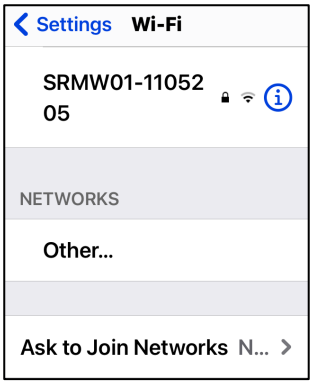
SRM-W01 Commissioning & Feature Guide

The SRM receives its power directly from an AC supply. When the AC supply is first switched on the blue LED will flash briefly followed shortly by a steady ON state – indicating that some form of WiFi is active.

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. Thus it is a fallback method to access the module directly if not accessible 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:

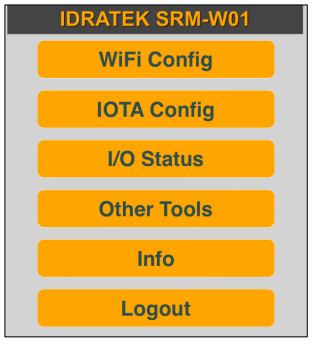
Assuming you have successfully joined the module's AP you will be able to access the on board configuration web server. To do so you should use a browser on your device and browse to **http://20.0.0.1** or as labelled on the module if different.

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
I/O Status
Allows basic interaction with and viewing states of input output functions provided by this module
Other Tools
Other module settings/functions such as calibrating dimmer range, 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.

Step 5

Option 1 - Commissioning into Cortex:

This step assumes that Cortex has a route to the same WiFi network either via an MRG type gateway module or via the Cortex direct to WiFi software option. As with commissioning of wired modules Cortex must be in a network stopped state. The commissioning process is then similar to what would be followed for a wired SRH module:

Reset the module either via it's reset button or via the Other Tools | Module Reset button if connected to it via a browser. After a few seconds (whilst module initialises and connects to Wifi) a standard commissioning dialogue should appear in Cortex. This will guide the remainder of the process for commissioning the module into the Cortex database. This process includes assigning a unique IDRANet ID to the module. WiFi modules are assigned a hexadecimal ID in the E000 to EFF0 range.

Add Analogue input objects

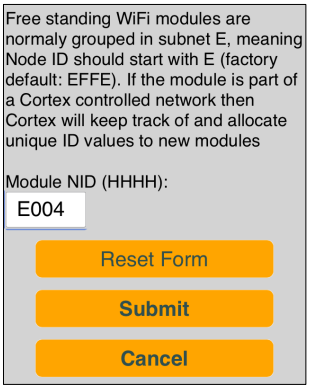
At the date of this document the native SRH object in Cortex does not automatically include provision for the analogue inputs so these must be added to the parent object manually. To do this, in Cortex select the SRH parent object then visit Tools | Design Network | Add Idratek Network Object. In the list of module types that appears, go to the bottom until you find a sub section titled Variant Sub Objects. Click on this and select Analogue Inputs. An analogue inputs container will then be added under the SRH parent object. This container will have provision for two inputs. You should use the first to represent the general purpose analogue input and the second for the RSSI measurement. Appropriate icons can later be picked to suit these two signals.

Analogue Input value mapping

The default analogue signal objects default to a quadratic 1:1 mapping which is not suited either to a thermistor nor the form of the RSSI signal. So you must alter the mapping function accessible from the behaviour menu of each analogue input object (icon is a 1:1 incline graph). For a thermistor input typically you would select the NTC thermistor beta function and appropriate values for the thermistor and pulldown resistor depending on your choices for these components. For the RSSI signal choose the Quadratic option with a=0, b=2.56 and c= -256

Option 2: Manual ID Commissioning:

If the module is to be used in some other automation context it will still need to have some form of unique identifier programmed into it. Addressing the module, even via MQTT, still relies on the IDRATEK concept of the Node ID (NID). This is a two byte value expressed in hexadecimal (so 0000-FFFF). Furthermore, under IDRATEK conventions, only values between E000 and EFF0 should be used for stand alone wireless modules. In Option 1 above, Cortex maintains a list of NID values it has handed out for a given database so it can pick a new value for every new module introduced. If you are conducting the process manually you will just have to be careful to avoid duplication. The default NID for this type of module is EFFE, so you will need to change this manually to some other value, such as E004, by visiting the IOTA Config | IDRANet ID page via the browser. Upon clicking the Submit button the new value will be saved and the module will then become addressable via MQTT or such using that NID (See Advanced Topics section)



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 to it by the router. If you have forgotten this 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 and visit the Info page in order to find this. Alternatively it may be more convenient to utilise the Cortex WiFi module management utility.

