MeshCore Repeater Diagnostics via Serial Console

The MeshCore serial console provides direct access to repeater state and diagnostic information. Connecting via USB to a deployed repeater is the most reliable way to diagnose problems that cannot be addressed remotely.

Connecting to the Serial Console

On Windows: use PuTTY or the Arduino Serial Monitor. On Linux/Mac: use `screen` or `minicom` as a serial terminal.

```
# Linux/Mac
screen /dev/ttyUSB0 115200


# Windows (PuTTY): Connection Type = Serial, Speed = 115200, COM port varies
```

MeshCore boards use 115200 baud for the serial console, including RAK boards. (If you have configured an attached GPS module, that module's own UART may run at a different baud rate such as 9600 - but that is a separate interface from the console.)

Key Diagnostic Commands

These are commands from the on-device MeshCore serial CLI (see docs.meshcore.io/cli_commands). There is no single combined "status" command - health information is split across several `get ...` and `stats-...` commands.

Command	Output	Use for
`ver`	Firmware version	Confirm firmware build
`stats-core`	Battery voltage, uptime, queue depth	Overall health check

Command	Output	Use for
`get radio` / `get freq`	Radio config (frequency, bandwidth, SF, coding rate)	Verify radio settings match the network
`neighbors`	Up to the 8 most recently heard nodes, with timestamp and SNR	Verify which nodes are reaching this repeater
`stats-packets`	Packet counters: Received and Sent	Identify traffic/routing problems
`stats-radio`	Noise floor, last RSSI/SNR, airtime, receive errors	Signal quality of last received packet
`log`	Prints a captured rx log (must first be started with `log start`, stopped with `log stop`, cleared with `log erase`)	Capture and review received-packet activity
`reboot`	(restarts device)	Recover from hung state

Note: `contacts` and `list` are companion/client-side concepts, not repeater serial commands - on a headless repeater, use `neighbors` instead. The `log` command is a packet/rx capture you start and stop, not an always-on event log.

Interpreting Stats Output

The `stats-packets` command is a useful diagnostic tool. The firmware exposes Received and Sent counters. A healthy repeater shows:

- **Sent count tracking received traffic** - the repeater is relaying packets it is meant to forward. (The documented counters are Received and Sent; there is no separate "forwarded" or "dropped" counter, so judge activity from the Sent counter rising alongside Received rather than from a "forward rate" metric.)
- **Drops in MeshCore** - MeshCore drops a flood packet when it exceeds the `flood.max` hop ceiling (default 64), and its loop detection (`loop.detect`) drops a packet whose path shows this repeater's own ID/hash repeated. This is not a Meshtastic-style per-packet hop counter decrementing to zero. In MeshCore's path-based routing, a repeater also intentionally does *not* retransmit packets whose embedded path does not include it - these appear as non-forwarded traffic but are correct behavior, not a loop. Distinguish this selective non-forwarding from a genuine duplicate-flood loop before acting.
- **Increasing received count over time** - Confirms the repeater is hearing traffic from the network

Common Issues and Diagnostics

Symptom	Check	Fix

Received count stays at zero	Check antenna connection, verify radio settings match network	Reconnect antenna; verify radio parameters with `get radio` and `get freq` (frequency, bandwidth, SF, coding rate) and confirm they match the USA/Canada preset values shown in the app
Sent count zero despite receives	Verify device is running repeater firmware variant and that forwarding is enabled	Reflash with repeater firmware; confirm `set repeat on`
Battery voltage declining	Check solar panel output, charge controller LVD setting	Clean panel, verify charge controller settings
Rebooting frequently	Check for low battery voltage causing brownout	Size battery correctly; check charge controller
Not appearing in client node list	Repeater may not be sending flood adverts; check the flood advert interval with `get flood.advert.interval`	Send a flood advertisement with `advert` and confirm a flood advert interval is set, e.g. `set flood.advert.interval 12` (hours). Zero-hop adverts (`advert.zerohop` / `set advert.interval`) are local only. There is no `advert_hops` parameter.

Revision #3
Created 2026-05-03 05:46:19 UTC by Mesh America Admin
Updated 2026-06-09 00:56:30 UTC by Mesh America Admin