MeshCore Official FAQ

- FAQ: 1. Introduction
- FAQ: 2. Initial Setup
- FAQ: 3. Server Administration
- FAQ: 4. T-Deck Related
- FAQ: 5. General
- FAQ: 6. Troubleshooting
- FAQ: 7. Other Questions:

FAQ: 1. Introduction

1.1. Q: What is MeshCore?

A: MeshCore is a multi platform system for enabling secure text based communications utilising LoRa radio hardware. The project lists its intended use cases as Off-Grid Communication, Emergency Response & Disaster Recovery, Outdoor Activities, Tactical Security including law enforcement and private security, and IoT sensor networks. (source) *Editorial note: these are vendor-listed use cases. For emergency, disaster, or tactical use, treat MeshCore as a supplemental best-effort text channel only - it offers no guaranteed delivery or quality-of-service and its channel encryption has known limitations, so it should not be relied on as a primary channel for life-safety traffic.*

MeshCore is free and open source:

- MeshCore is the routing and firmware etc, available on GitHub under MIT license
- There are clients made by the community, such as the web clients, these are free to use, and some are open source too
- The cross platform mobile app developed by Liam Cottle for Android/iOS/PC etc is free to download and use
- The T-Deck firmware is developed by Scott at Ripple Radios, the creator of MeshCore, is also free to flash on your devices and use

Some more advanced, but optional features are available on T-Deck if you register your device for a key to unlock. On the MeshCore smartphone clients for Android and iOS/iPadOS, you can unlock the wait timer for repeater and room server remote management over RF feature.

These features are completely optional and aren't needed for the core messaging experience. They're like super bonus features and to help the developers continue to work on these amazing features, they may charge a small fee for an unlock code to utilise the advanced features.

Anyone is able to build anything they like on top of MeshCore without paying anything.

1.2. Q: What do you need to start using MeshCore?

A: Everything you need for MeshCore is available at:

- Main Website: https://meshcore.io
- Firmware Flasher: https://flasher.meshcore.io
- MeshCore Firmware on GitHub: https://github.com/meshcore-dev/MeshCore
- MeshCore Companion Web App: https://app.meshcore.nz

- MeshCore Map: https://map.meshcore.io
- Liam Cottle's MeshCore Technical Presentation

You need LoRa hardware devices to run MeshCore firmware as clients or server (repeater and room server).

1.2.1. Hardware

MeshCore is available on a variety of 433MHz, 868MHz and 915MHz LoRa devices. For example, Lilygo T-Deck, T-Pager, RAK Wireless WisBlock RAK4631 devices (e.g. 19003, 19007, 19026), Heltec V3, Xiao S3 WIO, Xiao C3, Heltec T114, Station G2, Nano G2 Ultra, Seeed Studio T1000-E. More devices are being added regularly.

For an up-to-date list of supported devices, please go to https://flasher.meshcore.io

To use MeshCore without using a phone as the client interface, you can run MeshCore on a LiLygo's T-Deck, T-Deck Plus, T-Pager, T-Watch, or T-Display Pro. MeshCore Ultra firmware running on these devices are a complete off-grid secure communication solution.

1.2.2. Firmware

MeshCore offers four firmware roles: BLE Companion, USB Serial Companion, Repeater, and Room Server. Each is described below.

1.2.3. Companion Radio Firmware

Companion radios are for connecting to the Android app or web app as a messenger client. There are two different companion radio firmware versions:

1. **BLE Companion**

BLE Companion firmware runs on a supported LoRa device and connects to a smart device running the Android or iOS MeshCore client over BLE

1. **USB Serial Companion**

USB Serial Companion firmware runs on a supported LoRa device and connects to a smart device or a computer over USB Serial running the MeshCore web client

1.2.4. Repeater

Repeaters are used to extend the range of a MeshCore network. Repeater firmware runs on the same devices that run client firmware. A repeater's job is to forward MeshCore packets toward their destination. Unlike a simple flood-everything mesh, MeshCore repeaters follow embedded paths for direct messages once a path is known - but they do still forward flood-routed traffic such as adverts and channel (group) messages that have no specific destination.

A repeater can be remotely administered using a T-Deck running the MeshCore firmware with remote administration features unlocked, or from a BLE Companion client connected to a smartphone running the MeshCore app.

1.2.5. Room Server

A room server is a simple BBS server for sharing posts. T-Deck devices running MeshCore firmware or a BLE Companion client connected to a smartphone running the MeshCore app can connect to a room server.

Room servers store message history on them and push the stored messages to users. Room servers allow roaming users to come back later and retrieve message history. With channels, messages are either received when it's sent, or not received and missed if the channel user is out of range. Room servers are different and more like email servers where you can come back later and get your emails from your mail server.

A room server can be remotely administered using a T-Deck running the MeshCore firmware with remote administration features unlocked, or from a BLE Companion client connected to a smartphone running the MeshCore app.

When a client logs into a room server, the server pushes up to the 32 most recent messages the client has not yet seen.

Although room server can also repeat with the command line command `set repeat on`, it is not recommended nor encouraged. A room server with repeat set to `on` lacks the full set of repeater and remote administration features that are only available in the repeater firmware.

The recommendation is to run repeater and room server on separate devices for the best experience.

FAQ: 2. Initial Setup

2.1. Q: How many devices do I need to start using MeshCore?

A: If you have one supported device, flash the BLE Companion firmware and use your device as a client. You can connect to the device using the Android or iOS client via Bluetooth. You can start communicating with other MeshCore users near you.

If you have two supported devices, and there are not many MeshCore users near you, flash both to BLE Companion firmware so you can use your devices to communicate with your near-by friends and family.

If you have two supported devices, and there are other MeshCore users nearby, you can flash one of your devices with BLE Companion firmware and flash another supported device to repeater firmware. Place the repeater high above ground to extend your MeshCore network's reach.

After you flashed the latest firmware onto your repeater device, keep the device connected to your computer via USB serial, use the console feature on the web flasher and set the frequency for your region or country, so your client can remote administer the repeater or room server over RF:

```
set freq {frequency}
```

The repeater and room server CLI reference is here: https://github.com/meshcore-dev/MeshCore/wiki/Repeater-&-Room-Server-CLI-Reference

If you have more supported devices, you can use your additional devices with the room server firmware.

