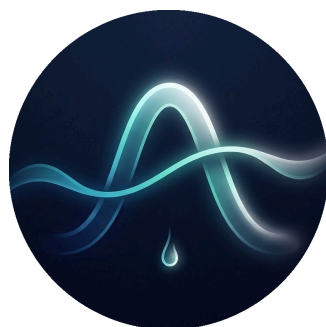




AetherSDR

User Manual

Native. Open. Yours.

Version main @ d833725c6

Built 8 October 2026

<https://docs.aethersdr.com>

AetherSDR User Manual

Version main @ d833725c6, built 8 October 2026.

This manual is generated from the online documentation at <https://docs.aethersdr.com>, which is always the most current version. The online pages are searchable and link to each other directly; in this book a cross-reference prints the number of the page it refers to, and a web address is printed in a footnote.

AetherSDR is free software, licensed under the GNU General Public License v3.0. Source code, releases and issue tracking:
<https://github.com/aethersdr/AetherSDR>.

FlexRadio, SmartSDR and other product names are trademarks of their respective owners. AetherSDR is an independent project and is not affiliated with or endorsed by FlexRadio Systems.

Contents

1	Introduction	1
2	Start Here	5
2.1	Your First Session	5
2.2	Installation	8
2.3	First Connection	11
2.4	Before You Transmit	14
3	Your Radio	18
3.1	Supported Radios	18
3.2	FlexRadio	20
3.3	Hermes-Lite 2	23
3.4	Networked Icom	28
3.5	ANAN-G2	32
3.6	RTL-SDR	34
3.7	KiwiSDR and Web-888	36
3.8	Demo Mode	39
4	Your Computer	43
4.1	Linux	43
4.2	macOS	47
4.3	Windows	50
5	Operating	54
5.1	Panadapter Controls	54
5.2	VFO Widget	61
5.3	RX Controls	64
5.4	TX Controls	68
5.5	Meters	72
5.6	Aetherial Audio	74
5.7	Multi-Slice Operation	83
5.8	Split Operation	85
5.9	Diversity and ESC	87
5.10	TNF (Tracking Notch Filters)	90
5.11	Memory Channels	92
5.12	Profile Management	94
5.13	XVTR (Transverters)	97
5.14	Workspace Canvas	98
6	CW & Digital Modes	102
6.1	CWX Panel	102

6.2	CW Decoder	104
6.3	DVK Panel	107
6.4	RTTY Operation	110
6.5	RADE Digital Voice	113
6.6	D-STAR (ThumbDV)	115
6.7	AetherModem Packet Radio	118
6.8	Copy Assist	121
6.9	WSJT-X Integration	125
7	Connecting Other Software	128
7.1	DAX Virtual Audio	128
7.2	DAX IQ Streaming	130
7.3	CAT Control	132
7.4	TCI Server	135
7.5	MQTT Station Automation	141
7.6	Automation Bridge and MCP	144
8	Spotting & Tools	148
8.1	SpotHub	148
8.2	PSK Reporter Map	155
8.3	AetherSweep	158
8.4	Net Scheduler	160
8.5	AetherClock and GPS	161
8.6	Callsign Lookup	163
9	Station Accessories	166
9.1	Peripherals	166
9.2	Amplifiers	170
9.3	TGXL Tuner Control	173
9.4	ShackSwitch	175
9.5	Green Heron Everyware	176
9.6	USB Cable Management	178
10	Controllers	181
10.1	StreamDeck	181
10.2	FlexControl Tuning Knob	183
10.3	Ulanzi Dial	186
10.4	USB Control Surfaces	189
10.5	MIDI Controller Mapping	192
10.6	CTR2 Proxy	196
11	Remote Operation	199
11.1	SmartLink Setup	199
11.2	Manual Connection	201
11.3	Tailscale Remote Access	203
11.4	Low Bandwidth Connections	208
11.5	Multi-Flex	211
12	Settings	214
12.1	Radio Setup	214
12.2	Audio Settings	217
12.3	Settings and Backups	220

12.4	Keyboard Shortcuts	224
12.5	Themes and Theme Editor	228
12.6	Slice Colors	230
12.7	Accessibility	231
12.8	Firmware Update	232
13	Reference	235
13.1	Menu Reference	235
	Keyboard Shortcuts	240
14	Understanding	241
14.1	DSP Noise Mitigation	241
14.2	BNR GPU Noise Removal	248
14.3	NR2 Noise Reduction	250
14.4	GPU Rendering	252
15	Help	256
15.1	Troubleshooting	256
15.2	Support and Logging	261
15.3	Runtime Monitor	266
16	Contributing	269
16.1	Contributing Guide	269
16.2	Docs Style Guide	271
16.3	Translating the Docs	274
16.4	AI-Assisted Development	275
16.5	Building from Source	277
17	Generated Reference	280
17.1	Generated Reference	280
17.2	Default Keyboard Shortcuts	280
17.3	MIDI Controller Actions	284
17.4	FlexControl Actions	288
17.5	Ulanzi Dial Actions	289
17.6	Stream Deck+ and HID Controller Actions	290
17.7	Log Categories	293
17.8	TCI Commands	294

CHAPTER 1

Introduction

Native. Open. Yours.

AetherSDR is an open-source client for **FlexRadio** transceivers with native builds for Linux, macOS and Windows, including an aarch64 build for the Raspberry Pi. FlexRadio is the supported target; the Hermes-Lite 2, networked Icom radios, the ANAN-G2 and RTL-SDR dongles are also supported at an earlier stage, each labelled with its status. No radio? Demo Mode ([p. 39](#)) runs the full interface against a built-in simulator.

These docs come in two versions, chosen from the version menu at the top of every page. **Stable**, the default, describes the latest release. **Next (main)** tracks `main`: it describes what has landed since that release and will ship in the next one. Release notes for every version are in the `CHANGELOG`¹.

Prefer paper? Download the printable manual² (PDF).

Where are you starting from?

If you...	Start with
Are brand new, or have no radio connected yet	Your First Session (p. 5), then Installation (p. 8)
Own a FlexRadio	FlexRadio (p. 20) → First Connection (p. 11) → Before You Transmit (p. 14)
Own a Hermes-Lite 2, Icom, ANAN-G2 or RTL-SDR dongle	Supported Radios (p. 18), then your radio's page
Run Linux, macOS or Windows and want the platform specifics	Linux (p. 43) · macOS (p. 47) · Windows (p. 50)
Want FT8, RTTY or other digital modes	WSJT-X Integration (p. 125), DAX Virtual Audio (p. 128), CAT Control (p. 132), TCI Server (p. 135)
Operate remotely	SmartLink Setup (p. 199), Tailscale Remote Access (p. 203), Manual Connection (p. 201)
Have a problem	Troubleshooting (p. 256), then Support and Logging (p. 261) to report it
Want to help with the code or docs	Contributing Guide (p. 269), Docs Style Guide (p. 271)

Start Here

- Your First Session ([p. 5](#)) — Hands-on tutorial with no radio needed: tune, filter, reduce noise and save a memory in Demo Mode
- Installation ([p. 8](#)) — Download and install on Linux, macOS (14.4+) or Windows
- First Connection ([p. 11](#)) — Connect to your radio for the first time

¹<https://github.com/aethersdr/AetherSDR/blob/main/CHANGELOG.md>

²<https://docs.aethersdr.com/AetherSDR-Manual.pdf>

- Before You Transmit (p. 14) — Checklist before your first transmission: TX slice, callsign, mic, power, amplifier, keying paths

Your Radio

- Supported Radios (p. 18) — FlexRadio models and firmware, other radio families' status, and what works where
- FlexRadio (p. 20) — Start here with a FlexRadio: models, firmware, setup order and every Flex feature page
- Hermes-Lite 2 (p. 23) — Experimental: setup, hardware, calibration, receive, transmit and limits
- Networked Icom (p. 28) — IC-7300MK2 (supported), IC-705, IC-9700 over RS-BA1: connecting, PC Audio, CWK, XFC
- ANAN-G2 (p. 32) — Experimental, receive-only: connect options, droop correction, what works
- RTL-SDR (p. 34) — Experimental, receive-only USB dongles: availability, modes, receiver corrections
- KiwiSDR and Web-888 (p. 36) — Public KiwiSDR/Web-888 receivers as receive-only slice antennas, with Receive Sync
- Demo Mode (p. 39) — Explore the full UI with no radio: built-in simulator, noise and fault injection

Your Computer

- Linux (p. 43) — AppImage install, Wayland/xcb, PipeWire, udev rules, file locations and Linux-only fixes
- macOS (p. 47) — DMG install, DAX driver, Input Monitoring, MNR, file locations and macOS-only fixes
- Windows (p. 50) — Installer or portable, TCI instead of DAX, Direct3D, file locations and Windows-only fixes

Operating

- Panadapter Controls (p. 54) — Spectrum, waterfall, 3D view, Display panel, Mini-Pan, Center Lock
- VFO Widget (p. 61) — Frequency display, flag tabs, adaptive RX filter, slice management
- RX Controls (p. 64) — Filters, AGC, squelch, mute, RIT/XIT
- TX Controls (p. 68) — Power, ATU pre-tune, MOX, TUNE, TX filter warnings, Quindar tones
- Meters (p. 72) — S-Meter, SmartMTR, cross-needle PWR, Radio Vitals, Antenna Health
- Aetherial Audio (p. 74) — The AetherTX and AetherRX audio chains: gate, EQ, compressor, de-esser, tube, exciter, reverb, limiter, presets
- Multi-Slice Operation (p. 83) — Working with multiple slices, Slice Link
- Split Operation (p. 85) — SPLIT/SWAP, Split Up offsets, Monitor TX, remembered audio, Split QSY closing
- Diversity and ESC (p. 87) — Diversity mode and Enhanced Signal Clarity
- TNF (Tracking Notch Filters) (p. 90) — Creating and managing notch filters
- Memory Channels (p. 92) — Saving, recalling, importing and editing memories
- Profile Management (p. 94) — Global, TX and Mic profiles; import and export
- XVTR (Transverters) (p. 97) — Transverter setup and operation
- Workspace Canvas (p. 98) — Experimental free layout: placed pans and applets, named workspaces, canvas windows

CW & Digital Modes

- CWX Panel (p. 102) — Keyboard-to-CW keying (CWK on Icom)
- CW Decoder (p. 104) — Real-time Morse decode of received and transmitted CW
- DVK Panel (p. 107) — Digital Voice Keyer: 12-slot recording and playback for contesting
- RTTY Operation (p. 110) — Built-in RTTY decoder, Mark/Space lines, external software
- RADE Digital Voice (p. 113) — FreeDV RADE AI digital voice, sync/SNR, FreeDV Reporter
- D-STAR (ThumbDV) (p. 115) — Local D-STAR digital voice using a PC-attached ThumbDV/DV3000U

- AetherModem Packet Radio (p. 118) — Built-in AX.25: APRS, WIDE1-1 digipeater, KISS TNC, BBS terminal and mailbox
- Copy Assist (p. 121) — On-device speech-to-text for received voice (whisper.cpp)
- WSJT-X Integration (p. 125) — FT8/FT4 over TCI, or CAT + DAX

Connecting Other Software

- DAX Virtual Audio (p. 128) — Virtual audio channels for digital mode apps
- DAX IQ Streaming (p. 130) — Raw I/Q streaming at 24–192 kHz
- CAT Control (p. 132) — The CAT applet: Rigctld, TS-2000 and Flex dialects on up to 8 ports
- TCI Server (p. 135) — TCI over WebSocket: CAT, audio, IQ, CW and spots
- MQTT Station Automation (p. 141) — Control rotators, switches and Node-RED flows over MQTT
- Automation Bridge and MCP (p. 144) — Local automation bridge and MCP server: token, observe-only mode, gated transmit

Spotting & Tools

- SpotHub (p. 148) — DX Cluster, RBN, WSJT-X, POTA, FreeDV Reporter, N1MM, EiBi, SpotCollector, Smart Spot Filtering
- PSK Reporter Map (p. 155) — Who hears you: 2D map or 3D globe, weather radar, night lights
- AetherSweep (p. 158) — In-panadapter SWR analyzer with CSV export
- Net Scheduler (p. 160) — Recurring net reminders with one-click Tune Now
- AetherClock and GPS (p. 161) — WWV/WWVB time-signal decoder and the GPS & Station Location dashboard
- Callsign Lookup (p. 163) — QRZ.com lookups, cache and CW-decoder contact cards

Station Accessories

- Peripherals (p. 166) — Add, remove and auto-connect station accessories; 4O3A access codes; PGXL and Antenna Genius
- Amplifiers (p. 170) — ACOM, SPE Expert, VK3AMP and KPA1500 amplifiers; LP-100A wattmeter
- TGXL Tuner Control (p. 173) — 4O3A Tuner Genius XL: direct or relayed, autotune, antenna switch
- ShackSwitch (p. 175) — Network antenna switch with dummy-load protection
- Green Heron Everywhere (p. 176) — Green Heron Everywhere antenna switch and rotator control
- USB Cable Management (p. 178) — CAT, BCD, Bit, LDPA and Passthrough cables on the radio's USB ports

Controllers

- StreamDeck (p. 181) — Native Stream Deck+ over USB; other models via TCI or the automation bridge
- FlexControl Tuning Knob (p. 183) — FlexControl knob and the on-screen AetherControl
- Ulanzi Dial (p. 186) — Ulanzi Dial knob and keys: wheel actions, key mapping, PTT and CW keying
- USB Control Surfaces (p. 189) — RC-28, PowerMate, Contour Shuttle, Stream Deck+ and serial PTT/CW interfaces
- MIDI Controller Mapping (p. 192) — Map any MIDI controller, with Learn mode and profile import/export
- CTR2 Proxy (p. 196) — Relay a CTR2-Max controller to the radio over Wi-Fi (USB pending CTR2 firmware)

Remote Operation

- SmartLink Setup (p. 199) — Operate your radio over the internet
- Manual Connection (p. 201) — Connect by IP across routed networks, VPN source path, diagnostics

- Tailscale Remote Access (p. 203) — Remote operation behind CGNAT through a container in the radio (FLEX-8000/Aurora), on Tailscale’s free plan
- Low Bandwidth Connections (p. 208) — Reduce traffic for VPN, LTE and metered links
- Multi-Flex (p. 211) — Operating alongside SmartSDR or Maestro

Settings

- Radio Setup (p. 214) — The searchable settings tree, page by page
- Audio Settings (p. 217) — Line out, headphones, PC audio devices
- Settings and Backups (p. 220) — Where settings live, keychain credentials, backups, `--config`, resetting properly
- Keyboard Shortcuts (p. 224) — Operating shortcuts (off by default), PTT hold, editor, Minimal Mode
- Themes and Theme Editor (p. 228) — Switch themes live, edit tokens, import and export `.aethertheme` files
- Slice Colors (p. 230) — Per-slice colours for VFOs, panadapter overlays and meters
- Accessibility (p. 231) — Screen readers, keyboard navigation, announced reasons for dimmed controls
- Firmware Update (p. 232) — Upload FlexRadio firmware from AetherSDR

Reference

- Menu Reference (p. 235) — Every menu item, the title bar, radio tabs and the About window
- Keyboard Shortcuts (p. 224) — Operating shortcuts (off by default), PTT hold, editor, Minimal Mode

Understanding

- DSP Noise Mitigation (p. 241) — All seven client-side noise-reduction engines plus the radio’s own
- BNR GPU Noise Removal (p. 248) — In-process NVIDIA Maxine denoising on a local RTX/GeForce GPU
- NR2 Noise Reduction (p. 250) — Client-side spectral noise reduction parameter reference
- GPU Rendering (p. 252) — QRhi spectrum and waterfall on GPU (OpenGL/Metal/D3D11), GPU selection

Help

- Troubleshooting (p. 256) — Common issues and solutions
- Support and Logging (p. 261) — Help → Support & Diagnostics..., log categories, support bundles, crash reports
- Runtime Monitor (p. 266) — CPU, threads and memory; Radio Health; Network Diagnostics

Contributing

- Contributing Guide (p. 269) — How to contribute code and features
- Docs Style Guide (p. 271) — How pages are laid out and written: page types, skeleton, status, Known issues
- Translating the Docs (p. 274) — Adding and maintaining a translation of these docs
- AI-Assisted Development (p. 275) — Contributing with AI tools; AGENTS.md
- Building from Source (p. 277) — Compile AetherSDR yourself (Qt 6.12)

Start Here

2.1 Your First Session

This tutorial walks you through AetherSDR for the first time using the built-in demo, so you need no radio, no antenna and no licence. In about 20 minutes you will connect, tune, change mode and filter, add a panadapter, clean up a noisy band, change the display, save a memory and disconnect. Every step tells you what you should see, so you know it worked.

The demo is a simulator that generates its own receive audio and spectrum. **It cannot transmit**, so nothing you do here reaches the air. It also has fewer features than a real radio: one slice, USB and LSB audio, and no DAX. See Demo Mode (p. 39) for everything it offers.

2.1.1 Requirements

- AetherSDR installed on Linux, macOS or Windows. See Installation (p. 8).
- Speakers or headphones on your computer.

2.1.2 Step 1: Start AetherSDR

1. Start AetherSDR.

You should see the **Connect to Radio** window. If it does not open (for example because AetherSDR connected to a radio you used before), choose **File** → **Connect to Radio...** On Windows and Linux the menus are behind the ≡ button at the left of the title bar; on macOS they are in the system menu bar.

2.1.3 Step 2: Connect to the demo

1. In the Connect to Radio window, choose **On This Network**.
2. Under **Available radios**, find **AetherSDR Demo**, labelled “**Simulator (not on the air)**”. It always sorts below any real radios.
3. Select it and click **Connect Selected Radio**.

You should see the main window come to life:

- a **panadapter**: the spectrum on top and the waterfall scrolling below;
- a **VFO** for slice **A** on **14.100 MHz** in **USB**, with a 100–2900 Hz filter;
- a tab for the demo in the title bar;
- the **Demo Noise** applet in the applet panel, with **Pink / hiss** and **Birdie carrier** lit.

You should hear a soft hiss with a steady tone in it: the band noise and the birdie carrier.

If the demo is not in the list, tick **Show the AetherSDR demo simulator** in the Connect to Radio window.

The demo panadapter is a narrow 8 kHz window that follows your frequency. A real radio shows much wider spans.

2.1.4 Step 3: Tune

1. Point at the spectrum and turn the mouse wheel one notch.

You should see the frequency change by **100 Hz**, the slice's step size.

2. In the **RX** applet, double-click the frequency, type **14.074** (a frequency in MHz) and press Enter.

You should see the VFO move to **14.074 MHz**.

You can also double-click anywhere on the spectrum or waterfall to tune there. See Panadapter Controls (p. 54) for every mouse action.

2.1.5 Step 4: Change the mode

1. On the VFO, open the mode list (it shows **USB**) and choose **LSB**.

You should see the mode read **LSB**, and the shaded passband on the panadapter move from above the frequency line to below it.

2. Choose **USB** again.

You should see the passband move back above the frequency line.

The mode list offers every mode a FlexRadio supports, but the demo's receiver is built for USB and LSB.

2.1.6 Step 5: Change the filter

1. In the **RX** applet, find the filter preset buttons and click **2.4K**.

You should see the shaded passband on the panadapter narrow, and the filter width in the RX applet header change to match.

2. Click **2.9K** to widen it again.

See RX Controls (p. 64) for the presets in each mode.

2.1.7 Step 6: Add a panadapter

The demo has a single slice, so the **+RX** button does not add a second one here. On a real radio it does; see Multi-Slice Operation (p. 83). The demo can show up to four panadapters, so add one instead.

1. Choose **Tools** → **Add Panadapter...**

You should see the **Panadapter Layout** window with **Choose panadapter layout:** and a set of layout buttons.

2. Click **A / B (2 pans)**.

You should see a second panadapter appear with the first. It has no slice on it: in the demo, extra panadapters are extra views of the same simulated band.

3. Choose **Tools** → **Add Panadapter...** again and click **Single (1 pan)** to go back to one panadapter.

2.1.8 Step 7: Clean up a noisy band

Make some noise, add a voice to listen to, then let AetherSDR's noise reduction work on it.

1. In the **Demo Noise** applet, click the **noisy-qth** preset.

You should see **Pink / hiss**, **Power-line** and **SMPS hash** light up, and **you should hear** a buzz and a rough hash on top of the hiss.

2. Click **Voice (speech)** to turn it on.

You should hear a voice in the noise. Hover over any source for a tooltip that explains the real-world cause.

3. Choose **Settings** → **AetherRX...** On the **AetherNR** tab, click **NNR**.

You should see the **NNR** button light and the **AetherNR** stage's tick box become ticked. **You should hear** the noise drop behind the voice.

4. Click **BYPASS** at the foot of the AetherRX window, then click it again.

You should hear the unprocessed noise while BYPASS is on, and the cleaned-up audio when you turn it off. NNR works on every platform with nothing to download. Try the other methods too. A method your computer cannot run is greyed out; hover over it for the reason (for example **MNR** outside macOS). The first time you choose **NR2**, it spends a while optimising itself in an **FFTW Wisdom** window. See DSP Noise Mitigation (p. 241).

2.1.9 Step 8: Open the Display panel

1. In the column of buttons at the left edge of the panadapter, click **Display**.

You should see the Display panel open, with controls such as **Heat Map**, **Grid** and FFT averaging.

2. Click **Grid**.

You should see the frequency and dB grid lines disappear from the panadapter. Click **Grid** again to bring them back.

3. Click **Display** again to close the panel.

See Panadapter Controls (p. 54) and Themes and Theme Editor (p. 228) for more ways to change the look.

2.1.10 Step 9: Save and recall a memory

1. In the same button column, click **Memory**, then **Add Memory**.

You should see a **Save Memory** window showing the current frequency, mode and filter.

2. Type a name, for example `My first memory`, and click **Save**.

You should see Saved “**My first memory**” to **memories**. in the status bar.

3. Tune somewhere else, for example by scrolling the mouse wheel a few notches.
4. Choose **Tools** → **Memory...** and double-click the row for your memory.

You should see the VFO return to the frequency you saved.

The demo keeps its memories on this computer. A FlexRadio keeps them on the radio. See Memory Channels (p. 92).

2.1.11 Step 10: Disconnect

1. Choose **File** → **Disconnect**.

You should see the demo session close. Open the **File** menu again and **Disconnect** is greyed out, because nothing is connected.

Because you disconnected yourself, AetherSDR does not reconnect to the demo the next time it starts.

2.1.12 Going further

- **Fault injection:** the **Fault Injection** row in the Demo Noise applet shows how AetherSDR handles trouble: a high SWR, a dropped slice, a stalled spectrum, a lost connection or a malformed message. **Clear** undoes them. A dropped slice comes back only after you disconnect and reconnect. See Demo Mode (p. 39).
- **Other presets:** **storm**, **night-40m** and **cw-in-noise** each set up a different band to practise on.
- **Your own radio:** connect it with First Connection (p. 11), then work through Before You Transmit (p. 14) before you key up for the first time.

2.1.13 See also

- Demo Mode (p. 39)
- First Connection (p. 11)
- Before You Transmit (p. 14)
- Panadapter Controls (p. 54)
- DSP Noise Mitigation (p. 241)
- Menu Reference (p. 235)

2.2 Installation

AetherSDR ships native builds for Linux, macOS and Windows. Every platform is built, tested in CI and released together, from the same source.

Download the latest release from GitHub Releases¹. The pre-built binaries are the recommended way to install. To build it yourself, see Building from Source (p. 277).

For everything about running AetherSDR on one operating system, see the platform pages: Linux (p. 43), macOS (p. 47) and Windows (p. 50).

2.2.1 Requirements

Platform	File	Notes
Linux x86_64	AetherSDR-* -x86_64.AppImage	Single file, no install. <code>chmod +x</code> and run.
Linux ARM	AetherSDR-* -aarch64.AppImage	Raspberry Pi and ARM laptops. <code>chmod +x</code> and run.
macOS Apple Silicon	AetherSDR-* -macOS-apple-silicon.dmg	M1 and later, macOS 14.4 or newer. Signed and notarized.
macOS Intel	AetherSDR-* -macOS-intel.dmg	Intel Macs, macOS 14.4 (Sonoma) or newer. Signed and notarized.
Windows Installer	AetherSDR-* -Windows-x64-setup.exe	Setup wizard with Start Menu shortcut and uninstaller.
Windows Portable	AetherSDR-* -Windows-x64-portable.zip	No install. Extract and run AetherSDR.exe.

All release binaries are built with Qt 6.12.

- **Linux x86_64:** the AppImage is built on an Ubuntu 22.04 base, so it runs on that release and on newer distributions.
- **Linux aarch64 (Raspberry Pi):** requires **glibc 2.38 or newer**. **Raspberry Pi OS Trixie** is the supported Pi baseline. Raspberry Pi OS **Bookworm** (glibc 2.36) cannot run it. The ARM build has GPU spectrum rendering like the other platforms.
- **macOS:** Apple Silicon and Intel have separate DMGs. Both need **macOS 14.4 (Sonoma) or newer**; Intel Macs that cannot run Sonoma cannot run current releases.

2.2.2 Setup

Linux

More on running AetherSDR on Linux: Linux (p. 43).

```
chmod +x AetherSDR-*-x86_64.AppImage
./AetherSDR-*-x86_64.AppImage
```

The AppImage bundles Qt and AetherSDR's other libraries; nothing else needs installing.

The AppImage is self-contained. If you built from source and want an application-menu entry and icon, `sudo cmake --install build` installs the binary, `.desktop` file and icon. See Building from Source (p. 277).

Wayland and XWayland

The AppImage runs **natively on Wayland**, which renders sharply under fractional scaling and avoids an XWayland crash some compositors produce when opening dialogs. If native Wayland is not available it falls back to XWayland automatically. On a headless Wayland session with no monitor attached (for example a Raspberry Pi reached over VNC), AetherSDR starts on XWayland (xcb) automatically, because native Wayland cannot render the panadapter there.

To override the choice, set `QT_QPA_PLATFORM` yourself. Your setting always wins:

¹<https://github.com/aethersdr/AetherSDR/releases/latest>

```
QT_QPA_PLATFORM=xcb ./AetherSDR-*.AppImage # force XWayland
QT_QPA_PLATFORM='wayland;xcb' ./AetherSDR-*.AppImage # force native Wayland
```

macOS

More on running AetherSDR on a Mac: macOS (p. 47).

Open the DMG for your Mac and drag AetherSDR to **Applications**. Both DMGs are signed and notarized by Apple, so Gatekeeper verifies them with no extra steps.

The **Intel DMG has no Copy Assist** (speech-to-text), and it draws the panadapter with the CPU renderer rather than the GPU. The Apple Silicon DMG has both. See Copy Assist (p. 121) and GPU Rendering (p. 252).

Windows

More on running AetherSDR on Windows: Windows (p. 50).

Choose either:

- **Installer** (...-setup.exe): adds a Start Menu shortcut and an uninstaller.
- **Portable** (...-portable.zip): extract anywhere and run AetherSDR.exe.

Both carry the Microsoft Visual C++ runtime with the application. **No Visual C++ Redistributable install is needed**, and repairing or removing the system redistributable does not affect AetherSDR.

The Windows builds are GPG-signed but not Authenticode-signed, so Windows SmartScreen may warn the first time you run them. Verifying the signature (below) confirms the download is genuine.

Your settings live in %LOCALAPPDATA%\AetherSDR\ whichever package you use. See Settings and Backups (p. 220).

2.2.3 After installing

- Your First Session (p. 5): a guided first run
- First Connection (p. 11): connect to your radio
- Demo Mode (p. 39): try AetherSDR with no radio
- Settings and Backups (p. 220): where your settings are kept

2.2.4 Reference

Verifying downloads

Linux and Windows downloads are GPG-signed, and each release includes .asc signatures and a signed SHA256SUMS.txt. macOS DMGs are Apple-notarized.

```
curl -sSL https://raw.githubusercontent.com/aethersdr/AetherSDR/main/docs/RELEASE-SIGNING-KEY.pub.asc | gpg --import
gpg --verify AetherSDR-vX.Y.Z-x86_64.AppImage.asc AetherSDR-vX.Y.Z-x86_64.AppImage
```

The expected result is “**Good signature from AetherSDR Release Signing**”. Full instructions, including the key fingerprint and Windows steps, are in docs/VERIFYING-RELEASES.md².

Debug symbols

Each release also carries per-platform debug-symbol archives (...-symbols.tar.xz). You do not need them to run AetherSDR; they let developers read crash reports. See Support and Logging (p. 261).

2.2.5 Known issues

- On Ubuntu 26.04 under native Wayland, the AppImage’s panadapter and waterfall can update very slowly (#4725³).

²<https://github.com/aethersdr/AetherSDR/blob/main/docs/VERIFYING-RELEASES.md>

³<https://github.com/aethersdr/AetherSDR/issues/4725>

- On Linux under native Wayland, the GPU spectrum can occasionally freeze the whole window at startup, seen on a Raspberry Pi 5 (#4704⁴).
- With the software (CPU) spectrum renderer on a Raspberry Pi 5, painting the spectrum takes nearly all of the interface thread, so the window can stop responding to clicks under extra load (#5192⁵).

2.2.6 Troubleshooting

AetherSDR won't start: error while loading shared libraries: libOpenGL.so.0

Ubuntu 26.04 no longer installs libopengl0 by default, and AetherSDR's GPU spectrum links against it.

1. Run `sudo apt install libopengl0`.
2. Start AetherSDR again.

The ARM AppImage won't run on a Raspberry Pi

Raspberry Pi OS Bookworm has glibc 2.36, and the aarch64 AppImage needs glibc 2.38 or newer.

1. Check your glibc version with `ldd --version`.
2. Move to Raspberry Pi OS Trixie, the supported Pi baseline.

The panadapter draws incorrectly

Your GPU driver does not render AetherSDR's GPU spectrum correctly. This is worth checking first on a Raspberry Pi whose Mesa driver is newer than its hardware.

1. Start AetherSDR with software OpenGL; no rebuild is needed: `AETHER_NO_GPU=1 ./AetherSDR-*.AppImage`.
2. See GPU Rendering (p. 252) for details.

PC audio misbehaves on a PipeWire system

The PipeWire library is installed but PipeWire's client configuration is not. AetherSDR then keeps Qt's audio backend off PipeWire at startup instead of crashing.

1. Install your distribution's PipeWire package: `pipewire` on Arch, `pipewire-bin` on Debian/Ubuntu.
2. Restart AetherSDR.

Windows SmartScreen warns the first time you run AetherSDR

The Windows builds are GPG-signed but not Authenticode-signed.

1. Verify the download's signature as described in Verifying downloads (p. 9) and docs/VERIFYING-RELEASES.md⁶.
2. If the signature is good, the download is genuine and you can let it run.

2.2.7 See also

- Linux (p. 43)
- macOS (p. 47)
- Windows (p. 50)
- Building from Source (p. 277)
- Troubleshooting (p. 256)
- GitHub Releases⁷

⁴<https://github.com/aethersdr/AetherSDR/issues/4704>

⁵<https://github.com/aethersdr/AetherSDR/issues/5192>

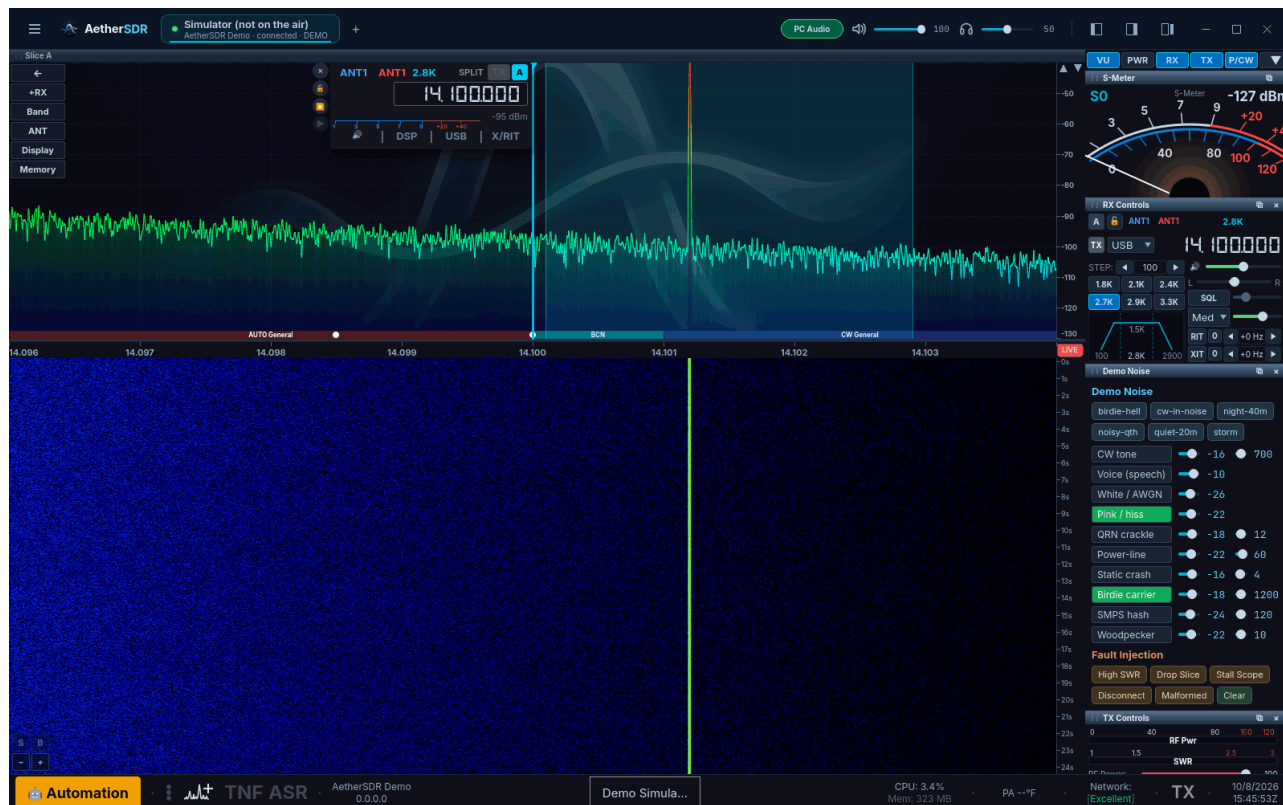
⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/VERIFYING-RELEASES.md>

⁷<https://github.com/aethersdr/AetherSDR/releases/latest>

2.3 First Connection

Status: Supported · **Applies to:** FlexRadio

This page walks through connecting to a FlexRadio for the first time. AetherSDR also connects to other radio families, and has a demo mode that needs no radio at all; see Supported Radios (p. 18) and Demo Mode (p. 39).

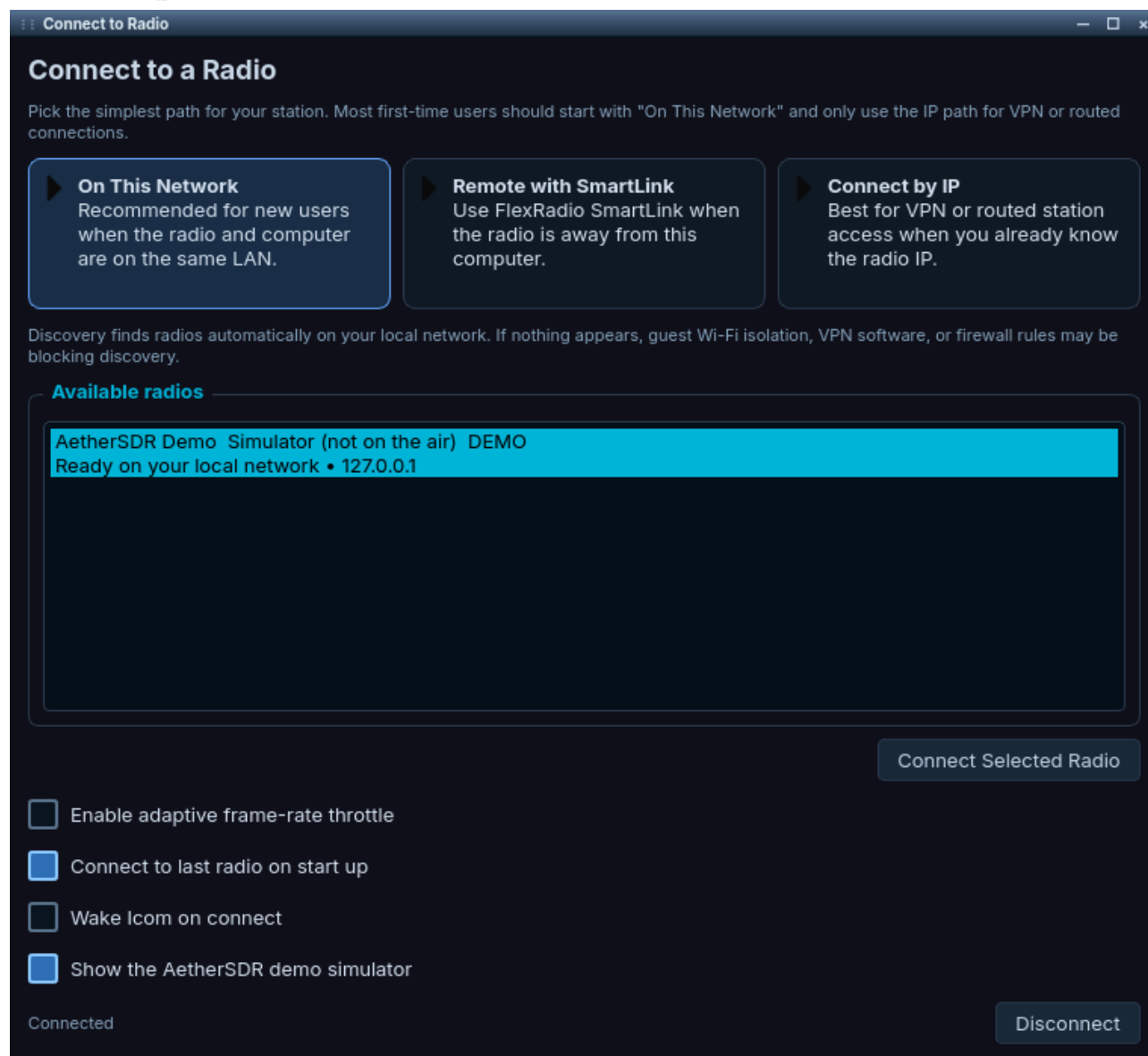


The AetherSDR main window connected to the demo simulator: title bar along the top, panadapter and waterfall on the left, applet panel on the right, status bar along the bottom.

2.3.1 Requirements

- A FlexRadio transceiver (FLEX-6000, FLEX-8000 or Aurora series) powered on
- Your computer on the same network as the radio
- AetherSDR installed (Installation (p. 8))

2.3.2 Setup



The Connect to Radio window with the AetherSDR demo simulator listed under On This Network.

1. **Launch AetherSDR.** The **Connect to Radio** window opens. You can reopen it at any time with **File → Connect to Radio...**, or from the + button beside the title bar's radio tabs.
2. Choose **On This Network** ("Recommended for new users when the radio and computer are on the same LAN").
3. Your radio appears under **Available radios** within a few seconds.
4. Select it and click **Connect Selected Radio**.
5. The spectrum and waterfall appear, and audio starts playing through your default output device.

If no radio appears, the window shows "No local radios found yet" with **Retry Discovery**, **Remote with SmartLink**, **Connect by IP** and **Open Network Diagnostics** buttons. See Troubleshooting (p. 13).

The list can also include:

- **Hermes-Lite 2** and **ANAN-G2** radios on the same LAN, and RTL-SDR dongles plugged into this computer (on builds with the RTL-SDR backend).

- **AetherSDR Demo**, labelled “Simulator (not on the air)”. It always sorts below real radios. Untick **Show the AetherSDR demo simulator** to hide it.

2.3.3 What to Expect

After connecting, you’ll see:

- **Spectrum** (top) showing the RF environment
- **Waterfall** (bottom) scrolling in real time
- **VFO widget** with frequency, S-meter and controls
- **Title bar** with a tab for the radio; its dot pulses with each heartbeat and turns red with “link lost” if the connection drops
- **Applet panel** (right) with the RX, TX, P/CW and other applets

Controls that the connected radio cannot use are hidden or dimmed with a reason. Continue with Your First Session (p. 5).

2.3.4 Other Ways to Connect

Auto-connect

With **Connect to last radio on start up** checked (the default), AetherSDR reconnects to the last radio on the next launch if it is reachable. Untick it in the Connect to Radio window to choose a radio yourself each time. If startup auto-connect gives up, the Connect to Radio window opens again.

Connect by IP (routed networks)

If your radio is on a different subnet or behind a VPN, discovery cannot see it. Choose the **Connect by IP** card, set **Radio type:**, enter **Radio IP:** and click **Connect by IP**. See Manual Connection (p. 201).

SmartLink (remote operation)

Choose **Remote with SmartLink** to reach a FlexRadio over the internet. See SmartLink Setup (p. 199).

2.3.5 Known issues

- On a FLEX-8600 on macOS, receive audio can stutter badly on the first connection until the radio is restarted (#5700⁸).
- On Windows, a FLEX-6300 that SmartSDR finds on the network may not appear under **On This Network** (#6221⁹).

2.3.6 Troubleshooting

The radio does not appear in the list

Discovery broadcasts do not cross routers or most VPNs, and guest Wi-Fi isolation, VPN software or firewall rules can block them.

1. Make sure the radio and computer are on the same subnet.
2. Check the radio is reachable: `ping <radio-ip>`.
3. Allow UDP port 4992 through the firewall (for example `sudo ufw allow 4992/udp`).
4. Click **Open Network Diagnostics** to narrow it down.
5. If the radio is on another subnet or behind a VPN, use **Connect by IP**; see Manual Connection (p. 201).

The radio refuses the connection

The radio rejected AetherSDR’s client registration (for example with error F3000001). AetherSDR closes the connection, does not retry on its own, and returns to the Connect to Radio window with the radio’s error.

1. Follow the message shown: typically another client must disconnect first, or the radio needs a restart.

⁸<https://github.com/aethersdr/AetherSDR/issues/5700>

⁹<https://github.com/aethersdr/AetherSDR/issues/6221>

2. If the **Connected Stations** dialog appears, select a station and click **Disconnect Station**; see Multi-Flex (p. 211).

Connected, but no audio

The PC audio path is off, or the wrong output device is selected.

1. Check that **PC Audio** is on in the title bar.
2. Open **Settings** → **Radio Setup...** → **Audio** and check the output device. See Audio Settings (p. 217).
3. Check that the slice and the title-bar speaker are not muted.

Spectrum, but no waterfall

The waterfall needs data from the radio before it starts drawing.

1. Wait a few seconds.

2.3.7 See also

- Your First Session (p. 5)
- Supported Radios (p. 18)
- Manual Connection (p. 201)
- SmartLink Setup (p. 199)
- Demo Mode (p. 39)
- Troubleshooting (p. 256)
- Support and Logging (p. 261)

2.4 Before You Transmit

Work through this checklist before you key up with AetherSDR for the first time, and again after a big change to your station, your computer or AetherSDR's settings. Each item says what to check and links to the page that explains it. Items marked **(FlexRadio)** apply only to a FlexRadio; the rest apply to every radio AetherSDR can transmit on.

The checks are written so you can do them without putting a signal on the air. Where you do need to transmit to check something, use low power into a dummy load or a clear frequency, and identify as your licence requires.

2.4.1 Requirements

- A radio that can transmit: a FlexRadio, a Hermes-Lite 2 or a recognised networked Icom. The ANAN-G2, RTL-SDR dongles, KiwiSDR/Web-888 receivers and the demo are receive-only. See Supported Radios (p. 18).
- AetherSDR connected to it. See First Connection (p. 11).
- A licence that covers the bands and modes you plan to use.

2.4.2 The checklist

The radio and the transmit slice

- ☐ **The right radio is connected.** Its tab is active in the title bar, and it is not the demo ("Simulator (not on the air)"). See First Connection (p. 11).
- ☐ **You know which slice transmits.** The transmit slice has the **red TX badge** on its VFO. Switching the active slice does not move transmit; click a grey **TX** badge to move it. See Multi-Slice Operation (p. 83) and VFO Widget (p. 61).
- ☐ **The transmit slice is on a frequency and mode you are licensed for.** Check the band plan strip along the bottom of the panadapter. See Panadapter Controls (p. 54).
- ☐ **Split is what you expect.** If a red **SPLIT** badge shows, you transmit on the second slice, not where you are listening. Click the red **SPLIT** badge to end the split. See Split Operation (p. 85).

Your station identity

- ☐ **Your callsign is set in Settings → Radio Setup... → Radio → Callsign:** (on macOS, **AetherSDR → Preferences... → Radio**). AetherSDR uses it for PSK Reporter, WSPR and spotting; on a FlexRadio it is also stored on the radio. See Radio Setup (p. 214).
- ☐ **Your callsign and grid are set wherever a feature transmits them.** AetherSDR has no single station grid; each feature that sends one has its own fields:
 - RADE Digital Voice (p. 113) end-of-over frames and FreeDV Reporter: **SpotHub → FreeDV, Callsign** (or **Use radio**) and **Grid Square** (or **Use GPS**). See SpotHub (p. 148).
 - The WSPR beacon in **Tools → PSK Reporter...: TX call:** and **Grid:**. See PSK Reporter Map (p. 155).
 - APRS in AetherModem Packet Radio (p. 118): **MY CALLSIGN**, and **GRID** when the radio has no GPS fix.
 - D-STAR: **MYCALL**, which defaults to the radio's callsign. See D-STAR (ThumbDV) (p. 115).
 - Digital-mode programs such as WSJT-X use the callsign and grid in their own settings.

