Afterburner MQTT

The Afterburner can connect to an MQTT broker allowing the ability to integrate information from the Afterburner into a larger home automation scheme, or the exciting possibility of controlling and monitoring your heater via a remote connection over the Internet.

The MQTT broker can be hosted on your own home network, or elsewhere on the Internet.  
Internet based brokers obviously allow remote access easily as they are already beyond any local firewall, but nothing prevents you making your own private broker visible from the Internet by adding it to your exposed connections.  
The details of exposing a private broker are beyond the scope of this document as it is very dependent upon what site equipment is in use.

## Topics

MQTT operates by using topics to provide or collect information to/from elsewhere.  
In MQTT speak, this is Publishing and Subscribing.

Publishing is pushing topics to the broker.   
The broker maintains a list of clients that have expressed interest in a specific topic by subscribing to that topic.  
When new topics are published, they are then relayed by the broker to the clients subscribed to it.

It is quite feasible a topic may simply be lost at the broker if no one has subscribed to it.

In the afterburner, there are two conventions used for the topic names:  
“Global” topic that provides detailed information within a JSON encapsulated payload.  
“Specific” topics that only provide information about a specific entity.

The specific topics are only a subset of the full information available via the JSON topic, but are sufficient to provide remote control and monitoring of the Afterburner by simple MQTT clients.

## Topic prefix

It is important that the information to/from the Afterburner is uniquely identified.  
This is especially important should you have multiple Afterburner’s attached to the same broker, or indeed the final MQTT client.   
**It is vital for the prefix to be unique if you use a public broker.**

By default, as shipped, all MQTT topics are preceded with the prefix ***Afterburner***.

This should be adapted to ensure your unit is uniquely distinguishable.  
The topic prefix can be anything that makes sense to you.  
A guaranteed unique name could be adding the last 3 hex digits of the MAC address to the topic.  
(The first 3 MAC digits are constant and identify the vendor/device, ie Espressif ESP32)

Eg: if your MAC is 24:0A:C4:C0:AB:95, set the topic prefix to be ***AfterburnerC0AB95***.  
Do not include the colons, they may lead to unpredictable behaviour.

The following description uses ***topicPrefix*** to indicate this unique name.

## Connection State

Upon connection to a broker, it is generally recommended to publish a retained topic that indicates the connection is available to the Afterburner.

Likewise should the connection be lost, indication should also be provided the unit is not available.  
The broker offers a “last will and testament” feature that allows clients to be informed upon loss of the Afterburner’s connection.

The same topic name is used for both instances.

### Publish

When the Afterburner establishes the broker connection, the following topic is published and retained by the broker:

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/status | “online” |

### Last Will and Testament

When the Afterburner establishes the broker connection, the last will & testament will be defined. Should the broker connection unexpectedly disappear, the broker will then publish the following:

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/status | “offline” |

## Control and Status

The Global Topic is /JSONout  
The Specific Topics start with /sts *(after the topicPrefix)*

### Publish

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/JSONout | JSON encapsulated string providing full detailed status of the heater state |
| ***topicPrefix***/sts/Run | 0=heater OFF, 1=heater ON |
| ***topicPrefix***/sts/RunState | An integer enumeration indicating the current running state of the heater |
| ***topicPrefix***/sts/RunString | String that describes the running state of the heater |
| ***topicPrefix***/sts/TempDesired | Desired temperature in °C |
| ***topicPrefix***/sts/TempCurrent | Current temperature in °C – from sensor used for thermostat |
| ***topicPrefix***/sts/Temp2Current | Current temperature in °C – from optional second sensor |
| ***topicPrefix***/sts/Temp3Current | Current temperature in °C – from optional third sensor |
| ***topicPrefix***/sts/TempBody | Current temperature in °C of the heater’s casing |
| ***topicPrefix***/sts/ErrorState | An integer enumeration indicating the current error state of the heater |
| ***topicPrefix***/sts/ErrorString | String that describes the error state of the heater |
| ***topicPrefix***/sts/Thermostat | 0=fixed pump demand, 1=thermostat mode active |
| ***topicPrefix***/sts/PumpFixed | Desired pump delivery rate, in Hz |
| ***topicPrefix***/sts/PumpActual | Current pump delivery rate, in Hz |
| ***topicPrefix***/sts/FanRPM | Current fan RPM |
| ***topicPrefix***/sts/InputVoltage | Current input voltage to heater unit |
| ***topicPrefix***/sts/GlowVoltage | Current voltage applied to glow plug |
| ***topicPrefix***/sts/GlowCurrent | Current amperage applied to glow plug |
| ***topicPrefix***/sts/GPalg | Analogue input value applied to analogue input as a percentage |

### Subscribe

The Afterburner subscribes to the following two topics:

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/JSONin | JSON encapsulated command, using defined dictionary names |
| ***topicPrefix***/cmd/# | # = wildcard wildcard can be any valid JSON dictionary name and will be treated as if it were part of a JSONin command string |

A client can choose whether to publish commands using either /JSONin or /cmd/*JSONname*.  
eg. The both the following are valid usages:

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/JSONin | {“TempDesired”:22} |
| ***topicPrefix***/cmd/TempDesired | 22 |

The /JSONin topic can support multiple names, whereas the /cmd/xxx is singular

eg. This is valid for /JSONin, but must be sent as individual /cmd’s:

|  |  |
| --- | --- |
| **Topic** | **Payload** |
| ***topicPrefix***/JSONin | {“TempDesired”:22,”Run”:1,”Thermostat”:1} |

## JSON dictionary

The JSON names are defined in the JSON dictionary document.

Configuring MQTT

First and foremost, the Afterburner must be able to connect to a broker, which means it must be configured to use STA wifi mode.  
STA mode being when the Afterburner is connected to a local WiFi network.

Once connected, discover the Afterburner’s STA IP address using the OLED menus.

Using a telnet client, connect to the Afterburner (telnet port 23).  
This accesses the general debug information output, from there you can now access a MQTT configuration menu.

As debug information freely flows, it is of benefit to suppress the notifications that appear when JSON data is packaged for delivery.  
Hit ‘J’ to toggle off JSON reporting. This avoids unwanted distraction whilst using the MQTT menu.

Hit ‘M’ to present the MQTT setup menu

| **Keystroke** | **Function** |  |
| --- | --- | --- |
| 1 | Enter broker’s IP address The IP address must be entered using dotted notation |  |
| 2 | Enter broker’s port *(default 1883)* |  |
| 3 | Enter the username to access the broker |  |
| 4 | Enter the password to access the broker |  |
| 5 | Enter the topic prefix |  |
| 6 | Enter the desired Quality of Service |  |
| 7 | Enable or disable MQTT connection |  |
| <ENTER> | Save and exit the parameters |  |
| <ESC> | Abort any changes |  |