

Device Configuration Settings

The Device configuration section (**Config > Device**) controls fundamental node behavior: its role in the network, how it handles rebroadcasting, node info broadcasts, and administrative access. These are the most impactful settings for network performance and should be understood before deploying any node.

Access these settings in the [Meshtastic app](#) under **Settings > Radio Configuration > Device**, via the web interface at `http://<node-ip>/config`, or through the Python CLI with `meshtastic --set device.*`.

Role

Config key: `device.role`

Default: `CLIENT`

The Role setting is the single most important Device configuration choice. It tells the firmware how this node should behave in the mesh - specifically, how aggressively it should rebroadcast packets, whether it should send periodic node info, and how it prioritizes battery life versus network contribution.

Available Roles

CLIENT

The standard role for personal devices carried by users. CLIENT nodes:

- Participate normally in mesh routing (rebroadcast packets they receive)
- Send NodeInfo broadcasts on the configured interval
- Receive and send messages on all subscribed channels
- Display in other nodes' neighbor lists

Use when: You are deploying a personal handheld node, a node you carry daily, or any general-purpose device.

CLIENT_MUTE

Identical to CLIENT but with packet rebroadcasting disabled. A CLIENT_MUTE node does not relay other nodes' packets (no routing-layer rebroadcast), but it still sends and receives its own messages.

Use when: You have many devices in close proximity (e.g., all members of a team within radio range of each other). Adding non-muted CLIENT nodes in a dense cluster creates redundant rebroadcasts that waste airtime. Making most of them CLIENT_MUTE reduces channel utilization while maintaining full message delivery to those devices.

Also useful for: Devices behind NAT or VPN tunnels acting as monitoring receivers; test devices you don't want to affect mesh traffic.

ROUTER

Designed for fixed infrastructure nodes at high elevation or central locations. ROUTER nodes:

- Rebroadcast packets with the highest priority
- Use a longer NodeInfo broadcast interval to reduce overhead traffic
- Are favored in routing decisions - the mesh prefers to route through ROUTER nodes
- Disable the screen and reduce radio/BLE activity by default to save power (ROUTER enables Power Saving automatically on ESP32)

Use when: Deploying a node on a rooftop, tower, hilltop, or any other fixed elevated location whose primary job is to relay traffic for other nodes. ROUTERS form the backbone of a community mesh.

Important: Only set ROUTER on nodes that will genuinely be at good RF locations. A ROUTER with poor antenna placement will be preferred for routing but perform poorly, degrading the network. It's better to use CLIENT on a node that isn't actually a good relay point.

ROUTER_CLIENT (removed)

ROUTER_CLIENT was a hybrid role that combined ROUTER routing behavior with CLIENT-level NodeInfo and message participation. This role was deprecated and removed in firmware 2.3.15 and is no longer a selectable role in current firmware.

What to use instead: For a home node that both relays traffic and receives your own messages, use `CLIENT` (which performs smart rebroadcasting via managed flooding) or `ROUTER_LATE`. For a node that is purely infrastructure, use `ROUTER`.

REPEATER

The most minimal infrastructure role. **Note:** REPEATER is deprecated as of firmware 2.7.11; for new infrastructure deployments prefer `ROUTER` (or `ROUTER_LATE`). REPEATER nodes:

- Rebroadcast received packets once with prioritized timing similar to ROUTER, with minimal overhead
- Do NOT send NodeInfo broadcasts - they are intentionally invisible in the node list
- Do NOT participate in messaging - they cannot send or receive application-layer messages
- Consume minimal airtime overhead

Use when: Deploying a relay-only node in a gap in coverage where you only need packet forwarding and don't need the node to participate in messaging or appear in the network map. REPEATER nodes are ideal for embedded or remotely deployed relays with no user interaction.

Caution: Because REPEATER nodes don't send NodeInfo, they won't appear on your map or in your node list. This makes it harder to verify they are online. Plan for out-of-band monitoring (serial console, MQTT telemetry, physical LED) if using REPEATER in unattended deployments.

TRACKER

Optimized for asset and vehicle tracking use cases. TRACKER nodes:

- Send frequent position updates automatically
- Are optimized to conserve bandwidth by only broadcasting position data (they reduce other overhead)
- Prioritize position packets. Smart position broadcast (transmit when moving, reduce rate when stationary) is configured via the position settings (`position.position_broadcast_smart_enabled` and related fields) and applies regardless of role

Use when: Attaching a node to a vehicle, pet, person, or asset for location tracking. The TRACKER role signals to the firmware and other nodes that this device's primary purpose is position reporting.