2.2. Q: Does MeshCore cost any money?

A: All radio firmware versions (e.g. for Heltec V3, RAK, T-1000E, etc) are free and open source developed by Scott at Ripple Radios.

The native Android and iOS client uses the freemium model and is developed by Liam Cottle, developer of meshtastic map at meshtastic.liamcottle.net on GitHub and reticulum-meshchat on github.

The T-Deck firmware is free to download and most features are available without cost. To support the firmware developer, you can pay for a registration key to unlock your T-Deck for deeper map zoom and remote server administration over RF using the T-Deck. You do not need to pay for the registration to use your T-Deck for direct messaging and connecting to repeaters and room servers.

2.3. Q: What frequencies are supported by MeshCore?

A: It supports the 868MHz range in the UK/EU and the 915MHz range in New Zealand, Australia, and the USA. Countries and regions in these two frequency ranges are also supported. Exact legal frequencies, channel plans, and duty-cycle limits vary by country (e.g., EU 863-870 MHz has duty-cycle restrictions; US uses 902-928 MHz). Always select the preset for your specific country and confirm it against your national regulator before transmitting.

Use the smartphone client, or the repeater setup feature on the web flasher, to set your radio's RF settings by choosing the preset for your region.

Recently, as of October 2025, many regions have moved to the "narrow" setting, aka using BW62.5 and a lower SF number (instead of the original SF11). For example, USA/Canada (Recommended) preset is 910.525MHz, SF7, BW62.5, CR5. These exact values change over time - confirm the current recommended preset with your regional community or the live MeshCore preset list before relying on a specific frequency.

After extensive testing, many regions have switched or about to switch over to BW62.5 and SF7, 8, or 9. Narrower bandwidth setting and lower SF setting allow MeshCore's radio signals to fit between interference in the ISM band, provide for a lower noise floor, better SNR, and faster transmissions.

If you have consensus from your community in your region to update your region's preset recommendation, please post your update request on the #meshcore-app channel on the MeshCore Discord server to let Liam Cottle know.

2.4. Q: What is an "advert" in MeshCore?

A:

Advert means to advertise yourself on the network. In Reticulum terms it would be to announce. In Meshtastic terms it would be the node sending its node info.

MeshCore allows you to manually broadcast your name, position and public encryption key, which is also signed to prevent spoofing. When you click the advert button, it broadcasts that data over LoRa. MeshCore calls that an Advert. There's two ways to advert, "zero hop" and "flood".

- Zero hop means your advert is broadcasted out to anyone that can hear it, and that's it.
- Flooded means it's broadcasted out and then repeated by all the repeaters that hear it.

MeshCore clients only advertise themselves when the user initiates it. A repeater sends a flood advert once every 12 hours by default. This interval can be configured using the following command:

```
set flood.advert.interval {hours}
```

The separate `set advert.interval {minutes}` command controls the local zero-hop advert timer.

2.5. Q: Is there a hop limit?

A: Internally the firmware has maximum limit of 64 hops. In real world settings it will be difficult to get close to the limit due to the environments and timing as packets travel further and further. Practical reliable delivery is a handful of hops; latency and packet loss accumulate quickly, so do not design emergency relay chains around large hop counts. We want to hear how far your MeshCore conversations go.

FAQ: 3. Server Administration

3.1. Q: How do you configure a repeater or a room server?

A: - When MeshCore is flashed onto a LoRa device is for the first time, it is necessary to set the server device's frequency to make it utilize the frequency that is legal in your country or region.

Repeater or room server can be administered with one of the options below:

- After a repeater or room server firmware is flashed on to a LoRa device, go to and use the web user interface to connect to the LoRa device via USB serial. From there you can set the name of the server, its frequency and other related settings, location, passwords etc.

!image

- Connect the server device using a USB cable to a computer running Chrome on the MeshCore web tool (as of 2026, configuration is at https://config.meshcore.io and flashing at https://flasher.meshcore.io; verify which host currently provides the serial console), then use the `console` feature to connect to the device

- Use a MeshCore smartphone clients to remotely administer servers via LoRa.

- A T-Deck running unlocked/registered MeshCore firmware. Remote server administration is enabled through registering your T-Deck with Ripple Radios. It is one of the ways to support MeshCore development. You can register your T-Deck through the Ripple Radios registration page (check the current MeshCore / Ripple Radios site for the active registration link).

3.2. Q: Do I need to set the location for a repeater?

A: While not required, with location set for a repeater it will show up on the MeshCore map in the future. Set location with the following command:

`set lat`

`set lon`

You can get the latitude and longitude from Google Maps by right-clicking the location you are at on the map.

3.3. Q: What is the password to administer a repeater or a room server?

A: The default admin password to a repeater and room server is `password`. Use the following command to change the admin password:

```
password {new-password}
```

3.4. Q: What is the password to join a room server?

A: The default guest password to a room server is `hello`. Use the following command to change the guest password:

```
set guest.password {guest-password}
```

⚠ **Security warning — change both defaults before deploying.** The default admin password (`password`) and guest password (`hello`) are publicly known. A repeater or room server left on these defaults can be remotely administered, reconfigured, or hijacked by **anyone within radio range** — a serious risk for community or emergency infrastructure. Immediately set a strong, unique admin password (and guest password) on every deployed node, before it goes on the air, using the commands above.

3.5. Q: Can I retrieve a repeater's private key or set a repeater's private key?

A: You can issue these commands to get or set a repeater's private key using a USB serial connection.

`get prv.key` to print a repeater's private key on the serial console

`set prv.key` to set a repeater's private key on the serial console

Reboot the repeater after `set prv.key` command for the new private key to take effect.

3.6. Q: The first byte of my repeater's public key collides with an exisitng repeater on the mesh. How do I get a new private key with a matching public key that has its first byte of my choosing?