Microphone and transmit audio

- ☐ **The microphone path is the one you mean to use.** On a FlexRadio, **Mic source** in the **P/CW** applet chooses MIC, BAL, LINE, ACC or PC; for a microphone on this computer, choose **PC**. On a radio that takes its transmit audio over the network, such as a networked Icom, turn on **PC Audio** in the title bar. See TX Controls (p. 68).
- ☐ **The computer's Input device is your real microphone**, not a virtual device such as a DAX microphone. Check **Settings → Radio Setup... → Audio**. See Audio Settings (p. 217).
- ☐ **The Mic level gauge in the P/CW applet moves when you speak.**
- ☐ **AetherTX is set up, if you use a computer microphone.** AetherTX shapes only a microphone on this computer. Open **Tools → AetherTX...**, check which stages are ticked, and set the **Gate** stage so it opens on your voice and closes between words. Use **REC / PLAY** to record and hear your processed audio **without keying the radio**. See Aetherial Audio (p. 74).

Power, tuner and antenna

- ☐ **RF Power and Tune Power are at sensible levels** in the **TX** applet (both 0–100 %). Start low for a first test. See TX Controls (p. 68).
- ☐ **(FlexRadio) The radio's power limits suit your station: Max Power** on the **Transmit** page of Radio Setup, and per-band RF and tune power and PTT inhibit in **Settings → TX Band Settings...**. See Radio Setup (p. 214).
- ☐ **The transmit antenna is the one you mean.** The RX applet header shows the RX and TX antenna buttons, with the names you gave them in **Settings → Radio Setup... → Antennas**. See RX Controls (p. 64).
- ☐ **The antenna is matched.** Run the radio's **ATU** from the TX applet (FlexRadio), or your external tuner, at low tune power, and check the **SWR** gauge. A Tuner Genius XL has its own applet; see TGXL Tuner Control (p. 173). **Tools → Pre-tune ATU Bands...** keys the transmitter repeatedly, so make sure the bands are clear first.
- ☐ **External amplifiers are in STANDBY** until everything above checks out, and while you tune or sweep. The amplifier applets have **STANDBY / OPERATE** buttons; the PGXL has a **STBY** key. See Amplifiers (p. 170) and Peripherals (p. 166).
- ☐ **(FlexRadio) TUNE will not key anything you don't want it to: Settings → Inhibit during TUNE ▶** chooses which outputs (ACC TX, TX1, TX2, TX3) are held while the radio tunes. See TX Controls (p. 68).

Digital modes

- ☐ **The program's audio route works.** On Linux and macOS with a FlexRadio, DAX Virtual Audio (p. 128) gives each slice a virtual audio device. On Windows, and on radios without DAX, use the TCI Server (p. 135). See WSJT-X Integration (p. 125).
- ☐ **CAT or TCI control works before you transmit.** Changing frequency and mode in the program should move the transmit slice in AetherSDR, and the program should key the slice you expect. See CAT Control (p. 132) and TCI Server (p. 135).
- ☐ **(FlexRadio) The TX filter passes your tones.** If AetherSDR shows a TX filter card on the panadapter when you transmit, widen **TX Filter** in the **Phone** applet. See TX Controls (p. 68).
- ☐ **Your computer's clock is accurate.** FT8 and similar modes fail when the clock is off by more than about a second. Keep the computer synchronised with NTP. AetherClock and GPS (p. 161) shows how far the computer's clock is from WWV/WWVB or the radio's GPS.

Keying paths and timers

- ☐ **You know every way AetherSDR can be keyed.** Check each one you have set up:
 - **Keyboard shortcuts** are off by default. If you turn them on in **Settings** → **Keyboard Shortcuts**, **Space** is push-to-talk while held and **T** toggles MOX. See Keyboard Shortcuts (p. 224).
 - Controller buttons can be bound to PTT, MOX or TUNE: USB Control Surfaces (p. 189) (including serial PTT and foot switches), MIDI Controller Mapping (p. 192), Ulanzi Dial (p. 186), StreamDeck (p. 181) and the FlexControl Tuning Knob (p. 183).
 - CAT and TCI clients can key the radio. See CAT Control (p. 132) and TCI Server (p. 135).
 - The automation bridge cannot key the transmitter unless **Allow TX via MCP** is ticked, which is off by default. Leave it off unless you need it. See Automation Bridge and MCP (p. 144).
- ☐ **(FlexRadio) The radio's transmit time-out and interlocks are set: Timeout (sec):** and the **Interlocks - TX REQ** group on the **Transmit** page of Radio Setup. See Radio Setup (p. 214).
- ☐ **You can see when you are transmitting.** A green M:SS transmit timer appears in the title bar, just left of **PC Audio**, for each of your own transmissions, and the VFO marker turns orange/red. If an interlock blocks transmit, a notification says why. See TX Controls (p. 68).

First transmission

- ☐ **The frequency is clear,** or you are on a dummy load.
- ☐ **Press MOX (or your PTT) briefly at low power** and watch the **Fwd Power** and **SWR** gauges in the TX applet. Unkey at once if the SWR is high or the power is not what you expect.

2.4.3 Troubleshooting

The radio keys but puts out almost no RF in DIGU/DIGL

The TX filter's low and high cut leave out your transmit audio.

1. In the **Phone** applet, widen the **TX Filter** with the **Low Cut** and **High Cut** buttons until it covers your tones.
2. Transmit again and check that the panadapter warning card no longer appears.

Transmit audio is missing with Mic source set to PC

The computer's selected microphone is not your microphone, for example a virtual DAX microphone.

1. Open **Settings** → **Radio Setup...** → **Audio** and choose your real microphone as the **Input** device.
2. Check that the **Mic level** gauge in the **P/CW** applet moves when you speak.

Space or another key does not key the radio

Operating keyboard shortcuts are off by default.

1. Turn them on with **Settings** → **Keyboard Shortcuts**.
2. Check the binding in **Settings** → **Configure Shortcuts...**

A notification says transmit is blocked

An interlock is holding transmit off, for example an amplifier warming up, the ATU busy, or a band the radio does not allow.

1. Read the notification for the reason.
2. Wait for the amplifier or ATU to finish, or move to a band you may transmit on, then key again.

2.4.4 See also

- TX Controls ([p. 68](#))
- Aetherial Audio ([p. 74](#))
- Amplifiers ([p. 170](#))
- WSJT-X Integration ([p. 125](#))
- Keyboard Shortcuts ([p. 224](#))
- Your First Session ([p. 5](#))

Your Radio

3.1 Supported Radios

AetherSDR is a client for **FlexRadio** transceivers, and FlexRadio is the supported target. It also drives several other radio families through a vendor-neutral backend layer. Those families are newer, and each is labelled with its status below.

3.1.1 Where to go next

- **FlexRadio:** FlexRadio (p. 20), then First Connection (p. 11) and Your First Session (p. 5).
- **Another radio family:** Hermes-Lite 2 (p. 23), Networked Icom (p. 28), ANAN-G2 (p. 32), RTL-SDR (p. 34) or KiwiSDR and Web-888 (p. 36).
- **No radio:** Demo Mode (p. 39).

3.1.2 FlexRadio (supported)

Works with any FlexRadio transceiver, including:

- **FLEX-6000 series:** FLEX-6300, FLEX-6400, FLEX-6400M, FLEX-6500, FLEX-6600, FLEX-6600M, FLEX-6700
- **FLEX-8000 series:** FLEX-8400, FLEX-8400M, FLEX-8600, FLEX-8600M
- **Aurora series:** AU-510, AU-510M, AU-520, AU-520M
- ML-, CL- and RT-series devices

Firmware: the active test target is FLEX-8600 firmware 4.2.18 (SmartSDR protocol v1.4.0.0). Earlier 4.x firmware works; **v3.x is unsupported**. See Firmware Update (p. 232).

The rest of this documentation describes FlexRadio operation unless a page says otherwise. Start with First Connection (p. 11).

3.1.3 Other Radio Families

Family	Status	Connects by	Page
Hermes-Lite 2	Experimental	LAN discovery, or Connect by IP	Hermes-Lite 2 (p. 23)
Networked Icom (IC-705, IC-7300MK2, IC-9700 ...)	Early; the IC-7300MK2 is supported	Connect by IP	Networked Icom (p. 28)
ANAN-G2	Experimental, receive-only	LAN discovery, or Connect by IP	ANAN-G2 (p. 32)
RTL-SDR USB dongles	Experimental, receive-only	USB discovery (builds with the RTL-SDR backend)	RTL-SDR (p. 34)
KiwiSDR and Web-888	Public receivers used as receive-only antennas for a slice	Radio Setup → Antennas	KiwiSDR and Web-888 (p. 36)

Family	Status	Connects by	Page
Demo	Built-in simulator, cannot transmit	The Connect to Radio list	Demo Mode (p. 39)

None of the other families is a supported family yet. The IC-7300MK2 over its built-in Ethernet (RS-BA1) is the one non-Flex model supported on its own; every other Icom model keeps the early, experimental treatment.

While a Hermes-Lite 2, an ANAN-G2 or an Icom other than the IC-7300MK2 is connected, an **EXPERIMENTAL** badge shows in the title bar, and an “Experimental radio support” notice appears when you connect. Tick “Don’t show again” in the notice to stop it for that family.

3.1.4 Setup

How you reach each family from the **Connect to Radio** window (**File** → **Connect to Radio...**):

- **On This Network** lists FlexRadio, Hermes-Lite 2 and ANAN-G2 radios on your LAN, RTL-SDR dongles on USB, and the demo.
- **Connect by IP** has a **Radio type:** list (FlexRadio, Hermes-Lite 2, ANAN-G2, Icom (network), RTL-SDR (USB)). A networked Icom is always reached this way. See Manual Connection ([p. 201](#)).
- **KiwiSDR and Web-888** receivers are configured under **Tools** → **Configure KiwiSDR...** and chosen as a slice’s RX antenna.

3.1.5 Reference

What works where

This table lists only what has been confirmed for each family. A ? means not yet confirmed for that family, not that it is known to work or fail.

	FlexRadio	Hermes-Lite 2	Networked Icom	ANAN-G2	RTL-SDR	Demo
Transmit	Yes	Yes (SSB voice, CW, data)	Yes on recognised models; none on an unrecognised model	No	No	No
Receivers / slices	As reported by the radio	Up to 4 (A–D)	Per model	1	1	1
Panadapters	As reported by the radio	One per receiver	1	1	1	?
DAX virtual audio and IQ	Yes	No	No	No	No	No
TCI server	Yes	Yes	Yes (receive audio, FT8, WSPR)	?	?	?
CAT (rigctl)	Yes	Yes	?	?	?	?
SmartLink	Yes	No	No	No	No	No
Multi-Flex (several clients)	Yes	No	No	No	No	No
Firmware update from AethersDR	Yes	No	No	No	No	No
Memory channels	On the radio	On this computer	On this computer, with import from the radio	On this computer	On this computer	On this computer

	FlexRadio	Hermes-Lite 2	Networked Icom	ANAN-G2	RTL-SDR	Demo
Radio Health dialog	Yes	Yes	Yes	Yes	Yes	?

Notes:

- **Receivers on the HL2:** up to four receivers at spans up to 192 kHz, three at 384 kHz. The span is one value for the whole board.
- **Icom panadapters:** the IC-9700, IC-7610 and IC-785x offer one panadapter, the single scope stream AetherSDR implements.
- **Memory channels:** a FlexRadio keeps its memories. Every other family uses one shared bank stored on this computer. On an Icom whose model supports it, the radio's own memories can be read into that bank; the IC-9700's radio memories are read-only. See Memory Channels (p. 92).

Controls on non-Flex radios

On a Hermes-Lite 2, ANAN-G2, RTL-SDR or Icom, every control does one of three things:

1. **Works** through the radio's own route.
2. **Is dimmed with its reason** in the tooltip and the screen-reader description. For example, the title-bar headphone controls say "this radio has no headphone output".
3. **Refuses with a warning** and a once-per-session notice before anything is sent. This covers, for example, picking an antenna the radio does not publish, WFM, DAX TX, and split or CWX controller bindings.

FlexRadio-only surfaces, such as the Profile Manager, DAX, SmartLink and the ATU chain, are hidden on other radios. GPS controls appear only when the radio reports GPS hardware (a Flex with the GPSDO option, or the IC-705).

Radio Setup gains a page for some families: **Calibration** and **HL2 Hardware** (Hermes-Lite 2), **RTL Receiver** (RTL-SDR) and **Droop Correction** (ANAN-G2).

Peripherals

Supported external devices include the 4O3A/FlexRadio PGXL power amplifier and TGXL tuner, plus ACOM S-series, SPE Expert and VK3AMP amplifiers. See Peripherals (p. 166) and Amplifiers (p. 170).

3.1.6 See also

- FlexRadio (p. 20)
- First Connection (p. 11)
- Manual Connection (p. 201)
- Demo Mode (p. 39)
- Peripherals (p. 166)
- README: Supported Hardware¹

3.2 FlexRadio

Status: Supported · **Applies to:** FlexRadio

AetherSDR is a native client for FlexRadio transceivers on Linux, macOS and Windows. It speaks the SmartSDR protocol directly, so it runs alongside SmartSDR and Maestro and shares the radio's own profiles and memories. FlexRadio is AetherSDR's supported target: the rest of this documentation describes FlexRadio operation unless a page says otherwise.

¹<https://github.com/aethersdr/AetherSDR#supported-hardware>

This page is the starting point for Flex owners. It lists the supported models and firmware, the recommended order for getting on the air, and every page with Flex-specific features.

3.2.1 Requirements

Supported models

AetherSDR works with any FlexRadio transceiver, including:

- **FLEX-6000 series:** FLEX-6300, FLEX-6400, FLEX-6400M, FLEX-6500, FLEX-6600, FLEX-6600M, FLEX-6700
- **FLEX-8000 series:** FLEX-8400, FLEX-8400M, FLEX-8600, FLEX-8600M
- **Aurora series:** AU-510, AU-510M, AU-520, AU-520M
- ML-, CL- and RT-series devices

The number of slices and panadapters comes from the radio itself, so AetherSDR offers exactly what your model can do.

Firmware

- The active test target is **FLEX-8600 firmware 4.2.18** (SmartSDR protocol v1.4.0.0).
- Earlier **4.x** firmware works.
- **v3.x is unsupported.**

To update the radio from AetherSDR, see Firmware Update ([p. 232](#)) (experimental; updating with SmartSDR for Windows is still the recommended route).

Network

- For local operation, the computer and the radio on the same LAN. Discovery uses UDP port 4992.
- For a routed network or a VPN, the radio's IP address (Manual Connection ([p. 201](#))).
- For remote operation, SmartLink with port forwarding (SmartLink Setup ([p. 199](#))), or Tailscale on a FLEX-8000 series or Aurora radio (Tailscale Remote Access ([p. 203](#))).

3.2.2 Setup

Follow these pages in order the first time:

1. **Installation** ([p. 8](#)) — download and install AetherSDR for your computer (Linux ([p. 43](#)), macOS ([p. 47](#)) or Windows ([p. 50](#))).
2. **Your First Session** ([p. 5](#)) (optional) — learn the window on the built-in demo, with no radio and no risk of transmitting.
3. **First Connection** ([p. 11](#)) — find the radio under **On This Network** and connect.
4. **Audio Settings** ([p. 217](#)) — pick your speakers, headphones and microphone, and the PC Audio path.
5. **Before You Transmit** ([p. 14](#)) — the checklist to work through before you key up for the first time.

3.2.3 Using AetherSDR with a FlexRadio

Operating

- Panadapter Controls ([p. 54](#)) — spectrum, waterfall, the Display panel and up to 8 panadapters
- VFO Widget ([p. 61](#)) and RX Controls ([p. 64](#)) — frequency, mode, filters, AGC, squelch, RIT/XIT, and the radio's own DSP
- TX Controls ([p. 68](#)) — RF and tune power, TUNE, MOX, ATU and APD
- Meters ([p. 72](#)) — S-meter, SmartMTR, cross-needle power and SWR, Radio Vitals
- Multi-Slice Operation ([p. 83](#)) and Split Operation ([p. 85](#)) — several receivers, TX assignment and split
- Diversity and ESC ([p. 87](#)) — two-SCU diversity reception and beamforming on dual-SCU models
- TNF (Tracking Notch Filters) ([p. 90](#)) — radio-side tracking notches
- XVTR (Transverters) ([p. 97](#)) — transverter setup
- Aetherial Audio ([p. 74](#)) — the AetherTX and AetherRX client audio chains

Radio configuration

- Radio Setup (p. 214) — every settings page, including Transmit, Antennas, GPS and Peripherals
- Profile Management (p. 94) — Global, TX and Mic profiles, stored on the radio and shared with SmartSDR; import and export `.ssdr_cfg` packages
- Memory Channels (p. 92) — an editor for the radio's own memory bank
- Firmware Update (p. 232) — upload SmartSDR firmware from AetherSDR
- USB Cable Management (p. 178) — CAT, band decoder, bit, LDPA and passthrough cables on the radio's rear USB ports

Remote operation and sharing

- SmartLink Setup (p. 199) — operate over the internet with your SmartLink account
- Tailscale Remote Access (p. 203) — remote operation behind CGNAT on FLEX-8000 series and Aurora radios
- Manual Connection (p. 201) — connect by IP across routed networks and VPNs
- Low Bandwidth Connections (p. 208) — reduce traffic for VPN, LTE and metered links
- Multi-Flex (p. 211) — run alongside SmartSDR, Maestro or another AetherSDR

Digital modes and other programs

- DAX Virtual Audio (p. 128) — virtual audio channels for WSJT-X, fldigi and others (Linux and macOS)
- DAX IQ Streaming (p. 130) — raw I/Q at 24–192 kHz
- CAT Control (p. 132) — up to 8 CAT ports with Rigctl, TS-2000 and Flex dialects
- TCI Server (p. 135) — CAT, audio, IQ, CW and spots over one WebSocket
- WSJT-X Integration (p. 125) — FT8/FT4 over TCI, or CAT + DAX
- RADE Digital Voice (p. 113), D-STAR (ThumbDV) (p. 115), RTTY Operation (p. 110) and AetherModem Packet Radio (p. 118)
- CWX Panel (p. 102), CW Decoder (p. 104) and DVK Panel (p. 107) — CW keying, decoding and the voice keyer

Amplifiers, tuners and station accessories

- TGXL Tuner Control (p. 173) — 4O3A Tuner Genius XL, direct or relayed through the radio
- Peripherals (p. 166) — PGXL amplifier, Antenna Genius and other accessories, with 4O3A access codes
- Amplifiers (p. 170) — ACOM, SPE Expert, VK3AMP and KPA1500 amplifiers
- ShackSwitch (p. 175) and Green Heron Everywhere (p. 176) — antenna switches and rotators

Controllers

- FlexControl Tuning Knob (p. 183) — the FlexControl knob and the on-screen AetherControl
- CTR2 Proxy (p. 196) — relay a CTR2-Max controller to the radio
- USB Control Surfaces (p. 189), MIDI Controller Mapping (p. 192), StreamDeck (p. 181) and Keyboard Shortcuts (p. 224)

3.2.4 Reference

Features that work only with a FlexRadio, compared with the other radio families (from Supported Radios (p. 18)):

Feature	FlexRadio	Other families
DAX virtual audio and IQ	Yes	No
SmartLink	Yes	No
Multi-Flex (several clients)	Yes	No
Firmware update from AetherSDR	Yes	No
Profiles	Stored on the radio	Not offered
Memory channels	Stored on the radio	Stored on this computer

3.2.5 Troubleshooting

The radio does not appear under On This Network

The radio is on another subnet, or a firewall, VPN or guest Wi-Fi isolation is blocking discovery on UDP port 4992.

1. Check the radio answers: `ping <radio-ip>`.
2. Allow UDP port 4992 through the computer's firewall.
3. Use **Connect by IP** with **Radio type**: FlexRadio. See Manual Connection (p. 201).
4. **Open Network Diagnostics** in the Connect to Radio window helps narrow it down.

The radio refuses the connection

The radio rejected AetherSDR's client registration, for example with error F3000001. AetherSDR returns to the Connect to Radio window with the radio's error and what to do about it.

1. Follow the message: typically another client must disconnect first, or the radio needs a restart.
2. If multiFLEX is off and other clients are connected, use the **Connected Stations** dialog to disconnect one. See Multi-Flex (p. 211).

For anything else, see Troubleshooting (p. 256).

3.2.6 See also

- Supported Radios (p. 18)
- First Connection (p. 11)
- Before You Transmit (p. 14)
- Profile Management (p. 94)
- Troubleshooting (p. 256)
- FlexRadio²

3.3 Hermes-Lite 2

Status: Experimental · **Applies to:** Hermes-Lite 2

AetherSDR can operate a **Hermes-Lite 2** (HL2) over openHPSDR Protocol 1. Support is **experimental**: the radio receives and transmits, but FlexRadio remains AetherSDR's supported target (see Supported Radios (p. 18)). An **EXPERIMENTAL** badge shows in the title bar while an HL2 is connected.

The HL2 has no on-board DSP, so AetherSDR does the demodulation, filtering, AGC, noise blanking and transmit modulation on your computer, using WDSP.

3.3.1 Where to go next

- Before You Transmit (p. 14) and TX Controls (p. 68): SSB voice, CW, TUNE and data.
- Aetherial Audio (p. 74): the transmit EQ, PROC and ALC that act on the HL2.
- TCI Server (p. 135) and WSJT-X Integration (p. 125): FT8, WSPR and other data modes.
- CAT Control (p. 132): rigctl control, including RF gain.
- CW Decoder (p. 104), RTTY Operation (p. 110) and AetherModem Packet Radio (p. 118).
- TNF (Tracking Notch Filters) (p. 90) and DSP Noise Mitigation (p. 241).
- Memory Channels (p. 92): stored on this computer.

²<https://www.flexradio.com/>

3.3.2 Setup

Connecting

The HL2 is found automatically on your LAN: open **File** → **Connect to Radio...** → **On This Network**, select it and click **Connect Selected Radio**.

Across a VPN or another subnet, use **Connect by IP** with **Radio type**: set to **Hermes-Lite 2** (see Manual Connection (p. 201)).

- **Connect to last radio on start up** works with the HL2.
- **Nickname**: right-click the radio in the list → **Set Nickname...** / **Clear Nickname** (no need to connect), or **Settings** → **Radio Setup...** → **Radio** → **Nickname**: while connected. HL2 nicknames are stored on this computer, keyed by the radio's MAC address.
- The very first connect on a computer spends a while planning its DSP filters; the app stays responsive while it does. The result is saved, so later connects are quick.
- The status bar shows the gateway version.

Board variant and hardware: HL2 Hardware

Settings → **Radio Setup...** → **HL2 Hardware** describes the physical radio. It appears while an HL2 is connected, and the settings are stored against that radio.

Group	Controls
Local Audio	Codec : None — plain Hermes-Lite 2, AK4951 — HL2+ companion board, or SquareSDR 2 — codec on the mainboard . A board with a codec can play receive audio on the radio's own speaker; Speaker level : sets it.
Dither Bit	Dither bit and Random bit . On some variants the first box is relabelled to what that board uses the line for (internal loudspeaker or band-voltage output).
Companion Filter Board	Board : None — nothing on J16 , N2ADR — receive and transmit , or N2ADR — transmit only . Receive through the N2ADR 3 MHz high-pass adds the AM-broadcast high-pass on receive (80 m to 10 m). Choose None only when nothing is fitted, because then nothing filters the transmitter's harmonics.
Reference Clock and Tuner	External 10 MHz reference at CL1 locks the radio to a reference such as a GPSDO instead of its crystal. CL1 expects a 3.3 V logic-level clock: pad a typical 5 V GPSDO output with at least 6 dB of 50 Ω attenuation. Lock is not read back. Antenna tuner driven by the HL2 gateway raises the tune request during TUNE for an AH-4-protocol ATU; leave it off for a tuner on the N2ADR IO board.

The HL2 IO board's amplifier control follows the band.

Frequency calibration

Settings → **Radio Setup...** → **Calibration** corrects the HL2's crystal. It appears only on radios that cannot calibrate themselves.

1. Let the radio warm up for 15 minutes.
2. Pick a **Reference**: (WWV/WWVH 2.5–25 MHz, CHU 3.330 / 7.850 / 14.670 MHz, or **GPSDO** / **signal generator (custom)** with its frequency). Attenuate a GPSDO by at least 30 dB.
3. Zero-beat the reference with normal tuning, then click **Calibrate from current VFO**.
4. Fine-tune with **Trim**: – / + (steps of 0.1, 1 or 10 Hz at 10 MHz), or type the **Error**: in ppb ($\pm 50,000$ ppb). **Reset** returns to 0.

One value corrects every band, receive and transmit. It is stored per radio (by MAC), so a second HL2 does not inherit it. While **External 10 MHz reference at CL1** is on, the manual calibration is zeroed and disabled.

3.3.3 Receivers and Panadapters

- **Up to four receivers** (A–D), each with its own slice, panadapter, audio and S-meter. Add one with **Tools** → **Add Panadapter...**; close its pane to remove it.
- The receivers share one converter, so the **span (sample rate)**, **LNA gain**, **band** and **antenna** are **radio-wide**. Mode, passband, AGC and frequency are per receiver.
- **Span** snaps to 48, 96, 192 or 384 kHz. Four receivers run up to 192 kHz, three at 384 kHz. **Use low bandwidth mode** caps the span at 96 kHz. Only the pane holding the TX slice (otherwise the first docked pane) has live span buttons; the others are dimmed with the reason.
- **Band buttons** switch the hardware filters and preamp as well as the frequency.
- **Spots** from the DX cluster, RBN, WSJT-X, POTA and manual entry are drawn on the panadapter. Right-click a spot → **Remove Spot**.
- **Display: FFT AVG** really averages (about 10 ms per step); at 0 the trace is unaveraged. Manual **Black Level** and waterfall **NB Blank** work. **WtrFall Rate** is a rate in rows per second. FFT FPS and the dBm range are remembered across restarts.
- **Tools** → **Wideband Bandscope...** shows one raw record of the whole converter band (0 to 38.4 MHz) on demand.

3.3.4 RF Gain and the dBm Scale

- **RF gain** is the HL2's LNA, –12 to +48 dB. It is remembered per band; a band you have not visited starts at +20 dB.
- The dBm scale is derived from the LNA setting (full scale is +3 dBm at 0 dB LNA). It is not a calibrated measurement.
- **Auto RF gain** adjusts the LNA from the measured wideband headroom, in 6 dB steps, across the full range. If it cannot arm, the reason appears on the checkbox and in a notice.
- CAT L RF / 1 RF (rigctl) drive the panadapter RF gain across the same range; 0.5 is +18 dB.

Upgrading from an older version:

- A band you have not visited comes up at **+20 dB LNA** on every profile. A profile that carried a stuck lower value comes up hotter (26 dB in the reported case) on unvisited bands. Visited bands keep their gain.
- At the default +20 dB LNA, a given squelch level gates **32 dB higher** than before. Where Auto squelch is not available, a slice in Auto drops to Manual at its manual level.
- Selecting **DIGU** or **DIGL** now turns AGC off on that receiver. A DIGU/DIGL band-stack bookmark saved earlier can still restore its saved AGC mode.
- A profile that ticked **Auto RF gain** (always refused before) now arms on its first connect, and runs in 6 dB steps.

3.3.5 Receive

- **Latency:** outside CW the receive filter runs at minimum phase, so audio returns 44 ms after an unmute. In CW, onset is 85 ms; a notch or a passband under 100 Hz lengthens the filter.
- After you unkey, receive stays muted through the transmit/receive turnaround, so you don't hear the PA's own carrier.
- **S-meter** reads WDSP's average level. It holds steady across key-down and unkey.
- **AM and SAM** audio is DC-blocked and clean.
- **CW:** the passband is centred on the signal marker (a real BFO at your CW pitch).

Squelch

The **SQL** button works per mode family:

- **FM:** FM squelch.

- **AM, SAM, DSB, LSB, USB:** a level squelch with a threshold map measured on the HL2. It tracks RF gain dB for dB.
- **CW and data modes:** no squelch.

The panadapter SQL line and **Auto** squelch use the HL2's own scale.

AGC and APF

- AGC mode and threshold are remembered across restarts.
- **AGC off:** the **AGC-T** slider sets a fixed gain, 4–64 dB (10 dB by default). The AGC-off level is remembered per receiver.
- Selecting **DIGU** or **DIGL** turns AGC off on that receiver. Raise the AGC-off level if data audio is too quiet.
- **APF**, the CW audio peaking filter, is centred on your CW pitch. Its toggle and level are in the P/CW pane.

RIT and XIT

RIT and XIT work per receiver, ± 9999 Hz. When transmit passes to another receiver, that receiver's XIT applies.

Notches and noise blanker

- **Manual notches:** right-click the panadapter → **Add TNF at X MHz**. Drag a notch sideways to move it and up or down to change its width; right-click it to remove it. Notches keep tracking as you tune. See TNF (Tracking Notch Filters) (p. 90).
- **NB** runs WDSP's impulse blanker on the raw IQ, ahead of the demodulator. It is held during transmit.
- Client-side noise reduction works as on any radio; see DSP Noise Mitigation (p. 241).

3.3.6 Transmit

- **SSB voice** uses WDSP's TXA modulator. The EQ, PROC, TX low/high cut, eSSB, ALC and compression gauges act on the HL2.
- **The Mic Level slider is the transmit level.** The ALC only ever reduces. The Phone panel shows the **ALC Gain** the radio reports.
- **Data:** WSJT-X and similar programs work over the TCI Server (p. 135). Their power slider changes RF output, and the speech ALC does not raise client audio. The WSPR beacon defaults to –3 dBFS and is not moved by the mic slider.
- **CW** is keyed with host-timed edges, with sidetone. CW settings are restored after a restart.
- **TUNE** keys at **Tune Power**, and moving Tune Power during a tune changes the carrier.
- **Max Power: Settings** → **Radio Setup...** → **Transmit** → **Max Power** shows the HL2's rated watts, read-only. The power gauge uses a 5 W scale, and forward power reads near PEP on speech.
- **RF power slider:** the radio has 16 drive steps, so several slider positions do nothing and then one position adds about 1.25 dB.
- **Modes that refuse to transmit:** AM, SAM, DSB, FM, NFM, WBFM and DRM refuse MOX and TUNE, because the HL2 modulator is SSB only. A mode the HL2 cannot demodulate is refused everywhere.
- **TX drive and LNA gain** are remembered per band.

3.3.7 Other Features

- CW, RTTY and AetherClock decoders, and the QSO recorder.
- AX.25 packet with AetherModem (APRS, KISS TNC, terminal and mailbox).
- **Memory channels**, stored on this computer in the bank shared by radios without memory slots. See Memory Channels (p. 92).
- **Tools** → **Radio Health...** shows link, temperature, supply, PA current, gateway and the transmit voice chain.
- Connection health in the title bar, status bar and network statistics.

3.3.8 Reference

Not available on the HL2

- DAX, SmartLink or Multi-Flex.
- The status-bar **CWX**, **DVK** and **FDX** toggles are hidden: the HL2 has no radio-side CW text keyer, voice keyer or full duplex.
- FM repeater duplex and CTCSS encode.
- Controls the HL2 cannot honour are dimmed with their reason, or refuse with a notice.

3.3.9 Known issues

- Receivers B, C and D do not come back after you restart AetherSDR; only receiver A reconnects (#5777³).
- Graphic EQ band values are lost when you restart AetherSDR (#5611⁴).
- Keying with the HL2's hardware PTT jack does not show as transmit in AetherSDR, and does not carry host audio (#5998⁵).
- **TX Delay** and the other fields under **Settings** → **Radio Setup...** → **Transmit** → **Timings** do not hold a value and are not applied to the HL2 (#5370⁶).
- Each over starts with about 100 ms of keyed silence before your audio is transmitted (#6052⁷).
- Changing CW break-in or its delay shows a “nothing was sent to the radio” notice, although both settings take effect (#6153⁸).

3.3.10 Troubleshooting

The first connect takes a long time

On a computer's first connect, AetherSDR plans the HL2's DSP filters.

1. Wait for it to finish; the app stays responsive meanwhile.
2. Later connects reuse the saved result and are quick.

Span buttons are dimmed on a panadapter

The receivers share one sample rate, and only one pane controls it.

1. Change the span from the pane that holds the TX slice (or, if none does, the first docked pane).
2. If the span will not go above 96 kHz, untick **Use low bandwidth mode** in the Connect to Radio window and reconnect.

Data-mode audio is too quiet

DIGU and **DIGL** turn AGC off on that receiver, so the fixed AGC-off gain sets the level.

1. Raise the **AGC-T** slider on that receiver (4–64 dB, 10 dB by default).

MOX or TUNE is refused

The HL2 modulator is SSB only, so AM, SAM, DSB, FM, NFM, WBFM and DRM refuse to transmit.

1. Switch the slice to USB, LSB, CW or a data mode before keying.

The Calibration controls are disabled

External 10 MHz reference at CL1 is on, which zeroes and disables the manual calibration.

1. Open **Settings** → **Radio Setup...** → **HL2 Hardware**.
2. Untick **External 10 MHz reference at CL1** if you want to calibrate the crystal instead.

³<https://github.com/aethersdr/AetherSDR/issues/5777>

⁴<https://github.com/aethersdr/AetherSDR/issues/5611>

⁵<https://github.com/aethersdr/AetherSDR/issues/5998>

⁶<https://github.com/aethersdr/AetherSDR/issues/5370>

⁷<https://github.com/aethersdr/AetherSDR/issues/6052>

⁸<https://github.com/aethersdr/AetherSDR/issues/6153>

Moving the RF power slider changes nothing

The HL2 has 16 drive steps, so several slider positions fall on the same step.

1. Keep moving the slider; the next step adds about 1.25 dB.

3.3.11 See also

- Supported Radios (p. 18)
- Manual Connection (p. 201)
- Low Bandwidth Connections (p. 208)
- Before You Transmit (p. 14)
- docs/HERMES.md⁹: bring-up field notes for developers
- docs/architecture/hl2-frequency-calibration.md¹⁰

3.4 Networked Icom

Status: Early · **Applies to:** Networked Icom

AetherSDR can operate Icom transceivers that have a built-in network connection, using Icom’s CI-V commands over the RS-BA1 network protocol. No Icom software and no USB cable are needed.

The **IC-7300MK2** is supported and shows no EXPERIMENTAL badge. Every other Icom shows the **EXPERIMENTAL** badge in the title bar and an “Experimental radio support” notice when you connect. FlexRadio remains AetherSDR’s supported target; see Supported Radios (p. 18).

3.4.1 Where to go next

- Before You Transmit (p. 14) and TX Controls (p. 68).
- TCI Server (p. 135) and WSJT-X Integration (p. 125): FT8, WSPR and other digital modes.
- Aetherial Audio (p. 74): the transmit audio chain, with PC Audio on.
- CWX Panel (p. 102): CW text through the radio’s own keyer (CWK).
- CW Decoder (p. 104), RTTY Operation (p. 110) and AetherModem Packet Radio (p. 118).
- Memory Channels (p. 92): the shared bank, with import from the radio.
- USB Control Surfaces (p. 189): the Icom RC-28 remote encoder.

3.4.2 Requirements

Model	Network	Status
IC-7300MK2	Built-in Ethernet	Supported: no EXPERIMENTAL badge
IC-705	Wi-Fi	Early (experimental); verified against its CI-V guide
IC-9700	Ethernet	Early (experimental); works, with read-only memories
IC-7610, IC-785x, IC-905	Ethernet	Listed in the model chooser, not verified

The original IC-7300 has no network port and is not in the model list.

3.4.3 Setup

On the radio

Enable network (remote) control and create a network user name and password. AetherSDR uses the radio’s three UDP ports, which start at 50001 by default (control, CI-V, audio). The IC-705 must be on your Wi-Fi network; the IC-7300MK2 and IC-9700 use Ethernet.

⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/HERMES.md>

¹⁰<https://github.com/aethersdr/AetherSDR/blob/main/docs/architecture/hl2-frequency-calibration.md>

Connecting

Networked Icoms are not discovered automatically. Open **File** → **Connect to Radio...**, choose **Connect by IP** and set **Radio type** to **Icom (network)**. The form gains these rows:

Row	What to enter
Radio IP:	The radio's IP address or host name.
Icom user:	The network user name configured on the radio.
Icom password:	The network password configured on the radio. It is stored in the operating system's credential store, never in the settings file, and only after a successful connect.
Icom ports:	Standard (50001–50003) , or Custom NAT ports... with a First UDP port: . AetherSDR uses that port and the next two. Different port triplets let you reach several radios behind one NAT router.
Icom CI-V:	Auto-detect (recommended) asks the radio for its own CI-V address. You can also pick a model from the list, or Custom... and type a hex CI-V address: (for example A2, from the radio's MENU > SET > Connectors > CI-V). A wrong address is not silently corrected: the connection gets no frequency, panadapter or transmit.

Click **Connect by IP**. The model is identified over the wire, not from the radio's name. If the handshake fails, the message names the target and the likely cause.

Wake Icom on connect

Wake Icom on connect, near the bottom of the Connect to Radio window, is off by default. When it is on and the radio does not answer identification, AetherSDR wakes a supported model (IC-705, IC-7300MK2 or IC-9700) from standby once. It never puts the radio to sleep on disconnect. The radio's network interface must stay reachable in standby, so a Wi-Fi radio that has dropped off the network cannot be woken. For a custom CI-V address, pick the model in **Icom CI-V**: so AetherSDR knows which radio to wake.

3.4.4 Using a Networked Icom

AetherSDR treats an Icom differently from a FlexRadio:

- **The radio is authoritative.** It remembers its own frequency, mode and filter. AetherSDR reads the radio's state and levels when it connects and never pushes saved state onto it.
- **Unknown models fail closed.** Each verified model has its own command profile. A control that has not been proven on a model is hidden, and a radio AetherSDR does not recognise gets no panadapter and no transmit, rather than guessed defaults.

What you can do:

- **Panadapter** from the radio's own scope. Every Icom has **one panadapter**.
- **Receive and transmit:** RX audio, transmit, RF gain, preamp and attenuator (each separate from the display's RF gain), AF gain, TX monitor, VOX, RIT/XIT, manual notch and the ATU on models that have one. TUNE power changes made during a tune are kept.
- **Filters:** the radio's real IF width is read back, the filter buttons are ordered narrow to wide for the mode, twin PBT is supported, and TX low/high cut follow what the model supports.
- **Status bar:** the radio's **Network Radio Name** (the name set in its network settings) is shown, separate from model and callsign.
- **Radio Health (Tools → Radio Health...)** shows the radio's modulation inputs and levels.
- **Decoders:** the CW, RTTY and AetherClock decoders work. The CW decoder opens in CW.
- **Memory channels** use AetherSDR's shared bank on this computer, and the radio's own memories can be read in on models that support it. The IC-9700's radio memories are read-only.

PC Audio and data modes

The **PC Audio** toggle chooses the radio's DATA OFF modulation input: **WLAN** on the IC-705, **LAN** on the IC-7300MK2, and **MIC** when PC Audio is off. The radio's DATA MOD setting is left alone. On a model AetherSDR cannot verify, the toggle refuses.

DIGU, **DIGL** and **DFM** set the radio's DATA flag, so transmit audio comes from the computer, not the microphone. If the radio's modulation input is wrong, AetherSDR tells you what to change.

With PC Audio off, the Aetherial Audio TX controls are replaced by a note that names the fix.

FT8, WSPR and other digital modes

WSJT-X and similar programs work through the TCI Server ([p. 135](#)): FT8 decoding and PSK Reporter spotting, and WSPR transmit (proven on air from an IC-7300MK2). The built-in AX.25 modem can transmit on an Icom too. DAX is not available on an Icom.

CW text: CWK

On an Icom the status-bar CWX toggle reads **CWK**: CWX text is sent through the radio's own CI-V text keyer, on models that have one (IC-705, IC-7300MK2). It drives the same CWX Panel ([p. 102](#)).

FM repeaters and XFC

The FM controls set tone, duplex and offset where the model supports them. On an Icom, the FM **REV** button becomes **XFC**, a momentary transmit-frequency check; it always releases when hidden or on disconnect. The IC-705 and IC-9700 support the extended tone registers (including DTCS); the IC-7300MK2 supports tone and tone squelch, and its offset controls are disabled.

Controllers

The Icom RC-28 remote encoder is supported (**Settings** → **Icom RC-28 Remote Encoder...**); see USB Control Surfaces ([p. 189](#)).

3.4.5 Reference

IC-7300MK2

- TX bandwidth table; CW speed 6–48 WPM; CW pitch in 5 Hz steps.
- Squelch works in CW and data modes. The panadapter squelch line uses the MK2's own measured scale, and **Auto SQL is not available** on the MK2.
- AGC threshold, AGC Off, AM carrier, VOX delay and repeater offset are disabled. TUNE is refused in CW and CW-R.
- The preamp and ATT buttons update about 0.2 s after a change.

IC-705

- 2 m and 70 cm are available alongside HF.
- **WFM** (broadcast FM) is receive-only.
- GPS location and NTP time controls. The power gauge uses a QRP scale.
- AetherSDR warns if the radio's MOD Input is not WLAN.

IC-9700

- Three bands with their own power ceilings (100 / 75 / 10 W).
- CTCSS (TX and RX), DTCS, dial-lock sync, supply voltage, PA drain current, continuous compression and LAN MOD level.
- If CI-V stalls, AetherSDR tries a bounded automatic recovery before a full reconnect.
- Its MAIN/SUB dual-receiver layout is not modelled.

Not available on a networked Icom

- DAX, SmartLink or Multi-Flex.
- More than one panadapter.
- Anything a model has not been proven to support: those controls are hidden, dimmed with a reason, or refuse with a notice.

3.4.6 Known issues

- On the IC-7300MK2, transmit audio drops out for about a third of a second roughly every 1.2 seconds while the radio stays keyed (#6169¹¹).
- Squelch can stick off: once its level reaches 0, SQL cannot be turned back on from AetherSDR (IC-7300MK2, IC-705, IC-9700) (#6172¹²).
- The speech processor (PROC) slider snaps back to DX+ after you move it (IC-7300MK2, IC-705) (#6171¹³).
- CW speed, break-in and pitch do nothing on models without a CW text keyer, such as the IC-9700 (#6110¹⁴).
- On the IC-9700, the panadapter trace can sit on the bottom line and look blank on a quiet band with the preamp off (#4814¹⁵).
- On the IC-9700, the waterfall rate returns to 100 after you restart AetherSDR (#5996¹⁶).

3.4.7 Troubleshooting

Connected, but no frequency, panadapter or transmit

The **Icom CI-V**: address is wrong. AetherSDR does not correct a wrong address silently.

1. Disconnect and open **Connect by IP** with **Radio type**: set to **Icom (network)**.
2. Set **Icom CI-V**: to **Auto-detect (recommended)**.
3. If you use **Custom...**, check the address on the radio under MENU > SET > Connectors > CI-V and type it as hex (for example A2).

Connect by IP fails with a handshake error

The radio is not accepting network control from AetherSDR. The message names the target and the likely cause.

1. Check that network (remote) control is enabled on the radio.
2. Check **Icom user**: and **Icom password**: against the network user configured on the radio.
3. Behind a NAT router, check that **Icom ports**: matches the three forwarded UDP ports.

The radio is in standby and does not answer

AetherSDR does not wake the radio unless you ask it to.

1. Tick **Wake Icom on connect** near the bottom of the Connect to Radio window.
2. For a custom CI-V address, also pick the model in **Icom CI-V**:
3. On an IC-705, make sure the radio is still on your Wi-Fi network in standby; a radio that has dropped off the network cannot be woken.

Data-mode transmit warns about the modulation input

The radio's modulation input does not match the PC Audio setting.

1. Toggle **PC Audio** off and on to select the right input.
2. Or set it on the radio's front panel: MENU > SET > Connectors > MOD Input (WLAN on the IC-705, LAN on the IC-7300MK2).

3.4.8 See also

- Supported Radios (p. 18)
- Manual Connection (p. 201)
- TCI Server (p. 135)
- USB Control Surfaces (p. 189)

¹¹<https://github.com/aethersdr/AetherSDR/issues/6169>

¹²<https://github.com/aethersdr/AetherSDR/issues/6172>

¹³<https://github.com/aethersdr/AetherSDR/issues/6171>

¹⁴<https://github.com/aethersdr/AetherSDR/issues/6110>

¹⁵<https://github.com/aethersdr/AetherSDR/issues/4814>

¹⁶<https://github.com/aethersdr/AetherSDR/issues/5996>

- [docs/architecture/icom-capability-profiles.md](#)¹⁷
- [docs/architecture/aetherd-icom-civ-backend-design.md](#)¹⁸

3.5 ANAN-G2

Status: Experimental, receive-only · **Applies to:** ANAN-G2

AetherSDR can receive with an Apache Labs **ANAN-G2** over openHPSDR Protocol 2. Support is **experimental and receive-only**: one receiver, one panadapter, no transmit yet. FlexRadio remains the supported target (see Supported Radios (p. 18)). An **EXPERIMENTAL** badge shows in the title bar while a G2 is connected.

3.5.1 Where to go next

- Panadapter Controls (p. 54): FFT AVG, Black Level and NB Blank on the G2's panadapter.
- RX Controls (p. 64): tuning, AF gain, mute and balance.
- DSP Noise Mitigation (p. 241): client-side noise reduction.
- Memory Channels (p. 92): stored on this computer.

3.5.2 Setup

The G2 is found automatically on your LAN: open **File** → **Connect to Radio...** → **On This Network**, select it and click **Connect Selected Radio**.

To reach it across a VPN or another subnet, use **Connect by IP** with **Radio type**: set to **ANAN-G2** (see Manual Connection (p. 201)). That form also holds the G2's connection options, which apply on both paths:

Option	Default	What it does
Sample rate:	48 ksp/s	DDC0 sample rate: 48, 96, 192, 384, 768 or 1536 ksp/s. It is also the starting width of the panadapter. Higher rates use more of the radio's Ethernet link.
Select ADC:	ADC0 (ANT1/2/3)	Which receive chain feeds the receiver: ADC0 (ANT1/2/3), behind the switched antenna relays, or ADC1 (RX2) , wired straight to the RX2 jack.
Dither / Random	On	The ADC's linearisation bits. Leave them on unless you are testing.
ADC0 RF filter bypass / ADC1 RF filter bypass	On	Routes the antenna around the front-end filter bank. Leave these checked: no band filter is selected in this version, so unchecked means nothing reaches the ADC.
Send RX audio to the radio's speaker	Off	Also plays receive audio through the G2's own speaker and headphone jack. Takes effect on the next connect.

