

Device Configuration Guide

Preparing every device and service on its own end so the dashboard can talk to it
— the checklist to run once, before enabling connectors.

VERSION

1.1.0 Beta 52

UPDATED

August 8, 2026

DO THIS

Once per device

WHAT'S INSIDE

1

Network foundations

Reserved IPs and the address worksheet

3

Audio & RF

Shure receivers, dLive x2, Smaart

5

Streaming platforms

YouTube, Facebook, Church Online

7

Spotify & assistant

Walk-in music and the top-bar assistant

9

Final verification

Test every connector before Sunday

2

Presentation & planning

ProPresenter x3, Planning Center, BoxCast

4

Lighting & broadcast

ETC Ion, HELO, HyperDecks, Ross

6

Power, network & encoder

UPS + switch SNMP, port maps, encoder stats URL

8

QR stickers on gear

Print, place, scan, revoke

1 Network foundations

Every integration depends on one thing: devices that never change address. Reserve a fixed IP (DHCP reservation or static) for each device below, write them down, and enter the same addresses in Admin → Integrations and Admin → Devices.

DEVICE	SUGGESTED ADDRESS	PORT THE DASHBOARD USES
Dashboard host	10.0.5.20	4400 (HTTP)
ProPresenter — GFX 1 / 2 / 3	10.5.70.100 / .101 / .117	1025 (HTTP)
Shure receivers (ULXD 1–4)	10.5.70.22 – .25	2202 (TCP)
dLive — FOH / Monitors	10.0.10.10 / 10.0.10.12	51325 (TCP)
ETC Ion	10.5.70.90	3032 (OSC/TCP)
AJA HELO	10.5.70.105	80 (REST)
HyperDeck 1 / 2	10.0.20.25 / .26	9993 (TCP)
Ross switcher	10.0.20.10	7788 (RossTalk)
Smaart rig	10.0.10.40	26000 (WebSocket API)

! If production gear lives on an isolated VLAN, the dashboard host needs a network interface (or a routed path) into that VLAN and internet access for the cloud APIs. A host with two NICs is the clean answer.

i While you're at it, fill the worksheet into the dashboard: each device's record in Admin → Devices takes the switch/port it's plugged into, the cable label, and the MAC address. That's what turns the Status page's switch port maps into "port 9 → ProgMng Computer" instead of just "port 9 up."

› How often each device is actually checked

Worth knowing before you tune anything in Admin → Alerts, because it sets the fastest the board could possibly notice a change.

STATE OF THE DEVICE	HOW OFTEN IT IS PROBED
Healthy, or newly added	Every 20 seconds — a TCP connect first, an ICMP ping when that fails. Unchanged, and it is what buys sub-20-second detection of a real failure.
Down for 2 minutes or more	Every 60 seconds .
Down for 10 minutes or more	Every 5 minutes .
Any answer at all, at any stage	Straight back to every 20 seconds. There is no gradual recovery and no cool-down — one reply resets it.

The reason is arithmetic rather than caution: a projector unplugged for the summer cost 120 probes an hour to re-learn a fact already known, multiplied by however many devices are off at once. **The cost of backing off is real and is stated rather than hidden** — a device that has been down past ten minutes can take up to five minutes to be seen coming back, so the Status page marks those rows with a “**checked every 5 min**” badge and prints “**not checked**” in the latency column rather than a figure. None of this is configurable, and nothing about alerting changed: the down window in Admin → Alerts (default 60 seconds) still decides when anyone is told.

- i **Switch port readings are cached briefly and shared.** Reading one switch's whole port table is a large number of SNMP round trips, and when a switch dies every device behind it crosses the down window in the same instant — so eight diagnoses used to start eight identical walks of the same switch within a millisecond of each other. A switch is now asked **at most once every 90 seconds**, requests that arrive together are collapsed into one, and the port-to-index mapping (which only changes when somebody re-cables) is held for hours. The visible consequence is that the port map panel states its own age — “**Cached reading — as of 40s ago**” — instead of looking live. That is disclosed at the point of use deliberately; it is not the switch being slow to answer.

2 Presentation and planning

› ProPresenter — on all three GFX machines

- 1 Open Settings → Network and check **Enable Network**. Leave the port at 1025. Do this on GFX 1 (Side Screens), GFX 2 (Video Wall), and GFX 3 (411 Room).
- 2 On GFX 1, create a Message named **Stage** (Messages panel → +) with a single text token — this is what dashboard stage messages fire. For parent paging, also create a Message named **Parent Page** (one text token) — paged codes appear through it on the side screens.
- 3 Name presentations to match Planning Center item titles (or add explicit pairs in Admin → PP ↔ PCO Links) so auto-advance fires reliably.

