

Internet Bridging and MQTT

A MeshCore room server runs as firmware on a single LoRa node (typically nRF52840 or ESP32 hardware). It is a store-and-forward node on the RF mesh and does **not** have a native internet, TCP, or MQTT bridge. There is no MeshCore feature that lets phone users without LoRa hardware join a room over the internet, and there is no MeshCore MQTT integration. If you want MeshCore-related traffic on the internet or in a monitoring stack, you bridge to a separate Linux host, or use Meshtastic's documented MQTT path on a co-located gateway. This page explains what MeshCore actually supports and how to do internet/monitoring integration correctly.

What "bridge" means in MeshCore

MeshCore's only built-in "bridge" is a **radio-layer link**, compiled into the firmware, used to join two co-located boards so they extend RF coverage. It is configured through the device CLI (over serial/BLE), for example:

- `set bridge.enabled <on|off>` - enable the compiled-in bridge (only present when bridge support is built into the firmware)
- The transport is a radio/serial link (RS-232 serial or ESP-NOW between two boards), not a TCP/internet connection

This bridge does **not** relay between LoRa radio and the internet, and it does not let off-mesh clients participate over TCP. MeshCore companion clients (phone, computer) connect to a node **locally** over BLE, USB, or serial - not over the internet to a remote room server. A person in another city cannot join your local mesh purely through a MeshCore room server; reaching the internet requires a separate gateway host (see below).

Getting mesh data to the internet (the supported paths)

MeshCore room-server firmware does not publish to MQTT. If you need mesh traffic on an MQTT broker, a dashboard, or the wider internet, use one of these real options:

- **Run a separate Linux gateway host.** Connect a MeshCore node by USB/serial to a small Linux machine (Raspberry Pi, mini-PC) and run your own software on that host to read the node's serial output and forward it wherever you like. The microcontroller

running the room server cannot host this software itself - it has on the order of 256 KB of RAM.

- **Use Meshtastic's MQTT path for an internet-connected mesh.** Meshtastic firmware *does* document native MQTT (uplink/downlink to a broker such as `mqtt.meshtastic.org` or a self-hosted Mosquitto). If internet integration is a hard requirement, a Meshtastic gateway node is the documented way to do it. See the Meshtastic MQTT pages in this book.

Do not expect MeshCore CLI keys such as a `mqtt:` config block, a `topic_prefix`, or `meshcore/...` topics - they do not exist in MeshCore firmware.

Monitoring stack (InfluxDB / Telegraf / Grafana) runs on a separate host

A time-series monitoring stack is a legitimate way to visualize mesh data, but every component runs on a **separate Linux host**, never on the nRF52840/ESP32 microcontroller of the room server:

1. Install InfluxDB (time-series database) on a Linux host.
2. Feed it from a real data source - for a Meshtastic mesh, Telegraf's MQTT input plugin can consume the Meshtastic MQTT topics and write to InfluxDB. (There is no MeshCore MQTT feed; for MeshCore you would write your own collector that reads the node's serial output on the gateway host.)
3. Install Grafana on the host and build dashboards (active nodes over time, node battery levels, message rates, coverage from GPS data).

This entire stack lives on the Linux host, not on the MeshCore room-server hardware.

Exposing a Linux gateway host to the internet (TLS)

If - and only if - you are running an actual web/dashboard service on a separate Linux gateway host (for example a Grafana instance or a custom collector's web UI), you can put it behind a reverse proxy with TLS. This nginx example terminates TLS in front of a local service; replace the upstream port with whatever your service actually listens on (e.g. Grafana's default `3000`). A MeshCore room server itself exposes no HTTP service, so there is nothing on the node to proxy to.

```
# Example nginx configuration on a Linux gateway host
# (fronting a real local service, e.g. Grafana on :3000)
server {
    listen 443 ssl;
    server_name mesh.yournetwork.com;

    ssl_certificate /etc/letsencrypt/live/mesh.yournetwork.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/mesh.yournetwork.com/privkey.pem;

    location / {
        proxy_pass http://127.0.0.1:3000; # your service's actual port
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Get free TLS certificates from Let's Encrypt using certbot. TLS protects the connection to that gateway host's service; it has nothing to do with the RF mesh, which is encrypted independently at the LoRa layer.

Security considerations for an internet-facing gateway host

Any Linux host you expose to the internet (broker, dashboard, collector) needs proper hardening:

- **TLS/SSL:** Terminate TLS on the gateway host's web/broker service so credentials and data are not sent in plaintext.
- **Authentication:** Require authentication on the service. An open dashboard or broker reachable from the internet attracts abuse.
- **Firewall rules:** Restrict access to only the ports the service needs; block everything else and front the service with a reverse proxy for TLS termination.
- **Rate limiting:** Apply per-client limits so a single client cannot flood the service.

Alerting on node failure (on the gateway host)

If you are collecting *Meshtastic* data via MQTT on your Linux host, you can alert when a node stops checking in using a small Python script (this consumes the Meshtastic MQTT topics, not any MeshCore topic):

```
import paho.mqtt.client as mqtt
import time

nodes = {}
ALERT_TIMEOUT_SECONDS = 3600 # Alert if not heard in 1 hour

def on_message(client, userdata, msg):
    node_id = msg.topic.split('/')[1]
    nodes[node_id] = time.time()

def check_timeouts():
    now = time.time()
    for node_id, last_seen in nodes.items():
        if now - last_seen > ALERT_TIMEOUT_SECONDS:
            print(f"ALERT: {node_id} has not been heard in over 1 hour!")

# Subscribe to the Meshtastic JSON topic tree on your broker
# (requires mqtt.json_enabled on the gateway; not available on nRF52)
client = mqtt.Client()
client.on_message = on_message
client.connect("localhost", 1883)
client.subscribe("msh/+/2/json/+/+")
client.loop_start()

while True:
    check_timeouts()
    time.sleep(300)
```

For a MeshCore-only deployment there is no MQTT feed to subscribe to; you would instead read the node's serial output (and its `stats-core` / `stats-radio` / `stats-packets` CLI output) on the gateway host and build alerting from that.

Revision #4

Created 2026-05-03 03:52:11 UTC by Mesh America Admin

Updated 2026-06-10 05:27:12 UTC by Mesh America Admin