3.5.3 Receiving with the G2

- **Receive** with live tuning and zoom. Changing the zoom changes the DDC rate live, without rebuilding the session.
- **Panadapter** computed by WDSP's display analyser: a ≥16384-point windowed FFT at one point per screen pixel. **FFT AVG** and **Weighted** averaging drive it, and noise-floor auto-adjust works.
- **S-meter**.
- **NB** (noise blanker): WDSP's impulse blanker, run on this computer before the demodulator.
- **RF Gain** from -31 to 0 dB drives the G2's ADC step attenuator, and is restored per radio.

¹⁷<https://github.com/aethersdr/AetherSDR/blob/main/docs/architecture/icom-capability-profiles.md>

¹⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/architecture/aetherd-icom-civ-backend-design.md>

- **AF gain, mute and balance** apply. With **Send RX audio to the radio's speaker** on, the receiver's mute and volume apply to both the computer and the radio's speaker, and the title-bar speaker button mutes the radio's speaker too.
- Manual **Black Level** and waterfall **NB Blank**. See Panadapter Controls (p. 54).
- **Tools → Radio Health...**

The panadapter is not dBm-calibrated: its levels are relative.

3.5.4 Droop Correction

The G2's DDC loses a little amplitude near the edges of the displayed span. AetherSDR applies a correction derived from the radio's own (Saturn) gatewayware automatically.

To measure your own radio instead, open **Settings → Radio Setup... → Droop Correction** while connected:

1. Disconnect the antenna or terminate it in a dummy load. The sweep uses the receiver's own noise floor as a flat reference, so a live signal biases the result.
2. Click **Start Sweep**. It steps through every DDC0 sample rate and takes several minutes. **Stop** ends it early.
3. Review the measured correction per rate, then click **Apply** to use and save it for this radio, or **Discard**.

Upgrading from an older version: G2 panadapter tones now read about **7.4 dB lower** than they did before the WDSP panadapter (a test tone that read -6.0 dB now reads -13.4 dB). Re-trim a saved reference level or waterfall black level once.

3.5.5 Reference

Not available on the G2

- **Transmit**. Keying is refused; transmit is a future phase.
- DAX, SmartLink or Multi-Flex.
- More than one receiver or panadapter.

Controls the G2 cannot honour are dimmed with a reason, or refuse with a notice; see Supported Radios (p. 18).

3.5.6 Known issues

- Weak signals sound quieter than in Thetis on the same radio unless the **AGC-T** slider is near the top (#5988¹⁹).
- DX cluster, RBN, WSJT-X, POTA and manual spots are never drawn on the panadapter (#6039²⁰).

3.5.7 Troubleshooting

Nothing is received at all

An **RF filter bypass** box is unchecked. No band filter is selected, so an unchecked bypass means nothing reaches the ADC.

1. Disconnect from the G2.
2. Open **Connect by IP** with **Radio type:** set to **ANAN-G2**.
3. Tick **ADC0 RF filter bypass** and **ADC1 RF filter bypass**, then connect again.

No audio from the radio's own speaker

Send RX audio to the radio's speaker is off, or was turned on during the session. It takes effect on the next connect.

1. Tick **Send RX audio to the radio's speaker** in the ANAN-G2 options.
2. Disconnect and connect again.
3. Check that the receiver and the title-bar speaker button are not muted; they mute the radio's speaker too.

¹⁹<https://github.com/aethersdr/AetherSDR/issues/5988>

²⁰<https://github.com/aethersdr/AetherSDR/issues/6039>

3.5.8 See also

- Supported Radios (p. 18)
- Manual Connection (p. 201)
- Panadapter Controls (p. 54)
- docs/architecture/anap2-backend-design.md²¹: background for developers

3.6 RTL-SDR

Status: Experimental, receive-only · **Applies to:** RTL-SDR

AetherSDR can use an RTL-SDR USB dongle as a receiver. Support is **experimental and receive-only**: one slice on one panadapter, demodulated on this computer. FlexRadio remains the supported target (see Supported Radios (p. 18)).

3.6.1 Where to go next

- Panadapter Controls (p. 54): FFT AVG, Black Level and NB Blank on the dongle's panadapter.
- RX Controls (p. 64): modes, filters and squelch.
- DSP Noise Mitigation (p. 241): client-side noise reduction.
- Memory Channels (p. 92): stored on this computer.
- Building from Source (p. 277): building with the RTL-SDR backend.

3.6.2 Requirements

The RTL-SDR backend is built only when the `librtlsdr` library is available, so whether your copy of AetherSDR has it depends on the package. If **Connect by IP** → **Radio type:** has no **RTL-SDR (USB)** entry, and no dongle ever appears in the radio list, your build does not include it. The Windows installer is built without it (#5800²²).

Building from source: the `ENABLE_RTL` option is on by default and takes effect when both `librtlsdr` and single-precision FFTW (`fftw3f`) are found. See Building from Source (p. 277).

3.6.3 Setup

1. Plug in the dongle.
2. Open **File** → **Connect to Radio...** → **On This Network**. Supported dongles on USB are listed with the network radios.
3. Select the dongle and click **Connect Selected Radio**.

3.6.4 Using an RTL-SDR

- **Modes:** AM, SAM, FM, FM-N, WFM, USB, LSB, CW and CW-R.
- **Tuning range:** 24 kHz to 1.766 GHz. Below 24 MHz the dongle uses direct sampling; HF reception on particular dongles (for example ones with an upconverter or the RTL-SDR Blog V4) has not been qualified.
- **Sample rates** up to 3.0 MS/s; 2.4 MS/s is the default.
- **Browsing the capture:** pan around the captured bandwidth and your slice keeps its frequency and settings. A slice that falls outside the usable capture is parked (silent) and resumes when it fits again.
- **Zoom** uses a continuous 65,536-point FFT, which gives 36.6 Hz bins at 2.4 MS/s.
- **FM and FM-N** have symmetric filters, 48 kHz audio and their own squelch. Squelch exists only in FM and FM-N; its level is a signal-level gate in dBFS per bin, not calibrated dBm.
- **Confirmed settings:** a change is shown only after the dongle confirms it, so a refused setting never becomes a saved one.

²¹<https://github.com/aethersdr/AetherSDR/blob/main/docs/architecture/anap2-backend-design.md>

²²<https://github.com/aethersdr/AetherSDR/issues/5800>

- Manual **Black Level**, waterfall **NB Blank** and **FFT AVG** work on the panadapter. See Panadapter Controls (p. 54).
- **Tools** → **Radio Health...** shows the dongle's status.

Only one receiver is available, even though the capture is wide.

3.6.5 Reference

RTL Receiver settings

Settings → **Radio Setup...** → **RTL Receiver** (shown while a dongle is connected) has a **Receiver corrections** group:

Control	Use
Frequency correction:	Your dongle's crystal error in ppm (−1000 to +1000). Click Apply frequency correction to send it.
Suppress IQ DC	Removes the spike at the centre of the capture. Off by default.
Device serial	The serial number the dongle reports.

The page shows the values the dongle actually applied and whether they were saved. Corrections are saved per dongle serial number. A dongle that reports no serial keeps its corrections for the current session only.

What is remembered

Frequency, mode, passband, sample rate and RF gain are saved per dongle, keyed by the serial number it reports. Dongles that report no serial share one family-wide entry, and so do two dongles that report the same serial.

Upgrading from an older version: saved RTL-SDR state used to be keyed by USB position. On the first connection after upgrading, a dongle whose only saved state is under the old key may start from defaults once. It then saves under its serial number (or the family-wide entry). The old entries are left in place but no longer read.

Not available on an RTL-SDR

- Transmit.
- DAX, SmartLink or Multi-Flex.
- More than one slice or panadapter.

3.6.6 Known issues

- An RTL-SDR Blog V4 on macOS shows only noise on the panadapter and in the audio, with no signals (#6016²³).
- DX cluster, RBN, WSJT-X, POTA and manual spots are never drawn on the panadapter (#6039²⁴).

3.6.7 Troubleshooting

No dongle in the list and no RTL-SDR (USB) radio type

Your copy of AetherSDR was built without the RTL-SDR backend.

1. Check **Connect by IP** → **Radio type:** for an **RTL-SDR (USB)** entry.
2. If it is missing, use a build that includes the backend, or build from source with `librtlsdr` and `fftw3f` installed (see Building from Source (p. 277)).

The slice goes silent while you pan

The slice has fallen outside the usable capture, so it is parked.

1. Pan back until the slice is inside the capture again; it resumes on its own.

A spike sits at the centre of the capture

This is the dongle's IQ DC offset.

²³<https://github.com/aethersdr/AetherSDR/issues/6016>

²⁴<https://github.com/aethersdr/AetherSDR/issues/6039>

1. Open **Settings** → **Radio Setup...** → **RTL Receiver**.
2. Tick **Suppress IQ DC**.

Stations appear off frequency

The dongle's crystal is off.

1. Open **Settings** → **Radio Setup...** → **RTL Receiver**.
2. Enter the error in ppm in **Frequency correction**.
3. Click **Apply frequency correction**.

3.6.8 See also

- Supported Radios (p. 18)
- Building from Source (p. 277)
- Panadapter Controls (p. 54)
- docs/rtl-m1-runtime.md²⁵
- docs/rtl-slice-settings.md²⁶

3.7 KiwiSDR and Web-888

AetherSDR can use a remote **KiwiSDR** or **Web-888** web receiver as a receive-only “virtual RX antenna” for a slice. While a slice uses one, its audio, S-meter and (optionally) its spectrum and waterfall come from that receiver instead of your radio. Public KiwiSDRs around the world can be browsed from inside AetherSDR.

These receivers never transmit, and AetherSDR inhibits transmit on a panadapter that is showing one.

3.7.1 Where to go next

- Diversity and ESC (p. 87): pair a KiwiSDR with your radio for diversity receive.
- SpotHub (p. 148): the KiwiSDR DX Community spot overlay.
- Memory Channels (p. 92): KiwiSDR receivers use the bank stored on this computer.
- Support and Logging (p. 261): KiwiSDR log categories for problem reports.

3.7.2 Setup

Adding a receiver

Open **Tools** → **Configure KiwiSDR...** It opens **Settings** → **Radio Setup...** → **Antennas**, where the **KiwiSDR RX Antennas** group lists your receivers.

Each receiver has:

Field	Use
Name	Required display name. It is how the receiver appears in antenna menus.
Server	hostname or hostname:port (the default port is 8073).
Password	Optional. Stored in the operating system's credential store. The status beneath it reads “Stored securely”, “Current session only” (builds without a credential store) or “No password stored”. To change it, type over the masked value; you don't need the old one.
Type	KiwiSDR or Web-888 . Entries saved before Web-888 support default to KiwiSDR.
Auto-connect	Connect to this receiver automatically.
Keep audio during TX	Off (default): the receiver's audio is silenced while you transmit and resumes immediately at un-key. The stream stays warm, so there is no gap or re-sync. On: keep listening while you transmit.

²⁵<https://github.com/aethersdr/AetherSDR/blob/main/docs/rtl-m1-runtime.md>

²⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/rtl-slice-settings.md>

Field	Use
Resume audio after TX delay	After unkey, wait out this receiver's stream delay before unmuting, so you don't hear your own delayed transmission. No effect while Keep audio during TX is on.

Each row also has **Connect** / **Disconnect** and a **Remove** (bin) button. To add a receiver by hand, fill in the empty row at the bottom (name, host :8073, optional password and type) and click **Add receiver**.

Import and export

Import... and **Export...** move the receiver list to and from a CSV file. Passwords are never exported; they stay in the credential store. Import merges on a matching server, so importing the same file twice changes nothing and keeps existing passwords. An optional RECEIVER_TYPE column sets KiwiSDR or Web-888.

Browsing public KiwiSDRs

Click **Browse public...** to open **Browse public KiwiSDR receivers**:

- A filter box (“Filter by name, location, or host...”) and **Refresh list**.
- Columns **Receiver**, **Location**, **Users**, **API** and **Limits**.
- Select one and click **Add selected**.

The directory comes from AetherSDR's own mirror at `cdn.aethersdr.com`, which the KiwiSDR maintainer asked for. It is refreshed hourly; AetherSDR never contacts `kiwisdr.com` for the list. Opening the browser reuses the list for 30 minutes; **Refresh list** always fetches. If the mirror is unreachable, the browser shows the list it already has and points to `kiwi-status.aethersdr.com`. Once a list is more than 6 hours old, its age is added to the status line, but it is still shown in full.

Out of respect for each operator's choice, receivers whose operator disabled API access (“web-only”), receivers that publish no policy, and receivers the directory flags are **not offered**. The status line counts how many are hidden for each reason.

Web-888 receivers are not in the public directory; add them by host name.

3.7.3 Using a Receiver on a Slice

Choose the receiver by name in the slice's RX antenna list: the RX antenna selector in the RX applet, or the antenna menu in the panadapter overlay. Choose a radio antenna again to return to your radio.

- Tuning the slice tunes the receiver. The receiver stays assigned across band changes, and the slice's previous mute state comes back when you stop using it.
- In CW the carrier is audible, offset by your CW pitch.
- The connection is released when no slice is using it.

The **KiwiSDR** (KSDR) applet lists each configured receiver with its type (a **Web-888** badge where it applies), its state and the slice it is assigned to. States include Disconnected, Connecting, Connected, Busy, Waiting, Monitoring, Camp ended and Error.

Busy receivers

Public receivers have a limited number of channels. When every channel is busy, AetherSDR waits for a free slot (“Waiting for a free KiwiSDR receiver slot”), can offer a monitor (camp) session on another user's channel, and says clearly when the receiver refuses you.

Flex / Kiwi display toggle

A small **Flex** / **Kiwi** button at the bottom-left of a panadapter with a KiwiSDR slice chooses where the spectrum and waterfall come from. It changes only the display; audio and meters are unchanged. While the Kiwi display is shown, transmit is inhibited on that panadapter: “Transmit is disabled because this panadapter is displaying a KiwiSDR receiver.”

The 3D stacked-trace spectrum works on KiwiSDR sources too.

3.7.4 Receive Sync and Diversity

A remote receiver's audio and waterfall arrive later than your own radio's. **Settings** → **Receive Sync** ▶ time-aligns them:

Item	Use
Sync Kiwi Audio & Display	Turns alignment on.
Manual Offset / Auto Assist	Set the offset yourself, or let AetherSDR measure it by cross-correlating the two receivers' audio and spectrum.
Delay KiwiSDR 50 ms / Delay Flex 50 ms / Reset Offset	Nudge the manual offset.
Latency ▶	Normal (360 ms), More Stable (520 ms) or High Jitter (1000 ms).

The menu also shows the current offset and a status line.

A KiwiSDR can take one side of a diversity pair, so you hear your radio and the remote receiver together. See Diversity and ESC (p. 87).

3.7.5 KiwiSDR DX Community Spots

The **Kiwi DX** toggle in **SpotHub** → **Display** overlays the KiwiSDR DX Community database on the panadapter. It is off by default. See SpotHub (p. 148).

3.7.6 Known issues

- Buzzing, crackling or popping can be heard while a KiwiSDR is a slice's RX antenna (#4993²⁷).
- On receivers that share their waterfall across all channels, AetherSDR can show "KiwiSDR waterfall unavailable" although the receiver sends one (#5656²⁸).

3.7.7 Troubleshooting

The receiver shows Busy or Waiting

Every channel on the receiver is in use.

1. Leave the slice on the receiver: AetherSDR waits for a free slot and connects when one opens.
2. Accept a monitor (camp) session if one is offered.
3. Or choose another receiver with **Browse public...**

The receiver shows Error

The receiver refused the connection. Error messages name their source: the receiver's own refusal codes and messages.

1. Read the message in the **KiwiSDR** applet.
2. Before reporting a problem, turn on the KiwiSDR logging categories in **Help** → **Support & Diagnostics...**; see Support and Logging (p. 261).

The public list is stale or will not refresh

AetherSDR's directory mirror is unreachable, so the browser shows the list it already has.

1. Check `kiwi-status.aethersdr.com`.
2. Click **Refresh list** again later.

Transmit is disabled on a panadapter

The panadapter is showing the KiwiSDR's spectrum and waterfall.

1. Click the **Flex / Kiwi** button at the bottom-left of the panadapter to switch the display back to your radio.

²⁷<https://github.com/aethersdr/AetherSDR/issues/4993>

²⁸<https://github.com/aethersdr/AetherSDR/issues/5656>

You hear your own transmission after unkeying

The receiver's audio is delayed, so it still carries your signal for a moment after unkey.

1. Open **Tools** → **Configure KiwiSDR...**
2. Tick **Resume audio after TX delay** for that receiver, and leave **Keep audio during TX** off.

The KiwiSDR's audio is out of step with your radio

A remote receiver's audio and waterfall arrive later than your own radio's.

1. Open **Settings** → **Receive Sync** ▶ and tick **Sync Kiwi Audio & Display**.
2. Choose **Auto Assist**, or set **Manual Offset** and nudge it with **Delay KiwiSDR 50 ms / Delay Flex 50 ms**.
3. On an unsteady link, choose a longer **Latency** ▶ setting.

3.7.8 See also

- Diversity and ESC ([p. 87](#))
- SpotHub ([p. 148](#))
- Supported Radios ([p. 18](#))
- docs/kiwisdr-public-directory.md²⁹
- docs/kiwisdr-cleanroom-design.md³⁰
- docs/web888-cleanroom-design.md³¹

3.8 Demo Mode

Demo mode runs the full AetherSDR interface against a built-in simulator, so you can explore the app with no radio attached. It generates its own receive audio and a matching panadapter. **It cannot transmit.**

²⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/kiwisdr-public-directory.md>

³⁰<https://github.com/aethersdr/AetherSDR/blob/main/docs/kiwisdr-cleanroom-design.md>

³¹<https://github.com/aethersdr/AetherSDR/blob/main/docs/web888-cleanroom-design.md>



The Demo Noise applet, shown only while connected to the demo simulator.

3.8.1 Where to go next

- Your First Session ([p. 5](#)): a guided tour you can follow on the demo.
- DSP Noise Mitigation ([p. 241](#)): practise noise reduction on the Demo Noise scenes.
- Panadapter Controls ([p. 54](#)): the spectrum and waterfall display.
- Memory Channels ([p. 92](#)): stored on this computer.

3.8.2 Setup

1. Open the **Connect to Radio** window (**File** → **Connect to Radio...**).
2. On **On This Network**, the list includes **AetherSDR Demo**, labelled “**Simulator (not on the air)**”.
3. Select it and click **Connect Selected Radio**.

The demo always sorts below real radios and is never chosen over one. There is no menu item or command-line flag; you connect to it like any radio.

To hide it, untick **Show the AetherSDR demo simulator** in the Connect to Radio window. The choice is remembered.

The demo's name is its “not on the air” safety label, so it cannot be renamed.

3.8.3 Using the Demo

- One slice on one panadapter, with synthetic receive audio and spectrum.
- The panadapter title bar, adaptive RX filter, auto-squelch, centre lock and band recall work as they do on a radio.
- Simulated CW is proper Morse at 18 WPM and 700 Hz.
- Memory channels are stored on this computer, in the bank shared with other radios that have no memory slots of their own.

Demo Noise applet

While connected to the demo, the **Demo Noise** applet lets you build a noisy band and practise with the noise-reduction tools (DSP Noise Mitigation ([p. 241](#))).

Presets set up a whole scene at once: quiet-20m, night-40m, storm, noisy-qth, birdie-hell and cw-in-noise.

Sources each have a toggle and a level slider, and some have a second control. Hover any source for a tooltip that explains the real-world cause and what to do about it. The CW tone slider glides its pitch without clicks.

Fault injection

The **Fault Injection** row lets you see how AetherSDR handles trouble without risking a real radio. The buttons are listed under Reference ([p. 41](#)).

3.8.4 Reference

Demo Noise sources

Source	Extra control	What it imitates
CW tone	Pitch (300–1200 Hz)	A wanted CW signal
Voice (speech)		A wanted SSB voice signal
White / AWGN		The receiver's own thermal noise
Pink / hiss		Atmospheric band noise
QRN crackle	Rate	Lightning static from distant storms
Power-line	50 / 60 Hz	Mains buzz and its harmonics
Static crash	Rate	Loud crashes from a nearby storm front
Birdie carrier	Frequency	A steady unwanted carrier
SMPS hash	Pulse rate	Switch-mode power supply hash
Woodpecker	Pulse rate	Over-the-horizon radar

Fault Injection buttons

Button	Effect
High SWR	Reports a high SWR
Drop Slice	Removes the slice
Stall Scope	Stops the spectrum stream
Disconnect	Drops the connection
Malformed	Injects a malformed message
Clear	Clears injected faults

3.8.5 Troubleshooting

AetherSDR Demo is not in the radio list

Show the AetherSDR demo simulator is unticked, and the choice is remembered.

1. Open **File** → **Connect to Radio...**
2. Tick **Show the AetherSDR demo simulator**.

3.8.6 See also

- Supported Radios ([p. 18](#))
- First Connection ([p. 11](#))
- Your First Session ([p. 5](#))
- DSP Noise Mitigation ([p. 241](#))

Your Computer

4.1 Linux

AetherSDR runs natively on Linux as a single-file AppImage, for x86_64 PCs and for aarch64 machines such as the Raspberry Pi. This page collects everything that is specific to Linux: installing, the features that differ from macOS and Windows, where your files live, and fixes for Linux-only problems. Everything else in these docs applies to Linux as written.

4.1.1 Requirements

	x86_64	aarch64 (Raspberry Pi, ARM laptops)
Download	AetherSDR-*-x86_64.AppImage	AetherSDR-*-aarch64.AppImage
Minimum system	Built on an Ubuntu 22.04 base, so it runs on that release and newer distributions	glibc 2.38 or newer. Raspberry Pi OS Trixie is the supported Pi baseline; Raspberry Pi OS Bookworm (glibc 2.36) cannot run it.
Graphics	OpenGL (Mesa or vendor drivers)	OpenGL; the 3D view works on the Raspberry Pi 5

The AppImage bundles Qt and AetherSDR's other libraries, so nothing else needs installing. Some features need a little help from the system:

- **PipeWire** for DAX Virtual Audio ([p. 128](#)).
- **A desktop keyring** (for example GNOME Keyring or KWallet) to remember passwords and tokens between sessions. See Settings and Backups ([p. 220](#)).
- **udev rules** for USB devices such as HID encoders and the Stream Deck+ (see Setup ([p. 43](#))).

4.1.2 Setup

Installing the AppImage

1. Download the AppImage for your machine from GitHub Releases¹.
2. Make it executable and run it:

```
chmod +x AetherSDR-*-x86_64.AppImage
./AetherSDR-*-x86_64.AppImage
```

3. The **Connect to Radio** window opens. Continue with First Connection ([p. 11](#)), or try Your First Session ([p. 5](#)) with no radio.

To check the download is genuine, verify its GPG signature as described in Installation ([p. 8](#)).

¹<https://github.com/aethersdr/AetherSDR/releases/latest>

The AppImage is self-contained. If you build from source instead and want an application-menu entry and icon, `sudo cmake --install build` installs the binary, `.desktop` file and icon. See [Building from Source](#) (p. 277).

Wayland and XWayland

The AppImage runs **natively on Wayland**, which renders sharply under fractional scaling. If native Wayland is not available it falls back to XWayland automatically. On a headless Wayland session with no monitor attached (for example a Raspberry Pi reached over VNC), it starts on XWayland (xcb), because native Wayland cannot render the panadapter there.

Set `QT_QPA_PLATFORM` yourself to override the choice. Your setting always wins:

```
QT_QPA_PLATFORM=xcb ./AetherSDR-*.AppImage      # force XWayland
QT_QPA_PLATFORM='wayland;xcb' ./AetherSDR-*.AppImage # force native Wayland
```

USB device permissions (udev)

Most USB accessories are root-only on Linux until a udev rule or group membership gives your user access.

Device	What to do	Page
RC-28, PowerMate, ShuttleXpress, ShuttlePro v2, Stream Deck+	Copy <code>packaging/linux/60-hid-encoders.rules</code> from the source tree to <code>/etc/udev/rules.d/</code> , then run <code>sudo udevadm control --reload-rules && sudo udevadm trigger</code>	USB Control Surfaces (p. 189), StreamDeck (p. 181)
Ulanzi Dial	Click Grant access in the mapper when it shows “... detected — needs permission”. AetherSDR installs <code>70-ulanzi-dial.rules</code> through <code>pkexec</code> .	Ulanzi Dial (p. 186)
CTR2-Max (USB mode)	Start offers to install the CTR2 udev rule if the device cannot be opened	CTR2 Proxy (p. 196)
ThumbDV / DV3000U	Official packages install <code>70-aethersdr-thumbdv.rules</code> ; otherwise join the serial group (below)	D-STAR (ThumbDV) (p. 115)
FlexControl knob, USB-serial PTT/CW interfaces	Add yourself to the serial-port group: <code>uucp</code> on Arch, <code>dialout</code> on Debian/Ubuntu/Fedora	FlexControl Tuning Knob (p. 183)

After changing your groups, log out and back in.

4.1.3 Using AetherSDR on Linux

What is different on Linux

- **Menus** are behind the `≡` button at the left of the title bar. Alt+letter mnemonics (Alt+F, Alt+S, ...) still open each menu. See [Menu Reference](#) (p. 235).
- **Radio Setup** is **Settings** → **Radio Setup**...
- **Window frame**: AetherSDR draws a frameless window with a 6 px resize band on the sides and bottom. **View** → **Frameless Window** (Ctrl+Shift+F) turns it off if your compositor mishandles it.
- **DAX virtual audio** is built in through PipeWire. Receive channels are native PipeWire streams, and they also appear through PipeWire’s PulseAudio and JACK layers. DAX IQ channels appear as capture devices named **AetherSDR DAX IQ n**. See [DAX Virtual Audio](#) (p. 128) and [DAX IQ Streaming](#) (p. 130).
- **CAT virtual serial ports** (PTY) are available as well as TCP ports, at `$XDG_RUNTIME_DIR/aethersdr/cat-A ... cat-H`. See [CAT Control](#) (p. 132).
- **Noise reduction**: **BNR** (NVIDIA Maxine, on an NVIDIA RTX/GeForce GPU) is available on Linux. **MNR** is macOS only and is greyed out. See [DSP Noise Mitigation](#) (p. 241) and [BNR GPU Noise Removal](#) (p. 248).

- **Copy Assist** can use a **Vulkan** GPU (NVIDIA, AMD or Intel). On a Raspberry Pi the **base** model is the default, and builds that include sherpa-onnx can run faster non-whisper models. See Copy Assist (p. 121).
- **Credentials** are kept in the desktop keyring (libsecret or KWallet), never in the settings database.
- **Screen reader**: Orca. See Accessibility (p. 231).

Software rendering

If the panadapter draws incorrectly on your GPU or driver, force software OpenGL without rebuilding:

```
AETHER_NO_GPU=1 ./AetherSDR-*.AppImage
```

This is worth trying first on a Raspberry Pi whose Mesa driver is newer than its hardware. See GPU Rendering (p. 252).

4.1.4 Reference

Where files live

What	Location
Settings database (AetherSDR.db), backups, quarantine	~/.config/AetherSDR/
Application logs and support bundles	~/.config/AetherSDR/logs/ (newest log also linked as aethersdr.log)
SpotHub source logs	~/.config/AetherSDR/spotify/
User themes	~/.config/AetherSDR/themes/
NR2 FFTW wisdom	~/.config/AetherSDR/aethersdr_fftw_wisdom
Copy Assist models	~/.local/share/AetherSDR/models/
BNR NVIDIA runtime	~/.local/share/AetherSDR/AetherSDR/nvidia-afx/current/
CAT virtual serial ports	\$XDG_RUNTIME_DIR/aethersdr/cat-A... cat-H

AetherSDR --config path prints the exact database path. The AETHER_SETTINGS_DIR environment variable points AetherSDR at a different configuration folder. See Settings and Backups (p. 220) and Support and Logging (p. 261).

Environment variables

Variable	Effect
QT_QPA_PLATFORM=xcb	Force XWayland
QT_QPA_PLATFORM='wayland;xcb'	Force native Wayland
AETHER_NO_GPU=1	Software OpenGL for the panadapter
AETHER_SETTINGS_DIR=<dir>	Use a different configuration folder
QT_LOGGING_RULES="aether.cat.info=true"	Raise a log category before the GUI opens

4.1.5 Known issues

- On Wayland, the GPU spectrum can intermittently freeze the whole window at startup, seen most on a Raspberry Pi 5 (#4704²).
- On Ubuntu 26.04 under native Wayland, the panadapter and waterfall can update very slowly (#4725³).
- On some X11 desktops with recent NVIDIA cards the AppImage shows no spectrum or waterfall and the Connect to Radio window never appears (#4998⁴).
- On a Raspberry Pi 5 with software rendering, the spectrum heat-map fill can use almost all of the main thread, so the window stops responding to clicks under extra load (#5192⁵).

²<https://github.com/aethersdr/AetherSDR/issues/4704>

³<https://github.com/aethersdr/AetherSDR/issues/4725>

⁴<https://github.com/aethersdr/AetherSDR/issues/4998>

4.1.6 Troubleshooting

AetherSDR will not start: error while loading shared libraries: libOpenGL.so.0

Ubuntu 26.04 no longer installs libopengl0 on the desktop image, and the GPU spectrum links against it.

1. Install it:

```
sudo apt install libopengl0
```

2. Start AetherSDR again. This is the GLVND desktop OpenGL runtime, not the legacy libgl1; both can be installed side by side.

The ARM AppImage does not run on a Raspberry Pi

The aarch64 AppImage needs glibc 2.38 or newer, and Raspberry Pi OS Bookworm has glibc 2.36.

1. Upgrade the Pi to **Raspberry Pi OS Trixie**.
2. Run the AppImage again.

A dialog crashes AetherSDR under XWayland (GLX BadAccess)

Some compositors produce this XWayland crash when a dialog opens.

1. Run AetherSDR natively on Wayland:

```
QT_QPA_PLATFORM='wayland;xcb' ./AetherSDR-*.AppImage
```

The panadapter is black over VNC or remote desktop

Native Wayland cannot render the panadapter on a headless session. AetherSDR detects a Wayland session with no display and switches to XWayland, unless you have set QT_QPA_PLATFORM yourself.

1. Remove your own QT_QPA_PLATFORM setting, or set QT_QPA_PLATFORM=xcb.
2. If the panadapter still draws wrongly, start with AETHER_NO_GPU=1.

PC audio misbehaves on a PipeWire system

The PipeWire library is installed but PipeWire's client configuration is not, so AetherSDR keeps Qt's audio off PipeWire at startup instead of crashing.

1. Install your distribution's PipeWire package: pipewire on Arch, pipewire-bin on Debian/Ubuntu.
2. Restart AetherSDR.

On source builds for Ubuntu or Linux Mint whose device lists show only "Dummy Output", see the distro notes in docs/BUILDING.md⁶.

A USB knob, dial or Stream Deck is not found

Your user cannot open the device until a udev rule or group grants access.

1. Install the rule or join the group listed in USB device permissions (p. 44).
2. Unplug the device and plug it back in.
3. If you changed groups, log out and back in (or reboot) before trying again.

Discovery finds no radio

A firewall is blocking discovery on UDP port 4992, or the radio is on another subnet.

1. Allow discovery through the firewall, for example `sudo ufw allow 4992/udp`.
2. If the radio is on another subnet or behind a VPN, use **Connect by IP**. See Manual Connection (p. 201).

⁵<https://github.com/aethersdr/AetherSDR/issues/5192>

⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md#distro-notes>

Passwords and tokens are forgotten after every restart

The build has no keychain support, or no desktop keyring is running, so credentials are kept for the current session only.

1. Use the AppImage, which includes keychain support.
2. Make sure a Secret Service keyring (for example GNOME Keyring) or KWallet is running in your session.

Reporting a crash

1. Run `coredumpctl list AethersDR`.
2. Run `coredumpctl info <PID> > aethersdr-crash.txt` and attach the text file to your report from **Help → File an Issue...**
3. Do **not** attach the core file itself: it can contain passwords and tokens.

See Troubleshooting (p. 256) and docs/debugging-crashes.md⁷.

4.1.7 See also

- Installation (p. 8)
- GPU Rendering (p. 252)
- DAX Virtual Audio (p. 128)
- Settings and Backups (p. 220)
- Support and Logging (p. 261)
- Troubleshooting (p. 256)
- docs/BUILDING.md⁸

4.2 macOS

AethersDR runs natively on macOS, with separate signed and notarized DMGs for Apple Silicon and Intel Macs. This page collects everything that is specific to macOS: installing, the features that differ from Linux and Windows, where your files live, and fixes for macOS-only problems. Everything else in these docs applies to macOS as written, with **Cmd** in place of **Ctrl** in shortcuts.

4.2.1 Requirements

	Apple Silicon	Intel
Download	AethersDR-*-macOS-apple-silicon.dmg	AethersDR-*-macOS-intel.dmg
macOS	14.4 (Sonoma) or newer	14.4 (Sonoma) or newer. Intel Macs that cannot run Sonoma cannot run current releases.
Panadapter rendering	GPU (Metal)	CPU renderer, deliberately, because the GPU path misbehaves on older Metal/OpenGL hardware
Copy Assist (speech-to-text)	Included, with Metal GPU acceleration	Not included in the DMG. Building from source on an Intel Mac still works.

4.2.2 Setup

Installing the DMG

1. Download the DMG for your Mac from GitHub Releases⁹.
2. Open it and drag **AethersDR** to **Applications**.

⁷<https://github.com/aethersdr/AethersDR/blob/main/docs/debugging-crashes.md>

⁸<https://github.com/aethersdr/AethersDR/blob/main/docs/BUILDING.md>

⁹<https://github.com/aethersdr/AethersDR/releases/latest>

3. Start AetherSDR from Applications. Both DMGs are signed and notarized by Apple, so Gatekeeper verifies them with no extra steps.
4. The **Connect to Radio** window opens. Continue with First Connection (p. 11), or try Your First Session (p. 5) with no radio.

Installing the DAX audio driver

DAX Virtual Audio (p. 128) on macOS uses the **DAX Virtual Audio Driver** that ships in the AetherSDR DMG. Install it from the DMG before you enable DAX. Without it, enabling DAX shows **DAX Audio Driver Missing**.

Allowing Input Monitoring

USB and Bluetooth HID controllers need macOS **Input Monitoring** permission. The first time AetherSDR opens one, macOS asks for it. Allow AetherSDR in **System Settings** → **Privacy & Security** → **Input Monitoring**.

This applies to the Ulanzi Dial (p. 186), the Stream Deck+ (StreamDeck (p. 181)) and the HID encoders in USB Control Surfaces (p. 189) (RC-28, PowerMate, Contour Shuttle, TMate 2). HID encoders are off by default in **Serial & Controllers** so that the prompt never appears unless you own one, and the Ulanzi Dial asks only once a dial is attached.

4.2.3 Using AetherSDR on macOS

What is different on macOS

- **Menus** are in the system menu bar at the top of the screen. See Menu Reference (p. 235).
- **Radio Setup** is **AetherSDR** → **Preferences...** (Settings → Radio Setup... on the other platforms). Search in it with ⌘F.
- **Window:** a native window with the standard traffic-light buttons, the tiling menu, Stage Manager and native full screen (Cmd+Ctrl+F). Cmd+M minimises the window.
- **DAX virtual audio** is built in through the DAX Virtual Audio Driver (above). See DAX Virtual Audio (p. 128).
- **CAT virtual serial ports** (PTY) are available as well as TCP ports, at ~/Library/Caches/AetherSDR/cat-A ... cat-H. See CAT Control (p. 132).
- **Noise reduction:** **MNR**, Apple's MMSE-Wiener spectral noise reduction on the vDSP/Accelerate framework, is available only on macOS. **BNR** (NVIDIA) is not available; use **DFNR** instead, which runs on any CPU. See DSP Noise Mitigation (p. 241).
- **Copy Assist** uses the **Metal** GPU on Apple Silicon. It is not in the Intel DMG. See Copy Assist (p. 121).
- **Credentials** are kept in the macOS Keychain, never in the settings database.
- **ThumbDV** appears as /dev/cu.usbserial-*. See D-STAR (ThumbDV) (p. 115).
- **Screen reader:** VoiceOver. AetherSDR holds animations still when **Reduce motion** is on. See Accessibility (p. 231).

4.2.4 Reference

Where files live

What	Location
Settings database (AetherSDR.db), backups, quarantine	~/Library/Preferences/AetherSDR/
Application logs and support bundles	~/Library/Preferences/AetherSDR/logs/ (newest log also linked as aethersdr.log)
NR2 FFTW wisdom	~/config/AetherSDR/aethersdr_fftw_wisdom
Copy Assist models	~/Library/Application Support/AetherSDR/models/
CAT virtual serial ports	~/Library/Caches/AetherSDR/cat-A ... cat-H
Crash reports	Console → Crash Reports , or ~/Library/Logs/DiagnosticReports/

To use the settings command line, run the program inside the app bundle, for example `/Applications/AetherSDR.app/Contents/MacOS/AetherSDR --config` path. See Settings and Backups (p. 220) and Support and Logging (p. 261).

4.2.5 Known issues

- Selecting the **Audio** page in **Preferences** can push the Preferences window behind the main window (#6277¹⁰).
- Without Input Monitoring permission, a Ulanzi Dial shows only “Disconnected”, with no hint that a permission is missing (#5247¹¹).
- DAX transmit through the macOS DAX driver can send audio in bursts, and can stop sending until AetherSDR is restarted; TCI is not affected (#5870¹²).
- Receive audio can become distorted after several minutes on some external audio interfaces until **PC Audio** is toggled (#6160¹³).
- Copy Assist can freeze the whole application for 15–20 seconds at a time on recent Apple Silicon Macs (#5107¹⁴).
- An RTL-SDR Blog V4 dongle shows only noise, with no signals (#6016¹⁵).

4.2.6 Troubleshooting

Enabling DAX shows “DAX Audio Driver Missing”

The DAX Virtual Audio Driver is not installed on this Mac.

1. Open the AetherSDR DMG again.
2. Install the **DAX Virtual Audio Driver** from it.
3. Enable DAX again.

A USB knob, dial or Stream Deck+ does nothing

AetherSDR does not have Input Monitoring permission, or HID encoders are switched off.

1. Open **System Settings** → **Privacy & Security** → **Input Monitoring** and allow AetherSDR.
2. For the RC-28, PowerMate, Contour Shuttle, TMate 2 or Stream Deck+, open **AetherSDR** → **Preferences...** → **Serial & Controllers** and turn on **Enable HID encoders / StreamDeck+ (RC-28, PowerMate, ShuttleExpress, ...)**.
3. Unplug the device and plug it back in.

Copy Assist is missing

You are running the Intel DMG, which ships without speech-to-text.

1. On an Apple Silicon Mac, install the Apple Silicon DMG.
2. On an Intel Mac, build AetherSDR from source to get Copy Assist. See Building from Source (p. 277).

BNR is not offered

BNR needs an NVIDIA GPU and is not available on macOS.

1. Choose **DFNR** in AetherRX’s **AetherNR** tab instead, or **MNR** for a native one-knob option. See DSP Noise Mitigation (p. 241).

A stored setting stops AetherSDR from starting

1. Quit AetherSDR.
2. In Terminal, run `/Applications/AetherSDR.app/Contents/MacOS/AetherSDR --config list` to find the value.

¹⁰<https://github.com/aethersdr/AetherSDR/issues/6277>

¹¹<https://github.com/aethersdr/AetherSDR/issues/5247>

¹²<https://github.com/aethersdr/AetherSDR/issues/5870>

¹³<https://github.com/aethersdr/AetherSDR/issues/6160>

¹⁴<https://github.com/aethersdr/AetherSDR/issues/5107>

¹⁵<https://github.com/aethersdr/AetherSDR/issues/6016>

3. Remove it with `--config unset <key>`. See Settings and Backups (p. 220).

Reporting a crash

1. Open **Console** → **Crash Reports**, or `~/Library/Logs/DiagnosticReports/`.
2. Attach the newest AetherSDR-*.ips file to your report from **Help** → **File an Issue...**. It names your user folder in file paths; edit that out if you prefer.

See Troubleshooting (p. 256) and docs/debugging-crashes.md¹⁶.

4.2.7 See also

- Installation (p. 8)
- DAX Virtual Audio (p. 128)
- DSP Noise Mitigation (p. 241)
- USB Control Surfaces (p. 189)
- Settings and Backups (p. 220)
- Troubleshooting (p. 256)

4.3 Windows

AetherSDR runs natively on 64-bit Windows, as an installer or as a portable ZIP. This page collects everything that is specific to Windows: installing, the features that differ from Linux and macOS, where your files live, and fixes for Windows-only problems. Everything else in these docs applies to Windows as written.

4.3.1 Requirements

Package	File	Notes
Installer	AetherSDR-* -Windows-x64-setup.exe	Setup wizard with a Start Menu shortcut and an uninstaller
Portable	AetherSDR-* -Windows-x64-portable.zip	No install. Extract anywhere and run AetherSDR.exe.

- Both packages carry the Microsoft Visual C++ runtime with the application. **No Visual C++ Redistributable install is needed**, and repairing or removing the system redistributable does not affect AetherSDR.
- The panadapter renders with **Direct3D 11**.
- For digital-mode programs, plan on the TCI Server (p. 135): AetherSDR ships no DAX audio driver on Windows (see below).

4.3.2 Setup

Installing

1. Download the installer or the portable ZIP from GitHub Releases¹⁷.
2. Run the installer, or extract the ZIP and run AetherSDR.exe.
3. The Windows builds are GPG-signed but not Authenticode-signed, so **Windows SmartScreen** may warn the first time you run them. Verifying the signature confirms the download is genuine; see Installation (p. 8) and docs/VERIFYING-RELEASES.md¹⁸.
4. The **Connect to Radio** window opens. Continue with First Connection (p. 11), or try Your First Session (p. 5) with no radio.

Both packages keep your settings in the same place, `%LOCALAPPDATA%\AetherSDR\`, so you can switch between them.

¹⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

¹⁷<https://github.com/aethersdr/AetherSDR/releases/latest>

¹⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/VERIFYING-RELEASES.md>

Connecting digital-mode programs

There is no built-in DAX audio driver on Windows. The DAX applet says so: “No built-in DAX driver on Windows. Use TCI, or SmartSDR DAX.”

- **TCI** carries audio and CAT over one connection and is the simplest route for WSJT-X. See TCI Server (p. 135) and WSJT-X Integration (p. 125).
- **FlexRadio’s own SmartSDR DAX** also works with a FlexRadio.
- **CAT** has no virtual serial ports on Windows. Point your program at one of the CAT applet’s TCP ports instead. See CAT Control (p. 132).

4.3.3 Using AetherSDR on Windows

What is different on Windows

- **Menus** are behind the ≡ button at the left of the title bar. Alt+letter mnemonics (Alt+F, Alt+S, ...) still open each menu. F11 toggles full screen. See Menu Reference (p. 235).
- **Radio Setup** is **Settings** → **Radio Setup...**
- **Window frame:** a Qt-drawn frame with the system shadow and resize borders, with AetherSDR’s own minimize, maximize and close buttons. Snap Layouts on hover over maximize does not appear yet.
- **GPU choice on hybrid laptops:** the discrete GPU is used by default. The integrated GPU is listed in **Display** → **SYSTEM** → **GPU**: but greyed out (“disabled (#1921)”), because it crashes when a panadapter is popped out or docked. See GPU Rendering (p. 252).
- **Noise reduction:** **BNR** (NVIDIA Maxine, on an NVIDIA RTX/GeForce GPU) is available on Windows; its runtime comes as a single self-contained download. **MNR** is macOS only and is greyed out. See BNR GPU Noise Removal (p. 248) and DSP Noise Mitigation (p. 241).
- **Copy Assist** can use a **Vulkan** GPU (NVIDIA, AMD or Intel). See Copy Assist (p. 121).
- **Credentials** are kept in Windows Credential Manager, never in the settings database.
- **USB devices** appear as COMn ports. A FlexControl knob that is unplugged and replugged may come back on a different COM port. The ThumbDV helper opens its port exclusively, so close any other program using it. See FlexControl Tuning Knob (p. 183) and D-STAR (ThumbDV) (p. 115).
- **Ulanzi Dial:** Windows cannot give AetherSDR exclusive control of the dial, so each key press also reaches Windows (for example as a media key). See Ulanzi Dial (p. 186).
- **Runtime Monitor:** Windows does not report per-thread state, so the State column shows a dash. See Runtime Monitor (p. 266).
- **Screen readers:** NVDA, Narrator and JAWS. See Accessibility (p. 231).

4.3.4 Reference

Where files live

What	Location
Settings database (AetherSDR.db), backups, quarantine	%LOCALAPPDATA%\AetherSDR\
Application logs and support bundles	%LOCALAPPDATA%\AetherSDR\logs\ (newest log also linked as an aethersdr.log shortcut)
NR2 FFTW wisdom	%APPDATA%\AetherSDR\aethersdr_fftw_wisdom
BNR NVIDIA runtime	%LOCALAPPDATA%\AetherSDR\AetherSDR\nvidia-afx\current\
Crash dumps (once enabled, below)	%LOCALAPPDATA%\CrashDumps\
SmartSDR firmware files, after installing SmartSDR	C:\ProgramData\FlexRadio Systems\SmartSDR\Updates\ (see Firmware Update (p. 232))

See Settings and Backups (p. 220) and Support and Logging (p. 261).