› Planning Center Services

- 1 Sign in at api.planningcenteronline.com/oauth/applications with an account that can view Services and control LIVE.
- 2 Personal Access Tokens → New Token → name it TechArts Dashboard. Copy the Application ID and Secret.
- 3 Open your Sunday service type in Services — the number after `/service_types/` in the address bar is the Service Type ID.
- 4 On Sundays, **operate PCO LIVE** — it drives the service clock and the report. No LIVE, no report (by design; see Admin → General → Service reports).

i **Improved in Beta 39 — reading the plan is far lighter on Planning Center than it was.** Fetching the item notes used to cost **one request per plan item** plus one for the category list: 21 requests for a twenty-item plan, every time the cache expired. It is now a **single request** that returns every item and every note together, and it returns identical data — the note text, its category and its ordering are all unchanged, and so is everything the dashboard shows. Nothing needs configuring; there is no setting for it. **Stated honestly:** this was measured against a real twenty-item plan and it is a large reduction, but it has not been proven to be the whole cause of the "it times out when both instances are running" report — the once-a-second plan poll is separate load and has not been counted. If you still see timeouts, say so rather than assuming this fixed it.

› BoxCast

- 1 BoxCast dashboard → Settings → Developers → create an API client; copy Client ID and Secret.
- 2 Grab the Channel ID from your channel's embed code (after `boxcast.tv/channel/`).
- 3 Nothing to configure on the encoder — the dashboard watches broadcast state through the cloud API. BoxCast only drives the service clock as a fallback when Planning Center isn't.

3 Audio and RF

› Shure receivers (ULXD, Axient Digital)

- 1 Give each receiver a reserved IP (you already have this if Wireless Workbench connects).
- 2 No setting to enable — command strings on TCP 2202 are always on. WWB and the dashboard can watch the same receivers at the same time (the dashboard uses one polite control session per receiver).
- 3 In the connector, list each receiver with a unit name (ULXD 1...4, AXIENT, PSM900) and channel count. Mic Assignments match on unit + channel.
- 4 Name channels in Wireless Workbench (or on the receiver front panel) — those names flow straight through to every dashboard screen automatically.

i **Step 4 is the one people skip, and it is the one you see.** The connector is enabled and pointed at all four ULXD4Q receivers (`10.5.70.22 - .25`), and the dashboard prefers the receiver's programmed Channel Name over the label in Mic Assignments. So a pack showing a generic name (`Mic 1 ... Mic 16`), the instrument packs `AG 1 / EG 1 / Bass Pack` , `Wireless 1 ... Wireless 4`) means one of two things at the hardware — and battery, RF and the name all arrive in the same poll, so the Mics page tells them apart in one glance. **Battery and RF showing, names generic** → the receivers are answering; no Channel Name is programmed on those channels. Set them in WWB. **Battery and RF blank as well** → the receivers aren't answering the dashboard host on port 2202 at all; fix the network path, the firewall, or network control on the receiver, and the names follow. The Wireless widget names any receiver that failed to connect on a "not reachable — check TCP 2202" chip.

› Allen & Heath dLive (FOH + Monitors)

- 1 MixRack → Utility → Control → Network: set the fixed IP. The MIDI-over-TCP port is 51325. The console must be running **MIDI over TCP/IP Protocol V2.0 or newer** — the dashboard's Test button now does a real channel-name round-trip, so a pass confirms the protocol (not just that the port is open); a console that answers TCP but stays silent gets a warning to check the firmware/protocol.
- 2 In Admin → Mic Assignments, fill the dLive channel column — that's where each person's fader label lands. Labels sync to both consoles on every Push live, as long as "Auto-label faders" is on for the dLive connector.

› Smaart — the real WebSocket API

The dashboard drives Smaart over its **WebSocket API (v3)** — the same protocol as the Smaart WebSocket API Client, not a web page or REST URL.

- 1 On the Smaart machine: **Options** → **API** → enable the API server. Note the **port** it shows (26000 in Rational's examples; use yours).
- 2 Run a **calibrated SPL input** in Smaart for the SPL meter (the API only exposes SPL for active calibrated inputs). For the live **RTA**, keep a **spectrum measurement active** too — the dashboard streams its frequency data into the RTA bars.