TAK

Designed for interoperability with TAK (Team Awareness Kit) - tactical situational awareness software used by military, law enforcement, SAR teams, and emergency management. TAK role nodes:

- Format outgoing position packets in a way compatible with TAK server ingestion
- Enable TAK-specific extensions in the Meshtastic packet format
- Work alongside ATAK (Android TAK) and WinTAK clients

Use when: Integrating Meshtastic into a TAK-based operational picture. Requires TAK server infrastructure or ATAK-capable devices on the network. Not useful for general community mesh deployments.

SENSOR

Optimized for sensor nodes that transmit telemetry data (temperature, humidity, air quality, power levels, etc.) rather than user messages. **SENSOR** nodes:

- Still participate in routing messages for other devices, but prioritize sending their own telemetry data; they can reduce routing/airtime and sleep between readings when `power.is_power_saving` is enabled
- May sleep between sensor readings
- Reduce rebroadcast activity to conserve battery (retransmit while awake only)

Use when: Deploying a battery-powered environmental sensor, weather station, or power monitor. The node's primary job is to report readings, not to relay traffic.

LOST_AND_FOUND

A specialized tracker role for lost-item finders (similar in concept to Apple AirTags or Tile trackers but on the Meshtastic mesh). **Lost_And_Found** nodes:

- Broadcast their location as a message to the default channel regularly to assist with device recovery
- Are designed for small form-factor nodes attached to items (bags, bikes, pets)
- The lost node broadcasts its own location to the default channel; anyone in range receives it and can use it to recover the device

Use when: Building a lost-item tracker on Meshtastic hardware. Not suitable for primary communication or infrastructure roles.

Note on role availability: The exact set of selectable roles (and their spelling) depends on your firmware version. Current roles include **CLIENT**, **CLIENT_MUTE**, **CLIENT_HIDDEN**, **ROUTER**, **ROUTER_LATE**, **REPEATER** (deprecated), **TRACKER**, **SENSOR**, **TAK**, **TAK_TRACKER**, and **LOST_AND_FOUND**. If a role above does not appear in your app's role dropdown, update to current firmware.

Serial Console Enabled

Config key: `security.serial_enabled` (this setting lives under the `security.*` namespace, not `device.*`)

Default: `true`

Controls whether the firmware exposes a serial console on the USB/UART port. When enabled, you can connect to the node via USB serial (commonly at 115200 baud) and interact with the device (view logs, run CLI commands). Disabling this prevents the Serial Console from initializing the Stream API.

Disable when: Deploying a node in an environment where unauthorized USB access is a concern, or to reduce power consumption marginally in deeply embedded deployments. Most users should leave this enabled - it is the primary recovery mechanism if you lose network access to the node.

Warning: If you disable Serial Console and also lose WiFi/Bluetooth access to the node, recovery may require a full firmware reflash.

Debug Log Enabled

Config key: `security.debug_log_api_enabled` (under the `security.*` namespace)

Default: `false`

When enabled, the firmware outputs verbose debug messages to the serial console. These messages include detailed information about packet processing, routing decisions, radio layer events, and subsystem state changes.

Enable when: Troubleshooting connectivity issues, investigating unexpected behavior, or developing integrations. Debug logging produces a high volume of output that can make normal serial console use harder to read.

Disable in production: Debug logging adds slight CPU overhead and can fill serial buffers. Leave disabled on deployed infrastructure nodes unless actively troubleshooting.

Rebroadcast Mode

Config key: `device.rebroadcast_mode`

Default: `ALL`

Controls which received packets this node will rebroadcast (relay) to other nodes. This setting interacts with the Role setting - Role determines the priority and frequency of rebroadcasting; Rebroadcast Mode determines what gets rebroadcast at all.

ALL

The default mode. This node rebroadcasts all received packets that pass the hop limit and deduplication filters, regardless of source or content. This is the correct mode for most nodes.

ALL_SKIP_DECODING

The node rebroadcasts all packets but skips attempting to decode their payload. This mode is primarily useful for REPEATER-role nodes where you want maximum rebroadcast speed and don't need the node to process message content. Slightly reduces CPU overhead per packet.

Use when: Operating a dedicated relay node where content processing is unnecessary and you want to minimize latency and CPU load.

LOCAL_ONLY

The node ignores observed messages from foreign meshes that are open or that it cannot decrypt, and only rebroadcasts traffic on its own local primary/secondary channels. LOCAL_ONLY filters by local channel membership / decryptability - it does *not* filter by hop count.