Environment variables

Set these in **System Properties** → **Environment Variables**, or in a Command Prompt before starting AetherSDR.exe from the same window.

Variable	Effect
AETHER_NO_GPU=1 or QT_OPENGL=software	Software OpenGL for every panadapter, instead of Direct3D 11
AETHER_SETTINGS_DIR=<folder>	Use a different configuration folder

4.3.5 Known issues

- On Windows 10 with **Frameless Window** on, click-to-tune, the panadapter right-click menu and the ≡ menu flash and close at once; turning off **View** → **Frameless Window** (Ctrl+Shift+F) avoids it (#6272¹⁹).
- On Windows 10 a thin white outline shows along the left, right and bottom window edges (#6266²⁰).
- The BNR runtime download can stall with an empty `.staging` folder under `nvidia-afx` and leave BNR unavailable (#4132²¹).
- The RC-28 encoder can take several seconds to change frequency; unticking **Use a Ulanzi Dial when detected** has cured it for some users (#6225²²).
- A Ulanzi D100H paired over Bluetooth can stay “Disconnected” in the mapper (#3485²³).
- Popping out a panadapter has crashed AetherSDR with an older AMD Radeon HD 7000 series graphics card (#5990²⁴).

4.3.6 Troubleshooting

Windows SmartScreen warns about AetherSDR

The Windows builds are GPG-signed but not Authenticode-signed, so SmartScreen does not recognise the publisher.

1. Verify the download’s GPG signature, following docs/VERIFYING-RELEASES.md²⁵.
2. If the signature is good, run AetherSDR anyway from the SmartScreen prompt.

WSJT-X or another digital-mode program gets no audio

There is no DAX audio driver on Windows.

1. Connect the program through the TCI Server (p. 135), which carries audio as well as CAT. See WSJT-X Integration (p. 125).
2. Or, with a FlexRadio, use FlexRadio’s SmartSDR DAX.

A logging or digital-mode program cannot find AetherSDR’s serial port

Windows has no CAT virtual serial ports.

1. In the CAT applet, enable a port with the **Rigctld**, **TS-2000** or **Flex** dialect.
2. Point the program at that TCP port on this computer. See CAT Control (p. 132).

The panadapter draws wrongly or shows “Spectrum renderer unavailable”

The Direct3D 11 renderer failed or your driver draws incorrectly.

1. Set `AETHER_NO_GPU=1` (or `QT_OPENGL=software`) and start AetherSDR again to use software OpenGL.
2. **Help** → **About AetherSDR** shows which renderer is in use. See GPU Rendering (p. 252).

¹⁹<https://github.com/aethersdr/AetherSDR/issues/6272>

²⁰<https://github.com/aethersdr/AetherSDR/issues/6266>

²¹<https://github.com/aethersdr/AetherSDR/issues/4132>

²²<https://github.com/aethersdr/AetherSDR/issues/6225>

²³<https://github.com/aethersdr/AetherSDR/issues/3485>

²⁴<https://github.com/aethersdr/AetherSDR/issues/5990>

²⁵<https://github.com/aethersdr/AetherSDR/blob/main/docs/VERIFYING-RELEASES.md>

A USB knob, encoder or Stream Deck+ does nothing

HID encoders are off by default.

1. Open **Settings** → **Radio Setup...** → **Serial & Controllers**.
2. Turn on **Enable HID encoders / StreamDeck+ (RC-28, PowerMate, ShuttleXpress, ...)**.
3. Unplug the device and plug it back in. See USB Control Surfaces ([p. 189](#)).

Reporting a crash

Windows keeps a crash dump only if local dumps are switched on.

1. In an administrator PowerShell, run:

```
$k = 'HKLM:\SOFTWARE\Microsoft\Windows\Windows Error  
Reporting\LocalDumps\AetherSDR.exe'  
New-Item -Force $k | Out-Null  
New-ItemProperty -Force $k -Name DumpType -PropertyType DWord -Value 1 | Out-Null
```

2. After the next crash, attach AetherSDR.exe.<pid>.dmp from %LOCALAPPDATA%\CrashDumps\ to your report from **Help** → **File an Issue...**, and say whether you ran the installer or the portable ZIP.

See Troubleshooting ([p. 256](#)) and docs/debugging-crashes.md²⁶.

4.3.7 See also

- Installation ([p. 8](#))
- TCI Server ([p. 135](#))
- WSJT-X Integration ([p. 125](#))
- GPU Rendering ([p. 252](#))
- Settings and Backups ([p. 220](#))
- Troubleshooting ([p. 256](#))

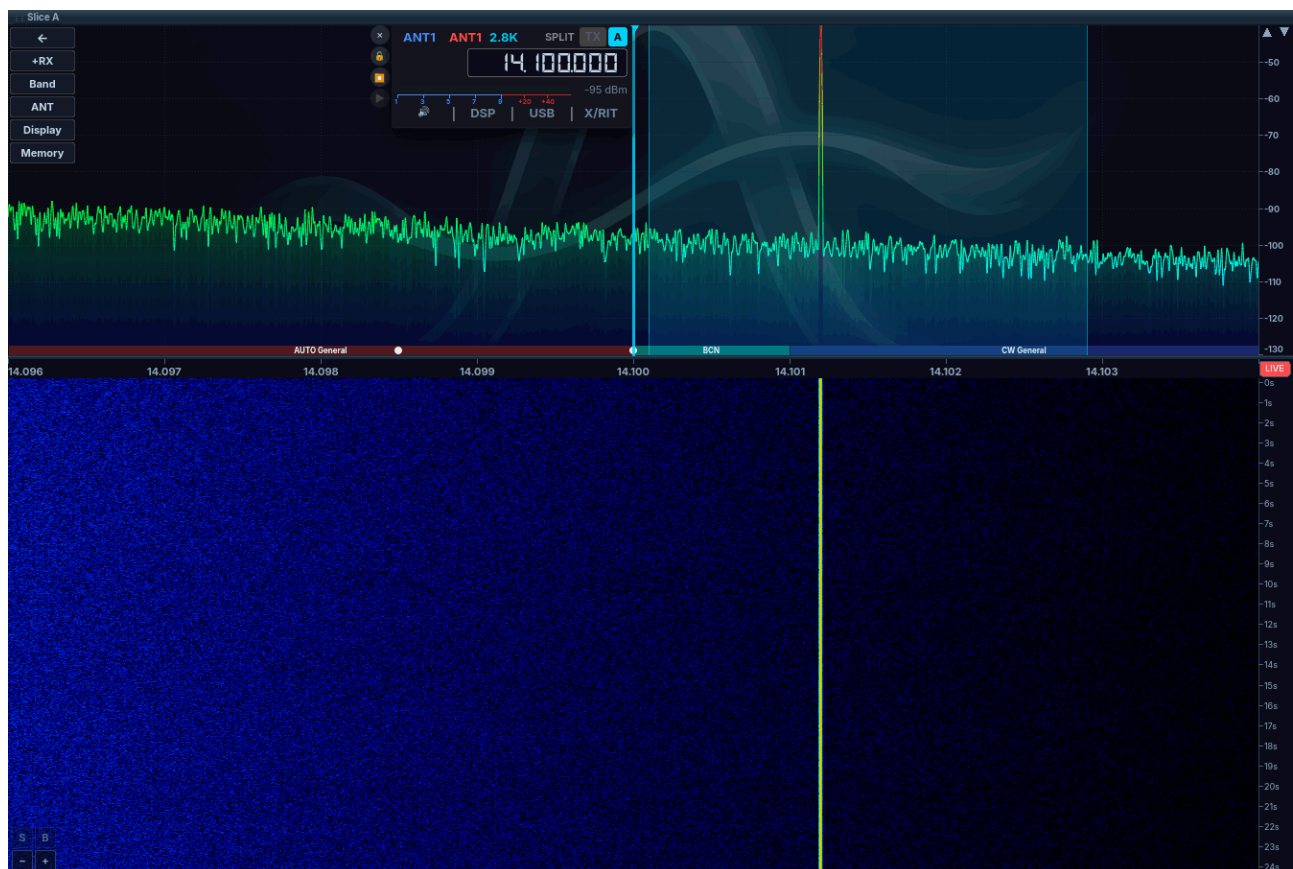
²⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

CHAPTER 5

Operating

5.1 Panadapter Controls

The panadapter is the main display: the FFT spectrum on top and the scrolling waterfall below. AetherSDR can show up to 8 panadapters (as many as the connected radio allows), stacked in the main window or popped out into their own windows.



A panadapter with its waterfall, the slice A VFO flag and the overlay menu down the left edge.

5.1.1 Using the panadapter

Mouse controls

Action	Spectrum / waterfall	Frequency scale	dBm scale	Waterfall time scale
Drag	Pan the display	Zoom the span	Shift the reference level	—
Ctrl+drag (Cmd on macOS)	—	—	Zoom the dB span (bottom stays anchored)	Change the waterfall scroll rate
Double-click	Tune to that frequency	—	—	—
Scroll wheel	Tune by the step size	—	—	—
Ctrl+scroll wheel	Zoom the span around the frequency under the cursor	—	—	—
Right-click	Context menu (below)	—	—	—

- **View → Single-Click to Tune** (off by default) makes a single click tune instead of a double-click.
- Clicking an empty part of a panadapter that has no slice on it adds a new slice there.
- Dragging a slice to the edge of the pan keeps panning the display.
- On a trackpad, vertical scrolling tunes, sideways swipes are ignored, and pinch-to-zoom changes the span where the system delivers a zoom gesture.
- If the wheel tunes the wrong way for your mouse or trackball, turn on **Radio Setup → Appearance & Behavior → Reverse mouse-wheel tuning direction**.
- Dragging any slider shows its value in a small popup.

Right-click menu

Right-click the spectrum or waterfall:

Item	What it does
Add Slice at X MHz	Opens a new slice at the clicked frequency.
Add TNF at X MHz	Places a tracking notch. See TNF (Tracking Notch Filters) (p. 90).
Add Spot at X MHz...	Adds your own spot.
Close Slice X	Closes the slice under the pointer.
Center Lock	Keeps this pan centred on a slice (below).
Link Slice ▶	Links two slices so tuning one retunes the other. See Multi-Slice Operation (p. 83).
Waterfall Time Markers ▶	Off, 15 seconds, 30 seconds, 1, 5, 10 or 15 minutes (below).
Show Tune Guides	Checkable display option.
Extended Frequency Line	Extends each slice's frequency line down through the waterfall.
Extended Passband	Extends each slice's passband shading down through the waterfall.
Extended TNF	Extends each notch down through the waterfall.
3D Slice Shadow	In 3D mode, draws slice shadows on the 3D surface.
↗ Pop out / ↖ Dock	Moves this panadapter into its own window, or back.

Right-click a **TNF marker** for Remove TNF, Width ▶, Depth ▶, Extended TNF and Make Permanent / Make Temporary. Right-click a spot you added for **Remove Spot**.

Center Lock

Center Lock keeps a panadapter centred on one slice as it tunes, independently for each pan.

- With one slice on the pan, the menu item reads **Center Lock Slice A** (checkable).
- With several slices, a **Center Lock** submenu lists them, active slice first.
- The lock survives band changes. When it is released, the pan shows “Center Lock released”.

- It can also be toggled from a keyboard shortcut (**Center Lock Active Slice**), MIDI, FlexControl and RC-28.

| **Upgrading from an older version:** Center Lock replaces the title-bar Pan Lock button.

Waterfall Time Markers

Thin lines across the waterfall at clock boundaries, each labelled with its UTC time. They stay pinned to their rows as the waterfall scrolls, pauses or resizes. Off by default; the choice is remembered per panadapter.

FFT/waterfall split

Drag the divider between the spectrum and the waterfall to resize them. The split is remembered.

dBm scale

- **Drag** the scale up or down to shift the reference level.
- **Ctrl+drag** (Cmd+drag on macOS) to zoom the dB span; the bottom stays anchored.
- Click the ▲ / ▼ arrows at the top to step the reference level by 10 dB.
- Labels use 1-2-5 steps.

3D Stacked Trace

Display → 3D VIEW → Spectrum: 3D Stacked Trace turns the spectrum into a perspective surface built from recent traces receding into the distance. Ridge height is measured from the noise floor and coloured from floor to peak, and single-frame impulse bursts are rejected. The waterfall, scales and every overlay (spots, memories, markers, band plan) still draw as normal, and the dBm scale stays on the right as a full-height axis. It works on KiwiSDR sources too, and uses the GPU where available (see GPU Rendering (p. 252)).

Band plan overlay

The strip along the bottom of the spectrum shows band segments, coloured by use. In the ARRL (US) plan, for example: blue for CW, red for data, orange for phone/SSB, cyan for beacons, purple for satellite, green for weak-signal SSB, and yellow for mixed or experimental segments. Dots mark spot frequencies (FT8, WSPR, QRP calling and so on); hover for a tooltip.

View → Band Plan sets:

- **Size:** Off, Small, Medium, Large or Huge.
- **Show Spots:** turns the spot dots on or off.
- **Region:** ARRL (US), IARU Region 1, IARU Region 2, IARU Region 3, RAC (Canada) or SSA (Sweden).

The ARRL and RAC plans mark licence classes by brightness. The ARRL plan's 60 m band follows the FCC order effective 13 February 2026. The selected plan is remembered.

Off-screen VFO indicator

When a slice is outside the visible span, an indicator at the edge shows its letter, TX status, a chevron and its frequency. **Click** it once to make that slice active and bring it into view. Right-click it for the slice menu.

Mini-Pan

Mini-Pan is a narrow, K4-style spectrum of the active slice in an applet (button **MINI**, off by default). It shows a ± 5 kHz window by default, or ± 10 kHz; right-click it to choose. The window is centred on the slice's passband (on SSB that is offset from the carrier); a hairline and readout mark the carrier.

It re-uses the main panadapter's data, so it uses no extra radio panadapter or slice, and its detail follows the main pan's zoom. It is an applet rather than a menu item so it is available in Minimal Mode; float it from its title bar to keep it on top of a logging program.

Multiple and detached panadapters

- **Tools → Add Panadapter...** or the **+PAN** icon in the status bar opens a layout picker and adds a pan if the radio has room.
- **Right-click → ↗ Pop out** moves a pan into its own window; **↶ Dock** puts it back.

- Pan layouts, including floating pans, are restored when you reconnect.
- With the Workspace Canvas (p. 98) on, **Import pop-outs onto canvas** places floating pans on the canvas.

Panadapter messages

Status cards appear over the spectrum for things like “Center Lock released”, a renderer problem, or KiwiSDR connection state. KiwiSDR status cards can be collapsed to a small pill but not closed.

Status bar

The status bar at the bottom of the window starts with the band-stack indicator and **+PAN** (add a panadapter), followed by indicators such as **TNF**, **CWX**, **ASR**, **DVK** and **FDX** (each only on radios and builds that support it). Click an indicator to toggle it.

View → FPS Meters (Ctrl+F) adds spectrum and waterfall frame-rate readouts to each pan.

5.1.2 Using the applet panel

The applet panel holds the radio controls (S-Meter, RX, TX, Phone, P/CW, EQ and many more).

Showing, hiding and moving the panel

Use the icons at the right of the title bar:

- **Dock left / Dock right** — put the panel on that side of the panadapters. Clicking the side it is already on hides it.
- **Pop out** — float the panel in its own window (also **Ctrl+Shift+S**). Popped-out applets have a pin for always-on-top.

The button bar

The top row of the panel holds **five favourite buttons** (by default VU, PWR, RX, TX and P/CW) and a ▼/▲ drawer button that shows the rest. Right-click the bar (or the drawer button) and choose **Customize Button Bar** to pick your favourites and hide buttons you don’t use; the first five active entries fill the bar and the rest go in the drawer. Hiding a button also closes its applet. Buttons for accessories you don’t have (tuner, amplifier, Antenna Genius and so on) appear only when that hardware is present.

Reordering applets

Drag an applet by its title bar (:: grip) to reorder it. The order is remembered. **View → Reset Applet Order** restores the default.

5.1.3 Reference

Screen layout

- **FFT spectrum** — real-time signal strength across the displayed span
- **Frequency scale** — between spectrum and waterfall, in MHz
- **Waterfall** — scrolling time-versus-frequency history
- **dBm scale** — right edge of the spectrum
- **Time scale** — right edge of the waterfall
- **Band plan** — coloured strip along the bottom of the spectrum
- **Overlay menu** — the button column at the left: **+RX**, **+TNF**, **Band**, **ANT**, **Display**, **Memory**, **DAX** (the DAX flyout holds the **IQ Ch** selector and the **WFM** demodulator, and only appears on radios with DAX)

Display panel

Click **Display** in the overlay menu. The panel is divided into sections and scrolls when the window is short. Settings apply to the panadapter you opened it from.



PANADAPTER

Heat Map Grid Wt Avg

FFT AVG: 0

FFT FPS: 25

FFT Line: 1.0

FFT Fill: 70

FFT Floor: Auto 75

WATERFALL

NB Blank: Off 1.15

Black Level: SW 50

WtrFall Gain: 50

WtrFall Rate: 100

BACKGROUND

Background: Choose... Clear Off

BG Opacity: 80

Color: 14.097 AUTO General 14.098

APPEARANCE

Grid: Auto

Scale text: 8 pt

Scheme: Default

3D VIEW

Spectrum: 2D Waterfall

3D Floor: 6

3D Gain: 70

3D Span: 100

SYSTEM

GPU: Auto (system default)

Clone to all Pans

Reset to Defaults

The Display panel, opened from the Display button on the panadapter overlay menu.

PANADAPTER

Control	Range / default	What it does
Heat Map	On by default	Colours the spectrum trace by signal strength instead of one colour.
Grid	On by default	Frequency and dB grid lines.
Wt Avg	Off by default	Weighted averaging that favours recent frames.
FFT AVG	0–100	Frame averaging. Higher is smoother but slower to react; 0 is unaveraged.
FFT FPS	5–60, default 25	Spectrum frame rate.
FFT Line	Off–5.0 px, default 1.0 px	Trace thickness, with its own colour button.
FFT Fill	0–100, default 70	Fill opacity under the trace, with its own colour button.
FFT Floor + Auto	1–99, default 75	With Auto on, the dBm scale follows the noise floor and keeps it at this height (% from top). Off by default.

WATERFALL

Control	Range / default	What it does
NB Blank + Off/On	5–95, default 15	Suppresses impulse-noise stripes in the waterfall. Higher values blank less.
Black Level + Off / SW / HW	0–100, default 50	Waterfall black point. The button cycles automatic black level: SW estimates the noise floor in AetherSDR and the slider sets the offset from it; HW uses the radio's own level (FlexRadio only; other radios cycle Off ↔ SW); Off is manual.
WtrFall Gain	0–100, default 50	Waterfall colour gain.
WtrFall Rate	1–100, default 100	Scroll speed: 1 is slowest, 100 fastest.

BACKGROUND

Control	What it does
Background: Choose...	Pick a background image. The picker can show small icons or thumbnails.
Clear	Back to the default logo background.
Off	No background image.
BG Opacity	0–100, default 80.
Color	Solid colour painted beneath the image.

APPEARANCE

Control	What it does
Grid	Frequency grid spacing (Auto adapts to the span).
Scale text	Size of the frequency-scale labels.
Scheme	Waterfall palette: Default, Grayscale, Blue-Green, Fire, Plasma, Purple, Glacier. Changing it recolours the waterfall already on screen, including scrollbar.

3D VIEW

Control	Range / default	What it does
Spectrum	2D Waterfall / 3D Stacked Trace	Switches to the 3D view (see 3D Stacked Trace (p. 56)).
3D Floor	0–24, default 6	How far below the noise floor the surface starts.
3D Gain	0–100, default 70	Ridge height.
3D Span	0–100, default 100	How far the nearest rows overhang the plot edges. 100 fills edge to edge; 0 gives a narrowing trapezoid.

SYSTEM

Control	What it does
GPU:	Which graphics adapter draws the spectrum. Only shown on computers with more than one GPU; takes effect after a restart (“Restart to apply”). See GPU Rendering (p. 252).
Clone to all Pans	Copies every Display setting from this pan to all other open pans: colours, fill, scheme, floor, 3D settings, background, and the radio-side values (FFT AVG, FPS, weighted average, waterfall rate, gain, black level).
Reset to Defaults	Resets this pan’s display settings, including turning the grid back on and the line width to 1.0 px.

5.1.4 Known issues

- Strong signals can show flat-topped, clipped peaks on the spectrum (#5685¹).
- With **FFT Floor Auto** off, the reference level can jump on its own when you tune to another frequency (#5525²).
- After switching profiles several times, **FFT Floor Auto** can stay highlighted but stop following the noise floor until you turn it off and on (#3186³).
- At a fractional UI scale (for example 85 %), mouse-wheel tuning can leave a stale, doubled strip along the top of the panadapter for about 15 seconds (#6270⁴).
- With **Frameless Window** off, a popped-out panadapter’s window title shows raw markup such as Slice A₁ (#6274⁵).
- The S-Meter applet can’t be dragged to a new place in the applet panel (#4347⁶).

5.1.5 Troubleshooting

Clicking the spectrum doesn’t tune

Tuning takes a double-click by default.

1. Double-click the frequency you want.
2. To tune with one click, turn on **View → Single-Click to Tune**.

The mouse wheel tunes the wrong way

Some mice and trackballs scroll the opposite way.

1. Open **Settings → Radio Setup... → Appearance & Behavior**.
2. Turn on **Reverse mouse-wheel tuning direction**.

The spectrum is drawn too low or too high

The reference level or dB span is set for a different band or signal level.

1. Drag the **dBm scale** up or down, or click its ▲ / ▼ arrows to step it 10 dB.
2. Or open **Display** and turn on **Auto** next to **FFT Floor**, so the scale follows the noise floor.

Impulse noise leaves stripes across the waterfall

Waterfall noise blanking is off or set too high.

1. Open **Display** in the overlay menu.
2. Turn on **NB Blank** under **WATERFALL** and adjust its slider. Higher values blank less.

My panadapters look different from each other

Display settings belong to the panadapter you opened them from.

¹<https://github.com/aethersdr/AetherSDR/issues/5685>

²<https://github.com/aethersdr/AetherSDR/issues/5525>

³<https://github.com/aethersdr/AetherSDR/issues/3186>

⁴<https://github.com/aethersdr/AetherSDR/issues/6270>

⁵<https://github.com/aethersdr/AetherSDR/issues/6274>

⁶<https://github.com/aethersdr/AetherSDR/issues/4347>

1. Open **Display** on the pan that looks right.
2. Click **Clone to all Pans** under **SYSTEM**.

A slice has disappeared off the edge of the display

The slice is outside the visible span.

1. Click the off-screen indicator at the edge of the spectrum. The slice becomes active and comes into view.
2. To keep a pan on one slice as it tunes, right-click and choose **Center Lock**.

A floating panadapter won't restore properly

The saved floating-window layout could not be put back as it was.

1. Turn on **View** → **Workspace Canvas** → **Enabled**.
2. Choose **View** → **Workspace Canvas** → **Workspaces** → **Import pop-outs onto canvas**. See Workspace Canvas (p. 98).

5.1.6 See also

- VFO Widget (p. 61)
- Multi-Slice Operation (p. 83)
- TNF (Tracking Notch Filters) (p. 90)
- GPU Rendering (p. 252)
- Workspace Canvas (p. 98)
- Menu Reference (p. 235)

5.2 VFO Widget

Every slice has a VFO widget (the “flag”) floating on the spectrum next to its frequency marker. It shows the frequency and a signal meter, and its tabs give quick access to the slice’s audio, DSP, mode, RIT/XIT and DAX settings.



The VFO flag for slice A, with its close, lock and record buttons on the left edge.

5.2.1 Using the VFO flag

Frequency display

- Shows the slice frequency in MM.KKK.HHH format.
- **Double-click** to type a frequency directly. Accepts 14.225 (MHz), 14225 (kHz), 14225000 (Hz) or 14.225.000 (dotted).
- On XVTR bands: 1446 = 144.6 MHz (the decimal goes after the third digit).

- Click the slice letter badge to **collapse** the flag to a compact label; click the collapsed label to expand it again. Right-click → Add Spot works in both forms.

Meter

Below the frequency is the signal meter. **Click it** to choose between the classic **S-Meter** bar and **SmartMTR**, which adds min/max extremes markers, numeric values and a transmit meter (Mic Level, SWR, Power or Compression). The choice applies to every flag. Full details are on Meters ([p. 72](#)).

Slice buttons

Beside the flag:

- ✕ — close the slice
- 🔒 / 📵 — lock the frequency (prevents tuning; a “LOCKED” notice appears if you try)
- ● — record slice audio (pulses red while recording)
- ▶ — play the recording (enabled once one exists)

Record & Play

Where recordings are made is set in **Settings** → **Radio Setup...** → **Audio** → **Recording**:

- **Client Side** (the default) records on this computer, including AetherSDR’s CW sidetone.
- **Radio Side** records in the radio (FlexRadio). On radios without a recorder, Radio Side is dimmed with the reason, and REC/PLAY record on this computer instead.

To transmit a radio-side recording as a voice keyer, press MOX first, then ▶.

Badges

- **TX** — red when this slice is the transmit slice; grey otherwise. Click a grey TX badge to move transmit to this slice.
- **SPLIT / SWAP** — split operation. Click **SPLIT** to create a transmit slice; on the transmit slice the badge reads **SWAP**. Right-click for the split menu. See Split Operation ([p. 85](#)).
- 🔗 — this slice is linked to another with **Link Slice**. See Multi-Slice Operation ([p. 83](#)).

Filter widen and narrow shortcuts

The **Filter Widen** / **Filter Narrow** actions (keyboard, MIDI or controller) change the filter width by 100 Hz in the right direction for the mode: USB, CWU, DIGU and RTTY move the upper edge; LSB, CWL and DIGL move the lower edge. See Keyboard Shortcuts ([p. 224](#)).

5.2.2 Reference

Tabs

Click a label along the bottom of the flag to open its tab. Controls the slice’s radio cannot use are dimmed, with the reason in the tooltip.

🔊 **Audio**

- **AF** slider with mute
- **Pan** (left/right) with a centre marker
- **SQL** — squelch on/off and level
- **AGC-T** — AGC mode (Off / Slow / Med / Fast) and threshold

DSP

The radio’s own noise-reduction and filter toggles for the current mode, with level sliders where they apply: **NR**, **NB**, **ANF**, **NRL**, **NRS**, **RNN**, **NRF**, **ANFL**, **ANFT**, plus **APF** in CW and **MN** (manual notch) on radios that have it. Right-click a button for a quick parameter popup.

The same tab has the **AetherRX** and **AetherTX** launchers, always the same width, which open AetherSDR's own receive and transmit audio chains. The client-side noise reduction methods (NR2, RN2, NR4, NNR, DFNR, BNR, MNR) live in **AetherRX**, not on this tab; see Aetherial Audio (p. 74) and DSP Noise Mitigation (p. 241).

The **DSP** tab label is colour-coded: green when any radio DSP or client noise reduction is active, grey when everything is off, cyan while the tab is open.

On FM, NFM, DFM and DSTR the tab becomes **OPT** (repeater offset and direction, simplex and reverse; CTCSS tones on FM, NFM and DFM only).

Mode tab

Labelled with the current mode (for example **USB**):

- mode selector and quick mode buttons;
- per-mode filter preset grid;
- **Marker**: button that cycles this slice's frequency marker Off → 1 px → 3 px, and a filter-edge toggle. These override the defaults set in **View** → **VFO Marker Size** (Off / 1 px / 3 px, default 3 px) and **View** → **VFO Filter Edge** (Show / Hide, default Show). Changing a View default updates every flag without overwriting a per-slice choice.

Adaptive RX filter (USB and LSB)

Under the filter presets in USB and LSB, **Adaptive RX filter** automatically fits the receive passband to the width of the signal you are hearing.

Control	Choices
Adaptive RX filter	On / off
Lo cut / Hi cut	The limits the fit may use
SNR	Sensitive, Normal (default), Strong
Speed	Fast, Normal (default), Slow
Splat	Tight, Normal (default), Wide — splatter rejection
Het	Off (default) / On — narrows away from a strong heterodyne at the passband edge

While a fit is applied, the filter readout shows **AUTO**. Adaptive fitting pauses while you transmit.

X/RIT

- RIT on/off, zero, step buttons and Hz readout
- XIT on/off, zero, step buttons and Hz readout

DAX

- **DAX Ch** — Off, or a DAX audio channel. The list follows the radio's capacity (up to 8 channels).

DAX IQ channels and the WFM satellite-data demodulator are in the panadapter's overlay **DAX** flyout, not on the flag. See DAX Virtual Audio (p. 128) and DAX IQ Streaming (p. 130).

5.2.3 Known issues

- With **Client Side** recording, the slice recording stops capturing audio when you transmit, so your own transmission is not recorded (#4141⁷).

5.2.4 Troubleshooting

Tuning does nothing and a “LOCKED” notice appears

The slice's frequency is locked.

1. Click the  button beside the flag to unlock it.

⁷<https://github.com/aethersdr/AetherSDR/issues/4141>

A control on the flag is dimmed

The slice's radio can't use that control.

1. Hover over the control; the tooltip gives the reason.

The filter readout says AUTO

Adaptive RX filter is on and is fitting the passband to the signal (USB and LSB only).

1. Open the mode tab (labelled with the current mode).
2. Turn **Adaptive RX filter** off to set the filter yourself.

► (play) is greyed out

There is no recording for this slice yet.

1. Click ● to record, then click it again to stop. ► is enabled once a recording exists.

The flag is a small label with no controls

The flag is collapsed.

1. Click the collapsed label to expand it again.

5.2.5 See also

- Slice Colors ([p. 230](#))
- Meters ([p. 72](#))
- Panadapter Controls ([p. 54](#))
- Split Operation ([p. 85](#))
- Diversity and ESC ([p. 87](#))
- Multi-Slice Operation ([p. 83](#))

5.3 RX Controls

The **RX** applet in the applet panel is the slice-centred receive control surface. It repeats the slice's identity at the top and gives you step size, filter, audio, squelch, AGC and RIT/XIT for the selected slice, with more room than the compact VFO flag. Radio-side DSP buttons (NR, NB, ANF, APF, NRL, NRS, RNN, NRF, ANFL, ANFT) are on the VFO flag's DSP tab; see VFO Widget ([p. 61](#)).

Most of this page describes a FlexRadio. On other radio families, controls the radio cannot honour are dimmed with the reason in their tooltip, or refuse with a notice; see Using the controls on other radios ([p. 67](#)) below.



The RX Controls applet for the active slice.

5.3.1 Using the RX applet

Header row

- **Slice badge** — the slice letter, in the slice’s colour (see Slice Colors (p. 230))
- **Lock icon** — tune lock
- **Antenna** — RX and TX antenna buttons. They show the local display names you set in **Settings** → **Radio Setup...** → **Antennas** (for example “80m Dipole”), while the radio still uses ANT1/ANT2.
- **Filter width** — current filter bandwidth (reads **AUTO** while the adaptive RX filter is fitting)
- **TX badge** — shows whether this slice transmits
- **Frequency** — double-click to enter directly

Step size

Step stepper with ◀ ▶ buttons. Available sizes: 10, 50, 100, 250, 500, 1000, 2500, 5000, 10000 Hz. Step sizes are mode-dependent.

Filter presets

Mode-dependent filter width buttons (FlexRadio):

- **USB/LSB**: 1.8K, 2.1K, 2.4K, 2.7K, 2.9K, 3.3K, 4K, 6K
- **CW**: 50, 100, 250, 400, 500, 600, 800, 1K
- **AM/SAM**: 5.6K, 6K, 8K, 10K, 12K, 14K, 16K, 20K
- **DIGU/DIGL**: 100, 300, 600, 1K, 1.5K, 2K, 3K, 6K
- **RTTY**: 250, 300, 350, 400, 500, 1K, 1.5K, 3K
- **DFM**: 6K to 20K in 2K steps
- **FM/NFM**: fixed by the radio

Radios that publish their own filter widths (such as networked Icom) show those instead.

Adaptive RX filter (USB/LSB)

An automatic filter that fits the passband to the station you are hearing. It is on the VFO flag’s mode tab, under the filter-width buttons, not in this applet: tick **Adaptive RX filter** and set the **Lo cut** / **Hi cut** bounds and the **SNR**, **Speed**, **Splat** and **Het** options. The filter readout shows **AUTO** while a fit is live, it is suspended while you transmit, and any manual filter change takes back control. See VFO Widget (p. 61).

AF gain, pan and mute

- **AF slider** (0–100) — slice audio volume
- **Pan slider** (L/R) — stereo balance with a centre marker
- **Mute** (speaker icon):
 - ▶ **single-click** mutes or unmutes this slice;
 - ▶ **double-click** mutes or unmutes **all** your slices.

AetherSDR waits briefly after a click to see whether it is a double-click. Set the wait in **Settings** → **Radio Setup...** → **Appearance & Behavior** → **Single-click delay** (0–1000 ms, with **Reset** for the platform default). At 0, single clicks act instantly and double-click actions are disabled.

Mute belongs to the radio: AetherSDR does not restore a slice’s mute itself when you reconnect.

Squelch

The **SQL** button cycles through three states:

State	Button	What the slider sets
Off	plain	—
SQL (manual)	lit	The squelch level

State	Button	What the slider sets
AUTO	amber, labelled AUTO	A margin of 5–20 dB above the measured panadapter noise floor (default 10 dB)

- In **AUTO**, squelch tracks the band’s noise floor as it changes, opening only for signals that stand the margin above it.
- In **SQL**, a threshold line shows on the panadapter while you adjust it and hides itself 3 seconds after you stop.
- Your manual level is remembered **per slice**, whichever control changes it (applet, VFO flag, MIDI or other controllers, the radio or another client). Auto tracking never overwrites it, so switching back from AUTO returns to your manual level.
- Switching a slice to **RTTY** turns squelch off, because squelch breaks the decode.
- The squelch scale, the panadapter SQL line and Auto SQL follow each radio’s own scale. On radios where Auto SQL is not available, a slice set to AUTO drops to manual at its manual level.

Upgrading from an older version: AetherSDR no longer saves the squelch level between sessions. On connect, each slice uses the level the radio reports.

AGC

- **Mode** — Off, Slow, Med, Fast
- **AGC-T slider** — the AGC threshold

AGC-T calibration: right-click the AGC-T slider, or use **Tools** → **Calibrate AGC-T...**, to calibrate the threshold against the measured noise floor on the active slice. It does not transmit.

On the Hermes-Lite 2, AGC runs on the computer: with AGC **Off**, the AGC-T slider sets a fixed gain (4–64 dB, default 10 dB) remembered per receiver, and **DIGU/DIGL open with AGC off**. See Hermes-Lite 2 (p. 23).

RIT / XIT

- **RIT** — Receiver Incremental Tuning (toggle, zero and step buttons)
- **XIT** — Transmitter Incremental Tuning (same layout)
- Step: 10 Hz per click

FM controls (FM/NFM)

When FM or NFM is selected the applet shows repeater controls:

- Repeater **Offset** (MHz)
- Offset direction: –, **Simplex**, +, and **REV** (listen on the repeater input)

Tone mode and tone value (CTCSS) are on the VFO flag’s FM tab. On networked Icom radios the **REV** button becomes **XFC**, a momentary transmit-frequency check.

5.3.2 Using client-side noise reduction

AetherSDR’s own noise-reduction methods run on your computer, one at a time: **NR2**, **RN2**, **NR4**, **MNR** (macOS only), **BNR** (NVIDIA RTX/GeForce GPU, Linux and Windows), **DFNR** and **NNR**. Choose a method and adjust it on the **AetherNR** tab of **Settings** → **AetherRX...**, which also opens from the **AetherRX** launcher on the VFO flag’s DSP tab. The launcher lights while a method is running. Right-clicking a DSP button gives a quick parameter popup.

- Each method processes the left and right channels separately, so pan and diversity are preserved.
- Client NR is for voice and turns itself off in CW, RTTY and digital modes.
- **BYPASS** at the foot of the AetherRX window turns off whichever method is running and restores it on the second click, for a quick A/B comparison.

See DSP Noise Mitigation (p. 241) and Aetherial Audio (p. 74).

5.3.3 Using CW receive aids (APF and decoder)

- **APF** (audio peaking filter) toggle and level are in the **P/CW** applet in CW modes, as well as on the VFO flag. The level slider is inactive until APF is switched on.
- The CW decoder panel appears below the waterfall in CW modes. See CW Decoder (p. 104).

5.3.4 Using WFM for satellite data

A software wide-FM demodulator for 9600-baud satellite data runs on the computer from DAX IQ. It is in the panadapter's **DAX** flyout: set **IQ Ch** first, then press **WFM** below it. The first time, choose a virtual audio cable as its output. It needs a radio with DAX IQ; other radios refuse with a notice. See DAX IQ Streaming (p. 130).

5.3.5 Using the controls on other radios

On the Hermes-Lite 2, ANAN-G2, RTL-SDR and networked Icom radios, every control either works through the radio's own route, is **dimmed with its reason** in the tooltip and screen-reader description, or **refuses with a notice** (once per session) before anything is sent. Examples that refuse when the radio cannot do them: antenna choices the radio has no port for, WFM, CWL, and NR/ANF from MIDI or keyboard shortcuts. On these radios the **NR cycle** keyboard/controller action goes straight to the client NR2 method, because the radio has no NR of its own. See Supported Radios (p. 18).

5.3.6 Known issues

- The squelch line, or Auto squelch, can appear in CW, RTTY and digital modes, where the SQL control is disabled (#6276⁸).
- Switching a slice from DIGU to USB can turn squelch on, silencing the audio until you switch it off (#3505⁹).
- On networked Icom radios, squelch can get stuck off once its level reaches 0, and only a non-zero change on the radio's front panel clears it (#6172¹⁰).

5.3.7 Troubleshooting

A slice is still muted after you reconnect

Mute belongs to the radio. AetherSDR does not change a slice's mute itself when you reconnect, so the slice keeps the mute state the radio reports.

1. Single-click the slice's **Mute** (speaker icon) in the RX applet to unmute it.
2. To unmute all your slices at once, double-click **Mute**.

Double-clicking Mute only mutes one slice

The single-click delay is set to 0, which disables double-click actions.

1. Open **Settings** → **Radio Setup...** → **Appearance & Behavior**.
2. Raise **Single-click delay** above 0, or press **Reset** for the platform default.

Squelch switched off when you changed to RTTY

This is by design: squelch breaks the RTTY decode, so switching a slice to RTTY turns it off.

1. When you leave RTTY, press **SQL** again if you want squelch back. Your manual level is remembered per slice.

The APF level slider does nothing

The level slider is inactive until APF is on.

1. Switch on **APF** in the **P/CW** applet or on the VFO flag.
2. Adjust the level.

⁸<https://github.com/aethersdr/AetherSDR/issues/6276>

⁹<https://github.com/aethersdr/AetherSDR/issues/3505>

¹⁰<https://github.com/aethersdr/AetherSDR/issues/6172>

5.3.8 See also

- VFO Widget ([p. 61](#))
- Slice Colors ([p. 230](#))
- DSP Noise Mitigation ([p. 241](#))
- Aetherial Audio ([p. 74](#))
- CW Decoder ([p. 104](#))
- Supported Radios ([p. 18](#))

5.4 TX Controls

Transmit is controlled from the **TX**, **Phone** (PHNE), **P/CW** and **EQ** applets, with the client-side voice chain in **Tools** → **AetherTX...** (see Aetherial Audio ([p. 74](#))). Receive-only radios (ANAN-G2 and RTL-SDR) do not transmit. Before you key up for the first time, read [Before You Transmit](#) ([p. 14](#)).



The TX Controls applet.

5.4.1 Using the TX applet

- **Fwd Power gauge** (0–200 W barefoot, 0–2000 W with a PGXL amplifier), with a **PEP** peak-hold tick
- **SWR gauge** (1.0–3.0)
- **RF Power slider** (0–100 %)
- **Tune Power slider** (0–100 %)
- **TX Profile dropdown** — select saved TX profiles
- **TUNE** — transmit a steady carrier for antenna tuning
- **MOX** — manual transmit
- **ATU** — run the antenna tuner
- **MEM** — use ATU memory
- **APD** — Adaptive Pre-Distortion (Active/Cal/Avail indicators). On FLEX-8000-series radios running fw 4.2.18+, APD also supports **External APD**: pick a sampler-port input via the sub-menu so the radio can train its predistortion against the output of an external linear amplifier rather than its own PA.

Hover **RF Pwr** or **SWR** for a floating badge with the exact value (for example 12 W or 1.12:1). Forward power drops to zero as soon as you unkey, and SWR does not show a stale reading while you receive.

An external amplifier or tuner adds its own applet: see [Amplifiers](#) ([p. 170](#)) and the PGXL section of [Peripherals](#) ([p. 166](#)) for amplifiers, and [TGXL Tuner Control](#) ([p. 173](#)) for the Tuner Genius XL.

TUNE

- **Right-click TUNE** to choose **Mono Tone** or **Two Tone** for the tune carrier.

- Changes to **Tune Power** made while TUNE is keyed take effect immediately, including on the Hermes-Lite 2 and networked Icom radios.
- TUNE and CW keying lock each other out: while TUNE is on, a paddle, straight key, keyboard CW or CWX text does not transmit, and TUNE will not start while a key is down or CWX is sending.
- **Settings → Inhibit during TUNE ▶** chooses which integrations are held while the radio is tuning.

ATU, MEM and pre-tuning

- **MEM** shows the tuner's real memory state as the radio reports it, not just your last click.
- ATU tune failures are reported in the status bar.

ATU band pre-tune sweep. Open it from **Tools → Pre-tune ATU Bands...** or right-click **ATU → Pre-tune bands....** It walks the band-plan segments of the bands you choose, tunes the ATU at each point and stores the result in the ATU memory, so later band changes recall a match instead of re-tuning. On channelised bands such as US 60 m it uses one point per channel.

- **MEM must be on**; otherwise the entry is disabled with “Enable MEM before running the pre-tune sweep”. It is hidden or disabled on radios without ATU memory, and a disabled entry says why.
- **Mode:** *Step (confirm each point)*, with Tune / Skip / Abort per point, or *Auto (run unattended)*.
- **License class** limits the points to sub-bands your class may use, where the active band plan has class data (the ARRL US plan has T/G/E).
- Safety nets: a warning with an estimated duration above 100 points; a warning when Auto-mode tune power is above 20 W; an audible cue on a per-point timeout; and after 3 consecutive failed or bypassed results on one band, the sweep skips the rest of that band and moves on.
- **The sweep keys the transmitter repeatedly. Make sure the bands are clear before you start.**

Clear ATU memories. **Tools → Clear ATU Memories...** or right-click **ATU → Clear ATU memories...** erases the stored tuner solutions after a confirmation.

5.4.2 Using the P/CW applet (Phone/CW)

- **Mic level gauge** (−40 to +10 dB) with peak hold
- **Compression gauge**
- **ALC gauge** — reads the software ALC meter; it appears in both the Phone and CW views
- **Mic profile dropdown** — select mic profiles
- **Mic source** — MIC, BAL, LINE, ACC, PC. Choose **PC** to use a computer microphone and Aetherial Audio (p. 74)'s AetherTX chain.
- **Mic level slider** (0–100), remembered across launches
- **+ACC** — accessory input toggle
- **PROC** — speech processor (NOR/DX/DX+)
- **DAX** — DAX audio toggle
- **MON** — monitor + volume slider

Hover the Level, Compression or ALC gauge for its exact value. In CW modes the applet switches to keyer controls, including the APF toggle and level.

5.4.3 Using the Phone applet

- **AM Carrier slider** (0–100)
- **VOX** toggle + level + delay
- **DEXP** — downward expansion (noise gate on the radio's mic path)
- **TX Filter** — Low Cut / High Cut step buttons

TX filter warning

If the TX low and high cut leave out your transmit audio, the radio keys but puts out almost no RF. AetherSDR measures the loss across the radio's TX filter and, when it reaches 40 dB or more, shows a card on the panadapter naming the passband and the measured attenuation, once per transmission. This matters most in DIGU/DIGL, where WSJT-X reports a normal cycle at zero power. CW is excluded, and the card is not available on the Hermes-Lite 2.

5.4.4 Using the EQ applet

- 8-band graphic equalizer (63 Hz – 8 kHz)
- ± 10 dB vertical sliders per band
- Independent **RX** and **TX** views; the view you chose is remembered
- **ON** toggle + reset button

5.4.5 Transmitting

- Press **MOX** (or use PTT) to transmit
- The VFO marker turns orange/red during TX
- The waterfall switches to FFT-derived rows during TX, scrolling at the same rate as on receive, masks the TX passband on every overlapping panadapter, and holds its dBm range until you return to receive
- Interlock problems that block transmit (amplifier warming up, ATU busy, band not allowed) appear as notifications, only when transmit is actually blocked

Transmit timer

While you transmit, a green M:SS timer (H:MM:SS after an hour) appears in the title bar, just left of the **PC Audio** button. It counts each key-down from 0:00, holds the final time for 15 seconds after you unkey and then fades out. It runs for your own transmissions (MOX, PTT, footswitch, VOX, CW and tune), never for TCI or DAX transmissions from other programs.

5.4.6 Using AetherTX (Aetherial Audio)

The client-side transmit voice chain — gate with RN2, EQ, split-band de-esser, compressor with Drive/Phase, tube, AetherVoice exciter, reverb and the final output stage with brickwall limiter, DC block, test tone and Quindar tones — lives in the **AetherTX** window, **Tools** → **AetherTX...** AetherTX needs a computer microphone (Mic source **PC**).

See Aetherial Audio (p. 74) for the stage reference and profiles.

Quindar tones

The AetherTX Final Output stage has a **QUIN** button that plays the Apollo-era chirp when you key and unkey: a 2525 Hz intro and a 2475 Hz outro.

