IDRANet SubNet ID1 (0-E): 1

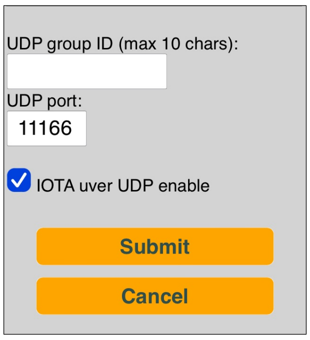
IDRANet SubNet ID2 (0-E): 2

Module NID (HHHH): 2FFB

There are 3 main routes for communicating commands to (and receiving data from) the module:

IOTA over UDP

This is a protocol which utilises UDP as the underlying means of communication. It allows direct communications between WiFi modules, bridged wired segments via IDRATEK gateway modules, and direct to Cortex (requires Cortex WiFi communications licence option). This is normally the method that is used to integrate a WiFi module into the wider IDRATEK system via Cortex so it is enabled by default. However if the module is only being utilised via MQTT or URL API communications then this channel can be disabled.


UDP group ID (max 10 chars):

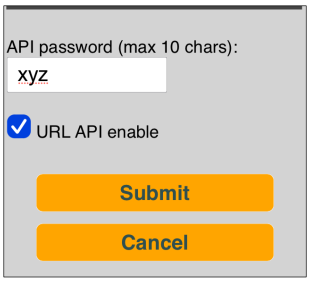
UDP port: 11166

✓ IOTA uver UDP enable

A non blank UDP group ID can be used to segment groups of modules sharing the same port number, such that you can operate multiple sub systems on the same UDP port independently of each other. Modules using a particular group name will not be visible to other groups. Using a non blank group ID can also provide an extra layer of security even if you are just operating a single group since it will make it more difficult for someone to clandestinely discover your modules even if they have access to the WiFi network.

Setting a different port number is another way to separate systems sharing the same underlying WiFi LAN. Unless otherwise indicated the default value for the UDP port number is: **11166**

URL API Access


API password (max 10 chars): xyz

✓ URL API enable

It is possible to send commands to the module via a URL format. For those familiar with it this might be better described as web hooks. Note that this uses a plain http request so is not a particularly secure communication channel if operated within a LAN which has public access to the WiFi router. The API password just prevents serendipitous access to the API, but it does not prevent someone with scanning tools from discovering the password. So it is best to leave the API access feature disabled unless you are confident about its usage context.

URL based commands are sent in the following general format:
http://**a.b.c.d**/IOTA/api/v1/**password/xxx...**
Where **a.b.c.d** is the module's IP address
password is the API password as described above
xxx… represents different command types as defined below:

1. **AllInf.json** - will cause the module to return a json formatted string to the browser, containing various items of information and signal states for the module itself.

2. **IdrPkt=**IDRANet protocol formatted packet (in HEXadecimal ASCII)/

This allows sending IDRANet packets to the wired network segment to which this module is attached. Knowledge of the IDRANet command structure and addressing is required to utilise this feature but essentially it is a scope limited form of the command structure used in the Cortex direct command line method (so can refer to that for some insight). The packet must start with 'FA' and adhere to the following format:

FAxxaabb**0400**command…

Where xx=**91** for a broadcast, **90** for a p2p command
aabb = the **NID** of the target device if xx=90, otherwise the **ZIDTID** if xx=91
command = IDRANet protocol command format, e.g. 3E0301 would toggle output 1 on a relay type module.

