FLAME BOSS MQTT PROTOCOL

January 29, 2020

License

Table of Contents

Overview

For programmers, here is how your application can communicate with your controllers over the Internet, over the LAN, and directly in Access Point mode.

Cloud Connect

When your Flame Boss device connects to the cloud it connects to one of multiple servers. You have to connect to the same server to communicate over MQTT to your Flame Boss device. If your device is online, get the MQTT host name with a GET HTTPS request to this URL:

https://myflameboss.com/api/v1/devices/8838/mqtt
(Replace 8838 with your device ID.)

The response will be a JSON object like this:

{"server":"s5.myflameboss.com"}

or if your device is not online, like this:

{"message": "not online"}

You can then open a TLS MQTT connection to this server on port 8883, or a non-encrypted MQTT connection on port 1883.

Cloud Authentication for Controllers

To connect to the Flame Boss Cloud MQTT broker, devices use a username that begins with "D-" and ends with the device ID (e.g. D-3796).

Devices are programmed with a password during production and there is no way for users to see it or change it.

Cloud Authentication for Users

To connect to the broker, users need information that is returned by the Flame Boss API. The following HTTP request will return the credentials.

Method	POST
URL	https://myflameboss.com/api/v4/sessions
Request Parameter Names	session[login] session[password]
Response Parameters	user_id username auth_token

The MQTT username has prefix "T-" followed by the user_id (e.g., T-1234). The password is the auth_token value.

Local Connect and Direct Connect

The controller also is a limited MQTT broker itself, in both station mode and in AP Mode. You can connect to the controller directly to both configure it or access data. However, while you are connected to your device directly, the cloud features are disabled. We recommend using the cloud connection when possible.

Direct Connect Connection

Join your device's AP with Wifi Settings (SSID: FB-<device ID>, no password) and use the following IP address and port:

AP Mode:
IP address: 192.168.4.1
Non-SSL port: 1883

Local Connect Connection

Within five minutes of turning on your Flame Boss device and it joining your local network in station mode, you can use Bonjour/mDNS to discover the IP address. Its host name is fb-<device ID>.local. Try this from the command line on a Mac or Windows computer:

```
ping fb-8838.local
```

You can also find the IP address by looking in the Advanced WiFi menu of your Flame Boss 200, 300, or 500.

Lastly, you could look in the DHCP configuration of your wifi router - it will show the IP addresses assigned to all connected devices.

Use the following port number:

Non-SSL port: 1883

Controller Authentication

Your controller has a Local Access setting that can be one of:

0 - No access
1 - Authenticated access
2 - Open access

If Local Access is Authenticated then the following credentials must be used:

username: fb
password: the device's PIN as a string with no leading zeroes

If Local Access is Open then any username and password will be allowed to connect.

Set Local Access to No Access if you never use Local or Direct Connect.

Uplinks and Downlinks

We use the following terms you'll need to know:

An *uplink* is a message published by the sensor or controller device.

A *downlink* is a message published by the cloud or by your application to the device.

Topics

All topics have the prefix

flameboss/<device_id>/

Uplinks will be published on these topics:

Topic	Application
flameboss/<device_id>/send/time	Controller sends its current time for synchronizing clocks.
flameboss/<device_id>/send/open	Main topic for most applications when cook is public
flameboss/<device_id>/send/data	Main topic for almost all user applications when cook is private
flameboss/<device_id>/send/fw	Upgrading firmware
flameboss/<device_id>/send/rc	Remote Control
flameboss/<device_id>/send/console	Remote Console
flameboss/<device_id>/send/adc	ADC Logging, see adc_recs

Downlinks are published on this topic:

flameboss/<device_id>/recv

Access Control

On the Flame Boss Cloud MQTT broker, for all devices added to the user's profile, users have publish access to the devices' recv topic for sending downlink messages, and subscribe access to the devices' send topics for receiving uplink messages.

All users have subscribe access to any device's open and time topics for receiving public cook data.

Payload Formats

Each message payload is formatted as a JSON object.

Each message has a "name" element.

Versioning

To work with future updates, applications should ignore any unknown message names and unknown attributes in messages.

Future versions may add new topics and new message names to add features without breaking backward compatibility.

Quality of Service

MQTT supports three levels of quality but Flame Boss only supports two. Flame Boss controllers use level 1 for most messages, but level 0 for some diagnostic messages.

Startup Messages

The following messages are sent by the controller each time it connects to the broker to make sure the cloud or app has the latest state of the device.