A: You can generate a new private key and specific the first byte of its public key here: https://gessaman.com/mc-keygen/

Having multiple repeaters with the same first byte ID does not negatively affect the mesh or its functionality. Flood and pathed packets will still reach their destinations. First byte ID collision makes traceroute and path analysis harder because these tools don't know exactly which of the two (or more) colliding repeaters is the one in the path.

Best practice is when you set up a new repeater, choose a public key that is not in use. If it is not possible to find a unique first byte for your repeater's public key, choose one that is unique within about 10 miles (16 km) to minimize collision with nearby repeaters.

3.7. Q: My repeater maybe suffering from deafness due to high power interference near my mesh's frequency, it is not hearing other in-range MeshCore radios. What can I do?

A: This appears to be related to the SX1262 radio's automatic gain control behavior (see the Semtech SX1262 datasheet for how AGC operates). As an empirical mitigation, you can periodically reset its AGC with the command below.

`set agc.reset.interval {seconds}`

The interval is specified in seconds and must be a multiple of 4. Example: `set agc.reset.interval 4` resets AGC every 4 seconds, which works well to cure deafness.

This is a very low cost operation. In the firmware, the AGC reset is implemented by setting `state = STATE_IDLE;` in the function `RadioLibWrapper::resetAGC()` in `RadioLibWrappers.cpp`.

3.8. Q: How do I make my repeater an observer on the mesh?

A: The observer instruction is available here: https://analyzer.letsmesh.net/observer/onboard

3.9. Q: What is multi-byte support? What do 1-byte, 2-byte, 3-byte adverts and messages mean?

A:

The original MeshCore protocol design uses the first byte of a repeater's public key to denote the repeater in a path. And with 1 byte for each repeater in the path, MeshCore packets can travel as many as 64 hops.

However, with 1 byte, there are only 254 unique IDs (exclude 00 and FF which are reserved). Many meshes group have multiple repeaters with the same first byte in their public keys. Packets

continue to pass through repeaters and the mesh is not harmed in anyway. It does make it harder for tools to analyze paths with duplicated repeater IDs.

Firmware version 1.14 and newer introduces the ability for repeaters to advert with 1-, 2-, or 3-byte adverts. Companions can also send out channel and direct messages with 1-, 2-, or 3-byte path. Adverts and messages sent in 1-byte path is compatible with repeater firmware older or newer than 1.14. Because each extra path-hash byte per hop consumes packet space, 1-byte paths reach up to 64 hops, 2-byte adverts and messages will travel up to 32 hops, and 3-byte adverts and messages will travel up to 21 hops.

3.9.1. Q: **What path hash sizes will my repeater forward?**

Repeaters running firmware 1.14+ repeat packets sent with 1-, 2-, or 3-byte path hash. Repeaters on firmware older than 1.14 only repeat 1-byte path hash packets and silently drop 2- and 3-byte packets.

3.9.2. Q: **What determines a packet's path hash size?**

The original packet sender determines the path hash size. The most common original sender is a companion app. The other common original sender is a repeater, when it broadcasts its advert.

3.9.3. Q: **How do I change my companion's path hash size?**

As of firmware version 1.14 and MeshCore app version 1.41.0, in the MeshCore app, you can set your companion's message path hash size in `Settings (gear icon)`, `Experimental Settings`.

Until your regional mesh has the vast majority of the repeaters updated to 1.14+ firmware, it is recommended to keep your companion at the default 1-byte because pre-1.14 repeaters will silently drop messages with larger path hashes.

3.9.4. Q: **What does the CLI command `path.hash.mode` do on a repeater?**

This CLI command `path.hash.mode` *only* controls the path hash size used in a repeater's own advert broadcasts. It does **NOT** affect which packets the repeater forwards. A repeater with firmware 1.14+ always forward 1-, 2-, and 3-byte packets regardless of this setting.

Usage: `set path.hash.mode {0|1|2}`:

```
┌──────────────────┬──────────────────────────┐
│ path.hash.mode │ Advert path hash size │
```

```
├──────────────┼──────────────────────────┤
| 0 | 1 byte (default) |
├──────────────┼──────────────────────────┤
| 1 | 2 bytes |
├──────────────┼──────────────────────────┤
| 2 | 3 bytes |
└──────────────┴──────────────────────────┘
```

It is safe to set your 1.14+ repeaters to mode 1 or 2.

3.9.5. Q: **Why use 2- or 3-byte path hash for adverts?**

A longer path hash helps tools like the LetsMesh.net Analyzer and MeshMapper disambiguate repeaters more reliably. With only 1 byte, the chance of different repeaters having the same first byte in their public key is high, making it harder to tell them apart in mesh network analysis. Since this only affects adverts, there's no downside. 2- and 3-byte adverts don't travel as far as 1-byte adverts, but it is not important for MeshCore nodes to hear a repeater's advert that are 21 or 32 hops away.

3.9.6. Q: **When can we move away from 1-byte path hash for channel and direct messages?**

You should move to send 2-byte or 3-byte channel and direct messages when the vast majority of the repeaters in your regional mesh are updated to firmware version 1.14 or newer. Setting your repeater's `path.hash.mode` to 1 (for 2-byte path hash) or 2 (for 3-byte path hash) now helps the community gauge to how many repeaters have updated to 1.14+. Please work with your MeshCore community together to decide when to switch to 2-byte path or 3-byte path for channel and direct messages.

FAQ: 4. T-Deck Related

4.1. Q: Is there a user guide for T-Deck, T-Pager, T-Watch, or T-Display Pro?

A: Yes, it is available on https://buymeacoffee.com/ripplebiz/ultra-v7-7-guide-meshcore-users

4.2. Q: What are the steps to get a T-Deck into DFU (Device Firmware Update) mode?

A:

1. Device off
2. Connect USB cable to device
3. Hold down trackball (keep holding)
4. Turn on device
5. Hear USB connection sound
6. Release trackball
7. T-Deck in DFU mode now
8. At this point you can begin flashing using the MeshCore web flasher at flasher.meshcore.io (use the Console / flash option).

