MeshCore Sensor CLI Reference

Overview

This page describes MeshCore's **actual** sensor support, which is deliberately minimal and differs substantially from the richer Meshtastic telemetry model. The most important facts to understand up front:

- **Sensor support is a compile-time feature.** Which sensor (if any) a node uses is selected when the firmware is built — there is no runtime command to pick or configure a sensor driver. A build only has sensor support if it was compiled with it (e.g. the `ENV_INCLUDE_*` build flags and example/sensor builds such as `simple_sensor` / SensorMesh). There is no separate deployable "SENSOR firmware variant" or role; the firmware roles are **Companion**, **Repeater**, and **Room Server**.
- **The entire sensor CLI is three commands:** a generic key/value store — `sensor list [start]`, `sensor get <key>`, and `sensor set <key> <value>`. These are only available when sensor support is compiled in.
- **Sensor readings are NOT broadcast in adverts.** A MeshCore advert carries only the node's name, optional position, and signed public key. Live sensor data is delivered on demand through the binary request/response protocol (`REQ_TYPE_GET_TELEMETRY_DATA`, CayenneLPP-encoded and permission-gated) — a client requests it; the node does not push it to the mesh.

Rich per-sensor runtime configuration is not currently available in MeshCore. There are no commands to select a sensor type, set its I2C address, set a per-sensor read/broadcast interval, enable/disable a sensor, trigger an immediate read, or apply a temperature offset at runtime. If you need those behaviors, they must be handled at compile time in the firmware or in the host application that consumes the telemetry. Always check the current documentation at docs.meshcore.io/cli_commands, because MeshCore is evolving and sensor support is expanding.

Accessing the CLI

MeshCore Repeater, Room Server, and sensor-capable nodes are configured over a USB/serial connection (using a terminal app, PuTTY, or the Arduino Serial Monitor) or over BLE via the MeshCore companion app or the `meshcore-cli` tool. Serial and BLE access to a node is at `115200 baud`. There is no documented "commands are case-insensitive" behavior — use the documented lowercase command names. Whether a given change takes effect immediately or requires a reboot is per-command (for example, `set radio` requires a reboot to apply), so do not assume every setting applies instantly.

The Sensor CLI (compile-time sensor builds only)

When a build includes sensor support, MeshCore exposes a small generic key/value store for sensor-related custom variables. These are the only sensor commands that exist:

Command	What it does
`sensor list [start]`	Lists the sensor custom variables as `key=value` lines (optionally starting from an index).
`sensor get <key>`	Prints the current value of one sensor variable.
`sensor set <key> <value>`	Sets the value of a sensor variable.

The keys available depend on what the compiled-in sensor build defines; this is a generic custom-variable mechanism, not a fixed set of sensor settings.

The following commands do **not** exist in MeshCore and will not work — do not use them (they are Meshtastic-style or invented): `set sensor type`, `set sensor addr`, `set sensor interval`, `set sensor enabled`, `set sensor temp_offset`, `sensor read`, `sensor status`, and `i2c scan`. The sensor (and its I2C address) is fixed at compile time; there is no runtime sensor-type table, enable/disable toggle, immediate-read command, status dump, or I2C scanner.

Reading Sensor Data (telemetry protocol)

There is no `sensor read` command and no human-readable console-output block. To obtain live sensor values, a client requests telemetry over the binary protocol: `REQ_TYPE_GET_TELEMETRY_DATA`. The node returns the data encoded as **CayenneLPP** (channel/type/value triplets), and access is permission-gated. In the Python library this is the `get_self_telemetry` path rather than a CLI

command. Decoding the CayenneLPP response into named values (temperature, humidity, pressure, etc.) is done by the requesting client, not printed by the node.

MeshCore Device CLI Reference

The real MeshCore node configuration commands (for Repeater / Room Server / sensor builds) include the following. See docs.meshcore.io/cli_commands for the authoritative and current list.

Command	Purpose
`set name <name>`	Set the node name.
`set tx <dbm>`	Set transmit power in dBm.
`set radio <freq>,<bw>,<sf>,<cr>`	Set radio parameters. Requires a reboot to apply.
`get <key>`	Read a device setting.
`advert`	Send an advert immediately.
`set flood.advert.interval <hours>`	Set the flood-advert interval, in hours (default 12 for Repeater, 0 for sensor).
`gps on` / `gps off`	Toggle GPS power state.
`set powersaving on` / `set powersaving off`	Change the power-saving mode (Repeater, persisted to prefs).
`reboot`	Reboot the node.
`erase`	Erase stored configuration.

Advert cadence is controlled by `set flood.advert.interval` (hours) — there is no per-second "sensor broadcast interval." Note again that adverts do not contain sensor readings.

Sensor Hardware and Wiring

Because the sensor driver is fixed at build time, the wiring below is about physically connecting a supported I2C sensor to the board, not about telling MeshCore which sensor to use. These I2C facts are accurate regardless of firmware.

- **BME280 / BME680 default I2C address:** `0x76` when SDO is tied low (to GND), or `0x77` when SDO is tied high (to VCC). The BME680 uses the same addressing as the BME280.
- **BME280 measures temperature, humidity, and pressure only** (no gas / air quality). The **BME680** adds a VOC gas / IAQ sensor.
- **Power the sensor from 3.3 V, not 5 V.** The BME280/BME680 and the nRF52/ESP32 I2C pins are 3.3 V devices; applying 5 V (or driving 5 V logic onto the 3.3 V SDA/SCL lines) can

permanently damage the sensor or the MCU.

For the cleanest integration on the RAK4631, use the **RAK1906 WisBlock Environmental Sensor module** (a Bosch BME680). It plugs directly into a WisBlock Base Board sensor slot with no soldering and no I2C wiring — the base board routes power and the shared I2C bus automatically. The RAK1906 is an I2C module and can go in any of the I2C sensor slots; slot A is just a convenient default.

If wiring a bare BME280/BME680 breakout to the RAK4631 GPIO header instead, use the WisBlock I2C1 bus:

Sensor pin	RAK4631 pin
VCC	3V3 (3.3 V — never 5 V)
GND	GND
SDA	WB_I2C1_SDA
SCL	WB_I2C1_SCL
SDO	GND (selects I2C address 0x76; tie to 3.3 V for 0x77)
CSB	3V3 (selects I2C mode)

Keep I2C wires short. If you must run the sensor on an extended cable, add 4.7 kΩ pull-up resistors on SDA and SCL.

Where to Go Next

For the authoritative, current CLI command list see docs.meshcore.io/cli_commands. Sensor support is a compile-time firmware feature, so which sensors and keys are available depends on the specific build. For host-side integration, the `meshcore_py` library and the MeshCore Home Assistant integration (`github.com/meshcore-dev/meshcore-ha`, domain `meshcore`) consume telemetry through the binary request/response protocol described above.

Revision #3
Created 2026-05-03 05:29:12 UTC by Mesh America Admin
Updated 2026-06-10 03:06:44 UTC by Mesh America Admin