Right-click it to open the editor:

- **Style** — Tone (classic chirp) or Morse (keyed K / BK)
- **Level** — -20 to 0 dBFS
- **Tone fields** — Intro / Outro frequency (2400–2700 Hz), duration (100–500 ms)
- **Morse fields** — WPM (20–60), pitch (400–1200 Hz)
- **Test intro / Test outro** — audition locally without keying

5.4.7 Known issues

- With **Mic source** set to **PC**, transmit audio can be silent after startup until you switch the source to **BAL** and back to **PC** (#5442¹¹).

¹¹<https://github.com/aethersdr/AetherSDR/issues/5442>

- On a remote FLEX-8600 over SmartLink, **TUNE** may not work while AetherSDR is connected to Genius peripherals (Antenna Genius, PGXL, TGXL) (#6001¹²).

5.4.8 Troubleshooting

The radio keys but puts out almost no RF in DIGU/DIGL

The TX filter's low and high cut leave out your transmit audio. When the loss reaches 40 dB or more, AetherSDR shows a card on the panadapter naming the passband and the measured attenuation.

1. In the **Phone** applet, use the **TX Filter Low Cut** and **High Cut** buttons to widen the filter until it covers your audio tones.
2. Transmit again and check that the warning card no longer appears.

Mic source is PC, but VOX does not open and transmit audio is missing

The computer's selected microphone (**Input**) is not your microphone, for example a virtual "DAX Mic" device.

1. Open **Settings** → **Radio Setup...** → **Audio**.
2. Choose your real microphone as the **Input** device.
3. Check that the **Mic level** gauge in the **P/CW** applet moves when you speak.

TUNE does not start

TUNE and CW keying lock each other out. TUNE will not start while a key is down or CWX is sending.

1. Release the paddle or straight key.
2. Wait for CWX to finish sending, or stop it.
3. Press **TUNE** again.

Pre-tune bands is disabled

The ATU pre-tune sweep needs ATU memory, so the entry is disabled with "Enable MEM before running the pre-tune sweep".

1. Turn on **MEM** in the TX applet.
2. Open **Tools** → **Pre-tune ATU Bands...** again. If the entry is still disabled, hover it: the radio may have no ATU memory.

A notification says transmit is blocked

An interlock is holding transmit off: for example an amplifier warming up, the ATU busy, or a band the radio does not allow.

1. Read the notification for the reason.
2. Wait for the amplifier or ATU to finish, or move to a band you may transmit on, then key again.

5.4.9 See also

- Before You Transmit (p. 14)
- Aetherial Audio (p. 74) — the AetherTX voice chain and its profiles.
- Meters (p. 72) — the cross-needle Power & SWR (PWR) applet and the HLTH antenna health trend.
- Amplifiers (p. 170) and TGXL Tuner Control (p. 173) — external amplifier and tuner applets.
- Split Operation (p. 85) — transmitting on a second slice.
- PSK Reporter Map (p. 155) — includes a one-shot **WSPR beacon** that transmits a single WSPR frame.

¹²<https://github.com/aethersdr/AetherSDR/issues/6001>

5.5 Meters

AetherSDR has several ways to read signal strength, power, SWR and the radio's health: the S-Meter applet, a meter on every VFO flag, a cross-needle power/SWR meter, Radio Vitals, Antenna Health, and hover readouts on the transmit gauges. Most meter applets live in the **Metering** group of the applet button bar.

Applet button	Title	What it shows
VU	S-Meter	Analog S-meter that switches to a transmit meter while you transmit
PWR	Power & SWR	Cross-needle forward/reflected power meter
MTR	Radio Vitals	PA temperature, supply voltage, fan speed
HLTH	Antenna Health	SWR and power trend with a health score (in <i>Antennas & Switching</i>)
LP100	LP-100A Meter	TelePost LP-100A wattmeter readings (see Peripherals (p. 166))

To add an applet to the bar or move it into the favourites row, right-click the button bar and choose **Customize Button Bar** (see Panadapter Controls (p. 54)).

5.5.1 Using the meters

S-Meter applet (VU)

An analog needle meter in the applet panel. While receiving it shows the selected receive meter; while transmitting it switches to the selected transmit meter.

Right-click the meter for its options:

Section	Choices
Face theme	Aether default (default), Classic warm, Dark-room upright, Graphite dark
TX Select	Power, SWR, Level, Compression
RX Select	S-Meter, S-Meter Peak
Peak Hold	Enabled (off by default)
Decay speed	Fast, Medium (default), Slow
Reset Peak Hold	Clears the held peak

Meter on the VFO flag (S-Meter and SmartMTR)

Every VFO flag has a meter strip under the frequency. **Click the strip** to open a selector with two views:

- **S-Meter** — the classic bar with S1–S9 and +20/+40/+60 markings.
- **SmartMTR** — a bar with analog-style needle ballistics and these extra options:

Option	Choices
Show extremes	Draws sliding-window minimum and maximum markers
Extremes speed	Slow, Medium, Fast (needs Show extremes)
Show values	None, Signal, Extremes
TX meter	None, Mic Level, SWR, Power, Compression — shown while transmitting (MOX, PTT or VOX)
Show meter type	Labels the TX meter with which quantity it is showing

The choice applies to every flag and is remembered. See VFO Widget (p. 61).

Power & SWR cross-needle meter (PWR)

A classic cross-needle wattmeter: separate **forward** and **reflected** needles, with printed SWR curves (1.1, 1.2, 1.3, 1.4, 1.5, 1.7, 2, 2.5, 3, 4, 8, ∞). Read SWR where the needles cross. The applet is off by default; open it from the **PWR** button.

- **Range** is automatic: $\times 1$ (20 W forward / 4 W reflected full scale) up to 20 W, $\times 10$ up to 200 W, and $\times 100$ above 200 W or whenever an amplifier is active. The face shows “RANGE 20 W $\times 1$ / 200 W $\times 10$ / 2 kW $\times 100$ ”.
- Pop it out from its title bar to get a floating meter window; the window can be shrunk down to a compact size.

Right-click the meter:

Section	Choices
Face theme	Classic warm, Dark-room uplight (default), Graphite dark
Display	Show Range (on by default)

The geometry behind the curves is documented in docs/cross-needle-meter-math.md¹³.

Radio Vitals (MTR)

Gauges for the radio’s hardware: **PA Temp**, **Supply Voltage** and **Main Fan**, each with its live value printed on the gauge. The °C / °F button in the “Radio Hardware” header switches the temperature unit and is remembered. On radios that report PA current, that gauge reads **PA Current** instead and the °C/°F button is hidden. The supply-voltage row is hidden on radios that do not report it.

Antenna Health (HLTH)

A scrolling trend of **SWR**, **RL** (return loss, computed from SWR) and **PWR** (smoothed forward power) while you transmit, with incident markers.

- A **0–100 health score** combines SWR, return loss, variance and power sag.
- The state reads **IDLE**, **RF IDLE**, **OK**, **WATCH** or **GROUND?**. **VAR** is the recent SWR spread; a wide spread can point to grounding or counterpoise problems.
- **SRC** shows where the reading comes from: **AMP** (Power Genius XL), **TUN** (Tuner Genius XL) or **RAD** (the radio), whichever is freshest.
- Click the graph to pause or resume it.

See also AetherSweep (p. 158) for an SWR sweep across the band.

Hover readouts on the transmit gauges

Hover over the TX applet’s **RF Pwr** or **SWR** gauge, or the P/CW applet’s mic **Level**, **Compression** or **ALC** gauge, to get a small badge with the exact value (for example 12 W or 1.12 : 1). It updates live, only one shows at a time, and it disappears about half a second after the pointer leaves. See TX Controls (p. 68).

5.5.2 Known issues

- The cross-needle **PWR** meter can read a different SWR from the TX applet’s SWR gauge and from external meters (#4436¹⁴).
- **Antenna Health** can show flat lines and no SWR or power during TUNE with a Tuner Genius XL in line (#4511¹⁵).
- The S-Meter applet can’t be dragged to a new place in the applet panel; it stays at the top (#4347¹⁶).

5.5.3 Troubleshooting

A meter applet isn’t in the button bar

The button is in the drawer or has been hidden. Buttons for accessories appear only when that hardware is present.

1. Click the ▼ drawer button in the button bar and look for it there.
2. If it isn’t there either, right-click the button bar, choose **Customize Button Bar** and turn the button on.

¹³<https://github.com/aethersdr/AetherSDR/blob/main/docs/cross-needle-meter-math.md>

¹⁴<https://github.com/aethersdr/AetherSDR/issues/4436>

¹⁵<https://github.com/aethersdr/AetherSDR/issues/4511>

¹⁶<https://github.com/aethersdr/AetherSDR/issues/4347>

The S-Meter needle holds a high reading

Peak Hold is on.

1. Right-click the S-Meter and choose **Reset Peak Hold**, or untick **Peak Hold** to turn it off.

The Antenna Health graph has stopped scrolling

The graph is paused. Clicking it pauses or resumes it.

1. Click the graph once to resume it.

5.5.4 See also

- VFO Widget (p. 61) (the per-flag meter)
- TX Controls (p. 68) (transmit gauges)
- Amplifiers (p. 170) (amplifier power and status meters)
- Peripherals (p. 166) (LP-100A and other external meters)
- AetherSweep (p. 158)
- Cross-needle meter maths¹⁷

5.6 Aetherial Audio

Aetherial Audio is AetherSDR's client-side audio processing. It runs on your computer, independent of the radio's own DSP, and comes as two windows:

- **AetherTX** — the transmit voice chain: gate, EQ, compressor, split-band de-esser, tube, AetherVoice exciter, reverb and a brickwall limiter. Open it from **Tools** → **AetherTX...**
- **AetherRX** — the receive chain: noise reduction, gate, EQ, compressor, tube, AetherVoice and an output meter. Open it from **Settings** → **AetherRX...**

The two chains keep completely separate settings, so you can shape your microphone for transmit and the band for listening without one disturbing the other. A docked **Aetherial Audio** applet in the applet panel gives you a compact view of both chains (see Using the docked applet (p. 78)).

Upgrading from an older version:

- The old *Aetherial Audio Channel Strip* window is gone. Its transmit half is now AetherTX and its receive half is AetherRX.
- Channel-strip presets are retired. The first time you open the new windows, each saved preset is split once into a TX profile and an RX profile, so nothing is lost.
- The **LIM** final limiter on the transmit Final Output stage is **off by default**, so drive level matches SmartSDR. Turn it on if you relied on it.
- **BYPASS** is remembered across restarts.
- The compressor's **Drive** and **Phase** are saved in AetherTX profiles.

¹⁷<https://github.com/aethersdr/AetherSDR/blob/main/docs/cross-needle-meter-math.md>



The AetherTX window: the transmit processing chain on the left, the selected stage's controls on the right.

5.6.1 Requirements

AetherTX needs a computer microphone. AetherRX has no requirements: it processes the receive audio the radio already sends you.

AetherTX processes a microphone connected to **this computer**. A mic plugged into the radio bypasses it. The window shows a notice that names the fix whenever the audio path is wrong:

- On a FlexRadio, set **Microphone source** to **PC** in the Phone/CW (P/CW) applet.
- On a radio that takes its transmit audio over the network (such as a networked Icom), turn on **PC Audio** in the title bar.

While the path is wrong, the docked Aetherial Audio applet replaces its TX controls with the same guidance, and AetherTX and the TX stage editors refuse to open. Receive views stay available. Fixing the path restores the controls.

DAX/TCI transmit audio (WSJT-X, fldigi and other digital-mode programs) and RADE digital voice never pass through AetherTX, so their tones reach the radio unshaped.

5.6.2 Using the AetherTX and AetherRX windows

AetherTX and AetherRX share one layout:

- **The tab column on the left is the chain.** Each stage has a row with an enable checkbox and a drag grip. Tick the box to switch the stage on, and drag a row to change the order the audio meets the stages. The order is saved.
- Click a row to show that stage. Every stage page has the same shape: switches along the top, the display under them, and every knob in one row at the foot.

- Knobs accept typed values as well as dragging.
- The foot of the tab column holds the live controls: a **REC / PLAY** pair, **BYPASS**, and a **Settings** gear for the profile library.

Right-clicking a DSP button elsewhere in the app (for example on the VFO flag) still opens a quick parameter popup without opening either window.

5.6.3 Using AetherTX

Signal path

The transmit voice chain runs at a fixed **48 kHz in floating point**. The microphone is captured at whatever rate the device negotiates, converted to float once and brought to 48 kHz. After the chain, a single high-quality conversion takes it to the radio's 24 kHz transport rate, with dither, before the one final conversion to 16-bit.

PC mic → (RN2) → Gate → EQ → De-esser → Compressor → Tube → Exciter → Reverb
→ Final Output (Quindar, limiter, meters) → radio

The middle seven stages are the reorderable chain; the order above is the default.

Stages

Tab	What it does
Gate	Downward expander or noise gate. Also holds the RN2 microphone noise reducer.
EQ	Parametric EQ with an optional reference target curve.
De-esser	Split-band sibilance control.
Compressor	Compressor with Drive/Phase peak-to-average (PAPR) controls and its own brickwall limiter.
Tube	Dynamic tube saturator.
Exciter	AetherVoice™ harmonic exciter.
Reverb	Freeverb room reverb. Off by default.
Final Output	Final limiter (LIM), DC block, test tone, Quindar tones, output meters with HOLD , and a waveform scope.

REC / PLAY: the transmit monitor

REC records up to 30 seconds of processed transmit audio without keying the radio. Click it again to stop early; playback then starts by itself. **PLAY** plays the capture back; click it during playback to cancel. The receive audio is muted while recording and playing back so you hear only your own voice.

Use it to tune the chain without putting a signal on the air: record a test phrase, listen, adjust a stage, record again. A phrase such as “The quick brown fox jumps over the lazy dog. Five or six big jet planes flew over the quiet countryside” covers most sounds.

Each recording is also written to `pudu_monitor.wav` in the system temporary folder, overwritten every time, so you can inspect it in an audio editor.

BYPASS

BYPASS switches every transmit stage off at once, including RN2, so the microphone reaches the radio unprocessed. Click it again to restore exactly the stages that were on. It is the quickest A/B test of what the chain is doing.

MIC and TX indicators

At the foot of the tab column, **MIC** lights when the PC microphone is selected and DAX is off, which means the chain really is in the transmit path. **TX** lights while you are transmitting on your own slice.

5.6.4 Using AetherRX

AetherRX processes the audio the radio is already sending you, so there is no routing to set up. The radio's own AGC, noise blanker and noise reduction run first; AetherRX adds finishing on top.

Stages

Tab	What it does
AetherNR	The client-side noise-reduction methods (NR2, RN2, NR4, MNR, BNR, DFNR and NNR) with their full settings. Always first. See DSP Noise Mitigation (p. 241).
Gate	Expander or gate, useful for knocking down hiss between overs.
EQ	Parametric EQ. The page also draws the slice's receive filter edges; dragging an edge changes the receive filter.
Compressor	Gentle levelling between loud and quiet stations.
Tube	Saturation for warmth, in small doses.
Exciter	AetherVoice presence and body.
Final Output	Output trim, MUTE and BOOST , output meter and waveform scope. Always last.

Gate, EQ, Compressor, Tube and Exciter are the reorderable chain. There is no de-esser or reverb on receive.

REC / PLAY and TX Playback

AetherRX's **REC / PLAY** pair records and plays back received audio through the same path as the record and play buttons on the VFO flag.

Right-click **PLAY** for **TX Playback**, which transmits the last client-side recording on the active slice. Select it again to stop. Playback is capped at the idle timeout set in the recording settings (**Settings** → **Radio Setup...** → **Audio**), and the transmitter is released if anything refuses or the radio disconnects.

BYPASS

RX **BYPASS** suppresses every receive stage at once, AetherNR included, so you hear the radio's audio unprocessed. Click it again to restore whatever was on, including the noise-reduction method that was running.

What is useful on receive

- **EQ:** pull harshness out of bright stations, add body to thin ones, tame a fatiguing midrange bump over a long contest. Boosting above your receive filter is wasted.
- **Gate in Expander mode:** knocks down inter-syllable hiss on a noisy band without the pumping of a hard gate.
- **Compressor at a low ratio** (1.5:1 to 2:1, 2–4 dB of reduction): evens out strength differences in a pile-up so you are not riding the AF gain.
- **Tube and Exciter:** a little warmth or presence on thin stations. Easy to overdo.

What to avoid: heavy compression lifts weak signals and noise along with strong ones, which is the radio's AGC job; and high tube drive just adds harmonic mud to a noisy band.

5.6.5 Using profiles

The **Settings** gear at the foot of each window opens its profile library (**AetherTX Settings** or **AetherRX Settings**). A profile is the whole chain: every stage's settings, which stages are on, and their order. An AetherRX profile also records the active noise-reduction method.

Button	Action
Save...	Store the chain as it is now under a name (asks before replacing an existing name)
Load	Apply the selected profile (double-click does the same)
Delete	Remove the selected profile
Import...	Read a profile from a JSON file and save it under a name you choose
Export...	Write the selected profile to a JSON file to share or back up

Loading a TX profile leaves the receive side alone, and the other way round.

Individual settings live in AetherSDR's settings database; you can inspect them with **Settings** → **Settings Browser...** (see Settings and Backups (p. 220)).

5.6.6 Using the docked applet

The **Aetherial Audio** applet in the applet panel (button **TXDSP**) is the compact companion to the two windows:

- **TX / RX** selects which chain is shown, and **BYPASS** bypasses that chain.
- The chain strip shows each stage as a box. Single-click a box to switch the stage on or off, double-click it to open that stage's editor, and drag it to reorder. The delay used to tell a single click from a double click is **Settings → Radio Setup... → Appearance & Behavior → Single-click delay**.
- On the TX side, ● / ► drive the same transmit monitor as AetherTX's REC / PLAY.
- Compact tiles for individual stages sit below the strip.

5.6.7 Reference

The ranges below are the knob ranges in the current windows.

Gate

Kills room noise, fan rumble and mic hiss in your pauses (TX), or hiss between overs (RX).

Mode: Gate (hard, snaps to 10:1 ratio and -40 dB floor) or **Expander** (gentle, snaps to 2:1 and -15 dB). Switching mode changes only ratio and floor; every other knob stays where you put it.

View: Level shows the live level history; **Curve** shows the static transfer curve.

RN2 (AetherTX only): the RNNoise microphone denoiser. It runs ahead of every chain stage, so noise is removed before the gate, compressor or tube can amplify it. Voice modes only, saved in the profile, and suppressed by **BYPASS**.

Peek: look-ahead, in steps of Off, 1, 1.5, 3 and 5 ms, so the gate opens before a transient instead of clipping its start.

Knob	Range	What it does
Thresh	-80 to 0 dB	Level below which the gate begins to close
Return	0 to 20 dB	Hysteresis: opens above Thresh, closes below (Thresh - Return)
Hold	0 to 500 ms	Time the gate stays open before release starts
Release	5 to 2000 ms	How fast the gate closes after the signal drops
Floor	-80 to 0 dB	Maximum attenuation the gate applies
Ratio	1 to 10 : 1	Steepness of the expansion

Example: SSB voice with room noise. Start in **Expander**. Set **Thresh** just above your noise floor and below your quietest syllables (around -35 dB), **Release** about 80 ms, **Hold** about 20 ms so sibilants don't stutter, **Floor** -15 dB so the room never goes completely dead, and **Return** about 3 dB so the gate doesn't flutter when your voice hovers near the threshold. If it pumps audibly, lengthen Release.

EQ

A parametric EQ with ten bands by default (high-pass, low shelf, peaking bands, high shelf and low-pass), each with frequency, gain and Q. Drag band handles on the response curve, or select a band and type values.

Toolbar controls:

- **Ref**: overlays an amber reference target curve: Off, AT&T 1959, Heil DX, Astatic D-104, Shure 444 or Heil HC-5. It is a visual guide only; it does not change the audio.
- **Smoothing**: fractional-octave smoothing of the analyser trace (display only).
- **Peak Hold**: freezes the analyser's peak trace.
- Filter family for the high-pass and low-pass slopes: **Butterworth**, **Chebyshev**, **Bessel** or **Elliptic**.
- **Reset** returns every band to its default.

When AetherTX's EQ is in use, leave the radio's own TX EQ flat.

Example: male voice, dynamic mic. High-pass at 80 Hz to remove handling noise; -3 dB around 200 Hz to cut mud; -2 dB around 500 Hz to scoop boxiness; +3 dB at 2 kHz and +2 dB at 3.5 kHz for presence; low-pass at your SSB filter width (about 3 kHz). For a bright condenser mic, do the opposite: a little low-mid warmth, pull back 2-4 kHz, roll off above 4 kHz.

De-esser (AetherTX)

Suppresses harsh “sss” and “shh” energy without dulling the rest of the voice. It is **split-band**: when the sibilant band crosses the threshold, only that band is turned down, so the rest of your voice keeps its level and your average power does not drop.

Control	Range	What it does
Freq	1 to 12 kHz	Centre of the sibilant band
Q	0.5 to 5	Width of the band (higher = narrower)
Threshold	-60 to 0 dB	Sidechain level where reduction begins
Amount	-24 to 0 dB	Maximum reduction of the sibilant band
Attack	0.1 to 30 ms	How fast reduction engages
Release	10 to 500 ms	How fast it recovers
Slope button	12 / 24 / 36 / 48 dB/oct	Steepness of the sibilant band; click to cycle. Default 24 dB/oct

Finding your sibilant frequency: sweep **Freq** while saying “sister silver cellophane” and park where the sibilance is loudest. Start with **Amount** around -6 dB and go further only if you still hear harshness. Too low a threshold leaves the de-esser working all the time and dulls the voice.

Compressor

Levels the signal: brings quiet syllables up and holds loud ones back.

- **Threshold:** the vertical fader on the left, which doubles as the input meter, so you can set the threshold against your own voice peaks.
- **Makeup:** drag the handle on the **Out** meter (-12 to +24 dB). It can also be adjusted from the keyboard.
- **Limiter / Ceiling:** the compressor’s own brickwall peak limiter and its ceiling (-24 to 0 dB, default -1 dB). The **Limiter** button glows red while it is clamping. This is separate from the **LIM** limiter on Final Output.

Knob	Range	What it does
Ratio	1:1 to 20:1	How hard to compress above the threshold
Attack	0.1 to 300 ms	Transient response
Release	5 to 2000 ms	Recovery speed
Knee	0 to 24 dB	Soft-knee width (0 = hard knee)
Drive	0 to +18 dB	AetherTX only. Gain before the threshold so the compressor works harder; makeup follows it, so average power rises with the peaks. Default 0
Phase	Off, 1-6 stages	AetherTX only. All-pass phase rotator that evens out lopsided voice peaks before compression, lowering peak-to-average ratio. 4 is the broadcast default

Drive and Phase together are the PAPR controls: they raise your average transmitted power without pushing peaks past the ceiling. They are hidden on receive.

Example: broadcast-style SSB. Threshold about -20 dB, Ratio 4:1, Attack 20 ms (keeps consonants crisp), Release 150 ms, Knee 6 dB, makeup until peaks sit around -3 dB on the Out meter, Limiter on with the ceiling at -1 dB. Aim for 3-6 dB of gain reduction on peaks; more than 10 dB is obvious compression that suits contests better than rag-chews.

Tube

Analog-style saturation for warmth and harmonic colour.

Model: **A** (symmetric, neutral warmth), **B** (hard clip into soft saturation, edgier) or **C** (asymmetric, even harmonics, warmest).

Control	Range	What it does
Drive	0 to 24 dB	How hard the signal hits the saturator
Tone	-1 to +1	Tilt filter: negative darkens, positive brightens
Bias	0 to 100 %	Asymmetry of the curve (more even harmonics)
Envelope	-100 to +100 %	Dynamic drive: negative eases drive on peaks, positive adds to it
Release	10 to 500 ms	Envelope-follower release
Output	-24 to +12 dB	Output gain
Dry/Wet	0 to 100 %	Blend of dry and saturated signal (parallel saturation)

Example: subtle warmth. Model A, Drive 6 dB, Tone +0.2, Bias 15 %, Dry/Wet 40 %, Output about -1 dB to match unity gain.

Exciter (AetherVoice™)

Adds sparkle at the top and body at the bottom. **Even** generates even harmonics with asymmetric shaping (warmer); **Odd** generates odd harmonics with symmetric shaping (edgier, cuts through noise).

Section	Knob	Range	What it does
Low	Drive	0 to 24 dB	Low-frequency enhancement drive
Low	Tune	50 to 160 Hz	Low-frequency corner
Low	Mix	0 to 100 %	How much low-frequency enhancement is added
High	Tune	1 to 10 kHz	High-frequency corner
High	Air	0 to 24 dB	Amount of high-frequency harmonic generation
High	Mix	0 to 100 %	How much high-frequency enhancement is added

The AetherVoice wordmark glows with the wet signal level, confirming the exciter is actually adding energy.

Pitfall: stacking the exciter on top of a bright EQ shelf and heavy tube drive gives a harsh, hyped sound. Pick one place to add brightness.

Reverb (AetherTX)

Freeverb room reverb. Reverb trades intelligibility for size, so it ships **off** and suits rag-chew or AM more than DX or contesting.

Knob	Range	What it does
Size	0 to 100 %	Room size
Decay	0.3 to 5 s	Tail length
Damp	0 to 100 %	High-frequency absorption in the tail (higher = warmer)
PreDly	0 to 100 ms	Gap before the tail starts
Mix	0 to 100 %	Dry/wet blend

Example: subtle broadcast space. Size 35 %, Decay 0.8 s, Damp 70 %, PreDly 15 ms, Mix 8 %. Keep the reverb last in the chain; a compressor after it squashes the tail. Pre-delay adds directly to the delay of the wet signal, which you will notice on quick overs.

Final Output (AetherTX)

The last stage before the radio.

Control	What it does
LIM	Final brickwall limiter at the end of the TX chain. Off by default

Control	What it does
DC	25 Hz high-pass that strips DC offset
TONE	1 kHz test tone injected at the head of the chain; right-click for frequency and level
QUIN	Quindar tones on PTT engage and release; right-click for the editor (below)
Trim	Output gain, -12 to +12 dB
HOLD	When lit, the PK / RMS / GR / CRST readouts latch the worst value since you engaged it. Click again to release

The horizontal meter shows the input peak, the output peak, gain reduction and the limiter ceiling as a draggable handle. **OVR** latches red on a clip and clears itself; **LIMIT** lights while the limiter is clamping. The waveform scope below the meter shows what you are sending.

Quindar tones

NASA's Apollo-era voice-loop chirp: a 2525 Hz tone when you key and 2475 Hz when you unkey. Right-click **QUIN** to set:

- **Style:** Tone (the classic chirp) or Morse (sends "K" and "BK").
- **Level:** -20 to 0 dBFS.
- **Tone:** intro and outro frequency (2400–2700 Hz) and duration (100–500 ms).
- **Morse:** speed (20–60 WPM) and pitch (400–1200 Hz).
- **Test intro** / **Test outro** audition the tone locally without keying.

Final Output (AetherRX)

Output **Trim** (-12 to +12 dB), **MUTE** and **BOOST**, with a level meter and a waveform scope, so you can match the processed receive path against the dry one.

Starting points by mic and voice

Tune from here; don't just load and transmit. Record yourself with REC/PLAY after every change.

Dynamic broadcast mic (e.g. SM7B), male voice

Stage	Setting
Gate	Expander, Thresh -40, Ratio 2.5, Release 80 ms, Floor -15
EQ	HPF 80, -3 @ 200, +3 @ 2k, +1 @ 3.5k, LPF 3.1k
De-esser	Freq 5k, Q 2, Threshold -25, Amount -6
Compressor	Ratio 4:1, Attack 25, Release 150, makeup +5, Limiter on at -1
Tube	Model A, Drive 6, Tone +0.2, Dry/Wet 40 %
Exciter	Even, low Mix 20 %, high Air 6, high Mix 20 %
Reverb	Off, or Size 30 %, Decay 0.7 s, Damp 70 %, PreDly 15 ms, Mix 6 %

Condenser mic, female voice

Stage	Setting
Gate	Expander, Thresh -45, Ratio 2, Release 60 ms
EQ	HPF 100, -2 @ 400, -2 @ 3k, +2 @ 5k, LPF 3.1k
De-esser	Freq 7k, Q 2.5, Threshold -22, Amount -8 (condensers are sibilance-heavy)
Compressor	Ratio 3:1, Attack 20, Release 120, makeup +4
Tube	Model C, Drive 9, Bias 20 %, Dry/Wet 45 %
Exciter	Even, low Mix 15 %, high Mix 10 %
Reverb	Off

Contest / DX punch

Stage	Setting
Gate	Gate mode, Thresh -35, Release 100 ms
EQ	HPF 200, +2 @ 500, -3 @ 1k, +5 @ 2.5k
De-esser	Freq 5k, Q 2, Threshold -20, Amount -9
Compressor	Ratio 8:1, Attack 10, Release 100, makeup +8, Limiter on; try Drive +6 and Phase 4
Tube	Model B, Drive 12, Dry/Wet 60 %
Exciter	Odd, high Air 12, high Mix 40 %
Reverb	Off

5.6.8 Known issues

- With the default AetherTX chain, transmit output can be well below the radio's rated power, and reaching full PEP needs a high **Trim** (#2876¹⁸).
- Turning on **RN2** in AetherTX can sharply reduce your transmit audio level and RF output (#4859¹⁹).
- With **Mic source** set to **PC**, transmit audio can be silent after startup until you switch the source to **BAL** and back to **PC** (#5442²⁰).

5.6.9 Troubleshooting**You changed a knob and nothing happened**

The stage is switched off, the chain is bypassed, or (on AetherTX) the chain is not in the transmit path.

1. Check that the stage's checkbox is ticked in the tab column.
2. Check that **BYPASS** is not lit.
3. On AetherTX, check that **MIC** is lit. If it is not, fix the audio path as described in Requirements (p. 75).

AetherTX won't open

The audio path is wrong. The message names the fix.

1. On a FlexRadio, set **Microphone source** to **PC** in the **P/CW** applet.
2. On a radio that takes its transmit audio over the network, turn on **PC Audio** in the title bar.

The monitor plays silence

REC only captures when the PC microphone is selected and DAX is off.

1. Set **Microphone source** to **PC**.
2. Switch DAX off.
3. Check that **MIC** is lit, then press **REC** again.

Your audio sounds worse, not better

You are probably over-processing.

1. Press **BYPASS** and listen to the dry signal.
2. If the dry signal is fine, switch stages off.
3. Start again with EQ and gentle compression, and add stages one at a time, recording with **REC / PLAY** after each change.

¹⁸<https://github.com/aethersdr/AetherSDR/issues/2876>

¹⁹<https://github.com/aethersdr/AetherSDR/issues/4859>

²⁰<https://github.com/aethersdr/AetherSDR/issues/5442>

Reordering the chain did nothing audible

Order matters most between dynamics stages. Gate before EQ and gate after EQ put the noise floor in different places; compressor before tube compresses the dry signal, compressor after tube compresses the saturation harmonics.

1. Move the gate, compressor or tube relative to each other.
2. Record with **REC / PLAY** before and after the change, and compare.

BYPASS didn't restore your previous state

BYPASS restores the stages that were on when you pressed it. A stage you switched by hand while bypassed counts as a manual change.

1. Release **BYPASS**.
2. Switch the stages you want back on by hand.

5.6.10 See also

- TX Controls ([p. 68](#)) — the TX, Phone and P/CW applets around the chain.
- RX Controls ([p. 64](#)) — receive controls, squelch and AGC.
- DSP Noise Mitigation ([p. 241](#)) — the noise-reduction methods on AetherRX's AetherNR tab.
- Audio Settings ([p. 217](#)) — PC audio devices and recording.
- Settings and Backups ([p. 220](#)) — where profiles and settings are stored.

5.7 Multi-Slice Operation

AetherSDR supports multiple simultaneous receivers (slices), on one panadapter or spread across several.

5.7.1 Requirements

The number of slices and panadapters comes from the connected radio, not from a built-in table, so AetherSDR offers exactly what your radio can do. Other radio families have their own limits: the Hermes-Lite 2 has up to four receivers (A–D), the RTL-SDR and ANAN-G2 one, and the IC-9700 one panadapter. See Supported Radios ([p. 18](#)).

5.7.2 Using multiple slices

Adding a slice

- Click **+RX** in the panadapter's overlay menu
- Or right-click an empty part of the panadapter → **Add Slice at X MHz**
- Clicking an empty panadapter also creates a new slice there instead of moving the active one
- A new slice appears with its own coloured VFO marker (see Slice Colors ([p. 230](#)))

To add another panadapter, use **Tools** → **Add Panadapter...**

Switching the active slice

- **Click the slice badge** on the spectrum to switch focus
- **Click the VFO marker line** to switch
- The VFO widget and the RX applet follow the active slice

TX assignment

- TX stays on its assigned slice until you move it
- **Red TX badge** — active TX slice
- **Grey TX badge** (in the VFO widget) — click to move TX to this slice
- Switching the active slice does **not** move TX
- The **Cycle TX Slice** keyboard and controller action steps TX through your slices

Split

Click **SPLIT** on a slice to listen there and transmit on a second slice beside it. See Split Operation (p. 85) for SWAP, Split Up, Monitor TX and remembered audio.

Slice Link

Two slices, even on different panadapters, can be tuned together. Right-click a panadapter → **Link Slice**: the submenu lists **Current Links** (choose one to unlink) and **Available Links** (choose a pair to link).

- Tuning either slice retunes the other, including KiwiSDR-sourced slices.
- Linked slices show a $\rightleftharpoons$ marker.
- The link is suspended while either slice is tune-locked; the panadapter says “Slice Link A to B suspended”.

Center Lock

Right-click a panadapter → **Center Lock Slice A** (or the **Center Lock** submenu when there are several slices) keeps that panadapter centred on the chosen slice as it tunes. Each panadapter locks independently, and the lock survives band changes. See Panadapter Controls (p. 54).

Closing a slice

- Click the **X** button floating beside the VFO widget
- Or right-click the slice on the spectrum → **Close Slice X**
- You cannot close the last remaining slice

Off-screen slices

When a slice is off the visible span, an arrow on the spectrum edge shows its letter, direction and frequency. **Single-click** the arrow to bring the slice back into view; right-click it for the slice menu.

Hermes-Lite 2

On the Hermes-Lite 2 the sample rate is shared by the whole board, so every panadapter shows the same span. Only the pane holding the TX slice (or the first docked pane) keeps live span buttons; the others are dimmed with the reason. See Hermes-Lite 2 (p. 23).

5.7.3 Known issues

- Adding a panadapter or a slice can make a loud pop in the speakers for about a second (#4973²¹).
- Clicking an off-screen slice’s arrow brings the slice to the edge of the panadapter rather than the centre (#2371²²).
- On the Hermes-Lite 2, receivers B, C and D don’t come back after AethersDR restarts; only one receiver appears (#5777²³).

5.7.4 Troubleshooting

A slice won’t close

It is the last slice; AethersDR always keeps one. The status bar says “Cannot close the last slice”.

1. Add another slice first if you want to replace this one, then close the old one.

Transmit stayed on the old slice

Switching the active slice does not move TX.

1. Click the grey **TX** badge on the slice you want to transmit on.

Linked slices stopped tuning together

One of the slices is tune-locked, so the link is suspended (“Slice Link A to B suspended”).

²¹<https://github.com/aethersdr/AetherSDR/issues/4973>

²²<https://github.com/aethersdr/AetherSDR/issues/2371>

²³<https://github.com/aethersdr/AetherSDR/issues/5777>

1. Click  beside that slice's VFO flag to unlock it. The link resumes.

A click added a new slice instead of moving the one I have

Clicking an empty panadapter (one with no slice on it) creates a slice there.

1. To retune an existing slice, use the panadapter that slice is on, or close the new slice with ✕.

5.7.5 See also

- Split Operation (p. 85)
- VFO Widget (p. 61)
- Slice Colors (p. 230)
- Panadapter Controls (p. 54)
- Diversity and ESC (p. 87)
- Supported Radios (p. 18)

5.8 Split Operation

Status: Supported · **Applies to:** FlexRadio

Split operation means listening on one frequency and transmitting on another, the usual way to work a DX station or a pile-up. AetherSDR does it with two slices: your receive slice and a transmit slice beside it on the same panadapter.

5.8.1 Using split

Starting and ending a split

1. On the slice you are listening with, click **SPLIT** on its VFO flag.
2. AetherSDR makes that slice the **receive** slice and creates a **transmit** slice on the same panadapter, **1 kHz up in CW** and **5 kHz up** in other modes. Transmit moves to the new slice.
3. The receive slice's SPLIT badge turns **red**. The transmit slice's badge becomes **SWAP**.
 - **SWAP** exchanges the two frequencies without changing which slice transmits, so you can jump to the DX station's listening frequency and back.
 - Click the red **SPLIT** badge again to end the split. The transmit slice is removed.

The two VFO panels of a split pair stay on opposite sides of the dual-VFO layout so they never cover each other. The SPLIT/SWAP badges follow each slice's actual role, so a diversity child slice is never mistaken for a split partner (see Diversity and ESC (p. 87)).

The badge menu

Right-click either badge (SPLIT or SWAP) for the controls that go with split operation.

Split Up 1 / 5 / 10 kHz

Moves the transmit slice that far above the receive slice. The receive frequency stays where it is. With no split running, the same choice starts one at that offset. **Split Up 1 kHz**, **Split Up 5 kHz** and **Split Up 10 kHz** are also keyboard shortcut actions (see Keyboard Shortcuts (p. 224)).

Monitor TX

Chooses what the **Monitor TX (Hold)** key does while you hold it:

- **Solo TX frequency** silences the receive slice so you hear only where you are about to transmit, like XFC, TF-SET or TXW on a conventional transceiver.
- **Hear both** makes both slices audible for the hold, the way a second receiver would.

Releasing the key puts both mutes back as they were. **Monitor TX (Hold)** is **not bound by default**: set a key in **Settings → Configure Shortcuts...** (and tick **Settings → Keyboard Shortcuts**, since operating shortcuts are off by default), or put it on a FlexControl, RC-28 or HID controller, where it toggles instead of holding. The menu shows the current binding, or “Not bound — set a key in Keyboard Shortcuts”.

Split QSY Option

Split QSY Option ▶ Close split on QSY (off by default) closes a split that AetherSDR created when something outside AetherSDR moves the receive slice (Slice A) by more than a threshold. The threshold is **1 to 200000 Hz, default 2 kHz**.

This is for a logging program that sends CAT FA / ZZFA directly to the radio through its own connection: when the logger QSYs you to a new spot, the old split goes away instead of transmitting on a stale frequency.

- Echoes of AetherSDR’s own tunes (its CAT, TCI, on-screen controls and SWAP) keep the split open.
- Moves at or below the threshold keep the split open and become the new reference point.
- Any other unmatched change to Slice A looks the same as the external CAT case and closes the split.
- Closing removes the transmit slice and returns TX to Slice A. It does not wait for you to finish transmitting.

Forget remembered audio

Clears the remembered audio arrangement (below). The menu also shows a summary of what is remembered, for example “TX unmuted, TX left, TX 40 %, RX right”.

Remembered audio

The transmit slice starts muted. If you change that (unmute it, set its level, pan it to one ear and the receive slice to the other), the next split you start comes back the same way, and keeps doing so until you change it.

- Only the transmit slice’s mute, level and pan are remembered, plus the receive slice’s pan if you moved it.
- The receive slice’s own volume and mute are never touched, and its original pan is restored when the split ends.
- Only your own edits teach it: the VFO flag, applets, keyboard and controllers. Changes arriving over CAT, TCI or from the radio do not.
- If you mute the transmit slice yourself before ending a split, the remembered arrangement is cleared, so muting it once returns to the original behaviour.
- Splits started by a logging program or another client over CAT, Hamlib or TCI are left alone.

Split from logging and digital-mode software

- Split is reported **per CAT client**. Several WSJT-X instances, each on its own CAT port, can all use **Split Operation: Fake It** at once. See CAT Control (p. 132) and WSJT-X Integration (p. 125).
- Over TCI, a client’s explicit split request still selects the transmit slice, so satellite and cross-band split work. See TCI Server (p. 135).

Controllers

FlexControl, AetherControl and other controller buttons can run **Split Active Slice** and **Monitor TX Frequency** actions. See FlexControl Tuning Knob (p. 183).

On radios that cannot do split, split controls and controller bindings refuse with a notice instead of silently doing nothing.

5.8.2 Known issues

- After closing a split and switching to another slice on a different band, clicking that band’s panadapter can retune Slice A instead of the slice you are on (#2102²⁴).

²⁴<https://github.com/aethersdr/AetherSDR/issues/2102>

5.8.3 Troubleshooting

Clicking **SPLIT** shows a notice and nothing happens

The connected radio can't do split, so AetherSDR refuses rather than doing nothing silently.

1. Use split on a FlexRadio (or in Demo mode). Other radio families don't offer it.

The split closed on its own after my logger changed frequency

Close split on QSY is on, and something outside AetherSDR moved Slice A by more than the threshold.

1. Right-click the **SPLIT** or **SWAP** badge and open **Split QSY Option**.
2. Untick **Close split on QSY**, or raise **QSY change threshold to close split** in the same submenu.

The Monitor TX key does nothing

Monitor TX (Hold) is not bound by default, and operating shortcuts are off by default.

1. Open **Settings** → **Configure Shortcuts...** and bind a key to **Monitor TX (Hold)**.
2. Turn on **Settings** → **Keyboard Shortcuts**.

A new split's transmit slice starts unmuted or panned to one side

AetherSDR remembered how you set up the transmit slice's audio last time.

1. Right-click the **SPLIT** or **SWAP** badge and choose **Forget remembered audio**.
2. Or mute the transmit slice yourself before ending the split; that also clears the remembered arrangement.

5.8.4 See also

- Multi-Slice Operation (p. 83)
- VFO Widget (p. 61)
- TX Controls (p. 68)
- Keyboard Shortcuts (p. 224)
- CAT Control (p. 132)
- WSJT-X Integration (p. 125)

5.9 Diversity and ESC

Status: Supported · **Applies to:** FlexRadio

Diversity reception uses two SCUs to receive the same signal from two different antennas simultaneously. **ESC (Enhanced Signal Clarity)** combines these signals with adjustable phase and gain to steer nulls toward interference and peaks toward desired signals — real-time antenna beamforming from your shack.

5.9.1 Requirements

Requirement	Details
Radio	A dual-SCU FlexRadio. AetherSDR reads diversity support from the radio model's capabilities (as FlexLib does), so dual-SCU models including the ML, CL, MLS and CLS variants show the DIV button
Antennas	Two antennas connected to different ports (e.g., ANT1 + ANT2, or ANT1 + RX_A)
License	DIV_ESC feature requires SmartSDR+ or higher
Firmware	v4.x recommended

Single-SCU radios (for example the FLEX-6400 and FLEX-6500) do not support diversity mode, and the DIV button does not appear. Diversity is a FlexRadio feature; it is not offered on other radio families.

5.9.2 Setup

1. Open the VFO widget's **Audio tab** (speaker icon)

2. Scroll to the bottom — the **DIV** button appears alongside the L/R audio pan slider
3. Click **DIV** to enable diversity reception

The radio will create a child slice on the second SCU. You'll see a second VFO overlay appear on the panadapter with a different antenna assignment.

5.9.3 Using diversity and ESC

How diversity works

When you enable DIV on a slice:

1. The radio creates a **child slice** on the second SCU, locked to the parent's frequency
2. Both SCUs receive the same signal from different antennas
3. The radio combines both signals into a single audio output on the parent slice
4. The child slice's independent audio is muted — you hear the combined result on the parent

The child slice appears in AetherSDR with its own VFO overlay (showing a different antenna), but its audio controls are irrelevant while ESC is active — the parent handles everything.

ESC controls

When DIV is active on a slice, the **ESC panel** appears below the DIV button in the VFO audio tab. ESC is only shown on the **parent** slice — the child slice shows DIV but not ESC controls.

ESC Toggle

Click **ESC** to enable Enhanced Signal Clarity beamforming. Without ESC enabled, diversity simply combines both antennas with default settings. With ESC, you can steer the combination.

Phase Slider (P)

The horizontal phase slider adjusts the phase offset between the two antenna signals:

- **Range:** 0° to 360° in 5° steps
- **Effect:** Rotates the beam pattern — different phase angles will null out interference arriving from different directions
- **Protocol:** Sent to the radio as radians (matching SmartSDR)

Gain Slider (G)

The vertical gain slider adjusts the amplitude balance between antennas:

- **Range:** 0.0 to 2.0
- **Default:** 1.0 (equal balance)
- **Effect:** Values below 1.0 favour the primary antenna, above 1.0 favours the secondary
- **Use case:** When one antenna picks up more noise than signal, reduce its contribution

Polar Display

The circular display shows the current phase and gain as a dot on a polar plot:

- **Angle** = phase (0° at top, clockwise)
- **Distance from centre** = gain (centre = 0.0, edge = 2.0)
- **Crosshairs** mark 0°/90°/180°/270° and the mid-gain reference circle (gain = 1.0)

As you adjust the phase and gain sliders, the dot moves in real time. You can also **click anywhere on the polar plot** to set phase and gain directly at that point.

ESC Signal Meter

Below the polar display, a level bar shows **signal strength after ESC processing** in dBm. This is a real-time readout from the radio's SLC/ESC meter (10 fps). Watch this meter while adjusting phase and gain to find the optimal null or peak:

- **Higher dBm** = more signal (or more noise)

- **Lower dBm** = deeper null

The goal is usually to find the phase angle where the ESC meter shows the most improvement in SNR — maximum desired signal with minimum interference.