4.3. Q: Why is my T-Deck Plus not getting any satellite lock?

A: For T-Deck Plus, the GPS baud rate should be set to **38400**. Also, some T-Deck Plus devices were found to have the GPS module installed upside down, with the GPS antenna facing down instead of up. If your T-Deck Plus still doesn't get any satellite lock after setting the baud rate to 38400, you might need to open the device to check the GPS orientation.

GPS on T-Deck is always enabled. You can skip the "GPS clock sync" and the T-Deck will continue to try to get a GPS lock. You can go to the `GPS Info` screen; you should see the `Sentences:` counter increasing if the baud rate is correct.

Source

4.4. Q: Why is my OG (non-Plus) T-Deck not getting any satellite lock?

A: The OG (non-Plus) T-Deck doesn't come with a GPS. If you added a GPS to your OG T-Deck, please refer to the manual of your GPS to see what baud rate it requires. Alternatively, you can try to set the baud rate from 9600, 19200, etc., and up to 115200 to see which one works.

4.5. Q: What size of SD card does the T-Deck support?

A: Users have had no issues using 16GB or 32GB SD cards. Format the SD card to **FAT32**.

4.6. Q: what is the public key for the default public channel?

A:

T-Deck uses the same key the smartphone apps use but in base64

`izOH6cXN6mrJ5e26oRXNcg==`

Note: This is the shared key for the public default channel and is known to everyone - it provides **NO privacy**. Anyone can read default-channel traffic. Never send sensitive information on the default channel; create a private channel with your own key for any confidential traffic.

There is no `=` key on the T-Deck's hardware keyboard. You can use the on-screen software keyboard to enter `=`. Tap the text box to enable the on-screen software keyboard.

In `izOH6cXN6mrJ5e26oRXNcg==`, the 3rd character is a capital letter `O` (as in Oscar), not the digit zero `0`.

The smartphone app key is in hex:

`8b3387e9c5cdea6ac9e5edbaa115cd72`

Source

4.7. Q: How do I get maps on T-Deck?

A: You need map tiles. Pre-downloaded map tile bundles (for Europe and the US) are sometimes shared as a way to support development. *(Note: the specific download links are not currently available here - check the MeshCore Discord or the official T-Deck guide for the current tile-bundle links.)*

Another way to download map tiles is to use a Python tile-download script to get the tiles for the areas you want. *(The script link is not currently provided here; the MeshCore community shares both a base script and a modified version with extra error handling and parallel downloads - ask in the MeshCore Discord for the current links.)*

4.8. Q: Where do the map tiles go?

Once you have the tiles downloaded, copy the `\tiles` folder to the root of your T-Deck's SD card.

4.9. Q: How to unlock deeper map zoom and server management features on T-Deck?

A: You can download, install, and use the T-Deck firmware for free, but it has some features (map zoom, server administration) that are enabled if you purchase an unlock code for \$10 per T-Deck device.

Unlock page: *(link not currently provided here - the unlock code is sold via the developer's store; see the official T-Deck guide linked in 4.1 for the current purchase page.)*

4.10. Q: How to decipher the diagnostics screen on T-Deck?

A: Space is tight on T-Deck's screen, so the information is a bit cryptic. The format is :

`{hops} l:{packet-length}({payload-len}) t:{packet-type} snr:{n} rssi:{n}`

See here for packet-type:

https://github.com/meshcore-dev/MeshCore/blob/main/src/Packet.h#L19

#define PAYLOAD_TYPE_REQ 0x00 // request (prefixed with dest/src hashes, MAC) (enc data: timestamp, blob)

#define PAYLOAD_TYPE_RESPONSE 0x01 // response to REQ or ANON_REQ (prefixed with dest/src hashes, MAC) (enc data: timestamp, blob)

#define PAYLOAD_TYPE_TXT_MSG 0x02 // a plain text message (prefixed with dest/src hashes, MAC) (enc data: timestamp, text)

#define PAYLOAD_TYPE_ACK 0x03 // a simple ack #define PAYLOAD_TYPE_ADVERT 0x04 // a node advertising its Identity

#define PAYLOAD_TYPE_GRP_TXT 0x05 // an (unverified) group text message (prefixed with channel hash, MAC) (enc data: timestamp, "name: msg")

#define PAYLOAD_TYPE_GRP_DATA 0x06 // an (unverified) group datagram (prefixed with channel hash, MAC) (enc data: data_type, data_len, blob)

#define PAYLOAD_TYPE_ANON_REQ 0x07 // generic request (prefixed with dest_hash, ephemeral pub_key, MAC) (enc data: ...)

#define PAYLOAD_TYPE_PATH 0x08 // returned path (prefixed with dest/src hashes, MAC) (enc data: path, extra)

Source

4.11. Q: The T-Deck sound is too loud, or can you customize the sound?

A: You can customise (and effectively quiet down) the sounds on the T-Deck by placing your own `.mp3` files - including silent or quieter clips - onto the `root` dir of the SD card. Replacing a sound file with a silent or low-volume clip is the way to reduce or disable that alert. The files are:

- `startup.mp3`
- `error.mp3`
- `alert.mp3`
- `new-advert.mp3`
- `existing-advert.mp3`

4.13. Q: What is the 'Import from Clipboard' feature on the t-deck and is there a way to manually add nodes without having to receive adverts?

A: 'Import from Clipboard' is for importing a contact via a file named 'clipboard.txt' on the SD card. The opposite, is in the Identity screen, the 'Card to Clipboard' menu, which writes to 'clipboard.txt' so you can share yourself (call these 'biz cards', that start with "meshcore://...")

4.14. Q: How to capture a screenshot on T-Deck?

A: To capture a screenshot on a T-Deck, long press the top-left corner of the screen. The screenshot is saved to the microSD card, if one is inserted into the device.

FAQ: 5. General

5.1. Q: What are BW, SF, and CR?

A:

BW is bandwidth - width of frequency spectrum that is used for transmission

SF is spreading factor - sets how many chips encode each symbol (2^SF); higher SF means longer airtime, greater range and sensitivity, but lower data rate.

CR is coding rate - from: https://www.thethingsnetwork.org/docs/lorawan/fec-and-code-rate/

TL;DR: default CR to 5 for good stable links. If it is not a solid link and is intermittent, change to CR to 7 or 8.