- 3 In Admin → Integrations → Smaart, enter the IP + API port, choose the SPL metric (default **SPL A Slow**) and RTA banding (default 1/3 octave), and press **Test connection**. It reports your exact Smaart product + version, whether it's supported, a live SPL value, and whether an RTA measurement is active. The meter shows the full metric set Smaart reports (SPL A/C, LAeq/LCeq/Leq, Peak).
- 4 **RTA scale (0.6.6)**. The RTA now draws Smaart's full **1/3-octave detail** (about 30 bars instead of 10) and refreshes several times a second, with a peak-hold cap above each bar. If the bars sit too low or peg at the top, tune **RTA scale floor / ceiling (dBFS)** in the same Smaart panel — they default to -70 and -20, which suits typical programme level.
- 5 If the rig is a separate PC, make sure its firewall allows inbound on the API port from the dashboard host.

i **Supported (API v3)**: full Smaart **v8.3.0 – v9.0.1**, Smaart Di 2.1+, Smaart SPL. SPL streaming works on **full Smaart** and **Smaart SPL** (Smaart Di has no calibrated inputs). Smaart 7 (v1) and 8.0–8.2 (v2) use an older API this connector doesn't support — update Smaart if you're on one.

4 Lighting and broadcast

› ETC Ion

Console Shell (Exit Eos → Settings) → Network → enable OSC RX and TX over TCP, framing OSC 1.0 (packet-length), port 3032. The dashboard is **read-only** — it subscribes and previews active + pending cues; it never fires anything.

› AJA HELO

Set a fixed IP in the HELO's web UI. No key needed — the dashboard reads stream and record state over the built-in REST interface, and the HELO Log widget tails the device's own [logwatch](#) page for the paper trail.

! Get the IP right — until 1.1.0 Beta 45 nothing would have told you it was wrong. The HELO connector's test discarded whatever error it got and answered "Connected" regardless, so the Pre-Service Readiness HELO row passed with the encoder unplugged, unreachable, or with the IP field left empty. As of Beta 45 the test reports what actually happened and names the address it tried, so a wrong or blank IP shows up as a red row instead of a green one. If that row goes red on your first run after updating, check the encoder before you doubt the dashboard: the reading is new, the fault may not be.

› Blackmagic HyperDecks

Fixed IP per deck (Setup → Ethernet). The text protocol on TCP 9993 is always on; transport, timecode, and loaded clip appear in Broadcast Gear.

› Ross switcher

Enable RossTalk in the switcher's network settings (port 7788). The dashboard verifies the link; deeper tally/status can be added later via Ross Dashboard/ogScript if you want it.

› OBS Studio

OBS → Tools → WebSocket Server Settings → enable, note the port (4455) and password. The dashboard reads current scene, stream/record state, and dropped frames.

5 Streaming platforms

PLATFORM	ON THEIR END
YouTube	Google Cloud Console → new project → enable YouTube Data API v3 → create an API key (restrict it to that API). Channel ID is in YouTube Studio → Settings → Channel → Advanced.
Facebook	developers.facebook.com → Business app on the church page → Page access token with pages_read_engagement → extend to a long-lived token (Token Debugger → Extend). Re-generate when it eventually expires.
Church Online	No public stats API — stand up any HTTPS endpoint returning attendance as JSON (Apps Script or a tiny Lambda reading your ChOP admin metrics), and paste the public + host chat embed URLs for the read-only chat widget.

6 Power, network & encoder telemetry

› UPS network cards (SNMP)

Works with any UPS whose network card speaks SNMP — APC/Schneider (NMC), CyberPower (RMCARD), Eaton, Tripp Lite, Vertiv. The dashboard reads the standard UPS-MIB and falls back to APC's PowerNet MIB automatically.

- 1 Give each UPS card a reserved IP.
- 2 In the card's web UI, enable SNMPv1/v2c with a read-only community string (default `public`; pick your own and note it). APC: Configuration → Network → SNMPv1 → Access. CyberPower: Network → Services → SNMP.
- 3 Admin → Integrations → UPS & Power (SNMP) → enable, then list each unit:
`[{"label": "F0H Rack UPS", "host": "10.0.60.11", "community": "public"}]` → Test connection.
- 4 Set the runtime warning in Admin → Alerts (default 10 minutes). ON BATTERY always alerts immediately.