Use when: You share RF space with neighboring meshes on the same modem preset and want to avoid relaying traffic from foreign meshes you are not part of. LOCAL_ONLY restricts rebroadcasting to your own configured channels.

KNOWN_ONLY

The node only rebroadcasts packets from nodes that appear in its local node database (nodes it has seen NodeInfo from). Unknown nodes - those that have never sent a NodeInfo the local node has received - are not relayed.

Use when: Running a private or semi-private mesh where you want to restrict relay behavior to known network members. This can reduce relay of stray packets from neighboring networks that share the same channel.

Node Info Broadcast Interval

Config key: `device.node_info_broadcast_secs`

Default: `10800` seconds (3 hours). This default is the same for all roles - it is not role-dependent.

How often (in seconds) the node broadcasts a NodeInfo packet - the announcement that includes the node's name, hardware model, and public key. NodeInfo packets allow other nodes to know you exist and update their neighbor lists.

Lower values: More frequent announcements - nodes see each other more quickly after joining the network, and the map updates faster. Costs more airtime.

Higher values: Less frequent announcements - reduces channel utilization overhead, especially important on busy networks. As user-tunable guidance, fixed infrastructure nodes can keep the default or longer intervals (up to UINT MAX) since their presence is stable; the firmware default is 10800 s for all roles.

Minimum: 3600 seconds (1 hour). The firmware enforces 3600 s as the minimum accepted value; lower values such as 300 s are not permitted and will be rejected or clamped.

Double-Tap as Button

Config key: `device.double_tap_as_button_press`

Default: `false`

On devices with an accelerometer (such as the RAK WisBlock with RAK1904 accelerometer module), enabling this option allows a physical double-tap on the device to simulate a button press. Useful for waking a device from sleep or triggering notifications on nodes that don't have physical buttons.

Enable when: Using a device in a case without easy access to physical buttons, or when you want tap-to-wake functionality. Requires compatible hardware with an accelerometer.

Managed Mode

Config key: `security.is_managed` (boolean; set via `meshtastic --set security.is_managed true`).

Managed Mode lives under the `security.*` namespace - it is not a `device.role` value and not `device.is_managed`.

Default: `false`

When Managed Mode is enabled, client applications are blocked from writing configuration to the radio locally (configurations may still be read). Once enabled, radio configurations can only be changed through PKC Remote Admin messages on firmware 2.5+ or the legacy Admin channel on firmware prior to 2.5. This is a security feature for remotely deployed infrastructure nodes.

Enable when: Deploying a ROUTER node (or other infrastructure node) that should be centrally managed and protected from unauthorized local modification. For example, a rooftop node that community members can physically access - Managed Mode prevents casual configuration changes.

Warning: Enabling Managed Mode without first verifying you can administer the node remotely (via PKC Remote Admin on firmware 2.5+, or a configured legacy admin channel on older firmware)

will lock you out of local configuration. Managed Mode blocks config writes over all local interfaces, including USB/serial. Confirm remote admin works before enabling this.

Admin Channel (Legacy) and Remote Admin

Config key: `security.admin_channel_enabled` (boolean, default `false` / disabled). This enables the insecure legacy admin channel. The older integer `admin_channel_index` field is deprecated.

Modern remote admin: `security.admin_key` (PKC Remote Admin keys)

On current firmware (2.5+), administrative access is normally provided by PKC Remote Admin: you authorize one or more public keys via `security.admin_key`, and only messages signed by those keys are accepted for administrative control. The legacy admin channel - enabled by the boolean `security.admin_channel_enabled` - is the older mechanism, where admin messages are protected only by a channel PSK. Admin messages can change device configuration, reboot the node, and perform other privileged operations.

If you must use the legacy admin channel, configure a dedicated secondary channel with its own strong PSK so administrative messages are kept separate from user traffic. For example:

- Channel 0: public mesh channel (LongFast or community key)
- Channel 1: private admin channel (strong PSK known only to node operators)
- Enable `security.admin_channel_enabled`

With this legacy setup, only operators with the channel 1 PSK can remotely reconfigure the node, while anyone on channel 0 can use the mesh normally. Note that the default public channel's PSK (`AQ==`) is publicly known, so never rely on channel 0 for admin.

Best practice for infrastructure nodes: Prefer PKC Remote Admin (`security.admin_key`) on firmware 2.5+ for separating and authenticating admin traffic. If you use the legacy admin channel, always give it a dedicated strong PSK. Never leave admin access on the default public channel in a production deployment.

Revision #7

Created 2026-05-03 05:55:55 UTC by Mesh America Admin

Updated 2026-06-09 02:00:23 UTC by Mesh America Admin