Forward Error Correction is a process of adding redundant bits to the data to be transmitted. During the transmission, data may get corrupted by interference (changes from 0 to 1 / 1 to 0). These error correction bits are used at the receivers for restoring corrupted bits.

The Code Rate of a forward error correction expresses the proportion of bits in a data stream that actually carry useful information.

There are 4 code rates used in LoRa:

4/5

4/6

4/7

4/8

For example, if the code rate is 4/7, for every 4 bits of useful information, the coder generates a total of 7 bits of data, of which 3 bits are redundant.

Making the bandwidth 2x wider (from BW125 to BW250) allows you to send 2x more bytes in the same time. Making the spreading factor 1 step lower (from SF10 to SF9) allows you to send 2x more bytes in the same time.

Lowering the spreading factor reduces the receiver's sensitivity, so the link tolerates less noise and has shorter range, in exchange for faster transmission. You could compare this to two people taking in a noisy place (a bar for example). If you're far from each other, you have to talk slow

(SF10), but if you're close, you can talk faster (SF7)

So, it's balancing act between speed of the transmission and resistance to noise.

things network is mainly focused on LoRaWAN, but the LoRa low-level stuff still checks out for any LoRa project

5.2. Q: Do MeshCore clients repeat?

A: No, MeshCore clients do not repeat. This is the core of MeshCore's messaging-first design. This is to avoid devices flooding the air ware and create endless collisions, so messages sent aren't received. Note the emergency-comms trade-off: because MeshCore clients never repeat, the network depends entirely on surviving repeater infrastructure. In a disaster where repeaters lose power, clients fall back to direct radio range only - plan redundant, powered repeaters.

In MeshCore, only repeaters and room server with `set repeat on` repeat.

5.3. Q: What happens when a node learns a route via a mobile repeater, and that repeater is gone?

A: If you used to reach a node through a repeater and the repeater is no longer reachable, the client will send the message using the existing (but now broken) known path, the message will fail after 3 retries, and the app will reset the path and send the message as flood on the last retry by default. This can be turned off in settings. If the destination is reachable directly or through another repeater, the new path will be used going forward. Or you can set the path manually if you know a specific repeater to use to reach that destination.

In the case if users are moving around frequently, and the paths are breaking, they just see the phone client retries and revert to flood to attempt to re-establish a path.

5.4. Q: How does a node discovery a path to its destination and then use it to send messages in the future, instead of flooding every message it sends like Meshtastic?

Routes are stored in sender's contact list. When you send a message the first time, the message first gets to your destination by flood routing. When your destination node gets the message, it will send back a delivery report to the sender with all repeaters that the original message went through. This delivery report is flood-routed back to you the sender and is a basis for future direct path. When you send the next message, the path will get embedded into the packet and be evaluated by repeaters. If the hop and address of the repeater matches, it will retransmit the message, otherwise it will not retransmit, hence minimizing utilization.

Source

5.5. Q: Do public channels always flood? Do private channels always flood?

A: Yes. Group/public channels are one-to-many broadcasts with no single destination, so there is no specific path to discover - they must flood. Repeaters can however deny flood traffic up to some hop limit, with the `set flood.max` CLI command. Administrators of repeaters get to set the rules of their repeaters.

Source

5.6. Q: what is the public key for the default public channel?

A: The smartphone app key is in hex:

`8b3387e9c5cdea6ac9e5edbaa115cd72`

T-Deck uses the same key but in base64

`izOH6cXN6mrJ5e26oRXNcg==`

The third character is the capital letter 'O', not zero `0`

Source

5.7. Q: Is MeshCore open source?

A: Most of the firmware is freely available. Everything is open source except the T-Deck firmware and Liam's native mobile apps.

- Firmware repo: https://github.com/meshcore-dev/MeshCore

5.8. Q: How can I support MeshCore?

A: Provide your honest feedback on GitHub and on MeshCore Discord server. Spread the word of MeshCore to your friends and communities; help them get started with MeshCore.

Support Liam Cottle's smartphone client development by unlocking the server administration wait gate with in-app purchase

Support Rastislav Vysoky (recrof)'s flasher web site and the map web site development through PayPal or Revolut

5.9. Q: How do I build MeshCore firmware from source?

A: See instructions here:

https://discord.com/channels/826570251612323860/1330643963501351004/1341826372120608769

Build instructions for MeshCore:

For Windows, first install WSL and Python+pip via: https://plainenglish.io/blog/setting-up-python-on-windows-subsystem-for-linux-wsl-26510f1b2d80

(Linux, Windows+WSL) In the terminal/shell:

```
sudo apt update
sudo apt install libpython3-dev
sudo apt install python3-venv
```

Mac: python3 should be already installed.

Then it should be the same for all platforms:

```
python3 -m venv meshcore
cd meshcore && source bin/activate
pip install -U platformio
git clone https://github.com/meshcore-dev/MeshCore.git
cd MeshCore
```

open platformio.ini and in `[arduino_base]` edit the `LORA_FREQ`. Set `LORA_FREQ` to your region's frequency (e.g. 910.525 for USA/Canada, 867.5 for EU) before building.

save, then run:

```
pio run -e RAK_4631_Repeater
```

then you'll find `firmware.zip` in `.pio/build/RAK_4631_Repeater`

5.10. Q: Are there other MeshCore related open source projects?

A: Liam Cottle's MeshCore web client and MeshCore Javascript library are open source under MIT license.

Web client: https://github.com/liamcottle/meshcore-web

Javascript: https://github.com/liamcottle/meshcore.js

5.11. Q: Does MeshCore support ATAK

A: ATAK is not currently on MeshCore's roadmap.

Meshcore would not be best suited to ATAK because MeshCore:

clients do not repeat and therefore you would need a network of repeaters in place

will not have a stable path where all clients are constantly moving between repeaters

MeshCore clients would need to reset path constantly and flood traffic across the network which could lead to lots of collisions with something as chatty as ATAK.

This could change in the future if MeshCore develops a client firmware that repeats.