Example: Let's assume that the gateway module has an IP address of 192.168.1.10 and let's say its URL API password is set to 'xyz'. Now, let's say we wish to send a command to toggle output 1 on a 4 channel QRH relay module which is connected to the wired network to which the gateway is attached, and say this QRH has a NID of 108A, then you can do this via a browser using the following URL:

http://192.168.1.10/IOTA/api/v1/xyz/IdrCmd=FA90108A04003E0301/

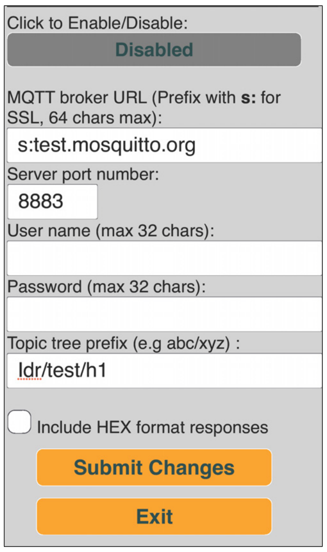
Note that packets which request information from a module will not result in such information being returned to the browser but may cause returns to the UDP channel.

IOTA over MQTT

MQTT is nowadays a well established method for communicating typically low volume low latency information over TCP/IP. It is also a way to circumvent the routing issue, i.e knowing which IP address to target a communication to, whether this be on a local area network or across the internet. It is based on the idea of having a communications 'broker' (go between) at some fixed and known URL. This broker then acts as the data router – in the sense that it can receive and buffer a communication from one device and then pass this on to another device when appropriate. The two devices both know the fixed URL of the broker and because the broker is acting as a go between they don't have to be concerned with working out the end device's IP address. Also the method uses what is known as a subscribe/publish model. This means a device wishing to send some information to interested targets 'publishes' this information to a specific 'topic' heading which the interested recipients also 'subscribe' to. Thus when the sender publishes an item, any recipients connected and subscribed to that topic will very shortly receive that information via the broker. Obviously the sender and recipients must have knowledge of the topic names (which are actually typically in the form of a tree path structure, a bit like folders on a PC). Various rules for topic naming and wild card options allow some level of targeting to selected groups of devices which might share commonality at higher levels in the topic path. A full description of MQTT is beyond the scope of this guide. You can find many helpful guides on the subject via an Internet search.

Connection parameters

Firstly you will need to specify the URL of your MQTT broker. This will not normally start with http:// as it is not an http protocol. If you are using a 3rd party broker you will be provided with information on the URL and the port number to use as well as user account credentials. If you are using an SSL connection to the broker then you should prefix the provided URL with **s:**. This is purely a requirement for the module firmware – it is not a standard convention. An example of using an SSL URL for a well known public test broker is shown here. You should then decide on a Topic tree Prefix. All communications sent via the MQTT channel from this module will then be published to a topic heading starting with this Prefix. It will also be used as the Subscription topic Prefix. In other words this module will also subscribe to the Topic tree which starts with this prefix.


Click to Enable/Disable: Disabled

MQTT broker URL (Prefix with **s:** for SSL, 64 chars max): s:test.mosquitto.org

Server port number: 8883

User name (max 32 chars):

Password (max 32 chars):

Topic tree prefix (e.g abc/xyz) : ldr/test/h1

☐ Include HEX format responses

More detail on the Include HEX option can be found in the Textual response section below.

Enable/Disable MQTT connection

To enable the MQTT connection to the broker click on the button at the top of the page. If the connection is successful the button will change colour and will say 'Enabled'. To disable a connection simply click on this button again to revert to the grey button showing 'Disabled'. Note: The connection state is memorised even if you don't click on the **Submit Changes** button

Data structures

In general IDRATEK modules are two way communicators. In other words you can both send commands to them AND receive responses. Responses might be due to a direct enquiry command or they might be auto generated as a result of some signal change or some other indirect reason ('Auto Response' in IDRATEK jargon). Auto response triggers are user definable on a signal by signal basis for a given module. For example you might enable an auto response to be generated every time a motion detector changes state, or if the light level changes by a certain amount and so on. The Radio Gateway Modules (MRG and BRG) are acting as relays for such messages between the wired and WiFi networks to which they are connected, such that for example a wired module can send a message out to the WiFi domain where it might then be captured by another gateway module and then passed down to a second wired network, or indeed such a message could be captured and acted upon directly by a stand alone wireless module (and vice versa). It is also possible for such messaging to be utilised by 3rd party applications.