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. This will then be followed by a module self reset.

Blue LED indications

- 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 LED
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)

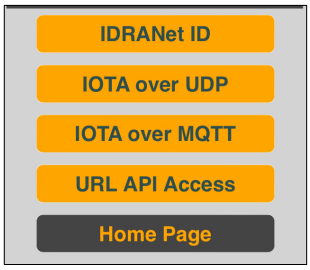
Update Firmware
Firmware updates are presently conducted manually. There is no update version check feature. You will simply be informed from time to time if there is an update and if so will be informed of a file name to enter in the relevant field. The updates themselves will normally be transmitted 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

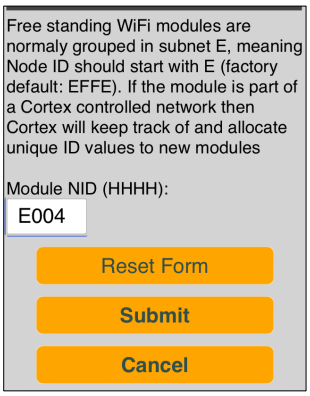
IOTA Config



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 ID

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 possesses an actual 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). By convention, WiFi modules are assigned a unique (to a particular system) NID value in the range E000 to EFF0. A factory state module will always initially have a NID of EFFE, so to avoid Cortex based assignment clashes you should only ever power up and commission one new module at a time. Alternatively you can assign NIDs manually using the module IDRANet ID settings page and then commission these into Cortex objects later. The danger of doing this is that if you are not careful to check the existing Cortex database then you run the risk of choosing a NID value that Cortex has already chosen in the past for some other module. Similarly, if at some later time you decide to change the module NID using the web server interface then you must be careful to change the NID of it's representative object in Cortex to match (and also check that this does not clash with any other IDs assigned in Cortex)

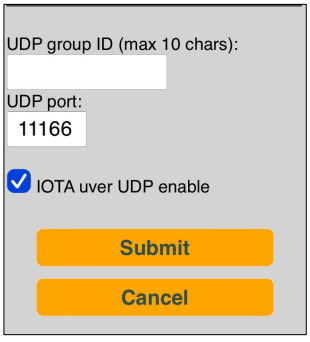


There are 3 main routes for communicating commands to (and receiving data from) the module:

IOTA over UDP

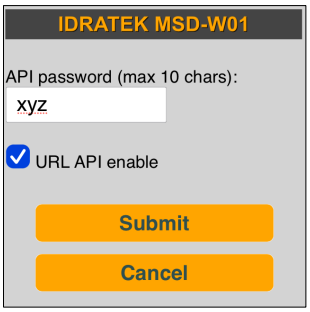
This is a protocol which utilises UDP as the underlying means of communication. It allows direct communications between WiFi modules, bridged wired segments via IDRATEK gateway modules, and direct to Cortex (requires Cortex WiFi communications licence option). This is normally the method that is used 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 for greater efficiency.

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



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 this 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.

2. **Relay/1/Action=p1/**

Where,
p1=On, Off or Toggle

Example: Say the IP address of the module is 192.168.1.12 and the password is xyz, then to switch the relay on you might use the following URL:
http://192.168.1.12/IOTA/api/v1/xyz/ Relay/1/Action=On/
to toggle you might use:
http://192.168.1.12/IOTA/api/v1/xyz/ Relay/1/Action=Toggle/

3. **IdrCmd=**IDRANet protocol Hex formatted command/

This command requires knowledge of the hexadecimal format commands fundamentally used to access detailed module functions, for example as used by Cortex and Reflex. It has been provided for potential specialist usage. For example, if the module IP address is 192.168.1.12 and the API password is xyz:

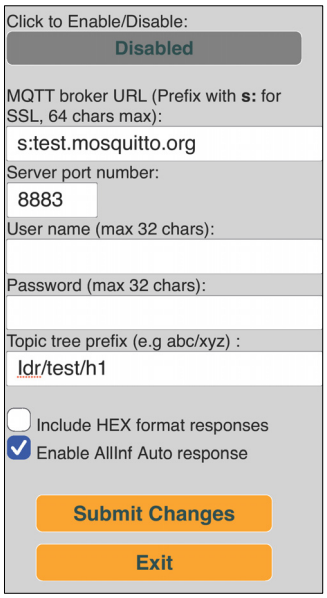
http://192.168.1.12/IOTA/api/v1/xyz/IdrCmd=3E0302/ - Commands a relay toggle operation

Note that commands which request information 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.