Finding the Best Null

1. Enable DIV and ESC
2. Listen to the signal with interference
3. Slowly sweep the phase slider from 0° to 360°
4. Watch the ESC meter — you'll hear interference peak and null as you rotate
5. Stop at the phase angle where interference is minimised
6. Fine-tune the gain slider to optimise further

Band Noise Reduction

Even without a specific interference source, diversity + ESC can reduce overall band noise:

1. Enable DIV and ESC
2. Set gain to 1.0 (balanced)
3. Sweep phase slowly while listening to background noise level
4. The noise floor will vary with phase — find the minimum

Tips

- **Different antenna types work best** — e.g., a vertical + a horizontal dipole give the most diversity benefit since they respond differently to noise sources
- **Same-band, different-polarisation antennas** are ideal for diversity
- **Phase settings are frequency-dependent** — what works on 20m may not be optimal on 40m
- **Client noise reduction keeps the two antennas apart.** Every AetherSDR noise-reduction method processes left and right separately, so if you pan the two diversity slices to opposite ears, each ear keeps its own antenna (see DSP Noise Mitigation (p. 241))
- **A diversity child is not a split partner** — the SPLIT/SWAP badges ignore it (see Split Operation (p. 85))
- **ESC state is not saved** — you'll need to re-adjust after band changes or reconnects

5.9.4 Reference

Protocol details

For developers and protocol researchers:

Command	Description
slice set <id> diversity=1	Enable diversity (creates child slice)
slice set <id> diversity=0	Disable diversity (removes child slice)
slice set <id> esc=on	Enable ESC beamforming
slice set <id> esc=off	Disable ESC beamforming
slice set <id> esc_phase_shift=<radians>	Set phase (0 to 2 π)
slice set <id> esc_gain=<float>	Set gain (0.0 to 2.0)

- ESC commands are only sent from the **diversity parent** slice (per FlexLib Slice.cs:3367)
- SmartSDR uses `esc=on/esc=off` (not 1/0)
- Phase is sent in **radians**, displayed as degrees in the UI
- Phase steps are 5° (~0.087 radians), matching SmartSDR pcap
- The radio echoes ESC state in slice status: `esc=0 esc_gain=1.000000 esc_phase_shift=0.000000`
- When ESC is enabled, the radio sets the child slice to `audio_mute=1 audio_level=0` — the child's audio is absorbed into the parent's ESC processing

- Feature gated by DIV_ESC license (status: license feature name=DIV_ESC enabled=1 reason=PLUS)

5.9.5 Known issues

- With diversity on for two slices on different panadapters, tuning one slice moves the other (#4891²⁵).
- Turning on DIV for a second slice silently breaks up the diversity pair that is already running, with no message (#5352²⁶).

5.9.6 Troubleshooting

The DIV button isn't there

Your radio model doesn't have dual SCUs, or AetherSDR hasn't detected it yet. The button only appears on dual-SCU models.

1. Check that your radio is a dual-SCU model (see Requirements (p. 87)).

The ESC controls are missing

ESC appears only on the **parent** slice, and only while DIV is on.

1. Open the **Audio** tab on the slice where you clicked **DIV**, not on the child slice.
2. If DIV is off, click **DIV** first; the ESC panel appears below it.

My ESC settings were lost

ESC state is not saved, so it resets after band changes and reconnects.

1. Re-adjust the phase and gain (see Finding the best null (p. 89)).

5.9.7 See also

- VFO Widget (p. 61)
- Multi-Slice Operation (p. 83)
- Split Operation (p. 85)
- DSP Noise Mitigation (p. 241)
- FlexRadio (p. 20)

5.10 TNF (Tracking Notch Filters)

Status: Supported · **Applies to:** FlexRadio, Hermes-Lite 2 (experimental)

Tracking Notch Filters remove persistent interfering carriers from the audio passband. A notch stays on its frequency as you tune, so it keeps removing the same interferer.

5.10.1 Using TNFs

Creating a TNF

- Right-click on the spectrum or waterfall → **Add TNF at XX.XXX MHz**
- Or click **+TNF** in the panadapter's overlay menu (creates a TNF at the current frequency)
- The TNF appears as a shaded region on the spectrum and waterfall

The **Add TNF** entry is absent on a radio with no notch engine, and once the radio has as many notches as it supports.

Managing TNFs

Hover a TNF marker for a tooltip with its depth and whether it is permanent. Right-click it for a menu headed by its frequency and width.

²⁵<https://github.com/aethersdr/AetherSDR/issues/4891>

²⁶<https://github.com/aethersdr/AetherSDR/issues/5352>

Move and resize

- Drag the marker horizontally to move it
- Drag vertically to change its width

Width and Depth

- Right-click → **Width**: 50, 100, 200 or 500 Hz (only the widths the radio can actually produce are listed)
- Right-click → **Depth**: Normal, Deep or Very Deep (FlexRadio)

Extended TNF

- Right-click → **Extended TNF** extends every notch's band down through the waterfall, the way Extended Passband does for the slice filter. It is a display preference that applies to all notches.

Make Permanent

- Right-click → **Make Permanent** – the radio keeps the TNF across power cycles. **Make Temporary** reverses it.
- Permanent TNFs display in **green**; temporary TNFs in **yellow**

Remove

- Right-click → **Remove TNF**

Hermes-Lite 2

The Hermes-Lite 2 supports the same notch gestures: right-click to add, drag to move and resize, right-click to remove. Its notches run on the computer rather than in the radio and keep tracking as you tune. Because the HL2 has no place to store them and its notches are full nulls, the **Depth** and **Make Permanent** entries are not offered. A notch (or a very narrow CW filter) lengthens the receive filter slightly, which adds a little audio delay. See Hermes-Lite 2 (p. 23).

5.10.2 Reference**Permanent vs temporary**

Type	Colour	Survives reboot	Default
Temporary	Yellow	No	Yes
Permanent	Green	Yes	No

TNF removal and permanent-state changes are applied straight away in the display; the radio does not send a status confirmation for these commands.

5.10.3 Troubleshooting**Add TNF is missing from the menu**

The radio has no notch engine, or it already holds as many notches as it supports.

1. Remove a notch you no longer need (right-click it → **Remove TNF**), then try again.
2. If the entry never appears, your radio doesn't offer notches. TNFs work on FlexRadio and Hermes-Lite 2.

My notches disappeared after the radio restarted

They were temporary (yellow). Temporary notches don't survive a power cycle.

1. Add the notch again.
2. Right-click it and choose **Make Permanent**. It turns green.

5.10.4 See also

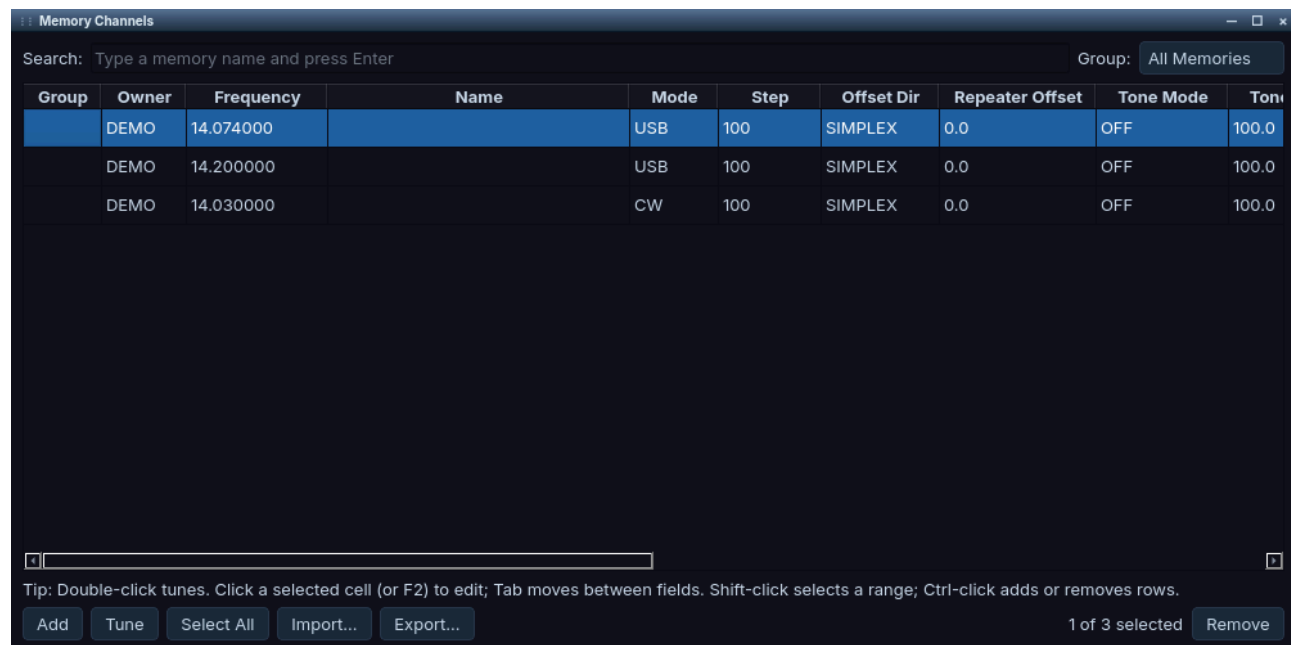
- Panadapter Controls (p. 54)
- DSP Noise Mitigation (p. 241)
- RX Controls (p. 64)
- Hermes-Lite 2 (p. 23)

5.11 Memory Channels

The Memory Channels dialog stores and recalls operating setups — frequency, mode, filter, repeater offset, tone, squelch, step and digital-mode offsets — for quick recall. On a FlexRadio it is an editor for the radio’s own memory bank.

Open it from **Tools** → **Memory...**

For quick recall while operating, the panadapter’s overlay **Memory** button opens a browse panel (see Memory browse panel (p. 93)).



The Memory Channels window with three memories added from the demo slice.

5.11.1 Using memory channels

Finding a memory

- **Search** — type a memory name and press Enter to jump to that entry; the table filters as you type
- **Profile** — show “All Memories” or only the memories of one global profile

Quick interactions

- **Double-click any row** — tunes the active slice to that memory
- **Shift-click** a range — selects multiple rows
- **Ctrl-click** (Command-click on macOS) rows — adds/removes rows from the selection
- **Edit in place** — click a selected cell (or press F2) to edit it; Tab moves between fields
- **Recall restores the whole setup** — mode, filter and slice settings, not just the frequency

Importing and exporting

Import... detects the file format automatically:

- **SmartSDR CSV** (the format **Export...** writes)
- **CHIRP-next generic CSV** — import only

The result reads “Imported N ... (format)”. Export always writes SmartSDR CSV.

To move a whole radio database (memories along with profiles) between SmartSDR and AetherSDR, use **Profiles** → **Import/Export Profiles...**; see Profile Management (p. 94).

Memory browse panel

The **Memory** button in the panadapter’s overlay menu opens a compact list (Frequency, Name) for recall without the full dialog. Its **Add Memory** button always stores the **active** slice.

When SpotHub’s **Memories** toggle is on, every memory also appears as a marker on the panadapter at its frequency. See SpotHub (p. 148).

Memory profiles

On a FlexRadio, memories are grouped by global profile. Use the **Profile** filter to see one profile’s memories; manage the profiles themselves in **Profiles** → **Profile Manager...** (see Profile Management (p. 94)).

5.11.2 Reference

Table columns

Sortable columns:

Column	What
Group	Grouping label (the profile the memory belongs to)
Owner	Callsign that created the memory
Frequency	Frequency in MHz
Name	User-assigned name (e.g. “K7ID Repeater”)
Mode	USB, LSB, CW, FM, etc.
Step	Tuning step in Hz
Offset Dir	Up / Down / Simplex
Repeater Offset	MHz
Tone Mode	CTCSS / DCS tone mode
Tone Value	Hz (CTCSS) or DCS code
Squelch, Squelch Level	Squelch on/off and level
RX Filter Low, RX Filter High	Receive filter edges
RTTY Mark, RTTY Shift	RTTY settings
DIGL Offset, DIGU Offset	Digital-mode offsets

Buttons

Button	Action
Import...	Import memories from a CSV file (see below)
Export...	Export the displayed memories to CSV
Add	Create a new memory from the active slice
Tune	Tune the active slice to the selected memory (also: double-click any row)
Select All	Select every visible row
Remove	Delete the selected memory or memories

The selection count is shown beside the buttons.

Where memories live

- **FlexRadio:** memories live on the **radio**. They persist across power cycles and every client connected to the radio sees them.
- **Radios without memory slots** (Hermes-Lite 2, KiwiSDR receivers, Demo mode): AetherSDR stores the memories on your computer, with the same dialog, browse panel, CSV import/export and panadapter markers. This is **one shared bank** used by all such radios, not one per radio.

- **Networked Icom:** where the radio supports it, a **Sync Memories** button reads the radio's memory channels into the list (on the IC-705, choose a memory group first). Where the radio's memories are read-only, Add, Import and in-place editing are disabled with the reason in the tooltip.

5.11.3 Troubleshooting

Some memories are missing from the list

The **Profile** filter is showing one profile's memories, or the search box is filtering the table.

1. Set **Profile** to "All Memories".
2. Clear the **Search** box.

Add, Import and editing are greyed out on my Icom

The radio's memories are read-only.

1. Hover over the greyed control; the tooltip gives the reason.
2. Where the dialog has a **Sync Memories** button, use it to read the radio's channels into the list, then recall them as usual.

Memories don't appear as markers on the panadapter

The markers come from SpotHub.

1. Open **Settings** → **SpotHub...** and turn on the **Memories** toggle. See SpotHub (p. 148).

5.11.4 See also

- Net Scheduler (p. 160) (reminders for recurring nets, with one-click tuning)
- Profile Management (p. 94)
- SpotHub (p. 148)
- Networked Icom (p. 28)

5.12 Profile Management

Status: Supported · **Applies to:** FlexRadio

Profiles save and restore FlexRadio state. AetherSDR supports all three SmartSDR profile types, and profiles are stored **on the radio**, so they are shared with SmartSDR and other clients.

AetherSDR's own preferences (layout, display, controllers) are separate and stay on your computer. See Settings and Backups (p. 220).

5.12.1 Requirements

- A FlexRadio must be connected. The Profiles menu, Profile Manager and PROF applet are hidden when another kind of radio is connected.
- Import and export also need a direct LAN connection (not SmartLink), and are not available while transmitting (MOX, TUNE, PTT).

5.12.2 Profile types

Global profiles

Save the whole radio state: panadapters, slices, frequencies, modes, filters, DSP, antennas and so on. Loading a global profile changes everything at once.

Transmit profiles

TX settings: power, EQ, compressor, mic settings. Keep several for different operating (contest, ragchew, digital).

Microphone profiles

Mic input settings: source, gain, bias, boost and compander level. Useful when you switch microphones.

5.12.3 Using the Profiles menu

Profiles in the menu bar:

- **Profile Manager...**: open the management dialog
- **Import/Export Profiles...**: back up or restore the radio's profile database as a SmartSDR .ssdr_cfg package
- the list of global profiles: click one to load it; the check mark follows the radio's active global profile

5.12.4 Using the Profile Manager

Profiles → **Profile Manager...** opens a non-modal window that remembers its size and position. Tabs: **Global**, **Transmit**, **Microphone**, **Auto-Save**.

On each profile tab:

- **New Profile Name**: type a name for a new profile
- **Load** (or double-click): apply the selected profile
- **Save** (Global tab): save the current state to the selected profile, or to a new one using the name field
- **Create** (Transmit and Microphone tabs): create a new profile. The radio cannot overwrite an existing TX or Mic profile directly; changes to the active one are captured by Auto-Save. If you try to create a name that already exists, AetherSDR tells you so and, if Auto-Save is off, offers **Enable Auto-Save**.
- **Delete**: remove the selected profile, after confirmation

A status line reports what the radio did: Saving "<name>"..., Saved profile "<name>"., "Radio refused save of ...", "Not connected — cannot save ...", or "No response from the radio ...". A blank name field never overwrites the highlighted profile.

Auto-Save

With **Auto-save profile changes** enabled on the Auto-Save tab, changes you make to TX and Mic settings are saved to the active TX and Mic profiles as you go.

5.12.5 Using the PROF applet (Profile Switcher)

The **PROF** applet ("Profile Switcher", in the Station group of the applet button bar, closed by default) has three drop-downs: **Global**, **TX** and **Mic**. Picking an entry loads that profile immediately. It follows the radio's active profiles, including changes made by other clients.

5.12.6 Importing and exporting profiles (.ssdr_cfg)

Profiles → **Import/Export Profiles...** moves profile libraries between radios, and between SmartSDR for Windows and AetherSDR, using SmartSDR-compatible .ssdr_cfg backup packages. **Both directions are carried out by the radio**; AetherSDR transfers the package but never edits its contents.

Export

1. On the **Export** tab, tick the **Radio Database Categories** to include: Global Profiles, TX Profiles and MIC Profiles (ticked by default), Memories, Preferences, TNF, XVTR and USB Cables. **Select All** ticks everything.
2. Choose a **Destination** (default name ASDR_Config_<timestamp>_v<version>.ssdr_cfg).
3. Click **Export**. The radio builds the package and AetherSDR saves it.

Import

1. On the **Import** tab, choose the **Configuration Package** and click **Import**.
2. A confirmation shows the package's firmware version against the radio's, and warns that:
 - imported profiles, memories and other settings can replace matching items on the radio, including defaults;
 - packages from newer firmware may fail or leave the radio database in an unexpected state;
 - packages that include Preferences can close or reopen slices and panadapters.

3. **Export Backup First** saves a backup of the radio before you go ahead; **Import** uploads the package and the radio applies it. Profile lists refresh when it finishes.

5.12.7 Workspaces and profiles

If you use the experimental Workspace Canvas (p. 98), a workspace can be bound to a global profile (**View** → **Workspace Canvas** → **Workspaces** → **Bind to radio profile**), so recalling that profile also switches the screen layout.

5.12.8 Multi-Flex

While a global profile is being restored, AetherSDR holds back its own slice, panadapter and waterfall changes until the radio has finished, so another client's profile load is not disturbed. See Multi-Flex (p. 211).

5.12.9 Reference

Profile operations are carried out by the radio with these commands:

- profile global save "name" / load / delete
- profile transmit create "name" / save / delete
- profile mic create "name" / save / delete

5.12.10 Known issues

- An export that is interrupted or invalid can replace an existing backup file and still report success (#5620²⁷).
- After repeated global profile changes, panadapter **FFT Auto** can stop adjusting the noise floor while still showing as on, until you turn it off and on again (#3186²⁸).
- DFNR can come back on after you turn it off, and saving a profile does not keep it off, because it runs in AetherSDR and is not stored in radio profiles (#4018²⁹).

5.12.11 Troubleshooting

The Profiles menu or PROF applet is missing

Profiles are a FlexRadio feature, and they are hidden when another kind of radio is connected.

1. Connect to a FlexRadio. The menu and applet appear.

Import/Export Profiles is unavailable

Import and export need a FlexRadio on a direct LAN connection, with nothing transmitting.

1. Connect over the LAN rather than SmartLink.
2. Turn off MOX, TUNE and PTT.
3. Open **Profiles** → **Import/Export Profiles...** again.

“Create” says a TX or Mic profile name already exists

The radio cannot overwrite an existing TX or Mic profile. Changes to the active one are captured by Auto-Save instead.

1. Click **Enable Auto-Save** when AetherSDR offers it, or tick **Auto-save profile changes** on the **Auto-Save** tab.
2. Load the profile you want to change, then make your changes. They are saved to it as you go.

5.12.12 See also

- Memory Channels (p. 92)
- Settings and Backups (p. 220): AetherSDR's own settings on this computer

²⁷<https://github.com/aethersdr/AetherSDR/issues/5620>

²⁸<https://github.com/aethersdr/AetherSDR/issues/3186>

²⁹<https://github.com/aethersdr/AetherSDR/issues/4018>

- Multi-Flex (p. 211)
- Workspace Canvas (p. 98)
- FlexRadio (p. 20)

5.13 XVTR (Transverters)

Status: Supported · **Applies to:** FlexRadio

AetherSDR supports transverters for VHF, UHF and microwave operation through the FlexRadio's XVTR configuration and its XVTA / XVTB antenna ports.

5.13.1 Setup

Configuring a transverter

1. Open **Settings** → **Radio Setup...** → **Transverters** (or click **XVTR** in the panadapter's Band panel, which jumps there).
2. Click **Create New Transverter**.
3. Fill in:
 - **Name** — e.g., "2m"
 - **RF Freq (MHz)** — transverter RF frequency (e.g., 144.000)
 - **IF Freq (MHz)** — radio IF frequency (e.g., 28.000)
 - **LO Freq (MHz)** — calculated for you (RF – IF)
 - **LO Error (MHz)** — local-oscillator calibration offset
 - **RX Gain (dB)** — transverter receive gain
 - **RX Only** — disable TX for receive-only transverters
 - **Max Power (dBm)** — maximum safe TX drive
4. The entry shows **Valid** or **Invalid** as the radio checks it. **Remove** deletes it.

Antenna ports

XVTR bands use the **XVTA** or **XVTB** antenna ports. The radio routes through the transverter when these ports are selected. You can give the ports friendly names in **Radio Setup** → **Antennas**; the radio still uses XVTA/XVTB.

5.13.2 Using transverter bands

Selecting a band

1. Click **Band** in the panadapter's overlay menu.
2. Your configured transverter bands appear as their own buttons in the band grid, between the HF bands and the utility row.
3. Click one to tune to it.

Band stacks are kept per transverter band.

Built-in 4 m and 2 m (FLEX-6500 / FLEX-6700)

On the FLEX-6500 (4 m, Region 1) and FLEX-6700 (4 m and 2 m), 4 and 2 band buttons appear in the band grid. They open a slice through the radio's XVTR path, using its built-in transverter or an external one, as SmartSDR does.

Entering a frequency

When on an XVTR band (antenna XVTA/XVTB or frequency above 54 MHz):

- Bare integers get the decimal inserted after the 3rd digit
- 1446 → 144.6 MHz
- 14696 → 146.96 MHz
- 4401 → 440.1 MHz

- Explicit decimals work normally: 144.6 → 144.6 MHz

Direct entry works above 450 MHz.

5.13.3 Reference

Microwave bands

AetherSDR knows the microwave bands for band-stack and bookmark grouping:

Band	Range (MHz)	Default (MHz)
13 cm	2300–2450	2304.100
9 cm	3300–3500	3456.100
5 cm	5650–5925	5760.100
3 cm	10000–10500	10368.100

All default to USB. They need a configured transverter; they are not band-grid buttons of their own. Common band-name aliases are accepted (for example 70cm for 440).

A radio can also advertise which bands it supports, and the band UI follows it.

5.13.4 Known issues

- Selecting a transverter band can put the VFO on the wrong frequency (for example 1317.100 MHz instead of 1296.100 MHz for a 1296 MHz transverter with a 28 MHz IF), and typing the right frequency snaps back (#6197³⁰).
- A band change from a CAT or TCI program onto a transverter band does not recall the band stack, so the saved **XVTA/XVTB** antenna port is not selected and the radio can key into the wrong port (#3543³¹).

5.13.5 See also

- Radio Setup (p. 214)
- Panadapter Controls (p. 54)
- CAT Control (p. 132)
- TCI Server (p. 135)

5.14 Workspace Canvas

Status: Experimental · **Applies to:** All radios

The workspace canvas is an alternative to the standard (“Classic”) layout. With it on, panadapters and applets become freely placed, resizable, layered items on a canvas, and the canvas can span several windows. Named workspaces remember which applets are open as well as where they sit, and a workspace can follow a radio profile.

The canvas is **off by default**. If you never turn it on, the Classic layout is unchanged and no canvas settings are written. Work remaining before the experimental label comes off: dragging items live between canvas windows, and more on-air time compared with the Classic layout.

5.14.1 Setup

View → Workspace Canvas ▶

³⁰<https://github.com/aethersdr/AetherSDR/issues/6197>

³¹<https://github.com/aethersdr/AetherSDR/issues/3543>

Item	What it does
Enabled	Turns the canvas on or off. The choice is remembered. If the canvas can't start, the status bar shows "Work-space canvas unavailable: <reason>".
Edit Layout	Switches between the two postures (below). Only available while the canvas is on, and not remembered across restarts.
Workspaces ▶	The workspace switcher and workspace actions.
Canvas Windows ▶	Extra canvas windows.

5.14.2 Using the canvas

Locked and edit postures

- **Locked** (the default) is for operating. Items stay put and every control works normally.
- **Edit** is for arranging. Turn it on with **View → Workspace Canvas → Edit Layout**, or **Edit layout** at the top of the canvas right-click menu.

Placing items

In edit posture:

- **Move** an item by dragging its title bar; it follows the cursor.
- **Resize** with the eight grips on the item's 8 px border. The contents stay clickable.
- **Snapping**: edges and centres snap to other items and to the canvas edges within 8 px, with guide lines drawn while you drag. **Snap to grid** adds an optional 96 × 54 grid with a gentler pull; edges of other items win over the grid.

Positions are stored as fractions of the canvas, so a layout made on a 3840 × 1600 screen comes back in proportion on 1920 × 1080.

Named workspaces

View → Workspace Canvas → Workspaces ▶ lists your workspaces with a check mark on the active one. Switching restores both which applets are open and where everything sits.

Item	What it does
New from current layout...	Copies what is on screen now.
New from Classic...	Starts from the Classic arrangement.
New blank...	Starts with an empty canvas.
Rename active...	Renames the current workspace.
Delete active...	Deletes it after confirmation ("Its arrangement is lost").
Import pop-outs onto canvas	Docks every floating applet window and floating panadapter onto the canvas, at positions mapped from where the windows were.
Bind to radio profile ▶	Links this workspace to a global radio profile, so recalling that profile switches to the work-space. See Profile Management (p. 94).

With the canvas off, the submenu just says "Enable the canvas to create workspaces".

Canvas windows

View → Workspace Canvas → Canvas Windows ▶ adds more top-level canvases, for example one per monitor.

- **New canvas window...** creates one; **Rename ▶** renames it.
- **Closing** a canvas window only hides it; its items are back exactly where they were when you reopen it.
- **Remove ▶** deletes a canvas window after confirmation; its items move back to the main canvas.
- To move an item to another window, use **Move to ▶** on the item. Dragging an item from one window to another is not supported yet.

Minimal Mode and the applet panel

- Entering **Minimal Mode** (Ctrl+Shift+M) while the canvas is on pauses the canvas, with the notice “Workspace canvas will resume after leaving minimal mode”. Leaving Minimal Mode brings the canvas back with its arrangement intact.
- **Ctrl+Shift+S** (pop out the applet panel) does nothing while the canvas is on, because the canvas owns the applets.

5.14.3 Reference

The canvas right-click menu

Right-click an empty part of the canvas:

Item	When	What it does
Edit layout	Always	Toggles edit posture.
Workspace ▸	More than one workspace	Switch workspace.
Add widget ▸	Always	Every applet, grouped by category. Applets already on the canvas are shown checked. Applets for hardware that isn’t detected (tuner, amplifiers, Antenna Genius, ShackSwitch, Demo Noise) are greyed rather than hidden.
Snap to grid	Edit	Turns the 96 × 54 grid on or off.
Undo last placement	Edit	Undoes the most recent placement.
Tidy layout	Edit	Tidies up the current arrangement.
Reset layout to Classic	Edit	Puts the canvas back to the Classic arrangement.

Right-click an item in edit posture for **Move to ▸** (another canvas window), **Return to panel**, **Bring to front**, **Send to back** (panadapters only offer Send to back) and **Hide band stack**.

Compatibility

A workspace file written by a newer AetherSDR release is refused rather than overwritten, so trying an older build won’t damage a newer layout.

5.14.4 Troubleshooting

Floating panadapters don’t come back where they were

A layout with floating panadapters did not restore properly.

1. Turn on **View → Workspace Canvas → Enabled**.
2. Choose **View → Workspace Canvas → Workspaces → Import pop-outs onto canvas**. Every floating applet window and floating panadapter is docked onto the canvas.

Items won’t move or resize

The canvas is in locked posture.

1. Turn on **View → Workspace Canvas → Edit Layout**, or choose **Edit layout** from the canvas right-click menu.
2. Arrange the items, then turn Edit Layout off again to operate.

The canvas vanished after entering Minimal Mode

Minimal Mode pauses the canvas.

1. Press **Ctrl+Shift+M** to leave Minimal Mode. The canvas comes back with its arrangement intact.

Ctrl+Shift+S does nothing

The canvas owns the applets while it is on, so the applet panel can’t be popped out.

1. Move the applet where you want it on the canvas, or turn off **View → Workspace Canvas → Enabled** to go back to the Classic applet panel.

5.14.5 See also

- Panadapter Controls ([p. 54](#))
- Menu Reference ([p. 235](#))
- Profile Management ([p. 94](#))
- Keyboard Shortcuts ([p. 224](#))

CW & Digital Modes

6.1 CWX Panel

Status: Supported · **Applies to:** FlexRadio, IC-7300MK2, IC-705 (early)

The CWX (CW Transmit) panel sends Morse code typed on your keyboard or stored in macros, keyed by the radio's own CW text keyer. On a FlexRadio it uses the radio's CWX keyer; on a supported Networked Icom ([p. 28](#)) radio the same panel drives the radio's CI-V text keyer and is labelled **CWK**.

6.1.1 Requirements

- **Mode:** CWX keys the **transmit slice**, so it is available when the TX slice is in **CW** or **CWL**. The indicator is greyed out otherwise, and switching the TX slice out of CW closes the panel.
- **Radio support:** the indicator is **hidden** on radios without a radio-side CW text keyer (for example the Hermes-Lite 2 ([p. 23](#))), and the panel closes with it.
- **On Icom (CWK):** supported on the IC-705 and IC-7300MK2. The speed range follows the radio, and the **Live** and **Setup** views are hidden because the CI-V keyer has no live keying or stored macros.

6.1.2 Using the CWX panel

Opening CWX

Click the **CWX** indicator in the status bar at the bottom of the screen (or use **Tools** → **CW Keyer**). The panel opens beside the panadapter.

The CWX panel and the DVK Panel ([p. 107](#)) share the same space — opening one closes the other.

Sending text

The panel has three views, chosen with the buttons at its top: **Send**, **Live** and **Setup** (see Views ([p. 103](#))). In **Send**, type text and send it; sent text appears as history bubbles above the entry line. In **Live**, characters are keyed as you type them.

Speed sets the keyer speed in WPM (5–100 on FlexRadio).

Right-click a sent bubble for **Resend** (sent again exactly as typed, per-word speed changes included) or **Clear History**.

When the CWX queue empties, AetherSDR releases the transmitter straight away, so the radio returns to receive right after the last character.

Macros and shortcuts

Key	Action
F1–F12	Send the matching macro. Works app-wide while the TX slice is in CW/CWL, even with the panel closed.
Escape	Abort the CW being sent

F1–F12 follow the transmit slice’s mode: in CW they fire CWX macros, in voice modes they play DVK recordings (see DVK Panel (p. 107)), and only one of the two responds. CWX macros 1–12 can also be assigned to FlexControl, MIDI and other controller buttons.

Keyboard and MIDI keying (local iambic keyer)

Beyond text entry, AetherSDR has a local iambic / straight-key keyer that respects the radio’s break-in setting. Three actions can be bound to keys or MIDI controls:

Action	Default key
Trigger straight key	unbound
Trigger CW Left Paddle (dits)	unbound
Trigger CW Right Paddle (dahs)	unbound

Bind keys under **Settings** → **Configure Shortcuts...** and turn on **Settings** → **Keyboard Shortcuts** (off by default), or map a MIDI control in MIDI Controller Mapping (p. 192). Further CW actions include speed up/down, sidetone, iambic on/off, iambic mode A/B, swap paddles and break-in.

- With **break-in** on, a paddle or straight-key press keys the radio directly. With break-in off, MOX must be on for the CW to go out.
- Element timing is scheduled against an exact clock with a sample-accurate sidetone, and iambic Mode B behaves correctly when both paddles are released together.
- The break-in delay holds when you change speed.
- CW keying and **TUNE** lock each other out: you cannot key CW while TUNE is on, or start TUNE while a key is down.

The sidetone plays on your selected audio output and follows the radio’s sidetone pitch and level.

To watch your own sending, turn on the **TX** decode of the CW Decoder (p. 104).

6.1.3 Reference

Views

View	What it does
Send	Type text and send it. Sent text appears as history bubbles above the entry line.
Live	Characters are keyed as you type them.
Setup	Delay, QSK, speed step and the twelve F1–F12 macros.

Setup settings

Setting	What it does
Delay	Break-in delay, 0–2000 ms
QSK	Full break-in on or off
Step	How many WPM each + or - speed prefix changes the speed (1–20)
F1–F12	Twelve macro texts

Special characters

The legend under the macros lists the special characters:

- **Prosigns:** = (BT), + (AR), ((KN), & (BK), \$ (SK)
- **Speed changes:** +word sends that word faster, -word slower; ++ / -- change by twice the step. A + or - only counts as a speed prefix at the start of a word — a standalone + is still AR.

6.1.4 Known issues

- Receive audio can go silent after a CW transmission and only return after a **TUNE** cycle (#5312¹).

6.1.5 Troubleshooting

Receive audio goes silent for about a minute after sending CWX

On FLEX firmware v4.2.18, a CWX break-in delay of 0 makes the radio mute all receive audio for roughly 70 seconds after CWX text is sent. The panadapter keeps running, but audio, the CW decoder and the S-meter go dead. This is a firmware defect, fixed in v4.2.20 (#5945²).

1. Update the radio to firmware v4.2.20 or later. See Firmware Update (p. 232).

The CWX indicator is greyed out

The transmit slice is not in a CW mode.

1. Switch the TX slice to **CW** or **CWL**.

The CWX indicator is missing

The radio has no radio-side CW text keyer, as on the Hermes-Lite 2 (p. 23). Use the local keyer instead (see Keyboard and MIDI keying (p. 103)).

A paddle or key press sends nothing

Break-in is off, so the keyer waits for MOX.

1. Turn on **break-in**, or turn on MOX before keying.
2. Check that **TUNE** is off; CW keying is locked out while it is on.

6.1.6 See also

- CW Decoder (p. 104)
- DVK Panel (p. 107)
- MIDI Controller Mapping (p. 192)
- Keyboard Shortcuts (p. 224)
- FlexControl Tuning Knob (p. 183)

6.2 CW Decoder

Status: Supported · **Applies to:** FlexRadio, Hermes-Lite 2 (experimental), Networked Icom (early; IC-7300MK2 supported)

AetherSDR includes a built-in CW (Morse code) decoder powered by gg Morse³ (MIT license). It automatically detects the CW tone pitch and keying speed, and displays decoded text in real time in a panel below the waterfall. It can decode what you receive, what you send, or both.

The decoder works on every radio family that has a CW mode: FlexRadio, the Hermes-Lite 2 (p. 23), and Networked Icom (p. 28) radios (it opens when an Icom slice is in CW).

6.2.1 Requirements

- The selected slice is in **CW** or **CWL**.

¹<https://github.com/aethersdr/AetherSDR/issues/5312>

²<https://github.com/aethersdr/AetherSDR/issues/5945>

³<https://github.com/ggemanov/ggmorse>

- On a FlexRadio, a **DAX RX channel** (1–8) assigned to the selected slice is recommended, so the decoder hears that slice alone. See Which audio is decoded (p. 105).

6.2.2 Setup

Decoding is controlled in **Settings** → **Radio Setup...** → **Phone & CW**, next to **Decode:**

Toggle	Default	What it decodes
RX	On	The received CW on the selected slice
TX	Off	Your own keying, from AetherSDR's sidetone — useful for checking paddle or bug timing

Both appear in the same panel. Your own sending is shown in **cyan**, so it stands apart from the confidence-coloured received text.

6.2.3 Using the CW decoder

When the selected slice is in **CW** or **CWL**, the decode panel appears below the waterfall, and it hides again when you switch to another mode.

Drag the grip on the panel's edge to resize it; the height is remembered. Right-click the text for font size and Clear.

How it works

The decoder processes the received audio on a separate worker thread. It detects the dominant tone frequency, then uses a Goertzel filter and timing analysis to decode Morse characters.

- **Auto pitch detection** within the pitch search range
- **Auto speed detection** within the speed search range
- **Cost function:** measures how well the detected on/off timing matches ideal Morse ratios; this drives the confidence colours

Retuning resets the decoder state; pitch and speed locks survive a retune.

Which audio is decoded

The decoder follows the **selected slice**, independently of speaker volume or mute:

- **On a FlexRadio, assign a DAX RX channel to the selected slice** (1–8). The decoder then reads that slice's audio alone, regardless of speaker gain, mute, or other audible slices.
- Without a DAX channel, it uses the shared receive audio, which mixes every audible slice and follows speaker gain and mute. The panel shows **RX: shared audio**; hover it for the explanation.

Reading the confidence colours

Received characters are colour-coded by decode confidence (see Confidence colours (p. 106)). Green text can be trusted. Yellow is usually correct. Orange and red should be read sceptically — they often appear during QSB, QRM, or with operators who have irregular spacing.

Callsign lookup

When the decoder copies a station identifying itself (DE <call>), and QRZ.com lookups are enabled, a **contact card** for that station appears beside the decoded text. Click the callsign to open its QRZ.com page. See Callsign Lookup (p. 163).

MQTT

Decoded text is published to the MQTT topic `aethersdr/cw/decode`, one JSON message per character, with "rx": false marking text from the TX decoder. See MQTT Station Automation (p. 141).

Tips for best results

- **Tune accurately:** Centre the CW signal in your filter passband. The decoder auto-detects the pitch, but a clean signal helps.

- **Use narrow filters:** CW filter widths of 200–500 Hz reduce noise and competing signals that confuse the decoder.
- **APF helps:** The Audio Peaking Filter (APF) narrows the audio around the CW pitch, giving the decoder a cleaner signal.
- **Give it a DAX channel:** On a FlexRadio, a DAX RX channel on the selected slice keeps other slices and your speaker settings out of the decode.
- **Narrow the search ranges or lock:** If the decoder wanders between signals, narrow the Pitch and WPM ranges, or lock pitch and speed once it has the right station.
- **Strong signals decode best:** Weak signals near the noise floor produce mostly red/orange text.
- Client-side noise reduction is voice-only and turns itself off in CW, so it does not affect the decoder.

6.2.4 Reference

Panel controls

Control	What it does
Stats	Detected pitch (Hz) and speed (WPM)
Sens	0–100 (default 30). Hides low-confidence characters: 0 shows everything, higher values show only confident decodes.
ⒺP	Lock the decoder's pitch at the current value
ⒺS	Lock the decoder's speed at the current WPM
Pitch	Pitch search range, 300–1200 Hz (default 500–700 Hz)
WPM	Speed search range, 5–60 WPM (default 15–40 WPM)
CPY ALL / CPY VIS	Copy all decoded text, or just the visible text, to the clipboard
A- / A+	Decoded-text size (8–32 px), remembered
CLR	Clear the decoded text
×	Close the decoder panel

Confidence colours

Colours are based on ggmorse's cost function:

Colour	Confidence	Cost range	Meaning
Green	High	< 0.15	Clean, well-timed CW
Yellow	Medium	0.15–0.35	Slightly imperfect timing
Orange	Questionable	0.35–0.60	May be incorrect
Red	Low	0.60–1.00	Likely noise or bad decode

Technical details

- Library: ggmorse⁴ by Georgi Gerganov (MIT license)
- Bundled directly — no external dependency
- Runs on a dedicated worker thread — does not block audio playback
- CPU usage: negligible

6.2.5 Known issues

- The decoder shows no text on some FLEX-8600 setups (#6158⁵).
- The pitch search range is not kept when a slice is reopened, even with both range locks engaged (#5921⁶).

⁴<https://github.com/ggerganov/ggmorse>

⁵<https://github.com/aethersdr/AetherSDR/issues/6158>

⁶<https://github.com/aethersdr/AetherSDR/issues/5921>

- There is no setting to stop the automatic QRZ contact card from appearing for decoded callsigns (#5454⁷).
- The squelch threshold line can appear on the panadapter in CW, although squelch is unavailable there (#6276⁸).

6.2.6 Troubleshooting

The decode panel doesn't appear

The panel follows the mode of the selected slice.

1. Select the slice you want to decode.
2. Switch it to **CW** or **CWL**.
3. Check that **RX** is on next to **Decode:** in **Settings** → **Radio Setup...** → **Phone & CW**.

The decode changes with speaker volume, mute or other slices

The decoder has no DAX channel on the selected slice, so it uses the shared receive audio. The panel shows **RX: shared audio**.

1. On a FlexRadio, assign a DAX RX channel (1–8) to the selected slice.
2. Check that **RX: shared audio** is no longer shown.

The decoder jumps between stations

Another signal inside the pitch or speed search range is pulling the decoder away.

1. Narrow the filter to 200–500 Hz, or turn on APF.
2. Narrow the **Pitch** and **WPM** search ranges around the station you want.
3. Once it copies the right station, lock pitch (**Ⓟ**) and speed (**Ⓢ**).

Text is mostly orange or red

The signal is weak, poorly tuned or irregularly sent.

1. Centre the signal in the filter passband.
2. Narrow the filter or turn on APF.
3. Raise **Sens** to hide low-confidence characters.

6.2.7 See also

- CWX Panel (p. 102) — sending CW from the keyboard
- RTTY Operation (p. 110) — the built-in RTTY decoder works the same way
- DAX Virtual Audio (p. 128)
- Callsign Lookup (p. 163)
- ggmorse⁹

6.3 DVK Panel

Status: Supported · **Applies to:** FlexRadio

The DVK (Digital Voice Keyer) panel enables recording and playback of up to 12 voice messages stored on the radio, commonly used for contesting and repetitive voice operations. DVK is a FlexRadio feature and requires the radio's SmartSDR+ subscription.

⁷<https://github.com/aethersdr/AetherSDR/issues/5454>

⁸<https://github.com/aethersdr/AetherSDR/issues/6276>

⁹<https://github.com/ggerganov/ggmorse>

6.3.1 Requirements

- **Radio:** a FlexRadio Aurora, FLEX-8000 series or FLEX-6000 series radio. The DVK indicator is **hidden** on radios that have no radio-side voice keyer, such as the Hermes-Lite 2 (p. 23) and Networked Icom (p. 28) radios, and the panel closes with it.
- **SmartSDR+ subscription:** DVK requires an active SmartSDR+ subscription on the radio. See SmartSDR+ licence (p. 109).
- **Mode:** DVK plays on the **transmit slice**, so it is available when the TX slice is in a **voice mode** (USB, LSB, AM, SAM, FM, NFM, DFM). The indicator is greyed out in digital and CW modes, and switching to a non-voice mode closes the panel.

6.3.2 Using the DVK panel

Opening DVK

Click the **DVK** indicator in the status bar at the bottom of the screen. The panel opens beside the panadapter.

The DVK panel and the CWX Panel (p. 102) share the same space — opening one closes the other.

Recording

1. Click a slot to select it (blue highlight).
2. Click **● REC** to start recording (button turns red, progress bar pulses red).
3. Click **■ STOP** to end recording.
4. The radio stores the recording and updates the duration.

Playback (on-air)

- Click **▶ PLAY** or press the **F-key button** to transmit the recording.
- The radio keys TX automatically during playback.
- The progress bar fills green in proportion to elapsed/total time.
- Status shows: Status: Playback 1 / 3.2s

Preview (local)

- Click **◀ PREV** to hear the recording locally without transmitting.
- The progress bar fills blue.
- Status shows: Status: Preview 1 / 2.1s

Managing slots

Right-click any slot row to access:

- **Rename...** — inline text editor (also available via double-click on name)
- **Clear** — erase the recording (greyed out for empty slots)
- **Delete** — remove the recording
- **Import WAV...** — upload a WAV file to this slot (see WAV file requirements (p. 109))
- **Export WAV...** — download the recording as a WAV file (greyed out for empty slots)

If an import or export fails, the destination is left intact — a failed export never leaves a half-written file over an existing one.

6.3.3 Reference

Recording slots

The panel shows 12 recording slots (F1–F12), each with:

- **F-key button** — click to toggle playback on-air
- **Recording name** — displayed next to the button
- **Duration** — shown on the right (or “Empty” if no recording)
- **Progress bar** — 3 px coloured bar during active operations

Empty slots appear dimmed. Slots with recordings show bright text.

Keyboard shortcuts

Key	Action
F1–F12	Play that slot on-air; press again to stop. Works while the TX slice is in a voice mode, even with the panel closed.
Escape	Stop any active recording, playback, or preview (or cancel a rename)

F1–F12 follow the transmit slice’s mode: in voice modes they play DVK recordings, in CW they send CWX macros (see CWX Panel (p. 102)), and only one of the two responds.

WAV file requirements

For importing WAV files, the radio requires:

- **Format:** 32-bit IEEE float
- **Channels:** 2 (stereo)
- **Sample rate:** 48,000 Hz
- **Max size:** 5 MB (~13 seconds)

Files that don’t meet these requirements are rejected with a specific error message.

Status bar indicator

The DVK label in the status bar has three visual states:

- **Bright cyan** — panel is open
- **Dim white** — available (voice mode), click to open
- **Dark grey** — unavailable (not in a voice mode, or not licensed)

SmartSDR+ licence

When the radio reports that the feature is not licensed, the **DVK indicator itself is disabled** — greyed out with an arrow cursor and the tooltip “Digital Voice Keyer — requires an active SmartSDR+ subscription”. The F1–F12 shortcuts and any open panel are gated the same way, and the panel status reads **Disabled (SmartSDR+ required)**.

AetherSDR only applies this once the radio positively reports the licence as disabled; if the radio never reports it, DVK stays available.

6.3.4 Known issues

- Cancelling a WAV import or export and starting another straight away can corrupt the new transfer (#5665¹⁰).

6.3.5 Troubleshooting

The DVK indicator is greyed out

The transmit slice is not in a voice mode, or the radio reports no SmartSDR+ licence.

1. Switch the TX slice to a voice mode (USB, LSB, AM, SAM, FM, NFM or DFM).
2. Hover the indicator. If the tooltip says it requires an active SmartSDR+ subscription, renew SmartSDR+ on the radio.

