Setting Up a Meshtastic MQTT-to-Internet Gateway

An MQTT gateway connects your Meshtastic mesh to the internet, enabling message delivery to non-LoRa clients, integration with home automation, and connection to the global Meshtastic MQTT network.

What the MQTT Gateway Does

A Meshtastic node in "MQTT gateway" mode:

- Receives LoRa packets on channels that have MQTT uplink enabled
- Forwards them to an MQTT broker (local or cloud)
- Receives messages from the MQTT broker and injects them into the LoRa network (on channels with downlink enabled)
- Bridges your mesh to the global Meshtastic network (if using the public MQTT server)

Note: uplink and downlink are configured **per channel** and both default to **OFF**. MQTT only carries traffic for a channel once the MQTT module is enabled *and* that channel has uplink (and/or downlink) turned on.

Requirements

- A WiFi-capable Meshtastic node (T-Beam, Heltec V3, T-Beam Supreme - all have WiFi)
- WiFi network with internet access at the gateway location
- An MQTT broker: either the public Meshtastic broker (mqtt.meshtastic.org) or a self-hosted one

Configuration Steps

Option A: Using the Meshtastic Public MQTT Server

The public broker `mqtt.meshtastic.org` is **not anonymous** - it requires username `meshdev` / password `large4cats` (these are also the firmware's built-in defaults).

```
# Configure via CLI:
meshtastic --set mqtt.enabled true
meshtastic --set mqtt.address mqtt.meshtastic.org
meshtastic --set mqtt.username meshdev
meshtastic --set mqtt.password large4cats

# Configure which channels to bridge (uplink/downlink are per-channel and default OFF)
meshtastic --ch-index 0 --ch-set uplink_enabled true
meshtastic --ch-index 0 --ch-set downlink_enabled true

# Configure WiFi (if not already done)
meshtastic --set network.wifi_ssid "YourSSID"
meshtastic --set network.wifi_psk "YourPassword"
```

Privacy note: The public MQTT server relays messages globally. Only use it for the default (unencrypted) channel unless you want your encrypted channel traffic relayed globally.

Option B: Self-Hosted Mosquitto MQTT Broker

```
# Install Mosquitto on Raspberry Pi or VPS:
sudo apt install mosquitto mosquitto-clients
sudo systemctl enable --now mosquitto

# Mosquitto 2.0+ defaults allow_anonymous to FALSE and binds localhost-only.
# A bare install rejects LAN clients until you define a listener. Create
# /etc/mosquitto/conf.d/local.conf with:
#   listener 1883 0.0.0.0
#   allow_anonymous true   # trusted LAN only; otherwise use a password_file
# For authentication instead of anonymous access:
```

```
#   mosquitto_passwd -c /etc/mosquitto/passwd youruser
#   password_file /etc/mosquitto/passwd
#   allow_anonymous false
sudo systemctl restart mosquitto

# Configure node to use your local broker:
meshtastic --set mqtt.address 192.168.1.100
meshtastic --set mqtt.enabled true
```

Verifying the Gateway is Working

```
# Subscribe to all Meshtastic MQTT topics and watch for packets
# (the public broker requires credentials):
mosquitto_sub -h mqtt.meshtastic.org -u meshdev -P large4cats -t "msh/US/2/e/#" -v

# Or on your local broker:
mosquitto_sub -h 192.168.1.100 -t "msh/#" -v
```

You should see packets appearing as mesh traffic is received. On the `/e/` (encrypted) topic these are raw binary protobuf-encoded ServiceEnvelope messages, not base64 text.

MQTT Packet Decoding

Meshtastic MQTT packets on the `/e/` topic are protobuf-encoded. The Python `meshtastic` package has **no turnkey MQTT class** - the documented approach is to subscribe with **paho-mqtt** and decode the payload with the bundled protobuf bindings (`meshtastic.protobuf`):

```
pip install meshtastic paho-mqtt

import paho.mqtt.client as mqtt
from meshtastic.protobuf import mqtt_pb2  # ServiceEnvelope lives in mqtt.proto

def on_message(client, userdata, msg):
    # The /e/ payload is RAW BINARY protobuf - feed it straight in, do not base64-decode it
    envelope = mqtt_pb2.ServiceEnvelope()
    envelope.ParseFromString(msg.payload)
    print(envelope)
```

```
client = mqtt.Client()
client.username_pw_set("meshdev", "large4cats")
client.on_message = on_message
client.connect("mqtt.meshtastic.org", 1883)
client.subscribe("msh/US/2/e/#")
client.loop_forever()
```

Note: `from meshtastic.mqtt import MQTT` does not exist - there is no built-in MQTT client class in the library.

Revision #3
Created 2026-05-03 06:33:49 UTC by Mesh America Admin
Updated 2026-06-10 05:35:28 UTC by Mesh America Admin