Source

5.12. Q: How do I add a node to the MeshCore Map

A:

To add a BLE Companion radio, connect to the BLE Companion radio from the MeshCore smartphone app. In the app, tap the `3 dot` menu icon at the top right corner, then tap `Internet Map`. Tap the `3 dot` menu icon again and choose `Add me to the Map`

To add a Repeater or Room Server to the map, go to the Contact List, tap the `3 dot` next to the Repeater or Room Server you want to add to the Internet Map, tap `Share`, then tap `Upload to Internet Map`.

You can use the same companion (same public key) that you used to add your repeaters or room servers to remove them from the Internet Map.

5.13. Q: Can I use a Raspberry Pi to update a MeshCore radio?

A: Yes.

Below are the instructions to flash firmware onto a supported LoRa device using a Raspberry Pi over USB serial.

> Instructions for nRF devices like RAK, T1000-E, T114 are immediately after the ESP instructions

For ESP-based devices (e.g. Heltec V3) you need:

- Download firmware file from https://flasher.meshcore.io
- Go to the web site on a browser, find the section that has the firmware up need
- Click the Download button, right click on the file you need, for example,
- `Heltec_V3_companion_radio_ble-v1.7.1-165fb33.bin`
- Non-merged bin keeps the existing Bluetooth pairing database
- `Heltec_v3_companion_radio_usb-v1.7.1-165fb33-merged.bin`
- Merged bin overwrites everything including the bootloader, existing Bluetooth pairing database, but keeps configurations.
- Right click on the file name and copy the link and note it for later use here is an example: `https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_ble-v1.7.1-165fb33.bin`
- Run:
- `wget https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_ble-v1.7.1-165fb33.bin` to download the firmware file for your device type. or the version you need - USB, BLE, Repeater, Room Server, merged bin or non-merged bin
- If the above wget command only downloads a very small file (10K bytes instead of more than 100K byte, use this command instead:
- `wget --user-agent="Mozilla/5.0" --content-disposition "https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_usb-v1.7.1-165fb33.bin"`
- Confirm the `ttyXXXX` device path on your Raspberry Pi:
- Go to `/dev` directory, run ls command to find confirm your device path
- They are usually `/dev/ttyUSB0` for ESP devices
- For ESP-based devices, install esptool from the shell:
- `pip install esptool --break-system-packages`
- To flash, use the following command:
- For non-merged bin:
- `esptool.py -p /dev/ttyUSB0 --chip esp32-s3 write_flash 0x10000 .bin`
- For merged bin:
- `esptool.py -p /dev/ttyUSB0 --chip esp32-s3 write_flash 0x00000 .bin`

Instructions for nRF devices:

For nRF devices (e.g. RAK, Heltec T114) you need the following:

- Download firmware file from https://flasher.meshcore.io
- Go to the web site on a browser, find the section that has the firmware up need
- You need the ZIP version for the adafruit flash tool (below)
- Click the Download button, right click on the ZIP file, for example:
- `RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip`
- Right click on the file name and copy the link and note it for later use here is an example: `https://flasher.meshcore.io/releases/download/companion-v1.7.1/RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip`
- Run:

- `wget https://flasher.meshcore.io/releases/download/companion-v1.7.1/RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip` to download the firmware file for your device type. or the version you need - USB, BLE, Repeater, Room Server, ZIP file only
- Confirm the `ttyXXXX` device path on your Raspberry Pi:
- Go to `/dev` directory, run ls command to find confirm your device path
- They are usually `/dev/ttyACM0` for nRF devices
- For nRF-based devices, install adafruit-nrfutil
- `pip install adafruit-nrfutil --break-system-packages`
- Use this command to flash the nRF device:
- `adafruit-nrfutil --verbose dfu serial --package RAK_4631_companion_radio_usb-v1.7.1-165fb33.zip -p /dev/ttyACM0 -b 115200 --singlebank --touch 1200`

To manage a repeater or room server connected to a Pi over USB serial using shell commands, you need to install `picocom`. To install `picocom`, run the following command:

- `sudo apt install picocom`

To start managing your USB serial-connected device using picocom, use the following command:

- `picocom -b 115200 /dev/ttyUSB0 --imap lfcrlf`

From here, reference repeater and room server command line commands on MeshCore github wiki here:

- https://github.com/meshcore-dev/MeshCore/wiki/Repeater-&-Room-Server-CLI-Reference

5.14. Q: Are there are projects built around MeshCore?

A: Yes, there are many. MeshCore's protocol is open source using the MIT license. The MIT license and the open source protocol makes it very easy for the MeshCore community to build new firmware for radios, applications on mobile devices, map tools, and analysis tools, and integration with other projects like Home Asistant.

As new MeshCore community projects become available on a weekly basis, we have stopped tracking them here in this FAQ. samuk maintains a very exhausive list of MeshCore community project at https://github.com/samuk/awesome-meshcore/blob/main/README.md. samuk accepts PRs and merges them regularly.

5.15. Q: Are there client applications for Windows or Mac?

A: Yes, the same iOS and Android client is also available for Windows and Mac. You can find them together with the Android APK here:

https://files.liamcottle.net/MeshCore

Both the Windows and Mac versions of the client app are fully unlocked and are free to use.

5.16. Q: Are there any resources that compare MeshCore to other LoRa systems?

A: Here is a list of MeshCore comparison resources:

The Comms Channel on YouTube:

https://www.youtube.com/watch?v=guDoKGs02Us

MeshCore Advantages by MCarper:

https://github.com/mikecarper/meshfirmware/blob/main/MeshCoreAdvantages.md

Meshcore vs Meshtastic by austinmesh.org

https://www.austinmesh.org/learn/meshcore-vs-meshtastic/

FAQ: 6. Troubleshooting

6.1. Q: My client says another client or a repeater or a room server was last seen many, many days ago.

A: This is almost always a clock/time-sync problem - see the shared answer under 6.2 below, which covers both this "last seen many days ago" symptom and the "not showing up at all" symptom.

6.2. Q: A repeater or a client or a room server I expect to see on my discover list (on T-Deck) or contact list (on a smart device client) are not listed.