Topic send/open

protocol

Topic send/time

cook

Topic send/data

id
time
set_temp_limits
set_temp
wifi
mqtt_server
mqtt_config
pid
meat_alarm
pit_alarm
console_config
device_temp
dc_input
labels (if supported on controller)
sound
temp_scale
wifi_module
local_access
overtemp_limits

Topic send/adc

adc_log

Topic send/rc

display
alarm

Topic send/nvram

nvram

Last Will and Testament (LWAT)

Topic: send/data

disconnected

This is the "last will and testament" message of the controller MQTT client. It is published automatically by the broker when the device is disconnected by the broker.

```
{
  "name": "disconnected"
}
```

Protocol

Topic: send/open

protocol

This uplink is published when a connection starts.

```
{
  "name": "protocol",
  "version": 2
}
```

Configuration Uplink Messages

Topic: send/data

local_access

```
{
   "name": "local_access",
```

```
    "value": <"authenticated" or "open">,
  }
```

time

```
  {
    "name": "time",
    "epoch": <integer: time in Unix epoch scale>
  }
```

id

```
  {
    "name": "id",
    "hw_id": <integer: device type id>,
    "device_id": <integer: flame boss device id>,
    "uid": <string: base64 encoded uid>,
    "pin": <integer, optional, only sent when device is in AP mode>
  }
```

temps

The temps message includes both configuration and measurement information because it includes the current set_temp along with current temperatures. See Measurement Uplinks for its description.

set_temp

```
  {
    "name": "set_temp",
    "min": <integer>,
    "max": <integer>
  }
```

wifi

```
  {
    "name": "wifi",
    "index": <integer 0 or 1>,
    "ssid": <string>
  }
```

meat_alarm

```
{
   "name": "meat_alarm",
   "sensor": <integer, 1-3>,
   "action": <"off", "on", or "keep_warm">,
   "done_temp": <integer>,
   "warm_temp": <integer>,
}
```

set_temp_limits

```
{
   "name": "set_temp_limits",
   "min": <integer>,
   "max": <integer>
}
```

pit_alarm

```
{
   "name": "pit_alarm",
   "enabled": <boolean>,
   "range": <integer>
}
```

labels

```
{
   "name": "labels",
   "values": <array of 4 strings, max 12 char each, e.g. [ "Pit", "Brisket", "Butt", "Turkey" ] >
}
```

sound

```
{
   "name": "sound",
   "config": <"off", "chirps", or "alarms">,
   “status”: <"alarm" or "off", >
}
```

pid

```
{
   "name": "pid",
   "p": <integer, p * 100>,
   "i": <integer, i * 1000>,
   "d": <integer>,
   "ff": <integer: learned duty cycle when no error from adaptive feed forward method>,
   "min_dc": <integer: minimum duty cycle>,
   "pvl": <integer: process value limit, caps output at this number * pit temp>
}
```

open_pit

```
{
   "name": "open_pit",
   "max_pause": <integer, max open pause time in sec>
}
```

device_temp

This message is sent when it changes 5 degrees C or 1 degree if device temp is high. (Not supported on 400.)

```
{
  "name": "device_temp",
  "value": <integer>
}
```

dc_input

This uplink is published when dc_input changes 0.1 volts.

```
{
  "name": "dc_input",
  "value": <integer: input voltage in decivolts>
}
```

temps

This uplink is published about every 30 sec when the temperatures are not changing, more often when any of the temperatures are changing.

```
{
   "name": "temps",
   "cook_id": <integer, see cook downlink>,
   "sec": <integer: epoch s of data point, might be earlier then ts if it was logged on device>,
   "temps": <array of integers: temperatures at sec in configured temp scale>,
   "set_temp": <integer>,
   "blower": <integer: blower duty cycle in .01% scale, 10000 = 100%>
}
```

mtemps

This uplink sends logged temperature data.

```
{
   "name": "mtemps",
   "cook_id": <integer, see cook downlink>,
   "data": Array of [<integer> x 7]
}
```

Each element of data contains integers in the following order:

1. sec
2. set temp
3. pit temp
4. meat temp 1
5. meat temp 2
6. meat temp 3
7. blower

meat_alarm_triggered

This uplink is published when the meat is done.

```
{
  "name": "meat_alarm_triggered",
  "sensor": <integer, 1-3>
}
```

pit_alarm_active

This uplink is published when pit alarm becomes active after being enabled. (It becomes active when the pit temp becomes nearly at the set temp.

```
{
  "name": "pit_alarm_active"
}
```

pit_alarm_triggered

This uplink is published when the pit temp goes out of range set by pit alarm if the pit alarm is active.

```
{
  "name": "pit_alarm_triggered"
}
```

vent_advice

This uplink is published when pit temp has been above set temp for a long time.

```
{
  "name": "vent_advice"
}
```

opened

This uplink is published when the controller detects the cooker is opened.

```
{
  "name": "opened"
}
```

closed

This uplink is published when the controller detects the cooker is closed.

```
{
  "name": "closed"
}
```