› Managed switches (SNMP) — and the port maps

- 1 Enable SNMP read access on each production switch (Cisco: `snmp-server community <string> RO` · Netgear/Ubiquiti/Aruba: Settings → Services → SNMP).
- 2 In Admin → Devices, check **SNMP** and **Port map** on each switch's  record, and fill the **Switch / port** field on every device plugged into it (e.g. `300-SW1 p9`). Clicking the switch on the Status page then shows every physical port: link up/down, the port's own name and the admin alias set on the switch, PoE delivering or not, and the device connected — matched from the switch's own MAC table plus those "Switch / port" labels.
- 3 That is the whole setup. Each switch is an ordinary monitored device, so it is pinged like everything else: if it goes down you get one alert with the auto-diagnosis attached, and anything behind it is named as sitting on that switch. There is no separate switch integration to configure — the **Network Switches (SNMP)** connector was removed in v1.1.0 Beta 18 because the port map above reads a superset of what it showed.
- 4 Then confirm it. **Port map** is already ticked on 7 devices in the live configuration, but no port map has ever been opened against a real switch here — nothing in the build environment can reach one. Click each switch on the Status page and check a port table actually comes back; if it doesn't, the error message names the failing step (community string, UDP 161, SNMP not enabled).

i **Not covered yet: chassis PoE budget in watts.** Per-port PoE ("is this port delivering power?") is in the port map; total budget used against the chassis maximum is not read by anything today. The `PoE budget warning (%)` setting under Admin → Thresholds is deliberately still there for the follow-up item — set it if you like, but nothing acts on it yet.

! SNMP is UDP port 161. If the dashboard host sits on a different VLAN, make sure UDP 161 is routed/allowed to the UPS cards and switch management addresses.

› Encoder telemetry (any JSON stats URL)

- 1 Find the stats endpoint: Teradek and Epiphan expose HTTP JSON stats; for OBS use the OBS connector instead; for anything else a 10-line script that writes JSON next to the encoder works.

- 2 Admin → Integrations → Encoder Telemetry → enter the URL and the JSON field paths (dot notation, e.g. `output.bitrate_kbps`). Only the bitrate path is required. A backup encoder (Resi or a spare) can be watched too — the dashboard alerts when the safety net goes down.
- 3 Set the bitrate floor in Alerts (default 2,500 kbps). Below it, the Encoder Health widget flips to DEGRADED and an alert fires.

7 Spotify & assistant

› Spotify (walk-in music)

- 1 Sign in at `developer.spotify.com/dashboard` with the church's Spotify account (Premium required for playback control) and create an app.
- 2 In the app's settings, add this Redirect URI **exactly**: `http://127.0.0.1:4400/api/spotify/callback`. As of November 2025 Spotify only accepts HTTPS or a **loopback literal** — a LAN IP like `10.5.70.x` and even the word `localhost` are rejected with "Insecure redirect URI." Use the `127.0.0.1` form shown here (this is also the exact string Admin → Integrations → Spotify now displays).
- 3 Paste the Client ID and Secret into the dashboard and wait a beat for the draft to save. Then do the one-time **Connect Spotify** from the server machine itself, with the dashboard open at `http://127.0.0.1:4400/admin` — not from a LAN IP. If you're working remotely, tunnel first: `ssh -L 4400:localhost:4400 <host>`, then open `http://127.0.0.1:4400/admin` on your own machine. Approve, then Test connection → Push live. (After connecting once, the widget works normally for everyone on the LAN; only the Connect step must be done over loopback.)
- 4 Playback control targets whatever device is playing (the lobby Mac, a rack Mac mini). Start Spotify on that machine once so it appears as a device.

› Tech Arts Assistant (the top-bar search)

Optional: create an API key at `console.anthropic.com` and paste it in Admin → Integrations → Tech Arts Assistant. Without a key the assistant still answers common questions from live data.

As of 0.6.6 the assistant is no longer a dashboard widget — it lives in the **search box in the top bar** (or press `⌘K` / `Ctrl-K`), so it works from every page instead of only where someone had placed the panel. Type a question and pick the "Ask" row.

8 QR stickers on gear

A sticker goes on a piece of equipment. Somebody standing in front of it on a Sunday morning scans it with their phone — a phone that has never signed in to anything — and reports that it is broken. The incident that lands in the log **names the gear**, because the sticker carried the identity. Nobody has to type "the amp in the amp room, the middle one, I think", and nobody has to find a volunteer who knows what it is called.

Managing stickers is an **Admin** job and lives on its own page at `/stickers` — link-only, not in the navigation. Scanning one is not; that is the whole point.