A:

- If your client is a T-Deck, it may not have its time set (no GPS installed, no GPS lock, or wrong GPS baud rate).
- If you are using the Android or iOS client, the other client, repeater, or room server may have the wrong time.

You can get the epoch time on and use it to set your T-Deck clock. For a repeater and room server, the admin can use a T-Deck to remotely set their clock (clock sync), or use the `time` command in the USB serial console with the server device connected.

6.3. Q: How to connect to a repeater via BLE (Bluetooth)?

A: You can't connect to a device running repeater firmware via Bluetooth. Devices running the BLE companion firmware you can connect to it via Bluetooth using the android app

6.4. Q: My companion isn't showing up over Bluetooth?

A: make sure that you flashed the Bluetooth companion firmware and not the USB-only companion firmware.

6.5. Q: I can't connect via Bluetooth, what is the Bluetooth pairing code?

A: the default Bluetooth pairing code is `123456`. Note that this is a well-known, fixed default - it is not a secret. For sensitive deployments, be aware that anyone within Bluetooth range (a short distance) could attempt to pair while the device is in pairing mode.

6.6. Q: My Heltec V3 keeps disconnecting from my smartphone. It can't hold a solid Bluetooth connection.

A: Heltec V3 has a very small coil antenna on its PCB for Wi-Fi and Bluetooth connectivity. It has a very short range, only a few feet. Try repositioning the device closer to the phone first. It is possible to remove the coil antenna and replace it with a 31mm wire (roughly a quarter-wave at 2.4 GHz), and the BT range is much improved with the modification. However, this is an advanced, irreversible mod requiring fine soldering on a tiny PCB antenna feed - it can permanently damage the board and voids the warranty, and soldering an unmatched wire can worsen SWR. Only attempt it if you are experienced.

6.7. Q: My RAK/T1000-E/xiao_nRF52 device seems to be corrupted, how do I wipe it clean to start fresh?

A:

1. Connect USB-C cable to your device, per your device's instruction, get it to flash mode:

- For RAK, click the reset button **TWICE**
- For T1000-e, quickly disconnect and reconnect the magnetic side of the cable from the device **TWICE**
- For Heltec T114, click the reset button **TWICE** (the bottom button)
- For Xiao nRF52, click the reset button once; if that doesn't work, double-click the reset button (two quick presses); if that still fails, disconnect the board from your PC and reconnect again (seeed studio wiki)

1. A new folder will appear on your computer's desktop
2. Download the `flash_erase*.uf2` file for your device on https://flasher.meshcore.io

- RAK WisBlock and Heltec T114: `Flash_erase-nRF52_softdevice_v6.uf2`
- Seeed Studio Xiao nRF52 WIO: `Flash_erase-nRF52_softdevice_v7.uf2`

1. drag and drop the uf2 file for your device to the root of the new folder
2. Wait for the copy to complete. You might get an error dialog, you can ignore it
3. Go to https://flasher.meshcore.io, click `Console` and select the serial port for your connected device
4. In the console, press enter. Your flash should now be erased
5. You may now flash the latest MeshCore firmware onto your device

Separately, starting in firmware version 1.7.0, there is a CLI Rescue mode. If your device has a user button (e.g. some RAK, T114), you can activate the rescue mode by hold down the user button of the device within 8 seconds of boot. Then you can use the 'Console' on https://flasher.meshcore.io

6.8. Q: WebFlasher fails on Linux with failed to open

A: If the usb port doesn't have the right ownership for this task, the process fails with the following error:

```
NetworkError: Failed to execute 'open' on 'SerialPort': Failed to open serial port.
```

Allow the browser user on it:

```
# setfacl -m u:YOUR_USER_HERE:rw /dev/ttyUSB0
```

FAQ: 7. Other Questions:

7.1. Q: How to update nRF (RAK, T114, Seed XIAO) companion, repeater and room server firmware over the air using the new simpler DFU app?

A: The steps below work on both Android and iOS as nRF has made both apps' user interface the same on both platforms:

1. Download nRF's DFU app from iOS App Store or Android's Play Store, you can find the app by searching for `nrf dfu`, the app's full name is `nRF Device Firmware Update`
2. On https://flasher.meshcore.io, download the **ZIP** version of the firmware for your nRF device (e.g. RAK or Heltec T114 or Seeed Studio's Xiao)
3. From the MeshCore app, login remotely to the repeater you want to update with admin privilege
4. Go to the Command Line tab, type `start ota` and hit enter.
5. you should see `OK` to confirm the repeater device is now in OTA mode
6. Run the DFU app,tab `Settings` on the top right corner
7. Enable `Packet receipt notifications`, and change `Number of Packets` to 10 for RAK, 8 for T114. 8 also works for RAK.
8. Select the firmware zip file you downloaded
9. Select the device you want to update. If the device you want to update is not on the list, try enabling `OTA` on the device again
10. If the device is not found, enable `Force Scanning` in the DFU app
11. Tab the `Upload` to begin OTA update
12. If it fails, try turning off and on Bluetooth on your phone. If that doesn't work, try rebooting your phone. If you keep getting failures at the "Enabling Bootloader" step, try forgetting the NRF board in your IOS or Andriod device's bluetooth settings and re-pair it through the DFU app.
13. Wait for the update to complete. It can take a few minutes.
14. It is strongly recommended that you install and use the OTAFIX bootloader at https://github.com/oltaco/Adafruit_nRF52_Bootloader_OTAFIX.
15. To update a companion node over OTA, it must be running companion firmware v1.15 or greater.
16. Please see the Meshcore Blog for additional information on OTA firmware flashing:

- https://blog.meshcore.io/2026/04/06/otafix-bootloader
- https://blog.meshcore.io/2026/04/02/nrf-ota-update

7.1.1 Q: Can I update Seeed Studio Wio Tracker L1 Pro using OTA?

A: You can flash this safer bootloader to the Wio Tracker L1 Pro

https://github.com/oltaco/Adafruit_nRF52_Bootloader_OTAFIX

After this bootloader is flashed onto the device, you can trigger over the air update using bluetooth by holding the button next to the D-Pad and then click the reset button. The follow the same OTA update instructions above. You can skip pass the `start ota` instruction and start the update using the DFU app.

