

Meshtastic MQTT Setup

MQTT lets a Meshtastic node forward all mesh traffic to the internet, making your local mesh visible on the network map, bridging messages to internet clients, and enabling monitoring and logging. This is what feeds online community maps (e.g. via the MapReport packet) that support Meshtastic map reporting.

Warning — read before enabling: By default, Meshtastic publishes packets to the MQTT broker **UNENCRYPTED**, even on channels that use a PSK, unless you set `mqtt.encryption_enabled` to true. Enabling MQTT uplink on the default LongFast channel publishes that traffic to the public broker where **anyone can read it**, because the default `AQ==` key is publicly known. Node positions are exposed regardless of PSK. Only uplink channels whose content you intend to be public, and never assume a PSK alone protects content sent over MQTT. The downlink/inject risk on public channels is covered below.

How MQTT works in Meshtastic

When MQTT is enabled on a node:

1. Every mesh packet received by the node is forwarded to an MQTT broker over WiFi or TCP
2. The MQTT broker stores and redistributes the messages to other subscribers
3. The public Meshtastic MQTT broker (mqtt.meshtastic.org) shares filtered traffic — zero-hop only, a limited set of portnums, and reduced location precision — which feeds community maps. It is not an unrestricted public feed, but anything it does publish is visible to anyone subscribed.
4. Optionally, messages from the internet (MQTT) can be injected back into the local radio mesh

Hardware requirement: Direct MQTT requires a WiFi-capable device (ESP32-based: Heltec V3/V4, T-Beam, etc.). nRF52840 devices (T-Echo, T-Deck, RAK4631) have no onboard WiFi and cannot connect to MQTT over their own network, but they can use the **MQTT Client Proxy** to relay MQTT through a connected phone running the app (note: JSON output is not supported on nRF52).

Connecting to the public Meshtastic MQTT broker

Via the [Meshtastic app](#)

1. Go to **Settings** → **Module Config** → **MQTT**
2. Enable MQTT: toggle ON
3. MQTT Server Address: `mqtt.meshtastic.org`
4. Username: `meshdev`
5. Password: `large4cats`
6. TLS Enabled: toggle ON (recommended)
7. Map Reporting Enabled: toggle ON to publish a map report so your node appears on community maps that consume MapReport packets
8. Save

Note on the public channel: the default LongFast channel uses the publicly known `AQ==` key and has no real authentication. If you leave uplink (and especially downlink) enabled on it, anyone on the public broker can see your traffic, and — if downlink is on — inject messages into your local mesh. See the downlink and security notes below before enabling.

Via CLI

```
meshtastic --set mqtt.enabled true
meshtastic --set mqtt.address mqtt.meshtastic.org
meshtastic --set mqtt.username meshdev
meshtastic --set mqtt.password large4cats
meshtastic --set mqtt.tls_enabled true
meshtastic --set mqtt.map_reporting_enabled true
```

Map reporting publishes a MapReport packet. Current firmware also exposes related sub-settings (for example position precision and the publish interval) under the `mqtt.*` namespace; check the field names in your installed firmware version, as they have changed over time.

Channel settings for MQTT

MQTT is enabled per channel. By default, the primary channel (channel 0) is configured to uplink to MQTT. Verify that your channel has **Uplink Enabled** set to ON:

```
meshtastic --ch-index 0 --ch-set uplink_enabled true
```

Important: A PSK on your channel does *not* mean MQTT uploads are encrypted. By default the gateway decrypts packets and uplinks them **unencrypted** to the broker, even on channels with a

custom PSK. To keep message content encrypted on the broker you MUST set `mqtt.encryption_enabled = true`. Even with encryption enabled, packet metadata and node positions may still be exposed. With encryption enabled, community maps see only reduced-precision node positions and cannot read message content; the public server further filters location precision.

Downlink: receiving messages from the internet

Downlink allows messages published to MQTT to be injected into the local radio mesh - enabling internet-connected users to send messages that appear on mesh nodes in your area:

```
meshtastic --ch-index 0 --ch-set downlink_enabled true
```

Security note: Only enable downlink on channels with PSK authentication if you want to control who can inject messages into your local mesh. The public LongFast channel has no authentication - anyone on the public MQTT broker can inject messages into your mesh if downlink is enabled on the default channel.

Running a private MQTT broker

For a community or organizational network, run your own Mosquitto broker instead of using the public one:

```
# Install Mosquitto
sudo apt install mosquitto mosquitto-clients

# Basic config: /etc/mosquitto/mosquitto.conf
listener 1883 localhost # Local only (use nginx/TLS for external)
listener 8883 # TLS port for internet clients
cafile /path/to/ca.crt
certfile /path/to/server.crt
keyfile /path/to/server.key
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Point your Meshtastic nodes to your broker's address instead of `mqtt.meshtastic.org`.

MQTT topic structure

Meshtastic publishes to topics of the form:

```
msh/{region}/2/e/{channel_name}/{node_id}    # protobuf (encrypted/binary) topic
msh/{region}/2/json/{channel_name}/{node_id}  # JSON topic (when JSON output is enabled)
```

Examples:

```
msh/US/2/json/LongFast/!abcd1234
msh/US/2/e/LongFast/!abcd1234
```

Subscribe to `msh/US/#` to receive all US region traffic. The 4th segment is the **channel name**; the packet type is *not* a topic segment. On `msh/{region}/2/e/...` the payload is a protobuf-encoded `ServiceEnvelope`; a JSON-encoded payload appears only on `msh/{region}/2/json/...` when JSON output is enabled, and the packet kind (nodeinfo, position, text, etc.) is carried in the JSON `type` field inside the payload.

Revision #5

Created 2026-05-03 03:54:16 UTC by Mesh America Admin

Updated 2026-06-09 12:12:52 UTC by Mesh America Admin