MQTT Topic Structure and Packet Format

Understanding Meshtastic's MQTT topic structure and packet encoding lets you build integrations, parse data, and troubleshoot gateway connectivity issues.

Topic Structure

Meshtastic publishes to MQTT topics using this hierarchy:

```
msh/{region}/2/{e|json}/{channel_name}/{user_id}

Examples:
msh/US/2/e/LongFast/!ab12cd34    # Encrypted packet, published by gateway !ab12cd34
msh/US/2/json/LongFast/!ab12cd34 # JSON-decoded packet (only if JSON enabled on the gateway)

Where:
 msh/      - Meshtastic prefix
 US/       - Region code (US, EU, etc.)
 2/        - Protocol version
 e/        - Encrypted protobuf (default)
 json/     - JSON decoded (optional, only if JSON output is enabled on the gateway)
 LongFast/ - Channel name (this segment is always present)
 !ab12cd34 - User/gateway node ID (the node that published this packet to MQTT, not
necessarily the originator)
```

The 4th segment is `e` (encrypted protobuf) or `json`; the 5th segment is the channel *name*; the last segment is the publishing (gateway) node's hex ID. The packet's *type* (text, telemetry, position, etc.) is a field inside the payload, not a topic segment - there are no `.../json/text/` or `.../json/telemetry/` topics.

Subscribing to All Traffic

```
# Subscribe to all Meshtastic packets from all US nodes:
mosquitto_sub -h mqtt.meshtastic.org -u meshdev -P large4cats -t "msh/US/2/e/#" -v

# Subscribe to a specific channel only:
mosquitto_sub -h mqtt.meshtastic.org -t "msh/US/2/e/CommunityMesh/#" -v

# Subscribe to JSON-decoded packets (if gateway has JSON enabled):
mosquitto_sub -h localhost -t "msh/US/2/json/#" -v
```

Packet Format

Each MQTT message payload is a protobuf-encoded `ServiceEnvelope` (defined in `meshtastic.protobuf.mqtt_pb2`) - the wrapper a gateway publishes - containing:

- **packet** - The MeshPacket (encrypted payload + routing info)
- **channel_id** - Channel name (e.g., "LongFast")
- **gateway_id** - Node ID of the gateway that published to MQTT

Enabling JSON Output on a Gateway Node

Security warning: JSON output publishes fully *decoded* (plaintext) packet content to MQTT, bypassing channel encryption entirely. The gateway can only emit JSON for packets it can decrypt (channels whose key it holds). Never enable `json_enabled` on a gateway that uplinks to a public or shared broker if any channel content is sensitive - anyone subscribed to that broker will read the cleartext.

```
# Enable JSON output (in addition to protobuf) on the GATEWAY node that publishes to MQTT.
# Only packets the gateway can decode (known channel key) are emitted as JSON:
meshtastic --set mqtt.json_enabled true

# JSON packets are published to: msh/REGION/2/json/CHANNELNAME/USERID
#   CHANNELNAME = the channel's name (e.g. LongFast)
#   USERID      = the gateway node's hex ID (e.g. !7efeee00)
# JSON payload example:
{
```

```
 "from": 2881537332,
 "to": 4294967295,
 "channel": 0,
 "type": "text",
 "id": 123456789,
 "rx_time": 1712000000,
 "hop_limit": 3,
 "payload": {
 "text": "Hello mesh!"
 }
}
```

Decoding Protobuf Packets in Python

```
import paho.mqtt.client as mqtt
from meshtastic.protobuf.mqtt_pb2 import ServiceEnvelope
from meshtastic.protobuf.portnums_pb2 import PortNum
import base64

def on_message(client, userdata, msg):
 try:
 se = ServiceEnvelope()
 se.ParseFromString(msg.payload)
 packet = se.packet
 print(f"From: !{packet.from_:08x}")
 print(f"To: !{packet.to:08x}")
 print(f"Channel: {se.channel_id}")
 # Decrypt and decode based on portnum for full payload
 except Exception as e:
 print(f"Parse error: {e}")

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()
```

Revision #3
Created 2026-05-03 10:48:06 UTC by Mesh America Admin
Updated 2026-06-09 12:08:51 UTC by Mesh America Admin