

ESP-Mesh-Lite App User Guide

ESP-Mesh-Lite is a provisioning and management tool for ESP Mesh devices. It runs in a web browser or as an Android app.

Basic features: device provisioning, MQTT connection, viewing devices and topology, and single-device, group-wide, or grouped control of lights.

Advanced features: fetch network topology from any node, migrate the entire network to a new router with the root node, and configure chained sub-networks.

This guide is for embedded / IoT enthusiasts and developers who want to get started quickly and become familiar with the app.

Launch the App: Enter Your Account First

When you open the app, you first see the account screen (similar to a login page). Enter your account and tap **Enter** to reach the main UI.

Item	Description
Default account	<code>admin</code> (you can tap Enter directly)
How to enter	Do not include a leading <code>/</code> . Use <code>admin</code> , not <code>/admin</code>
Memory	The last used account is remembered automatically

Why enter an account?

The account maps to the MQTT topic prefix and defines a separate communication channel:

- Account `admin` → topics such as `/admin/mesh/topology`, `/admin/device/.../control`
- Account `alice` → topics such as `/alice/mesh/topology`, `/alice/device/.../control`

Different accounts use different channels. Subscriptions, topology reports, and device control all run under their own prefix and do not interfere with each other. When multiple users share the same broker, each can use a different account to isolate devices and messages.

During provisioning, the current account is also sent to the device (as the device-side username). Therefore:

- The account used to log in to the app
- The account written to the device during provisioning
- The account used for later MQTT subscribe/control

These three must match. Otherwise you may connect to the broker but see no devices or get no response to control commands.

What You See in the Main UI

The top bar shows the app title and language switcher. Below that are the feature tabs:

Tab	Purpose
Provision	Send WiFi / Mesh parameters to devices over BLE
MQTT	Connect to a broker and communicate with provisioned devices
Devices	View device info and Mesh topology; single-device and group-wide light control
Groups	Manage and control devices in batches by group
Update	Update the app itself
Logs	View runtime and debug logs

Typical workflow: **enter account** → **provision** → **connect MQTT** → **manage and control in Devices / Groups**.

1. WiFi Provisioning

Send router credentials and Mesh network parameters to an ESP device in provisioning mode. Provisioning currently uses **BLE**. Make sure the firmware is in BLE provisioning mode (device name is usually similar to `PROV_XXXX`).

ProvisionMQTTDevicesGroupsUpdateLogs

WiFi Provisioning

Ensure the ESP Mesh device is in provisioning mode

Security2 username

wifiprov

See esp_prov.py: --sec2_username

Security2 password *

.....

See esp_prov.py: --sec2_pwd (default: abcd1234)

SoftAP SSID

Mesh

SoftAP SSID for inter-device communication, max 32 bytes

SoftAP password (optional, open if empty)

8-63 bytes (< 8 sets OPEN mode)

SoftAP password, 8-63 bytes (< 8 sets OPEN mode)

SoftAP Beacon Interval (TU)

500

Beacon interval in TU (1 TU = 1.024 ms), 100-60000, multiple of 100

Mesh network ID

1

Mesh network ID, range 1-255

Max mesh nodes

100

Maximum nodes allowed in the mesh, range 1-200

Broker URI (optional)

e.g. mqtt://192.168.1.1:1883

MQTT broker URI sent to the device with mesh_id.

Network segment (optional)

Subnet mask

11.0.0.0

Prefix must be a network address; mask must be a valid contiguous mask.

Subnet mask

255.255.255.0

Extra data (optional)

("key": "value") - sent as user_data in payload (JSON)

Custom JSON wrapped as user_data in the provisioning payload.

WiFi 名称 (SSID) *

Enter WiFi SSID

💡 会自动记住上次使用的 WiFi 名称

WiFi password

Enter WiFi password

Start provisioning

Steps

1. Open the app, enter your account, and enter the main UI (the channel name agreed with your devices; default is `admin`).
2. Power on the device and enter provisioning mode.
3. Open the **Provision** tab and fill in the fields below as needed (fields marked with ***** are required).

4. Tap **Start provisioning**, grant Bluetooth permission when prompted, and select the target device.
5. Wait for provisioning to finish. After the primary device is provisioned successfully, it zero-config provisions other devices. All devices connect to the router you configured and form a complete Mesh network.

LED states with default firmware (mqtt_light_control)

If you use the default **mqtt_light_control** firmware, you can tell the current state from the LED color:

LED state	Meaning
Blue blinking	In provisioning mode
Orange blinking	Provisioning data received
Green solid	WiFi connected

Note: Provisioning mode lasts about **5 minutes** only. If the device has been powered on for more than 5 minutes without being provisioned, it exits provisioning mode automatically. To provision again, power-cycle the device and wait until the blue blinking LED returns.

