bdts user guide

bdts adds a **Tailscale panel to the System page of the BirdDog PLAY's web UI**, so a device can be signed in to a tailnet — and configured afterwards — from the browser.

Before this, a PLAY patched with Tailscale came up **installed but unauthenticated**, and the only way to finish was to SSH in and run `tailscale up` by hand. That is a real gap rather than a convenience: the PLAY Patcher deliberately bakes **no auth key** into the package it builds — a key in a package downloaded through a browser is a key in cleartext — so every device it produces needs an interactive login it had no way to offer.

> **Before you rely on this:** it is **installed and running on a real BirdDog PLAY**. The System page is patched and served, the web UI survived the restart with its firmware-upload form intact, the panel's API answers off real tailscaled over both the LAN and the tailnet, and the auth gate refuses a missing or forged session. That is **one unit on one firmware**.
>
> Two things are still unproven. **No login has been completed through the panel** — the test unit was already signed in, and testing it would have logged a live node off the tailnet. And **the browser sending its session cookie to the panel's port is reasoned from how cookies are scoped, not measured**; if that turns out to be wrong, writes will refuse from a logged-in browser and the gate needs an explicit password field instead.
>
> This codebase was created with AI assistance, directed and reviewed by a human author.

Signing in

With no auth key, the panel starts an interactive login and shows the sign-in link as soon as tailscaled produces one; **the page then updates itself when the login completes.**

With a key, it connects directly — useful for unattended setup. The panel says plainly that **a key stays reusable until it expires**, which is the fact people forget when they paste one into a build sheet.

Settings, afterwards

Machine name, MagicDNS, accept-subnet-routes, Tailscale SSH, and choosing an exit node.

Changes go through `tailscale set`, not `tailscale up`. Running `up` on a live node **re-runs the whole login flow and silently resets anything not named on the command line** — which is how a

node loses its machine name and its route settings while somebody is only trying to change one thing.

What this hardware cannot do

The PLAY's kernel **has no TUN driver and no loadable modules**, so Tailscale runs in **userspace-networking mode**.

So this node **cannot advertise routes and cannot act as an exit node**, and **the panel says so rather than offering switches that would fail.** Using *someone else's* exit node is unaffected.

Inbound still works: SSH, the web UI and the device's own API are all reachable over the tailnet, because the netstack proxies inbound connections to local listeners.

Why writes need a login when the rest of the device does not

Reads are open. Writes require a valid web-UI session.

That is deliberately stricter than the device's own API, which is unauthenticated. The argument that "it adds no new exposure" does not carry here: **anyone able to reach an ungated Tailscale API could join the device to *their* tailnet and keep remote access to it indefinitely**, or log it out of yours. That is a different kind of thing from changing a frame rate.

The gate needs no second password, because **cookies are scoped by host and not by port** — the session cookie the browser already holds for the device is sent to this panel too.

The trap that makes it more than theatre

Validating the cookie means asking whatever issued it — re-requesting a protected page with the caller's cookie and seeing whether it comes back.

But that only proves something if the page is actually protected. On a device with **no web-UI password**, the unauthenticated request succeeds too, and a naive implementation would **wave everyone through while looking rigorous.**

So every check is **differential**: it probes *without* the cookie as well, and **if that also comes back authenticated it reports the device as unprotected and refuses the write** rather than pretending to have gated it.

`--allow-unprotected` overrides that for a trusted lab network, and **the panel says loudly when it is on.**

Everything fails closed: an unreachable web UI, an unrecognised page, a redirect to the login screen, a missing cookie.

If something is wrong

Symptom	Cause
Writes are refused and I am logged in	Either the device has no web-UI password — set one, or use `--allow-unprotected` on a trusted network — or the cookie is not reaching the panel, which is the one unproven assumption here.
The exit-node switches are missing	This kernel cannot be an exit node. Using another node as one still works.
A setting reverted after a change	Something ran `tailscale up` rather than `set`.
The panel does not appear on the System page	The wrong template was patched, or the firmware serves different markup.
The login link never appears	tailscaled has not produced one yet; the page updates itself when it does.