The DVK indicator is missing

The radio has no radio-side voice keyer, as on the Hermes-Lite 2 (p. 23) and Networked Icom (p. 28) radios. DVK is not available there.

A WAV import is rejected

The file doesn’t match the format the radio requires.

¹⁰<https://github.com/aethersdr/AetherSDR/issues/5665>

1. Convert the file to 2-channel, 32-bit IEEE float, 48 kHz.
2. Keep it under 5 MB (about 13 seconds).
3. Import it again with **Import WAV...**

6.3.6 See also

- CWX Panel (p. 102)
- Keyboard Shortcuts (p. 224)
- Audio Settings (p. 217)
- FlexRadio (p. 20)

6.4 RTTY Operation

Status: Supported · **Applies to:** FlexRadio, Hermes-Lite 2 (experimental)

RTTY (Radio Teletype) is a digital mode that encodes text as two alternating audio tones — a **Mark** tone and a **Space** tone. AetherSDR supports RTTY with the radio's **RTTY** mode, a **built-in RTTY decoder** that shows received text below the waterfall, and the usual DAX and CAT path for external RTTY software. The decoder also works on the Hermes-Lite 2 (p. 23).

Squelch switches itself off when a slice enters RTTY, because it would break the decode, and AetherSDR's client-side noise reduction is voice-only and stays off in RTTY.

6.4.1 Setup

Mark frequency

The radio's default Mark tone is **RTTY Mark Default** in **Settings** → **Radio Setup...** → **Phone & CW (Digital group)**. The default is **2125 Hz**, which is the worldwide amateur RTTY standard. Other common values are listed under Mark frequencies (p. 112).

The radio stores a global default (`rtty_mark_default`) that applies to all new RTTY sessions. AetherSDR reads this from the radio on connect.

Shift

The shift is the frequency difference between the Mark and Space tones. The standard amateur RTTY shift is **170 Hz**. Other shifts exist (for example 850 Hz for military/commercial) but are uncommon on the amateur bands.

The shift is set per slice via `rtty_shift` in the radio's slice status.

6.4.2 Using RTTY

Tuning with the Mark and Space lines

RTTY transmits binary data by switching between two audio frequencies, Mark and Space (see Mark and Space tones (p. 111)). The difference between them is the **Shift** — by default 170 Hz, the standard amateur RTTY shift.

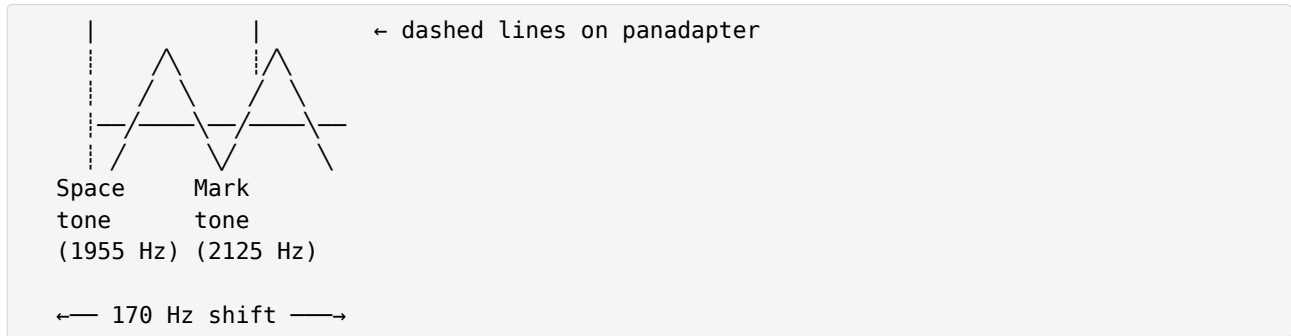
When you switch to RTTY mode, AetherSDR displays two dashed vertical lines on the panadapter labelled **M** (Mark) and **S** (Space). These lines show exactly where the two RTTY tones sit in the audio passband, and they move with your VFO frequency.

When you tune to an RTTY signal, position the two dashed lines so that the Mark line sits on the higher-frequency tone and the Space line sits on the lower-frequency tone. When both tones fall on their respective lines, the signal is properly tuned and can be decoded.

Panadapter view (RTTY mode):

S

M



The M/S lines appear only in **RTTY** mode. A DIGL slice shows the normal carrier marker, not RTTY cues.

Built-in RTTY decoder

When the selected slice is in **RTTY**, a decoder pane opens below the waterfall and shows the decoded text (Baudot/ITA2, with LTRS/FIGS shift tracking). Its controls are listed under Decoder controls ([p. 112](#)).

The stats bar updates about twice a second with the mark and space levels, the SNR and **LOCKED / UNLOCK**. Mark, Shift, Baud, REV and Sens are remembered.

Which audio is decoded

Like the CW Decoder ([p. 104](#)), the RTTY decoder follows the **selected slice**. On a FlexRadio, assign a **DAX RX channel** to that slice and the decoder reads it alone, regardless of speaker gain, mute or other slices. Without one it uses the shared receive audio and shows **RX: shared audio**.

Closing and reopening the pane

Closing the pane with **×** turns the decoder off, and it stays closed on later RTTY slices. To bring it back, open **Settings → Radio Setup... → Phone & CW** and set **RTTY Decode** (in the **Digital** group) to **Enabled**.

Using external RTTY software

Many operators use external RTTY software (MMTTY, fldigi and so on) connected through AetherSDR's DAX virtual audio and CAT control:

1. **Enable DAX** — Assign a DAX channel (for example DAX 1) to the slice and make sure DAX is running in the **DAX** applet. This routes the slice's receive audio to a virtual audio device your RTTY software can open.
2. **Set up CAT** — In the **CAT** applet, enable a port (rigctld, TS-2000 or Flex dialect, or a virtual serial port on Linux/macOS). Your RTTY software uses it to key the radio and read the frequency.
3. **Select RTTY mode** — Switch the slice to RTTY mode. The radio applies the correct filter bandwidth and enables its RTTY demodulator.
4. **Use DIGU/DIGL for AFSK** — If your RTTY software generates audio-frequency shift keying (AFSK) rather than expecting the radio to do FSK, use DIGU or DIGL mode instead of RTTY. The Mark/Space lines and the built-in decoder do not appear in DIGU/DIGL, since the demodulation is handled by your software.

See DAX Virtual Audio ([p. 128](#)) and CAT Control ([p. 132](#)) for setup details.

6.4.3 Reference

Mark and Space tones

Tone	Meaning	Default frequency
Mark (M)	Binary 1 — the “key down” tone	2125 Hz above the carrier
Space (S)	Binary 0 — the “key up” tone	$2125 - 170 = 1955$ Hz above the carrier

Decoder controls

Control	Options	What it does
Mark	Auto , 2125, 2210, 1700, 1275, 1000, 915, 850, 500 Hz	The Mark tone. Auto follows the slice's mark setting from the radio, live.
Shift	45, 50, 75, 100, 170 , 182, 200, 240, 425, 450, 500, 850 Hz	Mark-to-space shift
Baud	45.45 , 50, 75, 100, 110, 150, 300	Symbol rate
REV	On / off	Reverse polarity — space above mark (inverted signal)
Sens	0–100 (default 0)	Display squelch: hides characters the decoder is not confident about, so noise between transmissions doesn't fill the pane. 0 shows everything; about 38 hides what the stats bar calls UNLOCK; 100 keeps only near-certain copy. Display only — nothing is retuned.
CPY ALL	—	Copy all decoded text to the clipboard
CLR	—	Clear the text
×	—	Close the decoder pane

Mark frequencies

Mark frequency	Usage
2125 Hz	Amateur radio standard (default)
2295 Hz	Commercial / maritime RTTY
1275 Hz	Some European stations

Recommended RTTY software

Software	Platform	Notes
fldigi	Linux, macOS, Windows	Full-featured, supports many digital modes
MMTTY	Windows	Classic RTTY-specific decoder
WSJT-X	Linux, macOS, Windows	Primarily FT8/FT4, but connects via the same DAX/CAT path

Typical RTTY operating frequencies

Band	Frequency range	Notes
80m	3.580 – 3.600 MHz	RTTY/Data segment
40m	7.040 – 7.080 MHz	RTTY activity centre ~7.080
30m	10.130 – 10.150 MHz	RTTY ~10.143
20m	14.080 – 14.100 MHz	RTTY activity centre ~14.085
15m	21.080 – 21.100 MHz	RTTY ~21.085
10m	28.080 – 28.100 MHz	RTTY ~28.085

6.4.4 Known issues

- The squelch threshold line can still appear on the panadapter in RTTY, although squelch is off in that mode (#6276¹¹).

6.4.5 Troubleshooting

Mark/Space lines are in the wrong position

The Mark frequency may not match the station you're receiving. Most amateur stations use 2125 Hz Mark / 170 Hz Shift.

¹¹<https://github.com/aethersdr/AetherSDR/issues/6276>

1. If the lines don't align with the signal's two tones, check **RTTY Mark Default** in **Settings → Radio Setup... → Phone & CW**.

I don't see the M/S lines

The Mark and Space lines only appear in **RTTY** mode. They do not appear in **DIGU**, **DIGL**, or other modes.

1. Make sure the slice mode selector shows **RTTY**.

The decoder pane doesn't appear

The pane opens only for a selected slice in **RTTY** mode, and closing it with **×** turns the decoder off.

1. Select the slice and set it to **RTTY**.
2. If you closed the pane earlier, set **RTTY Decode** to **Enabled** in **Settings → Radio Setup... → Phone & CW**.

The decoder prints gibberish between transmissions

The decoder is printing noise while it has no lock.

1. Raise **Sens** on the decoder bar (around 38 hides unlocked copy).

Audio sounds like two alternating tones but no decode

The tones are not on the M and S lines, or the decoder settings don't match the signal.

1. Tune your VFO until both tones sit on their respective dashed lines.
2. Check that **Shift**, **Baud** and **REV** match the signal.
3. If you use external software, verify the DAX channel is routing audio correctly.

6.4.6 See also

- CW Decoder (p. 104)
- DAX Virtual Audio (p. 128)
- CAT Control (p. 132)
- WSJT-X Integration (p. 125)
- Hermes-Lite 2 (p. 23)

6.5 RADE Digital Voice

Status: Supported · **Applies to:** FlexRadio

AetherSDR includes RADE v1, an AI-based digital voice codec from the FreeDV¹² project. RADE uses a neural network to encode speech into an OFDM waveform for transmission over HF radio, then decode it back to speech on the receiving end. All the encoding and decoding happens on your PC; the radio just carries the waveform in a data mode.

The radio is set to **DIGU** or **DIGL** (SSB passthrough) — it just transmits and receives the raw OFDM waveform. On your PC:

- **TX path:** microphone → LPCNet feature extraction → RADE encoder → OFDM modulation → radio transmits
- **RX path:** receive audio from the radio → RADE demodulator → neural decoder → FARGAN vocoder → speaker

Contributed by @pepefrog1234

6.5.1 Requirements

- **A FlexRadio.** RADE's modem receives on a DAX channel, and only a FlexRadio offers one. On other radio families AetherSDR refuses to start RADE.

¹²<https://freedv.org/>

- **A build with RADE support.** RADE appears in the mode list only when AetherSDR is built with it, which the Linux, macOS and Windows release builds include.
- **Both stations** must be running RADE — one station transmits RADE OFDM, the other decodes it.
- Compatible with other FreeDV RADE v1 implementations (FreeDV GUI, etc.).

6.5.2 Using RADE

Turn RADE on

Select **RADE** from the mode dropdown in either the RX applet or the VFO widget. AetherSDR then:

- sets the radio to **DIGU** or **DIGL** using the band's usual sideband (LSB bands get DIGL; 60 m uses DIGU);
- sets the filter to **750–2250 Hz** (mirrored for DIGL);
- makes the RADE slice the **TX slice**.

RADE stays active across PTT cycles. It switches itself off if the radio or another client changes the slice to a mode other than DIGU/DIGL. If the RADE engine fails to start, the slice's previous mode, filter and TX assignment are restored.

You can also start RADE from **Tools → FreeDV Reporter...**: double-clicking a station tunes to it and switches the slice to RADE.

Read sync and SNR

While RADE is active, the VFO flag shows:

- a **RADE** badge with a sync LED next to the frequency — **green ●** when the modem is synchronised, grey ○ when it is not;
- a **RADE info row** below the frequency with the far-end **callsign** (when received), the decoded **SNR** and the **frequency offset**.

The same readouts are in the **RADE Status** applet (button **RADE**, in the applet drawer, closed by default), which shows “RADE inactive” when RADE is off.

The RADE modem uses pilot symbols for synchronisation and can acquire signals down to approximately 0 dB SNR.

End-of-over callsign

At the end of each over, RADE sends an **end-of-over (EOO)** frame that carries the transmitting station's callsign.

- **Receiving:** the far end's callsign is decoded from its EOO frame, shown in the RADE info row and the RADE Status applet, and posted to FreeDV Reporter when reporting is on.
- **Transmitting:** AetherSDR encodes **your** callsign into your EOO frame and holds the transmitter until the frame has played out. The callsign comes from **SpotHub → FreeDV**: the radio's callsign when **Use radio** is ticked, otherwise the **Callsign** you entered there. If neither is set, no callsign is sent.

Report to FreeDV Reporter

FreeDV Reporter is the FreeDV community's live view of who is on the air.

- Turn on reporting in **SpotHub → FreeDV** with **Enable FreeDV Reporter reporting when RADE is active**, and set **Callsign** (or **Use radio**), **Grid Square** (or **Use GPS**) and **Station Msg**.
- **Tools → FreeDV Reporter...** opens the reporter panel. Its **Message** field is the same setting as SpotHub's **Station Msg** — the two stay in sync, and changes go out live. The field is disabled while reporting is off.
- Double-click a station in the panel to tune to it and force RADE.

See SpotHub (p. 148) for the FreeDV spot feed.

6.5.3 Reference

Item	Value
Codec	RADE v1 (Radio Autoencoder) by David Rowe VK5DGR
Library	radae_nopy (https://github.com/drowe67/radae_nopy) (BSD-2 license), bundled
Dependency	Opus with FARGAN/LPCNet, built from a vendored snapshot — nothing is downloaded during the build
Resampling	r8brain-free-src polyphase resampler for all rate conversions
Mode selection	DIGU or DIGL following the band's conventional sideband
Filter	750–2250 Hz for DIGU, –2250 to –750 Hz for DIGL (set automatically)

6.5.4 Troubleshooting

“RADE Unavailable” when selecting RADE

The connected radio has no DAX audio. RADE's modem receives on a DAX channel, and only a FlexRadio provides one, so AetherSDR declines and resets the RADE control. The slice's mode is left as it was.

1. Connect to a FlexRadio to use RADE.

RADE is missing from the mode list

The build was made without RADE support.

1. Install a Linux, macOS or Windows release build, which include RADE.

RADE turned itself off

The radio or another client changed the slice to a mode other than DIGU/DIGL.

1. Select **RADE** again from the mode dropdown.

The far end receives no callsign

No callsign is set for the end-of-over frame.

1. Open **SpotHub** → **FreeDV**.
2. Tick **Use radio**, or enter your **Callsign**.

6.5.5 See also

- [SpotHub](#) (p. 148)
- [DAX Virtual Audio](#) (p. 128)
- [TX Controls](#) (p. 68)
- [FlexRadio](#) (p. 20)
- [FreeDV project](#)¹³
- [radae_nopy](#)¹⁴

6.6 D-STAR (ThumbDV)

Status: Supported · **Applies to:** FlexRadio

AetherSDR can operate D-STAR digital voice on a FlexRadio using a **ThumbDV** or **DV3000U** AMBE vocoder plugged into **your computer**. A small bundled helper, `aether-dv-waveform`, registers a **DSTR** mode with the radio, does the D-STAR framing on the PC, and uses the ThumbDV for voice encode and decode. The ThumbDV stays on the computer running AetherSDR; it does not go into the radio.

D-STAR is a registered trademark of Icom Inc. AetherSDR is not affiliated with or endorsed by Icom Inc.

¹³<https://freedv.org/>

¹⁴https://github.com/drowe67/radae_nopy

6.6.1 Requirements

- **A FlexRadio with waveform support**, reached directly on your network. The helper uses the Flex waveform interface, so it does not run over SmartLink (see SmartLink Setup (p. 199)), and it is not available on other radio families. On a radio without waveform support the D-STAR controls are hidden.
- **A ThumbDV or DV3000U** on a USB port of the computer. AetherSDR includes no software AMBE vocoder.
- **A build that includes the aether-dv-waveform helper**. In a build without it, the DSTR mode and the D-STAR controls are not shown.
- **Serial-port access to the ThumbDV:**
 - **Linux:** official packages install the udev rule `70-aethersdr-thumbdv.rules`, which gives the desktop user access to the ThumbDV (FTDI 0403:6015). Unplug and replug the ThumbDV after installing. On systems without systemd-logind, add yourself to the serial-device group (usually `dialout` or `uucp`) and log out and back in.
 - **Windows:** the ThumbDV appears as a COMn port. The helper opens it exclusively, so close any other program using it. The installer adds the helper's firewall rule.
 - **macOS:** the ThumbDV appears as `/dev/cu.usbserial-*`.

6.6.2 Setup

Open **Tools** → **Waveforms...** and find the **D-STAR** card under **Local Digital Voice**.

Control	What it does
ThumbDV device	Pick the ThumbDV's serial port, or type it (for example COM3, <code>/dev/ttyUSB0</code> , <code>/dev/cu.usbserial-...</code>). Refresh rescans. On Linux the stable <code>/dev/serial/by-id/...</code> name is preferred.
Auto-start	Start the D-STAR service automatically.
Station and routing	MYCALL , Suffix , URCALL , RPT1 , RPT2 and a 20-character Message .
Advanced	The helper's executable path. Leave it blank to use the bundled helper.

- A ThumbDV is only saved once it has answered with a valid DV3000 reply, so a wrong serial port is never remembered by mistake.
- **MYCALL** defaults to the radio's callsign. It must be 3–8 letters and digits, containing at least one of each. Suffix, routing and message fields are checked before the helper starts, so a malformed header is never transmitted.
- The default route is **CQCQCQ / DIRECT / DIRECT** (URCALL / RPT1 / RPT2).

6.6.3 Using D-STAR

Choose DSTR on a slice

Set the slice mode to **DSTR**. DSTR runs over the radio's DFM path, so the VFO flag's FM options apply: repeater offset, direction, simplex and reverse are shown, and CTCSS is hidden.

Only one digital-voice slice can be active at a time.

The D-STAR tab in AetherModem

Open **Tools** → **AetherModem...** and select the **D-STAR** tab. The header shows the service state, which slice is in DSTR (or **No DSTR slice**), and a **Start / Stop** button for the service.

Station & Route

Field	What it does
MYCALL / Suffix	Your callsign and optional 4-character suffix.
From	Direct or Repeater .
To	Local CQ , Specific station , Repeater area or Custom .
Access repeater	Repeater callsign and module, when going through a repeater.
Destination station	Destination callsign and module, when calling a specific station or repeater area.

Field	What it does
Advanced route details	The raw URCALL , RPT1 and RPT2 fields, plus the helper executable.

Voice Service shows the vocoder (**ThumbDV / DV3000**), the **ThumbDV device** with **Refresh**, and **Autostart at launch**.

D-STAR Traffic logs D-STAR activity (station, destination, URCALL / RPT1 / RPT2 and slice). Use the search box to filter it, or clear it. Below the log, type a **TX MESSAGE** of up to 20 characters and click **Set TX Message**; the **ACTIVE** line shows the message currently being sent.

The footer shows the **MODEM** state, the **ROUTE** and recent **ACTIVITY**.

Transmitting

- D-STAR is sent **only when you key the DSTR slice** (MOX, PTT or VOX). AetherSDR never keys the radio to start the service.
- A **TUNE** carrier stays a plain tune carrier; the helper does not send D-STAR data during TUNE.

Controllers and shortcuts

DSTR is available as a mode shortcut and as a MIDI mode action. While the D-STAR helper is not running, **Mode Up / Mode Down** on MIDI controllers skip over DSTR instead of stopping on it.

6.6.4 Known issues

- Received D-STAR audio is silent or stops after a short time, and transmissions are not recognised as D-STAR by the far station (#4277¹⁵).
- A DV3000U ThumbDV is not detected on macOS (#5780¹⁶).

6.6.5 Troubleshooting

No D-STAR card or tab, and no DSTR mode

The build has no `aether-dv-waveform` helper, or the connected radio has no waveform support.

1. Use a build that includes the helper.
2. Connect to a FlexRadio with waveform support, directly on your network.

The D-STAR service will not start

You are connected over SmartLink, which the helper does not support, or MYCALL is missing or invalid.

1. Connect to the radio directly on your network instead of through SmartLink.
2. Check **MYCALL**: 3–8 letters and digits, with at least one of each.

The ThumbDV is “in use”

Another program has the serial port open. The helper opens it exclusively.

1. Close the other program using the port.
2. Start the D-STAR service again.

Access denied to the ThumbDV on Linux

Your user has no permission on the serial device.

1. Install the official package, which adds the udev rule `70-aethersdr-thumbdv.rules`.
2. Unplug and replug the ThumbDV.
3. On systems without systemd-logind, add yourself to `dialout` or `uucp`, then log out and back in.

¹⁵<https://github.com/aethersdr/AetherSDR/issues/4277>

¹⁶<https://github.com/aethersdr/AetherSDR/issues/5780>

6.6.6 See also

- AetherModem Packet Radio (p. 118)
- MIDI Controller Mapping (p. 192)
- SmartLink Setup (p. 199)
- FlexRadio (p. 20)
- Design and protocol notes: docs/architecture/digital-voice-thumbdv-waveform.md¹⁷

6.7 AetherModem Packet Radio

Status: Supported · **Applies to:** FlexRadio, Hermes-Lite 2 (experimental), Networked Icom (early; IC-7300MK2 supported)

AetherModem is AetherSDR's built-in AX.25 packet modem. It decodes and transmits packet without an external TNC or a sound-card modem such as Dire Wolf, and on top of it sit an **APRS client**, a **WIDE1-1 fill-in digipeater**, a **KISS-over-TCP TNC** for other packet programs, a connected-mode **terminal** for calling a BBS, and a **personal mailbox** that other stations can connect to.

AetherModem works on FlexRadio, on the Hermes-Lite 2 (p. 23) and on Networked Icom (p. 28) radios. On a FlexRadio it transmits through DAX TX: the modem keys PTT itself and restores your previous DAX state afterwards.

6.7.1 Setup

Open it from **Tools** → **AetherModem...**

Window layout

The window has six tabs: **APRS**, **Digipeater**, **KISS TNC**, **Terminal**, **Mailbox** and **D-STAR** (the D-STAR tab is covered on D-STAR (ThumbDV) (p. 115) and is hidden on radios without waveform support). A slim status bar along the bottom shows the modem state, the receive gain and a packet-activity strip.

All AetherModem transmissions share one queue and key the radio one at a time, so the APRS client, digipeater, KISS clients, terminal and mailbox never talk over each other.

Modem and profiles

The modem controls are on the **APRS** tab:

Control	What it does
BAUD RATE	300 baud (HF, 1600/1800 Hz tones) or 1200 baud (VHF Bell 202, 1200/2200 Hz tones)
Enable Modem	Starts decoding on the attached slice
Autostart at launch	Starts the modem when AetherSDR starts

The 1200-baud receiver uses a Direwolf-derived AFSK demodulator. The 300-baud HF receiver is AetherModem's own multi-lane decoder.

Packet length, retry timing and preamble are derived from the baud rate, not fixed values sized for VHF. On HF the packet length defaults to 64 bytes. The 1200-baud preamble is shorter than the HF one; raise it if a transverter's T/R switching clips the start of a burst.

Tips for receive level: avoid clipping (peaks around -10 dBFS are fine) and don't chase every missed decode with AF gain once the tones are clearly visible.

¹⁷<https://github.com/aethersdr/AetherSDR/blob/main/docs/architecture/digital-voice-thumbdv-waveform.md>

6.7.2 Using AetherModem

APRS

The APRS tab is a lightweight APRS client.

- **MY CALLSIGN** (for example N0CALL - 9), **SYMBOL** and **PATH**.
- **BEACON**: Every *n min*, a status text, and **Beacon Now**.
- **POSITION**: taken from the radio's GPS when it has a fix (for example a FlexRadio with the GPSDO option). Without GPS, type a Maidenhead locator in **GRID** (for example JN48Qm, which fills the latitude and longitude) or enter **MANUAL LAT / LON**.
- **Station table**: TIME, STATION, SYMBOL, AGE, PKTS, GRID, DIST, CRS/SPD and STATUS / COMMENT, with symbol icons and decoded weather reports.

Messaging

Type the recipient in **To**, the text beside it, and press **Send APRS Msg**. Messages ask for an acknowledgement and are retried until acknowledged; incoming messages are acknowledged automatically, and digipeated duplicates are dropped. The  button opens the message view.

The ▼ button beside the To field addresses common APRS services and fills in the right syntax (hover an entry for its hint):

Entry	Service
SMS	Text a phone through the SMS gateway: @<number> <message>
EMAIL-2	Send an email: <address> <message>
WLNK-1	Winlink APRSLink: ? for help, or a one-line email/SMS
WXBOT	Weather forecast or METAR for a city, airport code or grid
ISS / ARISS	Path hint only: set the TX path to ARISS for the ISS digipeater

Internet gateways (APRS-IS) are not part of AetherModem.

Digipeater

A **WIDE1-1 fill-in digipeater**, for filling gaps in local APRS coverage.

- **Enable WIDE1-1 fill-in** arms it. It answers the first unused hop when that hop is WIDE1 - 1, and optionally your own call (**MYCALL**) or the legacy **RELAY** alias. It inserts your callsign with the “has been repeated” bit set.
- It needs the **1200 baud** profile and a valid callsign, and it must be **armed again each session**: your configuration is saved, but arming is never restored at startup.
- Switching to 300 baud, disabling the modem, changing the slice, disconnecting or closing AetherSDR disarms it.
- It does not decrement WIDE2 - n hops; it is a fill-in digipeater, not a wide-area one.
- The tab has its own **CALL**, **ALIAS**, **BEACON** (with **Now**), **SYMBOL**, **PATH** and **TEXT** fields, **MESSAGES**, **DIGIPEATS** and **SKIPPED** traffic graphs, and a **RAW MESSAGES** list in TNC-2 format.

KISS TNC

Turns AetherModem into a KISS-over-TCP TNC so other packet and APRS programs can use it, for example Xastir, YAAC, APRSdroid, UISS, or anything that talks to Dire Wolf's KISS port.

- **KISS TNC SERVER: Enable TNC**, **Start TNC on Startup** and **TCP PORT** (default **8001**, listening on all network interfaces). All three are saved.
- With **Start TNC on Startup**, the TNC runs in the background from launch and keeps running when the window is closed.
- Several programs can connect at once. Every decoded frame goes to all of them; frames they send are transmitted with the baud profile chosen on the APRS tab.

- Enabling the TNC turns the modem on, but a slice must be attached for traffic to flow.
- **TNC STATUS** shows the port, the number of clients and RX/TX frame counts.

Terminal

A connected-mode AX.25 client for calling a packet BBS or another station's mailbox.

- **MY CALLSIGN** and **CONNECT TO**, with **Connect**, **Cmd Mode** and **MHeard**.
- **LINK PARAMETERS: Retry s, Tries, Paclen, TXD flags** and **TX Tail ms**. Retry, Paclen and TXD default to **Auto**, worked out from the current baud rate; set a number to override.
- **Log session to file** records the session.
- Each session's measured round-trip time is logged (the **AX.25 Link** log category), which tells a mis-sized timeout from a poor channel.
- Asking to disconnect a second time ends a disconnect that is still retrying. Turning the modem off stops transmitting and drops the session.

Mailbox

A personal mailbox (PMS) in the style of the Kantronics KPC-3. One station at a time can connect to it over 1200-baud connected mode, read and leave messages, see who has been heard, and disconnect.

- **Enable Mailbox (PMS)**, the **LISTEN CALLSIGN** (full CALL - SSID, for example N0CALL - 10), and an optional **VANITY ALIAS** of up to 6 characters (for example AETBBS) that it also answers on.
- **WELCOME / PTEXT** greeting, and an optional **Send hourly beacon** with its own **BEACON TEXT**.
- **MAILBOX STATUS, STATISTICS** and **LAST CALLERS**.

Commands for callers (first letter or the full word): H help, B bye, I info, J heard list, L list, LM list mine, R n read, K n kill, S / SP call send, SB cat send bulletin, U users. End a message with /EX or Ctrl-Z on its own line. Address a public message to ALL.

Messages, callers and the heard list are kept in ~/.config/AetherSDR/pms/ (messages.json, callers.json, heard.json).

MQTT

Decoded frames are published on aethersdr/ax25/rx, and AetherModem transmits frames received on aethersdr/ax25/tx. See MQTT Station Automation (p. 141).

6.7.3 Troubleshooting

The **AetherModem** log category covers decoding, transmit and the KISS server; **AX.25 Link** covers connected-mode sessions. See Support and Logging (p. 261).

Nothing decodes

The modem is off, listening to the wrong slice, set to the wrong baud rate, or the receive level is clipping.

1. On the **APRS** tab, check **Enable Modem** is on.
2. Check the right slice is attached.
3. Match the **BAUD RATE** to the channel: **300 baud** on HF, **1200 baud** on VHF.
4. Lower the receive level until it no longer clips; peaks around -10 dBFS are fine.

A KISS client connects but nothing transmits

The modem has no slice attached, or the radio can't transmit.

1. Attach a slice to the modem. Enabling the TNC turns the modem on, but traffic only flows with a slice attached.
2. Check that the radio is able to transmit.

The digipeater is off after a restart

Arming the digipeater is never restored at startup; only its configuration is saved. Switching to 300 baud, disabling the modem, changing the slice or disconnecting also disarms it.

1. Make sure the modem is on the **1200 baud** profile and has a valid callsign.
2. On the **Digipeater** tab, tick **Enable WIDE1-1 fill-in** again.

The start of each 1200-baud burst is clipped

A transverter's T/R switching is cutting into the preamble, which is shorter at 1200 baud than on HF.

1. On the **Terminal** tab, under **LINK PARAMETERS**, set **TXD flags** (the preamble, in HDLC flags before every frame) to a number higher than **Auto** gives.

6.7.4 See also

- DAX Virtual Audio ([p. 128](#))
- D-STAR (ThumbDV) ([p. 115](#))
- MQTT Station Automation ([p. 141](#))
- Hermes-Lite 2 ([p. 23](#))
- Support and Logging ([p. 261](#))
- docs/MODEM.md¹⁸ and docs/HFMODEM.md¹⁹ (developer notes)

6.8 Copy Assist

Status: Supported · **Applies to:** All radios · **Platforms:** Linux, Windows, macOS (Apple Silicon)

Copy Assist is on-device speech-to-text for received voice. It transcribes what you are hearing on SSB, AM or FM into a live, scrolling text panel docked under the waterfall — an aid for weak or noisy copy, for accessibility, and for following nets and contests. It runs `whisper.cpp`²⁰ on your own computer, is **receive-only**, and never keys the transmitter.

Copy Assist is host-side, so it works with every supported radio family, including receive-only setups and sessions with TX disabled.

6.8.1 Requirements

- **Platform:** the Linux, Windows and Apple Silicon macOS builds include Copy Assist. The Intel macOS DMG ships without speech-to-text; building from source on an Intel Mac still works.
- **Mode:** a voice mode on the selected slice (see [When it is available \(p. 122\)](#)).
- **Model download:** the first time you enable it, a speech model downloads (see [Models \(p. 121\)](#)).
- **ONNX Runtime** for Silero VAD and speaker labels. The release builds include it.

6.8.2 Setup

Models

Model weights are **not bundled**. The selected model downloads the first time you enable Copy Assist — from Hugging Face, with AetherSDR's GitHub release as a fallback — and is SHA-256 verified before use. Models are cached in the app data folder (on Linux, `~/.local/share/AetherSDR/models/`).

Model	Size	Notes
tiny	74 MB	Fastest, roughest
base	141 MB	Default on CPU and ARM, including the Raspberry Pi 5
small	465 MB	Desktop CPU
large-v3-turbo	1.6 GB	Default when a usable GPU with enough free memory is found

¹⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/MODEM.md>

¹⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/HFMODEM.md>

²⁰<https://github.com/ggml-org/whisper.cpp>

Offline or air-gapped: copy a `ggml-*.bin` model file into the models folder by hand.

Custom model... loads any whisper `ggml-*.bin` or `.gguf` file you choose — a fine-tune or a different quantization. It loads directly, with no download or checksum, and is remembered as **Custom: <name>**.

sherpa-onnx model... (Linux and macOS builds that include sherpa-onnx) runs a non-whisper offline model such as Zipformer, Moonshine or Paraformer. Point it at the extracted model folder. These are much faster than whisper on CPU and ARM, so they are the practical choice on a Raspberry Pi.

GPU acceleration

The model runs on the GPU when one is available, otherwise on the CPU:

- **Vulkan** on Linux and Windows (NVIDIA, AMD or Intel).
- **Metal** on macOS (Apple Silicon).

If a GPU cannot be used — not enough free memory, or a driver fault — Copy Assist falls back to the CPU rather than crashing. The compute picker marks unusable devices (*unavailable*) and says when a fallback happened, without overwriting your saved choice.

A few faults inside the speech engine close the app outright. Copy Assist records which step it was in, so on the **next** launch:

- a GPU that failed while loading a model is left out, and the model runs on the next device;
- a failure while starting the engine or loading on the CPU keeps local Copy Assist off for that session (a remote server still works).

The settings window shows the reason under **Compute** with a **Try again next launch** button — use it after a driver update, or if something else closed the app. Updating AetherSDR clears the note automatically.

Remote server

Instead of the built-in engine, Copy Assist can send audio to **your own OpenAI-compatible /v1/audio/transcriptions endpoint** — whisper.cpp's `whisper-server`, `faster-whisper`, or any compatible server — so inference runs on another machine.

Choose **Remote server...** in the model picker and enter the endpoint URL (for example `http://host:8080/v1/audio/transcriptions`), an optional API key, and the model name. AetherSDR ships **no server and no default endpoint**; nothing leaves your computer unless you configure this. The API key is stored in the OS keychain, not in the settings database.

6.8.3 Using Copy Assist

Opening Copy Assist

- Click the **ASR** toggle in the status bar (between **CWX** and **DVK**), or
- **Tools → Copy Assist**.

Either one shows or hides the panel, which docks under the waterfall of the active panadapter. Drag the panel's grip to resize it; the height is remembered.

When it is available

- **Voice modes only:** USB, LSB, AM, SAM, FM, NFM and DFM. In CW, DIGU/DIGL and RTTY the toggle is dimmed.
- The mode check follows the **slice you have selected**, not the transmit slice. The CWX and DVK indicators follow the TX slice instead, so the three can legitimately disagree — with a CW transmit slice and a USB slice selected, CWX and ASR are both live.
- Selecting a slice that is not in a voice mode closes the panel and stops transcription. To resume, select the voice slice, reopen the panel and enable it again.
- A band-stack recall or a disconnect does **not** close an open panel, so transcription survives a band change.

- Whenever the panel is open, the ASR toggle is lit and clickable, so you can always close it by hand.

Transcribing

1. Open the panel and click **Disabled** so it reads **Enabled**.
2. The first time, the selected model downloads, is verified and loads. A loading indicator shows progress.
3. Transcribed text streams into the panel.

Text is **coloured by recognition confidence**, the same way as the CW Decoder (p. 104):

Colour	Confidence
Green	High
Yellow	Medium
Orange	Questionable
Red	Low

Copy Assist hears the audio after client-side noise reduction, so a suitable DSP Noise Mitigation (p. 241) setting can improve the transcript as well as your own listening.

Context

Context conditions each decode on the text of the previous confident segment, which helps with continuity of names, callsigns and topic. It applies live, so you can A/B it on the fly.

- A low-confidence segment is not carried forward, so one garbled over cannot poison the next.
- A real pause, **Clear**, or a retune starts fresh.
- Context works with the whisper models only. It is greyed out on the sherpa-onnx and remote backends.

The Queue readout

The status line shows **Queue: N s** — seconds of received audio not yet transcribed. It stays near zero when the engine keeps up and climbs (amber, then red) when it cannot, for example a large model on a Raspberry Pi.

The backlog cannot grow without limit. Once it reaches twice the Buffer setting (never less than 10 s), Copy Assist drops incoming audio until it has drained to half that, and the readout changes to **Queue: N s · dropped M s** in red. Those seconds were never transcribed, so the transcript has gaps. A retune or Disable resets both numbers. If you see drops, use a smaller model, the remote backend, a longer Buffer, or faster hardware.

6.8.4 Reference

Panel controls

Control	Range	What it does
Enabled / Disabled	—	Starts and stops transcription. Closing the panel turns it off.
⚙	—	Opens Copy Assist Settings (below).
Buffer	1–20 s	The longest stretch of audio collected before a decode is forced, even without a pause.
Sens	1–100 %	Voice-activity sensitivity. Higher picks up fainter speech.
Silence	100–2000 ms	How much trailing silence ends an utterance.
A– / A+	—	Transcript text size.
↵	—	Start a new line after each pause.
Context	Off by default	Carries context across segments (see below).
Clear	—	Clears the transcript. Also on the text area's right-click menu.

Buffer, Sens, Silence and the panel height are remembered.

Copy Assist Settings (⚙)

The ⚙ button opens a small modeless settings window that can stay open while you operate.

Setting	What it does
Model	Which speech model runs (see Models (p. 121)). Also Custom model... , sherpa-onnx model... and Remote server...
Compute	Which device runs the model: a GPU or the CPU. Shows Detecting... while GPU discovery runs in the background.
Language	The spoken language to transcribe (multilingual models only).
Save transcript to a file	Appends each finished utterance as a timestamped line. A date is inserted before the extension, so <code>net.txt</code> writes to <code>net-2026-07-21.txt</code> and rolls over daily. A frequency marker line is written when ASR starts, on every retune, and at the top of each day's file.
Use Silero VAD (ONNX)	Replaces the built-in energy detector with the small Silero neural voice-activity model, which segments real speech far better in HF noise. Downloads on demand; Browse... lets you use your own <code>.onnx</code> .
Label speakers (A/B/C...)	Tags each utterance with a speaker label ([A] ..., [B] ...) using a downloaded speaker-embedding model. Match threshold (0.00–1.00) sets how strict the matching is: higher splits voices more finely, lower merges them. Labels reset on retune or re-enable.
Boundary overlap	0–2000 ms, Off by default. When a long over is cut at the Buffer limit, the last N ms carry into the next segment so a word split at the cut is decoded whole, and the repeated words are removed. A few hundred ms is the practical setting. It does nothing useful for languages written without spaces (Chinese, Japanese, Thai).

Silero VAD and speaker labels need an ONNX Runtime build; the release builds include it.

6.8.5 Known issues

- On Apple Silicon, transcribing in heavy noise can freeze the whole app for several seconds at a time, worse with MNR on (#5107²¹).

6.8.6 Troubleshooting

The ASR toggle is dimmed

The selected slice is not in a voice mode. Copy Assist follows the slice you have selected, not the transmit slice.

1. Select a slice in USB, LSB, AM, SAM, FM, NFM or DFM.
2. Reopen the panel and click **Disabled** so it reads **Enabled**.

The panel closed and transcription stopped

You selected a slice that is not in a voice mode.

1. Select the voice slice again.
2. Reopen the panel and enable it again.

The transcript falls behind, or shows “dropped” in red

The engine cannot keep up with the audio. Once the backlog reaches twice the Buffer setting (never less than 10 s), Copy Assist drops audio, so the transcript has gaps.

1. Choose a smaller model in  → **Model**, or a sherpa-onnx model on CPU or ARM.
2. Or use a **Remote server...** on a faster machine.
3. Or set a longer **Buffer**.

Copy Assist runs on the CPU instead of the GPU

The GPU could not be used, because it lacks free memory or the driver failed, so Copy Assist fell back to the CPU. If the GPU failed during a crash, it is left out on the next launch.

1. Open  and read the reason under **Compute**. Unusable devices are marked (*unavailable*).
2. After a driver update, or if something else closed the app, click **Try again next launch** and restart AetherSDR.

²¹<https://github.com/aethersdr/AetherSDR/issues/5107>

Copy Assist stays off after the app closed unexpectedly

The speech engine faulted while starting or while loading on the CPU, so local Copy Assist is kept off for that session.

1. Open  and read the reason under **Compute**.
2. Click **Try again next launch** and restart AetherSDR.
3. In the meantime, a **Remote server...** still works.

The model will not download

The computer has no internet access, for example in an air-gapped shack.

1. Copy a `ggml-*.bin` model file into the models folder by hand (on Linux, `~/.local/share/AetherSDR/models/`).
2. Or choose **Custom model...** and point it at a whisper `ggml-*.bin` or `.gguf` file.

6.8.7 See also

- DSP Noise Mitigation ([p. 241](#)) — clean up the audio Copy Assist hears
- CW Decoder ([p. 104](#))
- Accessibility ([p. 231](#))
- Primary reference: `docs/asr-copy-assist.md`²²

6.9 WSJT-X Integration

WSJT-X (and JTDX, JS8Call and similar programs) needs two things from AetherSDR: **rig control** (frequency, mode, PTT) and **audio**. There are two ways to provide them:

Route	Rig control	Audio	Platforms
TCI (simplest)	TCI Server (p. 135)	Same TCI connection	Linux, macOS, Windows
CAT + DAX	CAT Control (p. 132) (Hamlib NET rigctl)	DAX Virtual Audio (p. 128) devices	Linux and macOS (FlexRadio)

On Windows AetherSDR does not ship a DAX audio driver, so use TCI there (or FlexRadio's own SmartSDR DAX).

6.9.1 Setup

Option 1: TCI

1. In AetherSDR, enable the TCI server: **Settings** → **Autostart TCI with AetherSDR**, or the **Enabled** button in the **TCI** applet. The default port is **50001**.
2. In WSJT-X, open **File** → **Settings** → **Radio**:
 - **Rig:** Expert Electronics (TCI) (needs a WSJT-X build with TCI support, such as WSJT-X Improved or WSJT-X 3.x)
 - **Network Server:** `localhost:50001`
 - **PTT Method:** CAT
3. Click **Test CAT**; it turns green.

Audio flows over the TCI connection, so no sound-card devices need setting up. PC Audio does not have to be on in AetherSDR for WSJT-X to receive audio. WSJT-X's **Pwr** slider acts as a digital attenuator on the transmitted audio.

Several WSJT-X instances can share one TCI server. Each one keys the slice it asked for, and a refused PTT is reported back so WSJT-X shows "TCI failed to set ptt" instead of keying the wrong slice. See TCI Server ([p. 135](#)).

²²<https://github.com/aethersdr/AetherSDR/blob/main/docs/asr-copy-assist.md>

Option 2: CAT and DAX (Linux and macOS)

1. AetherSDR

1. Put the slice in **DIGU** mode.
2. Give the slice a DAX channel: VFO widget → **DAX** tab → channel **1**.
3. Turn on **DAX**: the **Enabled** button in the **DAX** applet, or **Settings** → **Autostart DAX with AetherSDR**.
4. Turn on **CAT**: pop out the **CAT** applet, enable a **Rigctl** port (port **4532** on a fresh install) and switch the master **Enabled** on, or use **Settings** → **Autostart CAT with AetherSDR**.

2. WSJT-X audio

File → **Settings** → **Audio**:

- **Soundcard Input:** AetherSDR DAX 1
- **Soundcard Output:** AetherSDR TX

3. WSJT-X radio

File → **Settings** → **Radio**:

- **Rig:** Hamlib NET rigctl
- **Network Server:** 127.0.0.1:4532
- **PTT Method:** CAT

Click **Test CAT**; it turns green. The test can take a while: WSJT-X runs a test sequence on first connect.

4. Verify

- FT8 signals appear in the WSJT-X waterfall, and the **DAX 1** meter in the DAX applet moves.
- Click **Tune** in WSJT-X: the radio keys and the TX applet shows power.

6.9.2 Using WSJT-X with AetherSDR

Several instances at once

Run one WSJT-X (or JTDX, JS8Call, fldigi ...) per slice. Give each its own CAT port with a different **VFO A** slice, and its own DAX channel. For example:

Program	CAT port	VFO A	Audio input
WSJT-X #1	4532	A	AetherSDR DAX 1
WSJT-X #2	4533	B	AetherSDR DAX 2

Split is reported per CAT client, so every instance can use **Split Operation: Fake It**. Band changes from WSJT-X move the panadapter when the new frequency is off-screen or on another band; a tune within the visible span moves only the slice.

Decode spotting

AetherSDR can listen to WSJT-X's UDP decode messages and put the stations it decodes on the panadapter. Each spot lands on the band of the WSJT-X instance that reported it. See the WSJT-X tab on the SpotHub (p. 148) page for filters (CQ, CQ POTA, Calling Me) and colours.

6.9.3 Known issues

- FT8 transmit audio can drop out for about half a second roughly one second into each transmission (#5469²³).
- FT8 transmitted on slice B can show heavy ALC and be decoded far less often than the same signal on slice A (#5340²⁴).
- Over TCI, the mic profile, **MIC** level and **PROC** in the Phone/CW applet change the transmitted digital signal (#5844²⁵).

²³<https://github.com/aethersdr/AetherSDR/issues/5469>

²⁴<https://github.com/aethersdr/AetherSDR/issues/5340>

- Over TCI, the transmitted FT8 signal can show random jumps (#5133²⁶).
- TCI receive audio can stop after moving to a voice frequency with DAX off and back to a data frequency with DAX on (#4824²⁷).
- On macOS, DAX transmit can stop sending audio to the radio until AetherSDR is restarted (#5870²⁸).

6.9.4 Troubleshooting

Nothing decodes

The slice is not delivering audio to WSJT-X.