Re-enter provisioning mode on an already provisioned device

Use either method:

1. In the app, open a single-device, grouped, or group-wide control page and tap **Factory reset**.
2. Power-cycle the light **more than 3 times in a row** and wait until the LED returns to **blue blinking**.

Field reference

Security2 credentials

Used to establish a secure session with the device (same as Security2 in `esp_prov.py`).

Field	Required	Description
Security2 username	No	Default <code>wifipro</code>
Security2 password	Yes	Default <code>abcd1234</code> ; must match the device firmware configuration

If you changed the credentials in firmware, enter the same values here. Otherwise the session cannot be established.

SoftAP / Mesh network parameters

These parameters define how the Mesh network is formed internally. Devices in the same network should usually use the same values.

Field	Required	Description
SoftAP SSID	No	SoftAP name used for inter-device Mesh communication, max 32 bytes; common default is <code>Mesh</code>
SoftAP password	No	8–63 bytes; fewer than 8 bytes or empty means OPEN (no encryption)
SoftAP Beacon Interval (TU)	No	Beacon interval in TU (1 TU = 1.024 ms); range 100–60000, must be a multiple of 100; default <code>500</code>
Mesh network ID	No	Network identifier, range 1–255; use different IDs for different Mesh networks
Max mesh nodes	No	Maximum number of nodes allowed, range 1–200; default <code>100</code>

Cloud and custom settings (optional)

Field	Description
Broker URI	MQTT broker sent to the device, e.g. <code>mqtt://192.168.1.1:1883</code> (see important note below)
Network segment	Custom Mesh IP segment; prefix must be a network address (host bits all 0), mask must be a valid contiguous mask
Extra data	JSON sent as the <code>user_data</code> field in the payload, e.g. <code>{"key": "value"}</code>

Important (Broker URI): This field is optional, but you should fill it in correctly. **If the MQTT broker URI is missing or wrong, the device cannot connect to MQTT, and the mobile app / web UI cannot control Mesh devices over MQTT.** In that case you can only connect to a Mesh device SoftAP and control devices locally through SoftAP.

Target WiFi (core required fields)

Field	Required	Description
WiFi name (SSID)	Yes	Router name; the app remembers the last used SSID
WiFi password	No	Router password; leave empty for open networks

Provisioning tips

- The browser must support Web Bluetooth (Chrome / Edge recommended) and run under **HTTPS** or **localhost**.
- The Android APK also supports BLE provisioning with a similar experience to the web app.
- You only need to provision the primary device once; other nodes join through zero-config provisioning and connect to the same router to form the Mesh.
- If SoftAP password, Mesh ID, capacity, or similar values do not match the existing Mesh, new devices may fail to join correctly.

- If Security2 password is wrong, session setup usually fails. Check the firmware defaults first.
- **Broker URI must be correct.** Otherwise the device cannot connect to MQTT and the app / web UI must fall back to SoftAP local control.

2. MQTT Client

After provisioning, use MQTT to communicate with Mesh devices: subscribe to topology and node status, and send control commands. In a browser, use **WebSocket** (`ws://` / `wss://`).

Before connecting

The screenshot shows the 'MQTT Client' configuration page in the ESP-Mesh-Lite web interface. The page has a blue header with 'ESP-Mesh-Lite' and a language dropdown set to 'English'. Below the header is a navigation bar with tabs: 'Provision', 'MQTT' (selected), 'Devices', 'Groups', 'Update', and 'Logs'. The main content area is titled 'MQTT Client' and includes the instruction 'Connect to an MQTT broker for device communication'. The 'Broker address' section has a dropdown menu set to 'ws://', a text input field containing '192.168.3.71', and a 'Port' input field containing '9001'. A note states: 'Browsers support WebSocket only (ws:// or wss://); common ports: 9001 or 8083'. The 'Client ID' section has a text input field containing 'esp_mesh_web_1782184210787' and a note: 'Unique identifier; auto-generated if empty'. The 'Username (optional)' section has a text input field containing 'MQTT username'. The 'Password (optional)' section has a text input field containing 'MQTT password'. At the bottom, there is a blue 'Connect' button and a status indicator showing 'Disconnected'.

Steps

1. Open the **MQTT** tab.
2. Choose the protocol (`ws://` or `wss://`) and enter the broker address and port.
3. Optionally enter client ID, username, and password.
4. Tap **Connect**.
5. Once the status shows **Connected**, you can view and control devices on the Devices page and elsewhere.