IDRANet raw byte format payloads

It should be remembered that IDRATEK already have a well established integration framework which is based on the foundation of IDRANet data packets and structures. In order to maintain this framework the current fundamental implementation of MQTT is based on these structures. So data carried over the MQTT channel will be in the same format as that used on a wired IDRANet. Whilst this might not be so easily human readable it allows the existing rich integration structure to be used with minimal intervention. However to allow open access, the data carried on the MQTT channel is not encrypted (protection is conferred by the underlying carrier e.g SSL and/or closed LAN network with user credentials for access) so if you wish to use the MQTT data in other ways you are at liberty to decode and interpret the IDRANet command and data protocol into other formats.

Textual formats
To aid 3rd party application development it is possible to enable a more easily readable format for responses from wired modules relayed via the module. A HEXadecimal ASCII formatted version of the raw byte payloads can be enabled via the 'Include HEX format responses' option. This version will be published to a separate Topic channel (see Topics). Note that this is in addition to the raw byte version.

It is also possible to send commands to modules on the wired network in an abridged ASCII HEXadecimal format. This avoids the need to understand the details of forming a full IDRANet packet and is also easier to use in 3rd party applications which can only work with ASCII payloads, such as MQTT Explorer. This feature does not require enabling of any options, just that the Topic and Payloads adhere to particular forms. (See Sending IDRANet protocol packets … section below).

Topic tree structure
In order to facilitate message targeting in a manner consistent with IDRANet structures, a module will analyse the target address contained within the IDRANet packet of data which it is preparing to publish. In the IDRANet protocol there exist the concepts of Point to Point (p2p) and Broadcast (Bcst) targeted packets. P2p packets are targeted to specific modules on the IDRANet and the target address is specified by a 16 bit ID (Node ID – NID) unique to each module on a particular IDRATEK system. These IDs are usually assigned and registered into the Cortex database for a given system at the module commissioning stage. Bcst packets can be targeted either to ALL modules or to groups of modules depending on the values used in a two byte (16 bit) address field. In this case the first byte specifies a Zone ID and the second byte a Type ID. A value of 00 for both fields means ALL. Whether the target address is to be interpreted as p2p or bcst is controlled by a flag located elsewhere in the IDRANet packet structure.

Publish To Topics
So.. before it attempts to publish an MQTT bound packet the module checks to see whether the address field is to be interpreted as p2p or bcst and then sets up the topic name as follows:
If the packet is p2p then Topic path = **Prefix/p2p/SID/NID** (where SID is the most significant nibble of the NID expressed as a byte, and NID is the two byte target module NID value - both in hexadecimal format)
If the packet is bcst then Topic path = **Prefix/bcst/ZIDTID** (where ZIDTID is a two byte value in hexadecimal format)
For example say the chosen Prefix is **Idr/test/h1** and say a packet is to be sent p2p to a module whose NID is **E06C** then the topic to which this packet will be published will be: **Idr/test/h1/p2p/0E/E06C**. In this same context if the packet is to be broadcast to a group of modules with ZID=01 and TID=5A then the topic will be Idr/test/h1/bcst/015A
Note however: The gateway module will not publish to MQTT any packets whose destination is recognised to be solely within its own wired segment. By default for an MRG module this is defined as p2p packets to modules which have 0,1 or 2 as the top nibble of their NID value. This means that messages between modules on the connected wired network and indeed from such modules to/from Cortex will not be visible to the outside world.