More detail on the Include HEX and Enable AllInf options 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 enabled/disabled 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 the button changes state, or if the analogue input changes by a certain amount and so on. The default for this module is that all such auto response triggers are not enabled. For efficiency, IDRANet format responses generally only contain information relating to the signal or query which triggered them and the nature of the content is indicated by one or more of the bytes of information within the same data packet. This is quite different to typical MQTT formulations where, for example, you might expect to see a temperature value being published to a temperature topic, or even an entire data set containing multiple signal types being published to some more general topic (see AllInf.json below).

IDRANet 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 addressing structures. In order to maintain this framework the current implementation of MQTT including Topic and Payload formulation is based on these structures. Payload data carried over the MQTT channel primarily utilises the same format that is used on a wired IDRANet. Whilst this might not be 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. The payload for IDRANet format packets is encoded as raw byte values.

Textual formats
To aid 3rd party application development it is also possible to enable two types of more easily readable formats for responses from 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.

A feature is also provided to return a JSON formatted 'full' information set, containing comprehensive detail about the module and all the signals in one lump. This can be enabled via the 'Enable AllInf Auto response' option. This response will be generated in addition to any raw data responses and, if enabled, hexadecimal format responses - so long as the reason for the response is not the general AllInf.json query command (see below).

In summary: The Hexadecimal ASCII option just makes it easier to visualise payload content when using off shelf applications such as MQTT explorer. The AllInf option is useful when you want easier access to module data without having to understand how to interpret IDRANet protocols. Also the JSON format makes it easier to extract into applications which already have JSON decoding capabilities.

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 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
Before it attempts to publish an MQTT bound packet the module therefore 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 value, 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 **106C** then the topic to which this packet will be published will be: **Idr/test/h1/p2p/01/106C**. 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**

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

AllInf response data:
If enabled, the AllInf JSON formatted response is sent to **Prefix/bcst/AInfJS**

Subscribe To Topics
The module will subscribe to the following topics:
Prefix/p2p/mySID1/myNID
Prefix/bcst/myZIDmyTID
Prefix/bcst/00myTID
Prefix/bcst/myZID00
Prefix/bcst/0000

For example if the Prefix is **Idr/test/h1** and say this module has a NID of **E004** and group identifies with ZID=**05** and TID=**3C** then the subscriptions will be as follows:
Idr/test/h1/p2p/0E/E004

Idr/test/h1/bcst/053C
Idr/test/h1/bcst/0500
Idr/test/h1/bcst/003C
Idr/test/h1/bcst/0000

The rationale behind this scheme is to allow the existing IDRATEK communication framework to be used with minimal changes. Thus, if a module is Reflex programmed using existing rules and Cortex Tools then any such Reflexes can operate over an MQTT connection with no further intervention. It means for example you can Reflex program a button on Module A in location 1 to toggle the dimmer output on Module B at location 2 using a common MQTT broker between them even when the two modules are in entirely different locations and with no Cortex involved thereafter (assuming only that both modules have access to an internet connection at both locations).

Sending textual format commands to the module via MQTT
Whilst sending commands to an IDRATEK module is most comprehensively achieved using the raw bytes IDRANet protocol and requires knowledge of the command structures and parameters (not in scope of this document), it is nevertheless possible to send some commands in more easily composed textual forms.

Specifically: MQTT payloads sent to any of the subscribed to topics listed above will be interpreted as textual format commands if the first character is "**>**". The syntax following this character is similar to that for the URL API commands described before. For example, if the prefix is **Idr/test/h1** and say this module has a NID of **E004** then you can send a plain text command to it by sending to the topic: **Idr/test/h1/p2p/0E/E004**. The payload syntax might then be for example:

>Relay/1/Action=Toggle/
which would cause the output state to be toggled.

Or for example, the following will elicit a JSON format information packet return:
>AllInf.json

Alternatively, more powerfully, any IDRANet Hexadecimal format command can be issued directly to the module. This does require knowledge of command codes and parameters, but these can often be gleaned from the Cortex 'command insight' feature. One thing to be aware of however is that the module expects to receive only the command section of an equivalent direct command line in Cortex, since the addressing is already implicit in the Topic to which you are sending this command.

For example, If you wished to send a command to this module from the Cortex command line to toggle the relay, then the structure of that command in Cortex would look like this: FA90E00444003**E0302**. If you want to send the same command to this module via MQTT then you would first set the Topic to : **Idr/test/h1/p2p/0E/E004** (assuming example prefix as above), then send the following as the Payload: **>IdrCmd=3E0302/**

As another example, if you wish to enquire about the present analogue input level then you would use the same Topic and the Payload would be: **>IdrCmd=41/**. The analogue input value will then be returned via an MQTT message at the very least in raw byte form but could also be accompanied by HEX ASCII and/or full JSON info forms depending on options described in Textual Formats section.