› Print and place

- 1 Add the device in **Admin** → **Devices** first, with the name and room you want printed. The sticker takes a copy of both at the moment it is created, because paper cannot be updated later.
- 2 Open `/stickers`, pick the device from the list, and create the sticker. One active sticker per device is the normal case; a second is allowed if a big rack needs a copy at each end, and the page tells you when you already have one.
- 3 Press **Print** for a sheet of the stickers you have selected — or of every active one if you select nothing. Cut them out and stick them where somebody standing in front of the gear will see them, not on the back of a rack.
- 4 Scan one yourself with your own phone before you walk away, and check the page names the right device.

✓ **The page tells you when the wall and the record have drifted apart.** Rename a device in Admin, or move it to a different room, and every sticker printed before that says so on the `/stickers` page — "the device has been renamed to X since this sticker was printed" — with the printed wording quoted back at you. Reprint it when you see that. A sticker whose device has been deleted outright is called out too, and reports from it still land under the name printed on it.

› What a scan can and cannot do

Each sticker carries its own code, and that code **is** the credential — anyone who photographs the sticker holds it. That is deliberate, because the person holding the phone is a volunteer with no account, and it is why the code is made worth almost nothing to steal:

- **It can do exactly one thing:** file one problem report against the one device it was printed for. It cannot read the incident list, cannot see whether anything is up or down, cannot see an address or a port, cannot reach any other page, and cannot name a different device even if somebody tries.
- **It is rate-limited.** A handful of reports from one sticker in ten minutes and the next one is refused, with a sentence saying so and telling the person to find a member of the team. A photographed sticker is a nuisance for a few minutes rather than a way to bury a service report under noise.
- **It is revocable on its own.** Peel that sticker off, revoke that one code on `/stickers`, print a new one. Nothing else in the building is affected.
- **Everything it files is marked as coming from a sticker**, so a flood is traceable to one sticker and one revocation ends it.

What a scan does reveal, said plainly rather than left out: it answers with the device's name and room — the two things already printed on the sticker in the holder's hand — and it confirms that the code is live. Nothing else. A code that has been revoked, and a code that never existed, get exactly the same answer, so nobody can probe the difference.

i **Stickers do not expire, deliberately.** A guest cue-sheet link dies after a day because it is "here is Sunday's sheet". A sticker is glued to a rack for two years, and a code that quietly died would turn every sticker in the building into a dead end that nobody notices until the morning somebody needs it. **Revoking is the control here, not time** — which means an unrevoked sticker in a skip is still a working sticker. Revoke before you throw gear away.

! **Managing stickers needs a real Admin sign-in, even with access control switched off.** This changed in Beta 38 and it is worth knowing before you go looking for the page. The list shows every sticker's code — that is the whole set of wall-mounted credentials in one screen — so it is not something the network at large gets to read. Before this release, with the access switch off, those four actions answered anybody. Now they want a signed-in administrator or an Admin-level approved machine, switch on or switch off.

› Every QR code this dashboard ever printed was unscannable

This is worth saying out loud because somebody has almost certainly blamed their phone for it. Until Beta 38, **no QR code the dashboard produced could be decoded by anything** — not by a phone, not by a scanner app, not by anything. The codes looked perfect: correct shape, correct corner squares, a plausible-looking speckle of black and white. Three separate faults in the code generator all had to be fixed before one would read, and nothing had ever successfully scanned one, so nothing ever reported it.

This was never limited to stickers. **The volunteer board and the event screens print QR codes too**, and theirs were equally dead. If anyone has ever held a phone up to a code on one of those screens and concluded their camera was at fault, they now have an answer. Reprint anything with a QR code on it that is still in circulation.

9 Final verification

- 1 Admin → Integrations: enable each connector, enter the addresses from your worksheet, press **Test connection** on every card. Green checks across the board.
- 2 Admin → Devices: confirm every IP matches the worksheet — and fill each 📄 record (switch/port, cable label, MAC) — then watch the Status page go green. Click a switch and confirm the port map lines up with reality.
- 3 Press **Push live**, switch Mode to Live, and run a weeknight test: fire a linked presentation in ProPresenter and watch PCO advance; send a stage message and see it hit the ProPresenter stage screens; page a test parent code from `/paging`.
- 4 Print one QR sticker from `/stickers`, scan it with a phone that is **not** signed in to anything, and file a test report. Confirm the incident that arrives names the right device, then delete it. This is the only way to know a sticker works — and until Beta 38 no QR code this dashboard printed could be scanned at all, so it is worth doing once deliberately.

This document references dashboard version 1.1.0 Beta 52. What changed in each production release lives in the separate **Version History** document — beta releases included.