HEX formatted data:
If the HEX ASCII format response feature is enabled then published data packets are also sent to either **Prefix/p2pH/SID/NID** or **Prefix/bcstH/ZIDTID**

AllInf response data:
It is possible to interrogate the module for its own AllInf.json response via MQTT at any time. The resulting JSON formatted response is sent to **Prefix/bcst/AInfJS**

Subscribe To Topics
The MRG module will subscribe to the following topics:
Prefix/p2p/00/#
Prefix/p2p/mySID1/#
Prefix/p2p/mySID2/#
Prefix/bcst/#

So for the default configuration and an example prefix of Idr/test/h1, the subscriptions would be:
Idr/test/h1/p2p/00/#
Idr/test/h1/p2p/01/#

Idr/test/h1/p2p/02/#
Idr/test/h1/bcst/#

Sending IDRANet protocol packets to modules on the wired domain via MQTT

It is possible for a 3rd party application to send a full (raw byte form) IDRANet protocol packet to the wired network via MQTT. To do so the Topic must begin with one of the Subscribe To topics listed above and, in place of the # character, be followed by sub topics which target the desired module or modules. In order for the Payload to be interpreted as a raw byte format packet the first byte must have the special value of 85 (55 hex). For example say you wish to send a p2p packet to a module on the main network whose NID is **1008**. Say also that the MQTT topic prefix is **Idr/test/h1/**, then the topic you would construct to send this packet would be **Idr/test/h1/p2p/01/1008**. The payload would then be a string of raw bytes, the first of which has a value of 85, and the remainder adhering to the full IDRANet packet format as used on the wired network.

Alternative Textual Payload Format

An alternative textual payload format can also be used, which looks more like what a user might see if sending direct command packets from Cortex. This is provided to ease development work since Cortex also provides some level of insight into IDRANet command syntax.

The construction of the Topic remains as described above but if you then start the Payload with "**>IdrCmd=**" then the remainder of the payload will be expected to be just the command section of a packet, in ASCII HEX format. The reason the full packet is not required is because the address targeting information is already implicit in the Topic construction.

For example say you wish to send a p2p packet to toggle the channel 1 relay of a QRH module on the main network. Say the QRH module has a NID of **1008**. Say also that the MQTT topic prefix is **Idr/test/h1/**, then the topic you would construct to send this packet would be: **Idr/test/h1/p2p/01/1008**.
Now, instead of using the full raw byte format, the Payload could instead be: **>IdrCmd=3E0301/**
The gateway module will then internally construct the full raw byte packet using this information together with some default presumptions. In particular it will default the sender's NID to have a value of **0000.** This is ok for packets which are just sending action commands to wired modules. If however you are sending an interrogation command (such as what is the temperature?) then you will need to use a sender's notional NID appended to the Payload. See below

Commands which request information from wired domain modules

If a command is sent to a wired module asking for some information, then the wired module needs to know where to send the reply. When such commands are sent using the full raw byte format the sender's NID is included as part of the data in this packet, so the application formulating the raw packet can explicitly control where any responses are to be sent. However in the case of using the simplified ASCII format method outlined above the implicit assumption is a sender's NID of **0000**. To override this assumption you can append a sender's NID value to the end of command packet using the syntax: **/sender's NID/**. The actual value of the sender's NID would be chosen like you might chose the unique NID for any module – in other words something unlikely to be used by any other module and outside of the Subnet ID of the target wired network. Typically something in Subnet F would be suited - such as **FFFB**. So as an example, let's say we have a wired temperature sensor module whose NID is **101C** and we wish to request the temperature value from this module (the IDRANet Hex command for which is simply **45**), then assuming the same MQTT topic prefix as previous examples we would first construct the Topic as: **Idr/test/h1/p2p/01/101C** and the Payload would now be: **>IdrCmd=45/FFFB/**

The resulting data will be sent in a packet addressed to NID FFFB and would be visible as an MQTT return to a 3rd party application subscribed to **Prefix/#** for example, or **Prefix/p2p/0F/FFFB** if more specific topic filtering is required. You would then need to know how to decipher the contents of such a packet to extract the temperature data (outside scope of this document).