7.2. Q: How to update ESP32-based devices over the air?

A: For ESP32-based devices (e.g. Heltec V3):

1. On https://flasher.meshcore.io, download the **non-merged** version of the firmware for your ESP32 device (e.g. `Heltec_v3_repeater-v1.6.2-4449fd3.bin`, no `"merged"` in the file name)
2. From the MeshCore app, login remotely to the repeater you want to update with admin privilege
3. Go to the Command Line tab, type `start ota` and hit enter.
4. you should see `OK` to confirm the repeater device is now in OTA mode
5. The command `start ota` on an ESP32-based device starts a wifi hotspot named `MeshCore OTA`
6. From your phone or computer connect to the 'MeshCore OTA' hotspot
7. From a browser, go to http://192.168.4.1/update and upload the non-merged bin from the flasher

7.3. Q: Is there a way to lower the chance of a failed OTA device firmware update (DFU)?

A: Yes, developer `oltaco` has an enhanced OTA DFU bootloader for nRF52 based devices. With this bootloader, if it detects that the application firmware is invalid, it falls back to OTA DFU mode so you can attempt to flash again to recover. This bootloader has other changes to make the OTA DFU process more fault tolerant.

Refer to https://github.com/oltaco/Adafruit_nRF52_Bootloader_OTAFIX for the latest information.

Currently, the following boards are supported:

- Heltec Automation Mesh Node T114 / HT-nRF5262
- Nologo ProMicro NRF52840 (aka SuperMini NRF52840)
- Seeed Studio SenseCAP Card Tracker T1000-E
- Seeed Studio Wio Tracker L1
- Seeed Studio XIAO nRF52840 BLE
- Seeed Studio XIAO nRF52840 BLE SENSE
- RAK 4631
- RAK WisMesh Tag (new 28/11/2025)

7.4. Q: are the MeshCore logo and font available?

A: Yes, it is on the MeshCore github repo here:

https://github.com/meshcore-dev/MeshCore/tree/main/logo

7.5. Q: What is the format of a contact or channel QR code?

A:

Channel:

`meshcore://channel/add?name=&secret=`

Contact:

`meshcore://contact/add?name=&public_key=&type=`

where `&type` is:

`chat = 1`

`repeater = 2`

`room = 3`

`sensor = 4`

7.6. Q: How do I connect to the companion via WIFI, e.g. using a heltec v3?

A:

WiFi firmware requires you to compile it yourself, as you need to set the wifi ssid and password.

Edit WIFI_SSID and WIFI_PWD in `./variants/heltec_v3/platformio.ini` and then flash it to your device.

7.7. Q: I have a Station G2, or a Heltec V4, or an Ikoka Stick , or a radio with a EByte E22-900M30S or a E22-900M33S module, what should their transmit power be set to?

A:

For companion radios, you can set these radios' transmit power in the smartphone app. For repeater and room server radios, you can set their transmit power using the command line command `set tx`. You can get their current value using command line comand `get tx`

⚠ **WARNING: Set these values at your own risk. Incorrect power settings can permanently damage your radio hardware.**

⚠ **US FCC compliance — read before setting power:** In the United States, FCC Part 15.247 caps the conducted output of unlicensed 902–928 MHz LoRa devices at **1 W (30 dBm)** and effective isotropic radiated power (EIRP) at **4 W (36 dBm)** (the 1 W conducted limit assumes an antenna of up to 6 dBi gain; gain above 6 dBi requires a dB-for-dB reduction in conducted power). **Any setting above 30 dBm conducted is not legal for unlicensed US operation**, and pushing the radio's power amplifier to its hardware maximum can also damage the PA. Keep US settings at or below 30 dBm conducted (1 W), minus any antenna gain over 6 dBi. The rows in the table below that exceed 30 dBm are marked accordingly. Outside the US, follow your own region's limits — note that EU868 limits are far lower (typically 25 mW / 14 dBm ERP on most sub-bands), so the high-output figures shown for EU868 below are well above what EU rules permit and are listed only to document the hardware's capability.

Note on the "In-App Setting" column: The value you enter in the app or via `set tx` is a module-internal index, *not* the actual emitted power. It maps to the real radio output in a non-obvious, non-linear way (for example, an in-app value of 19 dBm on a Station G2 produces about 36.5 dBm of actual output, while 22 dBm in-app on a Heltec V4 produces only about 28 dBm). Always judge legality and safety by the **Target Radio Output** column, not the in-app number. The actual-output figures below come from community/vendor measurements; confirm against your specific board's datasheet before relying on them.

Device / Model	Region / Description	In-App Setting (dBm)	Target Radio Output	Notes
Station G2 Reference	US915 Max Output	19 dBm	36.5 dBm (4.46W) — EXCEEDS US FCC Part 15 limit (not legal for unlicensed US use)	
	US915 Max at 1dB compression point	16 dBm	35 dBm (3.16W) — EXCEEDS US FCC Part 15 limit (not legal for unlicensed US use)	1dB compression point
	EU868 Max at 1dB compression point	15 dBm	34.5 dBm (2.82W) — exceeds the US FCC Part 15 conducted limit, and is far above EU868 ERP limits (EU/other regions only — not legal at this level in the EU)	1dB compression point

Device / Model	Region / Description	In-App Setting (dBm)	Target Radio Output	Notes
	US915 1W Output	10 dBm	1W	Refer to your local government's requirements
	EU868 1W Output	9 dBm	1W	Refer to your local government's requirements
Ikoka Stick E22-900M30S	1W Model	19 dBm	1W	**DO NOT EXCEED** (Risk of burn out) data sheet
Ikoka Stick E22-900M33S	2W Model	9 dBm	2W — EXCEEDS US FCC Part 15 limit (not legal for unlicensed US use)	**DO NOT EXCEED** (Risk of burn out) data sheet Refer to your local government's requirements
Heltec V4	Standard Output	10 dBm	22 dBm (~0.15W)	
	High Output	22 dBm	28 dBm (~0.5W to 0.6W)	