Field reference

Field	Required	Description
Broker address	Yes	WebSocket address; common ports are 9001 and 8083
Port	Yes	Must match the broker WebSocket listening port
Client ID	No	Unique identifier; auto-generated if empty (e.g. esp_mesh_web_...)
Username / password	No	Leave empty if the broker has no authentication

After connecting

ESP-Mesh-Lite

LanguageEnglish

Provision

MQTT

Devices

Groups

Update

Logs

MQTT Client

Connect to an MQTT broker for device communication

Broker address *

ws://

192.168.3.71

Port

9001

Browsers support WebSocket only (ws:// or wss://); common ports: 9001 or 8083

Client ID

esp_mesh_web_1782184210787

Unique identifier; auto-generated if empty

Username (optional)

MQTT username

Password (optional)

MQTT password

Disconnect

Connected

Item	Description
Connect	Establish a WebSocket MQTT session
Disconnect	Close the connection to the broker
Connection status	Disconnected / connecting / connected (✔)

After a successful connection, the app automatically subscribes to relevant topics (such as `/mesh/topology`) to refresh topology and device status.

MQTT tips

- The web app cannot use `mqtt://` directly. You must use the broker WebSocket port.
- If the device-side broker is configured as `mqtt://...:1883`, the app still needs the same broker **WS port** (such as 9001).
- For local debugging, the address is usually a LAN IP (such as `192.168.x.x`). Make sure the phone / PC and broker are on the same network.
- Subscribe and control topics include the login account prefix. If you see no devices, verify that the account entered at app launch matches the one used during provisioning.

Run your own MQTT broker on the LAN

To set up an MQTT broker on your local network, you can use the following commands on a Linux host (Mosquitto example):

```
sudo apt-get install -y sshpass vlan mosquitto mosquitto-clients tcpdump nfs-common hostapd

# mosquitto
sudo tee /etc/mosquitto/conf.d/local.conf > /dev/null << EOF
# Listen port (default is 1883)
listener 1883
protocol mqtt

listener 9001
protocol websockets

# Allow anonymous connections (default is true; set to false and configure auth in
production)
allow_anonymous true
EOF

sudo systemctl restart mosquitto
```

After the configuration takes effect:

- **Provision page Broker URI** (sent to devices): `mqtt://<host-lan-ip>:1883`
- **App MQTT tab** (browser / phone connects to broker): protocol `ws://`, same host IP, port `9001`

Make sure Mesh devices, the phone / PC running the app, and the broker host are on the same LAN, and that the firewall allows `1883` and `9001`.

3. After Provisioning and MQTT

Devices

- View MAC, level, parent, IP, RSSI, and more.
- Switch between list view and **topology tree** to see Mesh hierarchy clearly.
- Online status is usually maintained by heartbeat / node reports; devices with no updates for a long time appear offline.
- Control a single device with on/off, HSV light control, rename, and more (depends on device firmware).
- In **topology view**, tap a root node to perform group-wide control on that Mesh network.

Groups

- Put multiple devices into the same group for group on/off or group light control.
- Useful for living-room light strips, multiple nodes in one room, and other "control together" scenarios.

Update and logs

- **Update:** Check for and install app updates.
- **Logs:** Useful for troubleshooting provisioning failures, MQTT disconnects, and control issues. Check here first when something goes wrong.

4. Fetch Topology and Control Devices via SoftAP

When no MQTT broker is available, or devices cannot connect to MQTT temporarily, connect to the **SoftAP of any Mesh device** to fetch the full network topology locally and control devices with single-device or group-wide control. This is one of the advanced features mentioned at the top: **fetch network topology from any node**.

Steps

1. In the phone Wi-Fi settings, connect to the SoftAP of any Mesh device (SSID / password match the SoftAP values used during provisioning; common default SSID is **Mesh**).
2. Open the app and go to **Devices**.
3. Enable **Fetch topology via SoftAP**.
4. After it succeeds, the full Mesh topology is shown.

Control options

After topology is loaded:

- **Single-device control:** Tap any device to control on/off, lighting, and more.
- **Group-wide control:** In **topology view**, tap a root node to control the whole network.

Tips

- SoftAP topology fetch currently works mainly on the **Android APK**. Connect to the device SoftAP in system settings first.
- This is a local control path that does not depend on MQTT. It is useful when the broker is not deployed, the address is wrong, or external network access is unavailable.
- If MQTT becomes available again, disable **Fetch topology via SoftAP** to return to the MQTT device list.

Recommended onboarding path

1. Open the app and enter your account (default `admin`; if multiple users share one broker, use your own account to isolate channels).
2. **Flash** firmware with Mesh-Lite provisioning support and confirm the device enters BLE provisioning mode.
3. On the **Provision** page, fill in Security2 and WiFi and complete a successful provisioning run (the account is sent to the device with the configuration).
4. On the **MQTT** page, connect to the broker and confirm the status shows **Connected**.
5. Open **Devices**, confirm nodes and topology appear, and try group-wide control by tapping a root node in topology view.
6. If MQTT is unavailable, connect to any node SoftAP instead and enable **Fetch topology via SoftAP** for single-device / group-wide control.
7. When needed, create groups under **Groups** for batch control.

Follow this path once and you will have covered the core workflow of the app.