probe_overtemp

```
{
   "name": "probe_overtemp",
   "sensor": <integer>
```

```
}
```

device_overtemp

```
{
  "name": "device_overtemp"
}
```

wifi_scan

This uplink is published at startup to show results of discovered access points, one message for each AP discovered.

```
{
 "name": "wifi_scan",
 "index": <integer>,
 "count": <integer>,
 "ssid": <string>,
 "rssi": <integer>,
 "bssid": <string>
}
```

mqtt

```
{
  "name": "mqtt_server",
  "timeout": <integer, seconds to timeout waiting for response from server>,
  "keepalive": <integer, seconds to wait between pings>,
  "index": <0 for user config, 1 for system config>,
  "host": <string: hostname of broker>,
  "ip": <optional, string: ipv4 address of broker, used if host is blank or dns lookup of host fails>,
  "port": <integer>,
  "tls": <boolean>,
  "username": <string>,
  "local_en": <boolean, true if local connect is enabled>
}
```

Note that timeout and keepalive are common settings that apply to both MQTT configurations.

console_config

```
{
  "name": "console_config",
  "logging": <boolean>
}
```

console

```
{
  "name": "console ",
  "input": <string>
}
```

Remote Control Uplinks (Topic: send/rc)

These messages are published on the "rc" topic.

display

```
{
  "name": "display",
  "lines": ["line1", "line2", "line3", "line4"],
  "cursor_on": <boolean>,
  "cursor": <array of 2 integers for row and column, omitted if cursor_on is false>
}
```

alarm

```
{
  "name": "alarm",
  "on": <boolean: true if alarm is on>
}
```

chirp

```
{
  "name": "chirp"
}
```

Remote Control Downlinks

press

```
  {
    "name": "press",
    "button": <string "back", "down", "up", and "next">,
    "held": <boolean, optional, true for a press and hold action, ignored unless button is
"next">,
    "times": <integer, optional, default 1, ignored if held is true>
  }
```

Diagnostic Downlinks

write

```
  {
    "name": "write",
    "type": <"nvram" for non-volitile memory or EEPROM>,
    "addr": <integer>,
    "data": <string, base64 encoded data>
  }
```

assertion_failed

```
  {
   "name": "assertion_failed",
   "index": <integer, 0 or 1>
   "file": <string>,
   "line": <integer>
  }
```

adc_log

```
  {
    "name": "adc_log",
    "mask": <integer, a bitmask of ports to log>
  }
```

overtemp_limits

```
{
  "name": "overtemp_limits",
  "probe": <integer>,
  "device": <integer>
}
```

temps

```
{
  "name": "temps"
}
```

test

```
{
  "name": "test",
  "op": <string>
}
```

These are only available when WiFi is configured for testing within Flame Boss offices.

The following values are valid for the “op” attribute:

assertion_failure
assertion_run_failure
exception
wd_timeout

Firmware Update Uplinks

These messages are published by devices on the fb_fw topic.

versions

```
{
  "name": "versions",
  "hw_id": <integer>,
  "app": <string>,
  "boot": <string>,
  "wifi": <string>,
  "next_addr": <integer: load address of next app>
}
```

dl_ack

```
{
   "name": "dl_ack",
   "id": <integer>,
   "offset": <integer>,
   "length": <integer>
}
```

Target received the dl_start or dl_block

Firmware Update Downlinks

These messages are published by the server on the recv topic.

dl_start

```
{
   "name": "dl_start",
   "id": <integer: id of firmware file>,
   "target": <string, "app", "boot", or "wifi">,
   "version": <string>,
   "addr": <integer>,
   "length": <integer>,
   "part": <integer>,
   "last_part": <boolean>,
   "crc": <integer, crc32 of whole download>
}
```

dl_block

```
{
   "name": "dl_block",
   "id": <integer>,
   "offset": <integer, offset in download, zero based>,
   "data": <string, base64 encoded for block>
}
```

Diagnostic Uplinks

adc_log

```
{
  "name": "adc_log",
  "mask": <integer, bitmask of ports being logged>
}
```

adc_recs

```
{
  "name": "adc_recs",
  "lost": <integer: count of lost values if net cannot keep up with request>,
  "data": [
    {
      "s": <integer: second>,
      "p": <integer: port>,
      "v": <integer: value>
    }
  ]
}
```

core

```
{
 "name": "core",
 "index": <integer>,
 "offset": <integer>,
 "last": <boolean>,
 "data": <binary string base64 encoded>
}
```

Several core messages are sent to the server if the target crashes. The messages contain the RAM contents at the time of the crash. Only supported headed platforms, not the 400.

error

```
{
  "name": "error",
  "message": <string>
}
```

nvram

```
{
  "name": "nvram",
  "sn": <integer, sequene number nvram>,
  "data": <string, base64 encoded memory>
}
```

Production Uplinks

ptest_passed

```
{
  "name": "ptest_passed",
  "uid": <string>,
  "device_id": <integer>
}
```

Production Downlinks

If manufacturing test has never passed device will subscribe to flameboss/0/recv and wait for its device ID to be assigned by the server.

id

Assigns device ID and aes_key (which is used for MQTT password). During production test, device must verify uid is correct before accepting new device_id, aes_key, and pin.

```
{
  "name": "id",
  "uid": <string, base64 encoded>,
  "device_id": <integer>,
  "aes_key": <string, base64 encoded>,
  "pin": <integer, PIN to be displayed on headed devices, for authenticating in station mode>
}
```

sn

Assigns serial number. Device must verify uid and device_id are correct before accepting sn.

```
{
```

```
  "name": "sn",
  "uid": <string, base64 encoded>,
  "device_id": <integer>,
  "sn": <integer>
}
```

Configuration Downlinks

time

```
{
  "name": "time",
  "epoch": <integer: time in Unix epoch scale>
}
```

id

Assigns pin and possibly other identification attributes from server.

```
{
  "name": "id",
  "uid": <optional, string, base64 encoded>,
  "device_id": <optional, integer>,
  "aes_key": <optional, string, base64 encoded>,
  "pin": <integer, PIN to be displayed on headed devices, for authenticating in station mode>
}
```

cook

```
{
  "name": "cook",
  "id": <integer: becomes the cook_id in the temps uplink>,
  "private": <boolean: determines whether topic …data or …open is used for several uplinks>
}
```

local_access

```
{
  "name": "local_access ",
  "value": < "authenticated" or "open">,
}
```

mqtt

```
{
   "name": "mqtt",
   "host": <string: hostname of broker to connect to, optional if ip included, blank by default>,
   "ip": <string: ipv4 address of broker, used if host is blank or dns lookup of host fails,
optional>,
   "tls": <boolean, optional, default is false>,
   "port": <integer, optional>,
   "username": <string, optional>,
   "password": <string, optional>
}
```

set_temp

```
{
   "name": "set_temp",
   "value": <integer>
}
```

Causes controller to send a temps uplink immediately

set_temp_limits

```
{
   "name": "set_temp_limits",
   "min": <integer, optional>,
   "max": <integer, optional>
}
```

pit_alarm

```
{
   "name": "pit_alarm",
   "enabled": <boolean, optional, no change if omitted>,
   "range": <integer, optional, no change if omitted>
}
```

Causes controller to send a pit_alarm uplink immediately

open_pit

```
{
  "name": "open_pit",
  "max_pause": <integer, max open pause time in sec>
}
```

meat_alarm

```
{
  "name": "meat_alarm",
  "sensor": <integer, 1-3>,
  "action": <"off", "on", or "keep_warm">,
  "done_temp": <integer>,
  "warm_temp": <integer>
}
```

alarm_ack

```
{
 "name": "alarm_ack"
}
```

sound

```
{
  "name": "sound",
  "config": <string: "off", "chirps", "alarms">
}
```

wifi

Device will switch to the new configuration immediately so be careful since this message can break communications and require a controller reset to recover.

```
{
  "name": "wifi",
  "mode": <"station" or "ap">,
  "ssid": <string, ignored if mode is ap>,
  "key": <string, ignored if mode is ap>
}
```

temp_scale

This message is not sent and is ignored on headless devices. It sets the scale used to show temps on the target display.

```
{
  "name": "temp_scale",
  "value": <"c" or "f">
}
```

labels

```
{
  "name": "labels",
  "values": <array of 4 strings, max 12 char each, e.g. [ "Pit", "Brisket", "Butt", "Turkey" ] >
}
```

sim_temp

```
{
  "name": "sim_temp",
  "sensor": <integer, 0-3 for temp probes, 4 for device temp, 5 for voltage input in decivolts>,
  "value": <integer for fake temp, null for switch to real sensing>
}
```

read

```
{
  "name": "read",
  "type": <string: "nvram">,
  "addr": <integer>,
  "len": <integer less than 128>
}
```

read_res

```
{
  "name": "read_res",
  "type": <string: "ram", "nvram">,
  "addr": <integer>,
  "data": <string>
}
```

reset

```
{
  "name": "reset",
  "type": <"wifi", "device", or "revert">
```

```
}
```

console_config

```
{
  "name": "console_config",
  "logging": <boolean>
}
```

Console Logging Uplink

Topic: .../console

No json object, just the console text.

Firmware Update

Topic	flameboss/<device_id>/update
Name	update_status
Attributes	target: string, "app", "boot", or "wifi" part: integer last_part: boolean version: string percent: integer