1. Check the slice is in **DIGU**.
2. With CAT and DAX: check the slice has a DAX channel, and that WSJT-X's input is that DAX device (not your microphone).
3. With TCI: check WSJT-X is connected to the right port.

PTT does nothing

WSJT-X is not keying through CAT, or a different slice holds TX.

1. Set WSJT-X's **PTT Method** to CAT.
2. Make sure the intended slice holds TX (red TX badge).

The frequency does not follow WSJT-X

WSJT-X is not talking to a running CAT port.

1. Check that the port in WSJT-X matches an enabled CAT port.
2. Check that the master CAT switch is on.

WSJT-X shows “Error in Sound Input”

DAX was switched off while WSJT-X was using it.

1. Turn DAX back on.
2. Restart WSJT-X.

Transmit audio does not reach the radio

WSJT-X sends to the wrong device, or the TX slice is not in a digital mode.

1. Set WSJT-X's **Soundcard Output** to **AetherSDR TX**.
2. Put the TX slice in **DIGU**.

6.9.5 See also

- CAT Control (p. 132)
- DAX Virtual Audio (p. 128)
- TCI Server (p. 135)
- SpotHub (p. 148)
- PSK Reporter Map (p. 155)
- WSJT-X home page²⁹

²⁵<https://github.com/aethersdr/AetherSDR/issues/5844>

²⁶<https://github.com/aethersdr/AetherSDR/issues/5133>

²⁷<https://github.com/aethersdr/AetherSDR/issues/4824>

²⁸<https://github.com/aethersdr/AetherSDR/issues/5870>

²⁹<https://wsjt.sourceforge.io/wsجتx.html>

Connecting Other Software

7.1 DAX Virtual Audio

Status: Supported · **Applies to:** FlexRadio · **Platforms:** Linux, macOS

DAX (Digital Audio eXchange) creates virtual audio devices on your computer that digital-mode programs (WSJT-X, fldigi, JS8Call, VARA and others) use as their sound card. Receive audio flows from a slice to the program, and the program’s transmit audio flows back to the radio, all through AetherSDR with no cables.

DAX is a **FlexRadio** feature. On other radio families the DAX applet, menu entry and overlay button are hidden; use the TCI Server (p. 135) for audio there.

7.1.1 Requirements

Platform	DAX
Linux	Built in. Needs PipeWire. Receive channels are native PipeWire streams; they also show up through PipeWire’s PulseAudio and JACK layers.
macOS	Built in, through the DAX Virtual Audio Driver that ships in the AetherSDR DMG; install it from there. If it is missing, enabling DAX shows “DAX Audio Driver Missing”.
Windows	No built-in DAX driver. The DAX applet says “No built-in DAX driver on Windows. Use TCI, or SmartSDR DAX.” Use the TCI Server (p. 135) (audio and CAT over one connection), or FlexRadio’s own SmartSDR DAX.

7.1.2 Setup

Turning DAX on

- The **DAX Audio** applet (applet-tray button **DAX**) has a **DAX: Enabled / Disabled** button at the bottom.
- **Settings** → **Autostart DAX with AetherSDR** is the same setting: while it is on, DAX starts every time you connect. This menu entry is not shown on Windows.

Assigning a slice to a DAX channel

The radio sends a slice’s audio to the DAX channel set on that slice. Choose it on the slice’s **VFO widget** → **DAX tab (DAX Ch)**. The channel list follows the radio’s capacity, and a slice on a channel the radio cannot back shows **Off**.

The panadapter overlay’s **DAX** flyout is for IQ instead: it holds **IQ Ch** and the **WFM** button (see DAX IQ Streaming (p. 130)).

Checking the devices (Linux)

```
pactl list sources short | grep aethersdr # DAX 1..N capture devices
pactl list sinks short | grep aethersdr # TX device
```

Stale devices left by a crash are cleaned up the next time DAX starts.

7.1.3 Using DAX

Transmit audio

When a digital program transmits in a digital mode (DIGU, DIGL), its audio from **AetherSDR TX** goes to the radio. Voice modes use your microphone. You don't switch anything by hand.

Wiring WSJT-X (Linux and macOS)

1. Put the slice in **DIGU** and set its DAX channel to **1**.
2. Turn DAX on, and turn on a **Rigctl** CAT port (see CAT Control (p. 132)).
3. In WSJT-X, **File** → **Settings** → **Audio**: Input **AetherSDR DAX 1**, Output **AetherSDR TX**.
4. **Settings** → **Radio**: Rig Hamlib NET rigctl, Network Server 127.0.0.1:4532, PTT Method CAT.

Full walk-through: WSJT-X Integration (p. 125).

Other programs

Program	Modes	Audio in	Audio out	Rig control
WSJT-X / JTDX	FT8, FT4, JT65, Q65, WSPR	AetherSDR DAX n	AetherSDR TX	Rigctl CAT port, or TCI
fldigi	PSK, RTTY, MFSK, CW, ...	AetherSDR DAX n	AetherSDR TX	Rigctl CAT port
JS8Call	JS8	AetherSDR DAX n	AetherSDR TX	Rigctl CAT port
VARA HF/FM	VARA	AetherSDR DAX n	AetherSDR TX	Rigctl CAT port
Dire Wolf	APRS, packet	AetherSDR DAX n	AetherSDR TX	—

For packet and APRS you may not need an external modem at all: see AetherModem Packet Radio (p. 118).

Two programs on two slices

Give each program its own DAX channel and its own CAT port (with a different **VFO A** slice).

7.1.4 Reference

Devices

AetherSDR offers **up to eight RX channels plus one TX channel**. The number of RX channels follows the connected radio's slice capacity (for example 8 on a FLEX-6700, 4 on a FLEX-6600 or 8600, 2 on a FLEX-6300 or 6400). Rows and devices above that number are not shown.

Device	Direction	Purpose
AetherSDR DAX 1 ... DAX 8	Capture (input)	Receive audio from whichever slice is assigned that DAX channel
AetherSDR TX	Playback (output)	Transmit audio to the radio

The DAX Audio applet

Each RX row shows **DAX n**, the slice currently assigned to that channel (or “—”), and a combined **level meter and gain slider**. The **TX**: row shows which slice holds transmit and the gain applied to audio coming from your program. Gains are saved per channel.

7.1.5 Known issues

- On macOS, a digital-mode program's transmit audio can reach the **AetherSDR TX** meter while the radio sends no RF (#4554¹).
- On macOS, DAX transmit can stop sending audio to the radio altogether until AetherSDR is restarted (#5870²).

- On macOS with a FLEX-8000-series radio, DAX receive audio can come out chopped and very quiet (#3837³).

7.1.6 Troubleshooting

The program hears no audio

The slice has no DAX channel, or the program is reading a different device.

1. On the slice's **VFO widget** → **DAX tab**, set a **DAX Ch**.
2. In the program, choose the matching **AetherSDR DAX n** input.
3. On Linux, `pactl list source-outputs` shows which device each program is reading.

Transmit audio does not reach the radio

The program sends to the wrong device, or the TX slice is in a voice mode.

1. Set the program's audio output to **AetherSDR TX**.
2. Put the TX slice in a digital mode (DIGU or DIGL).

A DAX row is missing

The connected radio has fewer slices than that channel number, so AetherSDR does not show it. Use a lower channel number.

WSJT-X shows “Error in Sound Input”

DAX was switched off while WSJT-X was using it.

1. Turn DAX back on.
2. Restart WSJT-X.

There is no DAX on Windows

AetherSDR ships no DAX driver on Windows.

1. Use the TCI Server (p. 135), which carries audio and CAT over one connection, or FlexRadio's own SmartSDR DAX.

7.1.7 See also

- DAX IQ Streaming (p. 130)
- CAT Control (p. 132)
- TCI Server (p. 135)
- WSJT-X Integration (p. 125)
- AetherModem Packet Radio (p. 118)

7.2 DAX IQ Streaming

Status: Supported · **Applies to:** FlexRadio

DAX IQ delivers raw I/Q samples from the radio's digital down-converter to other SDR programs. Regular DAX Virtual Audio (p. 128) carries demodulated audio from a slice; DAX IQ carries the unprocessed complex signal of a **panadapter**, before any mode, filter or AGC. AetherSDR offers **four IQ channels** at 24, 48, 96 or 192 kHz. DAX IQ is a FlexRadio feature.

Typical uses:

- **Second-opinion SDR software:** run SDR#, GQRX, SDR Console or similar alongside AetherSDR.
- **Skimmers:** CW Skimmer and SDC take IQ through the TCI Server (p. 135), up to four streams at once.

¹<https://github.com/aethersdr/AetherSDR/issues/4554>

²<https://github.com/aethersdr/AetherSDR/issues/5870>

³<https://github.com/aethersdr/AetherSDR/issues/3837>

- **Recording and analysis:** record raw IQ, or feed GNU Radio or your own DSP.
- **Satellite data:** the built-in WFM demodulator (below).

7.2.1 Setup

Choosing which panadapter feeds a channel

IQ channels belong to panadapters, not slices. In the panadapter overlay, open the **DAX** flyout and pick the channel in **IQ Ch** (None, 1–4). The stream itself is switched on in the DAX IQ applet.

Starting a stream

1. Open the **DAX IQ** applet (applet-tray button **IQ**, closed by default).
2. Choose a rate for the channel.
3. Press **On** for that channel.

7.2.2 Using DAX IQ

Getting IQ into other programs

On **Linux**, each running channel appears as a virtual capture device named **AetherSDR DAX IQ n** (PulseAudio/PipeWire source `aethersdr-iq-n`): float32 stereo, I on the left channel and Q on the right, at the rate you chose.

- **SDR#:** pick the audio IQ input, choose **AetherSDR DAX IQ 1**, and set the same sample rate.
- **GQRX:** use a PulseAudio I/Q input on **AetherSDR DAX IQ 1** at the matching rate.
- **GNU Radio:** an Audio Source on `aethersdr-iq-1` at the matching rate gives interleaved I/Q.

On other platforms, take IQ through the TCI Server (p. 135) instead, which is how CW Skimmer and SDC connect.

WFM demodulator (satellite data)

AetherSDR can FM-demodulate a panadapter’s DAX IQ stream on the computer, for G3RUH 9600-baud satellite data. The chain applies phase-continuous Doppler correction, resamples to exactly 48 kHz and uses a flat discriminator, then sends the audio to a virtual audio cable for a program such as HS-SoundModem.

1. In the panadapter overlay **DAX** flyout, set **IQ Ch** first.
2. Press **WFM**, directly below it. It works in any slice mode, on the flyout’s slice.
3. The first time, a **Select WFM Audio Output Device** dialog asks for the output. Choose a virtual cable (for example Hi-Fi Cable, BlackHole or a PipeWire null sink) and optionally remember the choice.

WFM needs a radio with DAX IQ; on other radios it is refused with a notice.

7.2.3 Reference

The DAX IQ applet

The applet has one row per channel, **IQ 1** to **IQ 4**:

Control	What it does
Rate	24k, 48k, 96k or 192k
Off / On	Creates or removes the IQ stream for that channel
Level meter	Signal level in dBFS over a –70 to –10 dBFS window. Normal signals sit low; a full bar means you are near overload.

The rate and on/off state of each channel are saved and restored at startup.

Higher rates use more of the radio’s resources and more network bandwidth.

Limits

- The radio reports how many IQ channels it can supply; four is the maximum.
- Receivers on the same panadapter share that panadapter’s IQ stream, so four independent band views need four panadapters.

7.2.4 Troubleshooting

A program sees no AetherSDR DAX IQ device

The device only exists on Linux, and only while that channel's stream is on.

1. In the panadapter overlay **DAX** flyout, set **IQ Ch** for the panadapter you want.
2. In the **DAX IQ** applet, press **On** for that channel.
3. On macOS or Windows, connect the program through the TCI Server ([p. 135](#)) instead.
4. Set the program's sample rate to the rate chosen in the applet.

WFM is refused with a notice

The connected radio has no DAX IQ.

1. Use a FlexRadio, which supplies DAX IQ.

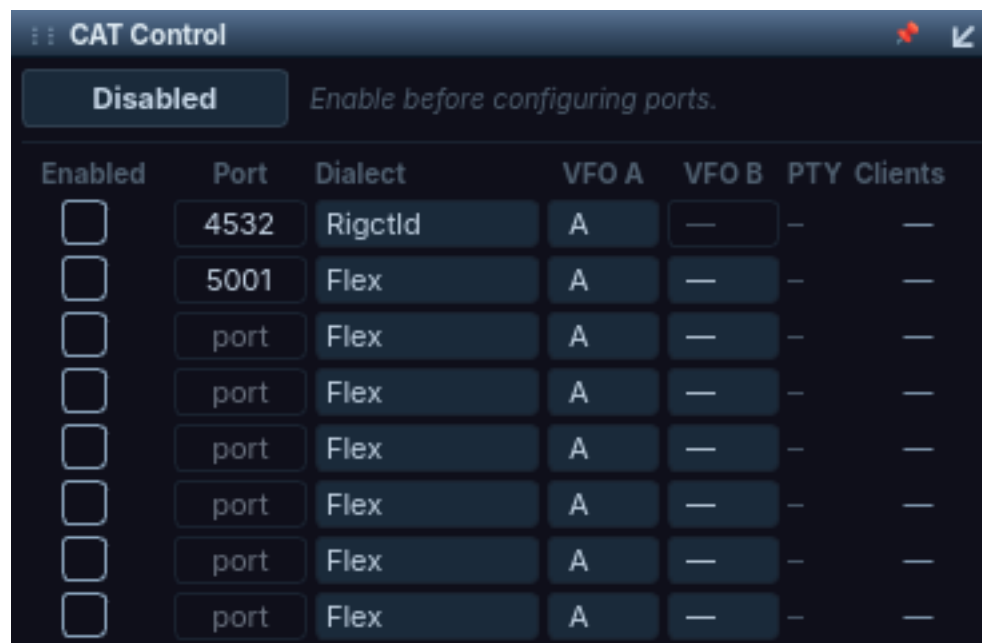
7.2.5 See also

- DAX Virtual Audio ([p. 128](#))
- TCI Server ([p. 135](#))

7.3 CAT Control

AetherSDR has a built-in CAT server, so logging and digital-mode programs can read and set frequency, mode, split and PTT without an external rigctl or a SmartSDR CAT install. It offers up to **eight CAT ports**. Each port speaks one of three dialects: Hamlib **Rigctl** (NET rigctl), Kenwood **TS-2000**, or **Flex** (FlexCAT, the SmartSDR CAT command set). On Linux and macOS every port can also appear as a virtual serial port.

CAT works with WSJT-X, JTDX, fldigi, JS8Call, Winlink/VARA, N1MM+, DXLog and most other Hamlib- or Kenwood-aware programs. For audio, pair it with DAX Virtual Audio ([p. 128](#)), or use the TCI Server ([p. 135](#)), which carries CAT and audio over one connection.



The CAT Control applet popped out of the applet panel, showing the full channel table.

7.3.1 Setup

The CAT applet

CAT has its own **CAT Control** applet (applet-tray button **CAT**, Integration category). Docked, it shows only the master **Enabled** / **Disabled** button. To configure ports, **pop the applet out** with the float button on its title bar. The floating view reads “Enable before configuring ports” and lists one row per port:

Column	Meaning
Enabled	Starts this port. A port runs only when the master switch is on, this box is ticked and a port number is set.
Port	TCP port, 1024–65535.
Dialect	Rigctld , TS-2000 or Flex .
VFO A	The slice this port controls as VFO A.
VFO B	The slice used as VFO B (dual-VFO dialects only; see Choosing a dialect (p. 133)).
PTY	The virtual serial port path (Linux and macOS only). Right-click → Copy PTY Name .
Clients	Number of programs connected to the port, or “—” when it is stopped.

While a port is running, its port number, dialect and VFO selectors are locked. Untick the port to change them.

The VFO selectors list one letter per receiver the connected radio offers, up to **A–H**. If you connect to a smaller radio, a slice choice it cannot show is kept and comes back when that slice exists again.

Turning CAT on

- **Settings** → **Autostart CAT with AetherSDR** is the same switch as the applet’s master **Enabled** button. While it is on, every enabled port starts when you connect to a radio.
- A fresh install has CAT switched off, with two rows pre-filled: port **4532** as **Rigctld** on slice **A**, and port **5001** as **Flex** with VFO A = slice A and VFO B = slice B. Both rows are unticked. The other six rows are empty.

Upgrading from an older version: there is no longer a DIGI applet or a fixed set of four servers on ports 4532–4535. The first time the CAT applet ran, your old rigctld and SmartCAT settings were moved to its first two rows.

7.3.2 Using CAT Control

Choosing a dialect

Dialect	Use it for	VFO B
Rigctld	Hamlib “NET rigctl” clients: WSJT-X, JTDX, fldigi, JS8Call, VARA/Winlink	Not used. Split is created on demand when the program asks for it.
TS-2000	Programs that only offer a Kenwood rig	Yes
Flex	Programs with a SmartSDR CAT / FlexRadio profile (Kenwood commands plus the ZZ extensions)	Yes

Switching a port between a single-VFO and a dual-VFO dialect keeps your VFO B choice; it reappears when you switch back.

The rigctld dialect follows the Hamlib NET rigctl command set: frequency, mode and passband, VFO, split, PTT, levels, functions, CTCSS/FM tones, `dump_state` and the long \ forms. See the Hamlib documentation⁴ for the commands themselves.

Testing a rigctld port

```
echo "f" | nc localhost 4532
# 14074000
```

⁴<https://hamlib.github.io/>

```
echo "F 7074000" | nc localhost 4532
# RPRT 0
```

Virtual serial ports (Linux and macOS)

Each CAT port also creates a pseudo-terminal, so programs that only offer a serial port can connect. The path is shown in the applet's **PTY** column:

Platform	Path
Linux	\$XDG_RUNTIME_DIR/aethersdr/cat-A ... cat-H (falls back to the user cache directory when there is no runtime directory)
macOS	~/Library/Caches/AetherSDR/cat-A ... cat-H
Windows	No virtual serial ports. Use the TCP ports.

The letter is the port's row (first row = **cat-A**), not the slice. The path is a per-user symlink to the current pseudo-terminal and is replaced atomically, so a program can keep the same path across restarts.

Upgrading from an older version: the serial ports used to be `/tmp/AetherSDR-CAT-A ... D`. They moved to the per-user paths above for security (GHSA-qxhr-cwrc-pvrn). Re-point `fldigi` and any other serial-port program at the new path.

7.3.3 Reference

Command behaviour

- **Band changes move the panadapter.** A tune that lands inside the visible span moves only the slice, so Doppler-stepping satellite software does not keep re-centring the pan. A tune outside the span, or to another band, re-centres or re-bands the panadapter. This applies to `WSJT-X` and `fldigi` band changes, `rigctld set_freq`, `FA/FB` and `VFO up/down` (#4876).
- **Locked slices refuse.** A CAT tune on a locked slice returns an error instead of a false success.
- **Split is tracked per client.** Each CAT client sees its own split state, so several `WSJT-X` instances (one per port) can all use **Split Operation: Fake It** (#4853).
- **Safety.** A bare `ZZTX`; reads the transmit state; it does not key. `TS-2000` satellite-mode TX is refused. Repeating `set_split_vfo` or `tx_enable` with the same value does not trip the radio's TX watchdog.
- **S-meter.** `rigctld l STRENGTH` reads the real S-meter of the addressed slice, in dB relative to S9.
- **Tuning step.** `set_ts 0` answers `RPRT -1` on every radio. On radios other than FlexRadio, `set_ts` sets AetherSDR's own tuning step.
- **RF gain on other radios.** On the Hermes-Lite 2 and other non-Flex radios, `L RF / l RF` scale across the radio's published RF-gain range (on the HL2, 0.5 = +18 dB). Until a range is published the command answers `RPRT -11` (#6013).
- **CW over rigctld.** `cwx send` and the clear commands go through the CWX model, so sidetone matches manual keying.

7.3.4 Known issues

- A cross-band tune from a CAT or TCI program moves the frequency but does not recall the radio's band stack, so the per-band antenna, mode and filters are not restored. On a transverter band this can leave the wrong antenna port selected (#3543⁵).
- CAT ports listen on every network interface with no password, so any machine that can reach a port can tune and key the radio (#5296⁶).

⁵<https://github.com/aethersdr/AetherSDR/issues/3543>

⁶<https://github.com/aethersdr/AetherSDR/issues/5296>

- MacLoggerDX on macOS can be slow to connect and can drop its connection (#5699⁷).

7.3.5 Troubleshooting

To capture a log for any CAT problem, turn on the **TCI / CAT / rigctld** category in **Help → Support & Diagnostics....** It logs the CAT servers and the virtual serial ports. See Support and Logging (p. 261).

The Clients column shows “—”

The port is not running.

1. Check that the master **Enabled** switch is on (or **Settings → Autostart CAT with AethersDR** is ticked).
2. Tick the port’s **Enabled** box.
3. Make sure the row has a port number set.

You can’t change a port’s number, dialect or VFO

The port is running, and a running port’s settings are locked.

1. Untick the port’s **Enabled** box.
2. Change the setting.
3. Tick **Enabled** again.

The program tunes the wrong slice

The port’s **VFO A** points at a different slice.

1. Pop out the CAT applet.
2. Untick the port, set its **VFO A** letter to the slice you want, and tick it again.

A serial-port program can’t open /tmp/AethersDR-CAT-A

The virtual serial ports moved to per-user paths.

1. Read the new path from the port’s **PTY** column (right-click → **Copy PTY Name**).
2. Point the program (fldigi, for example) at that path.

7.3.6 See also

- WSJT-X Integration (p. 125)
- DAX Virtual Audio (p. 128)
- TCI Server (p. 135)
- Split Operation (p. 85)
- Support and Logging (p. 261)
- Hamlib documentation⁸

7.4 TCI Server

AethersDR includes a built-in TCI (Transceiver Control Interface)⁹ server. TCI is an open protocol that carries CAT control, RX/TX audio, IQ data, CW keying and spots over a single WebSocket connection, so you don’t have to set up separate CAT ports, virtual audio devices and serial ports.

TCI is also the simplest way to connect WSJT-X on **Windows**, where AethersDR ships no DAX driver, and on radios that have no DAX at all, such as the Hermes-Lite 2.

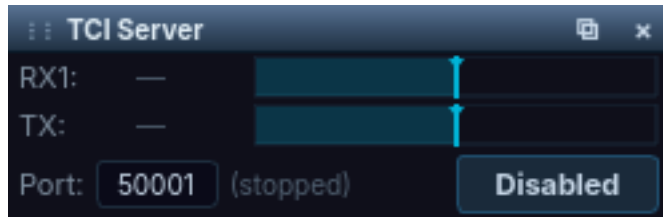
TCI was created by Expert Electronics for ExpertSDR3 and is now widely supported by SDR programs. It uses a WebSocket connection: text frames for commands and binary frames for audio and IQ. Any program on your computer or LAN can connect with no audio routing to configure.

⁷<https://github.com/aethersdr/AethersDR/issues/5699>

⁸<https://hamlib.github.io/>

⁹<https://github.com/ExpertSDR3/TCI>

Task	Traditional	TCI
CAT control	rigctl TCP or virtual serial port	Single WebSocket
RX audio	DAX virtual devices	Same WebSocket
TX audio	DAX TX device	Same WebSocket
IQ data	DAX IQ devices	Same WebSocket
Spots	Separate DX cluster connection	Same WebSocket
CW keying	Serial DTR/RTS or rigctl	Same WebSocket



The TCI Server applet in the applet panel.

7.4.1 Setup

Turning the server on

- **Settings** → **Autostart TCI with AetherSDR** starts the server on every radio connect.
- Or use the **Enabled** button in the **TCI Server** applet (applet-tray button **TCI**).

The TCI Server applet

Control	Meaning
RX1 ... RX8	One row per receive channel: the slice it carries and a combined level meter / gain slider. Rows above the radio's slice capacity are hidden.
TX	Level meter and gain for transmit audio arriving from TCI clients. Right-click it to open TX overflow handling (see TX overflow handling (p. 138)).
Port:	TCP port the server listens on. Default 50001 .
Enabled / Disabled	Starts or stops the server.

Connecting a TCI client

Point your program at:

```
ws://localhost:50001
```

or, from another machine on your LAN, `ws://<computer-ip>:50001`.

Testing with wscat

```
npm install -g wscat
wscat -c ws://localhost:50001
```

You'll see the init burst with the full radio state. Then try:

```
vfo:0,0;           # read the frequency
modulation:0;       # read the mode
vfo:0,0,14074000;   # tune to 14.074 MHz
modulation:0,digu;  # switch to DIGU
drive;              # read the drive level
drive:25;           # set drive
trx:0,true;         # key PTT
```

```

trx:0,false;          # unkey PTT
audio_start:0;        # start RX audio for receiver 0
audio_stop:0;         # stop it

```

7.4.2 Using TCI with other programs

WSJT-X

Use a WSJT-X build with TCI support (WSJT-X Improved, or WSJT-X 3.x).

1. **File** → **Settings** → **Radio**
2. **Rig:** Expert Electronics (TCI)
3. **Network Server:** localhost:50001
4. **PTT Method:** CAT
5. Click **Test CAT**; it turns green.

WSJT-X takes CAT, PTT **and** audio over TCI, so there are no sound-card devices to set up. WSJT-X's **Pwr** slider attenuates the transmitted audio. See WSJT-X Integration (p. 125).

MSHV

Settings → **Interface**, select **TCI**, enter localhost and port 50001, and enable audio streaming if wanted.

Log4OM v2

Settings → **Radio**, choose **TCI** and enter localhost:50001. Log4OM receives spots, and clicking a spot in AetherSDR sends `clicked_on_spot` back to it.

CW Skimmer and SDC

Point SDC or CW Skimmer at AetherSDR's TCI port. IQ flows as DAX IQ at 24, 48, 96 or 192 kHz (SDC negotiates 96 kHz), and `dds`: follows the panadapter centre. Up to **four** IQ subscriptions can run at once, so several skimmers can share one server. Receivers on the same panadapter share its IQ stream, so four independent bands need four panadapters. IQ is not offered as `int16` or over `wss`.

JTDX

JTDX has TCI support but is no longer maintained and can crash with some TCI servers. WSJT-X is recommended instead.

Other TCI clients

Any program that implements the TCI protocol¹⁰ can connect, for example:

- **HamDeck** — Stream Deck automation via TCI
- **TCI-Hamlib Adapter** (DL3NEY) — bridges TCI to Hamlib for programs that only support `rigctl`
- **eesdr-tci** — a Python TCI client library for scripting
- **AH-4-style tuner interfaces** and band-pass switch controllers that watch `tune`: (for example HB9DUT's `TCI_ICOM_Tuner_Interface`)

7.4.3 Using several clients at once

Several TCI programs can be connected at once, and each keys the slice it asked for:

- `trx` keys **the receiver the client named**. A plain `trx:0` resolves against the receiver the client declared in its `audio_start`, so a second WSJT-X instance does not take transmit from the first.
- An externally selected TX slice only wins when `split` was requested, or when VFO B set up a route for that exact receiver, so satellite and cross-band split still work.
- A receiver that cannot be resolved is **declined**, never keyed on slice A instead.

¹⁰<https://github.com/ExpertSDR3/TCI>

- **Every refused PTT is answered with `trx:<n>,false;`**, including a second client trying to key while another holds TCI PTT. WSJT-X shows this as “TCI failed to set ptt”.
- **Receiver numbers are stable.** If a slice closes mid-session, the other slices keep their receiver numbers, and the freed number goes to the next new slice.
- A VFO B request from one client does not retune another client’s slice.

7.4.4 Audio streaming

TCI audio is sent and received as binary frames on the same WebSocket.

- **Audio doesn’t flow until the client asks for it** with `audio_start:<receiver>;`.
- **PC Audio does not need to be on.** On a FlexRadio, TCI receive audio comes from DAX channels that the TCI server claims for itself; your speaker volume and mute don’t affect it. On other radios it comes straight from the slice.
- If a client reconnects more than 10 seconds later, its receive audio is re-bound to the right DAX channels automatically.
- TX audio from a client is resampled from the rate the client declares, so 8, 12 and 24 kHz programs are not mis-pitched.

Sample rate and format

The default is **48 kHz**. A client can request 8000, 12000, 24000 or 48000 Hz:

```
audio_samplerate:24000;
```

Any other rate (44.1 kHz, for example) is refused and the previous rate is echoed back. Each client has its own rate and converter.

Audio is float32 stereo by default. A client can ask for int16 and/or mono:

```
audio_stream_sample_type:int16;
audio_stream_channels:1;
```

TX overflow handling

Right-click the **TX** meter in the TCI Server applet to choose how transmit samples beyond ± 1.0 are treated:

Mode	Behaviour
Clip (saturating ± 1.0)	Default. Hard-clamps overshoots to ± 1.0 .
NaN guard (zero NaN/Inf only)	Passes samples through bit-exact, only zeroing NaN/Inf. Best for digital-mode tone fidelity.
Measure only (true bypass)	Never changes samples; counts overshoots for telemetry.

FT8/WSJT-X users who want exact tones can choose NaN guard or Measure only.

7.4.5 Reference

Server identity

The init burst announces `device:AetherSDR;` and `protocol:ExpertSDR3,1.5;`, followed by the full radio state and `ready;`.

Protocol coverage

AetherSDR implements the TCI 2.0 command set, including:

Category	Commands
VFO / mode	<code>vfo</code> , <code>modulation</code> , <code>rx_filter_band</code> , <code>dds</code> , <code>if</code> , <code>vfo_limits</code> , <code>if_limits</code> , <code>vfo_lock</code>
TX	<code>trx</code> , <code>tx_enable</code> , <code>tune</code> , <code>drive</code> , <code>tune_drive</code> , <code>tx_frequency</code> , <code>tx_gain</code>
RIT / XIT	<code>rit_enable/offset</code> , <code>xit_enable/offset</code> , <code>split_enable</code>

Category	Commands
Audio	volume, mute, rx_volume, rx_mute, rx_balance, mon_enable, mon_volume, mic_level
AGC / SQL	agc_mode, agc_gain, sql_enable, sql_level, lock
DSP	rx_nb_enable/param, rx_nr_enable, rx_anf_enable, rx_apf_enable, rx_bin_enable, rx_anc_enable, rx_dse_enable, rx_nf_enable
CW	cw_macros_speed, cw_keyer_speed, cw_macros_delay, cw_msg, cw_macros, cw_macros_stop, cw_terminal, keyer
Audio streaming	audio_start/stop, audio_samplerate, audio_stream_sample_type, audio_stream_channels, audio_stream_samples
IQ	iq_start/stop, iq_samplerate
Spots	spot, spot_delete, spot_clear, clicked_on_spot
Sensors	rx_sensors_enable, tx_sensors_enable, rx_channel_sensors, tx_sensors
Digital	digl_offset, digu_offset

Notes:

- volume is in dB, as the spec defines it; the older percent form is still accepted. mic_level is a global 0–100 %.
- The server also broadcasts which slice has focus in the GUI, plus DSP, squelch, RIT/XIT and TX power when they change.
- Malformed arguments (for example to volume: or modulation:) are rejected, never applied to slice 0.
- A vfo: confirmation reports the frequency the radio actually reached.
- **Panadapter spectrum forwarding:** subscribed clients receive the panadapter's FFT rows.

Tune notifications

tune:<trx>,true|false; is broadcast to **every** client on every tune edge, whether the tune came from the TX applet, any TCI client or the radio, and it follows the matching trx:. A tune: SET always gets an answer, and the init burst includes the current tune: state. A tune that a TCI client started stops when that client disconnects or the server stops. tune: -1, true keys nothing (#6200).

Sensor telemetry

```
rx_sensors_enable:true;    # S-meter updates
tx_sensors_enable:true;    # power, SWR, mic and ALC while transmitting
```

RX sensors broadcast each receiver's S-meter in dBm:

```
rx_channel_sensors:0,0,-97.3;
```

TX sensors broadcast during transmit, including ALC.

CW keying

```
cw_macros:0,CQ CQ CQ DE N0CALL;  # send text
cw_macros_stop;                  # stop
cw_keyer_speed:25;               # speed
keyer:0,true;                   # key down
keyer:0,false;                  # key up
```

7.4.6 Known issues

- If a client that holds PTT dies without closing its connection (power loss, a pulled cable, a dropped Wi-Fi link), the radio stays keyed until the operating system gives up on the TCP connection (#5985¹¹).

¹¹<https://github.com/aethersdr/AetherSDR/issues/5985>

- Transmit audio from a TCI client passes through the Phone/CW microphone chain, so the mic profile, **MIC** level and **PROC** change what a digital-mode client puts on the air (#5844¹²).
- TCI receive audio can stop after moving to a voice frequency with DAX off and back to a data frequency with DAX on, while CAT keeps working (#4824¹³).
- After a TCI voice keyer (for example OpsLog) transmits, DAX stays on, so the next microphone transmission has no audio until you turn DAX off (#5524¹⁴).
- A cross-band tune from a TCI client does not recall the radio's band stack, so the per-band antenna, mode and filters are not restored (#3543¹⁵).
- The TCI port listens on every network interface with no password, so any machine that can reach it can tune and key the radio (#5296¹⁶).

7.4.7 Troubleshooting

The **TCI / CAT / rigctld** category in **Help → Support & Diagnostics...** logs the TCI server. See Support and Logging (p. 261).

The server won't start because port 50001 is in use

Another program already listens on the TCI port.

1. Change the **Port:** field in the TCI Server applet.
2. Point each TCI client at the new port.

A client connects but gets no audio

Audio only flows after the client sends `audio_start`.

1. Check that the client has audio streaming turned on. WSJT-X sends `audio_start` automatically.
2. Test by hand with `wscat` (p. 136): send `audio_start:0`; and watch for binary frames.

WSJT-X shows “TCI failed to set ptt”

AetherSDR refused the PTT request. This happens when another TCI client already holds PTT, or when the requested receiver can't be resolved to a slice.

1. Stop the other client's transmission, then try again.
2. Check that each WSJT-X instance uses its own receiver.

TCI keys the wrong slice

The PTT routing needs a log to diagnose.

1. Start AetherSDR with `QT_LOGGING_RULES="aether.cat.info=true"`. This logs one line per PTT decision: the requested receiver, the mapped slice, the chosen slice, the slice holding TX and any cached route.
2. Reproduce the problem and include that log when you report it.

JTDX crashes

This is a known JTDX bug; JTDX is no longer maintained.

1. Use WSJT-X instead.

A client on another computer can't connect

A firewall is blocking the TCI port, or the client is using `localhost`.

1. Open the TCI port (TCP) in the firewall of the computer running AetherSDR.
2. Point the client at `ws://<computer-ip>:<port>`.

¹²<https://github.com/aethersdr/AetherSDR/issues/5844>

¹³<https://github.com/aethersdr/AetherSDR/issues/4824>

¹⁴<https://github.com/aethersdr/AetherSDR/issues/5524>

¹⁵<https://github.com/aethersdr/AetherSDR/issues/3543>

¹⁶<https://github.com/aethersdr/AetherSDR/issues/5296>

7.4.8 See also

- WSJT-X Integration (p. 125)
- DAX Virtual Audio (p. 128)
- DAX IQ Streaming (p. 130)
- CAT Control (p. 132)
- SpotHub (p. 148)
- TCI protocol specification¹⁷

7.5 MQTT Station Automation

AetherSDR connects to your station automation system over MQTT, the lightweight publish/subscribe protocol used by Node-RED, Home Assistant and many shack automation tools. You can show values from your station (rotator heading, selected antenna, amplifier state) on the panadapter, send commands with custom buttons, and let other software follow the radio through AetherSDR's own built-in topics.

Configuration lives in **Settings** → **MQTT...**; the **MQTT applet** is the live control surface.

7.5.1 Requirements

An MQTT broker. Any broker works. Popular options:

- **Mosquitto**: lightweight, installs in seconds on most Linux distributions (`sudo apt install mosquitto` or `sudo pacman -S mosquitto`)
- **Home Assistant**: its MQTT broker add-on
- **Node-RED**: can act as a broker via the `node-red-contrib-aedes` package

AetherSDR uses the Eclipse Mosquitto client library (libmosquitto), bundled with the app. No separate client installation is needed.

For a quick local test broker:

```
# Install
sudo pacman -S mosquitto      # Arch
sudo apt install mosquitto    # Debian/Ubuntu

# Start (unauthenticated, local only)
mosquitto -v

# Test subscribe in another terminal
mosquitto_sub -t "test/#" -v

# Test publish
mosquitto_pub -t "test/hello" -m "world"
```

Once that works, point AetherSDR at `localhost:1883`.

7.5.2 Setup

1. Open **Settings** → **MQTT...** (or **Settings...** in the MQTT applet).
2. On the **Broker** tab, enter the **Host** and **Port** (default 1883), and a **User** and **Password** if your broker needs them.
3. On the **Subscriptions** tab, click **Add** for each topic you want. Tick **Display** to show a topic's latest value on the panadapter.
4. Optionally add **Publish Buttons**.
5. Open the MQTT applet from the applet panel and click **On** to connect.

¹⁷<https://github.com/ExpertSDR3/TCI>

The status turns green when connected. If the broker goes away, AetherSDR reconnects automatically, backing off from 5 s up to 60 s. The On/Off state is remembered, so MQTT reconnects by itself at the next launch.

The password is stored in your operating system's keychain (macOS Keychain, Windows Credential Manager, or libsecret/KWallet on Linux), never in the settings database. Builds without keychain support keep it for the current session only.

Upgrading from an older version: MQTT used to be configured inside the applet, with comma-separated topics and a * prefix for panadapter display. Those settings are migrated automatically, once, into the MQTT Settings dialog; a * topic becomes a subscription with **Display** ticked. The password is moved into the OS keychain.

7.5.3 Using the MQTT applet

The applet shows **On/Off**, the connection status, a log of incoming messages, and your publish buttons. Click a button to publish its payload. **Settings...** opens the MQTT Settings dialog. Antenna alias topics are subscribed automatically.

Panadapter display

Topics with **Display** ticked show their latest value on the panadapter, so a beam heading or the selected antenna is visible without switching windows. Several displayed topics appear side by side.

7.5.4 Using the built-in topics

AetherSDR publishes and subscribes to its own topics under `aethersdr/`. Each can be enabled or disabled in the MQTT Settings dialog, except the antenna-alias topics, which are always on. The full list is under Built-in topics (p. 143).

To give the radio's antenna ports friendly display names in AetherSDR, publish to the built-in `aethersdr/antenna/name/+` or `aethersdr/antenna/names` topics.

Transmit topics: `cw/transmit` and `ax25/tx` key your transmitter when a message arrives. Leave them off unless you mean to transmit from automation, and secure your broker. A relay script that forwards `cw/decode` into `cw/transmit` must filter on the topic namespace, or it will loop.

7.5.5 Example: rotator control via Node-RED

- Node-RED reads your rotator controller and publishes `rotator/pos` with the current heading (e.g. 240).
- Node-RED subscribes to `rotator/cmd` and sends commands to the controller.

AetherSDR setting	Value
Subscription	<code>rotator/pos</code> , Display ticked
Button: CW	topic <code>rotator/cmd</code> , payload CW
Button: CCW	topic <code>rotator/cmd</code> , payload CCW
Button: Stop	topic <code>rotator/cmd</code> , payload STOP

The beam heading stays on the panadapter and you can turn the beam without leaving AetherSDR.

7.5.6 Example: antenna switching

For any MQTT-connected antenna switch (Node-RED, ESP32, Raspberry Pi, ...):

AetherSDR setting	Value
Subscription	<code>ant/selected</code> , Display ticked
Button: Hexbeam	topic <code>ant/select</code> , payload 1
Button: Vertical	topic <code>ant/select</code> , payload 2

AetherSDR setting	Value
Button: Wire	topic ant/select, payload 3

7.5.7 Example: SteppIR controller

AetherSDR setting	Value
Subscriptions	steppir/band and steppir/direction, Display ticked
Button: Normal	topic steppir/cmd, payload normal
Button: 180°	topic steppir/cmd, payload reverse
Button: Bi-Dir	topic steppir/cmd, payload bidir

7.5.8 Reference

MQTT Settings dialog

Broker

Field	Notes
Host / Port	Broker address; port defaults to 1883
User / Password	Optional; leave blank for unauthenticated brokers
TLS: Use TLS	Encrypt the connection
CA cert	Optional CA certificate file (Browse...); blank uses the system CA bundle

Subscriptions: a table of **Topic** and **Display** with **Add / Remove**. Ticking **Display** shows that topic's latest value on the panadapter. Below it, **Internal AetherSDR Topics** lists the built-in subscribe topics with an enable for each.

Publish Buttons: a table of **Label**, **Topic** and **Payload** with **Add / Remove**, up to 12 buttons, shown three per row in the applet. Below it, **Internal AetherSDR Topics** lists the built-in publish topics with an enable for each.

Built-in topics

Published by AetherSDR:

Topic	Default	Contents
aethersdr/cw/decode	On	CW decoder output, one JSON message per chunk: text, rx (false for the decoder watching your own transmitted CW), freq, and pitch_hz / speed_wpm when known. Published only while the CW decoder panel is showing decoded text.
aethersdr/radio/state	Off	Radio state as JSON: slice, freq, mode, tx, connected (false on disconnect), and the RF drive, its ceiling max_power_level and drive_confirmed. Fields are present only when known, so key off whether a field exists, not on zero.
aethersdr/ax25/rx	Off	Received AX.25 packet frames

Subscribed by AetherSDR:

Topic	Default	Effect
aethersdr/cw/transmit	Off	Transmits CW through the CWX keyer. Payload: {"text": "de N0CALL", "speed_wpm": 28, "pitch_hz": 600}; speed and pitch are optional. Needs a radio with a radio-side CW keyer.
aethersdr/ax25/tx	Off	Transmits an AX.25 packet
aethersdr/antenna/name/+ and aethersdr/antenna/names	Always on	Antenna display names (per port, or all at once)

Technical notes

- QoS 0 (fire-and-forget) is used for subscribe and publish.
- Reconnect backs off exponentially: 5 s initially, doubling up to 60 s.

7.5.9 Troubleshooting**Nothing is published on `aethersdr/cw/decode`**

The topic is published only while the CW decoder panel is showing decoded text.

1. Check that `aethersdr/cw/decode` is enabled under **Internal AetherSDR Topics** in **Settings** → **MQTT...**
2. Open the CW decoder panel (see CW Decoder (p. 104)) and make sure it is showing decoded text.

`aethersdr/radio/state` or a transmit topic does nothing

These built-in topics are off by default.

1. Open **Settings** → **MQTT...** and enable the topic under **Internal AetherSDR Topics**.
2. For `aethersdr/cw/transmit`, the radio must have a radio-side CW keyer.

The broker password is asked for again after a restart

The build has no keychain support, so the password is kept for the current session only.

1. Enter the password again in **Settings** → **MQTT...**, or use a build with keychain support.

7.5.10 See also

- CW Decoder (p. 104)
- CWX Panel (p. 102)
- AetherModem Packet Radio (p. 118)
- Automation Bridge and MCP (p. 144)
- Green Heron Everywhere (p. 176)

7.6 Automation Bridge and MCP

Status: Off by default · **Applies to:** All radios

The automation bridge is an opt-in, local command channel into a running AetherSDR. A program on the same computer can read the app's state (radio, slices, panadapters, the widget tree), capture any window or the panadapter as a PNG, and, if you allow it, operate controls. The AetherSDR source tree also includes an **MCP server** that exposes the bridge to AI assistants such as Claude Code, Cursor, Copilot, Codex CLI and Gemini CLI. It is mainly a developer and testing tool, but it is also a supported way for your own control-surface or automation software to drive AetherSDR. The full reference, including every verb, is `docs/automation-bridge.md`¹⁸.

7.6.1 Requirements

- A client on the same computer. The bridge listens on a local socket (a Unix domain socket on macOS and Linux, a named pipe on Windows). There is **no TCP port** and no network exposure.
- For the MCP server: Python 3 (no other dependencies).

Each request is one line (a bare command or a JSON object) and each reply is one line of JSON. On startup the app writes the socket path to `aethersdr-automation.json` in the system temp folder so clients can find it.

7.6.2 Setup

Open **Settings** → **Radio Setup...** → **Network** and find the **Advanced** group:

¹⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/automation-bridge.md>

Control	What it does
Agent Automation (MCP): Enabled / Disabled	Starts or stops the bridge. Off by default. Enabling it with no token creates one. The setting is saved, so the bridge comes back at the next launch.
Access Token: with Copy and Rotate	The shared secret clients must present. Rotate creates a new token and applies it immediately, locking out any client still using the old one.
Allow TX via MCP: “Enable transmit control”	Lets bridge clients key the transmitter. Off by default; turning it on asks for a one-time confirmation.
Observe only: “Read-only (block all driving)”	Blocks every action that changes anything; reads still work.

You can also start the bridge for a single run by launching with `AETHER_AUTOMATION=1`. In that case the Enabled/Disabled button is greyed out, because the environment variable wins.

Access token

Every verb except `ping` is refused without the matching token. The token is kept in your operating system’s secret store (macOS Keychain, Windows Credential Manager, or libsecret/KWallet on Linux), never in the settings database.

To give it to an MCP client, set `AETHER_MCP_TOKEN` **only in the environment of the shell that launches your assistant**, using a method that does not record it in shell history. The MCP server inherits it from there.

Keep the token out of shared configs. Do not put the literal token in a shell profile, in `.mcp.json`, or in any other MCP config file: that puts a live credential on disk and risks it landing in a commit or a backup. The token field’s tooltip in Radio Setup suggests pasting it into the MCP config; follow this page and the docs instead.

Headless or CI runs can pass `AETHER_MCP_TOKEN` to AetherSDR directly; it overrides the keychain.

The token decides which client you have deliberately let in. It is not a wall against other programs running as your own user once your keychain is unlocked.

MCP server for AI assistants

`tools/aether_mcp.py` in the AetherSDR repository (plain Python 3, no dependencies) wraps the bridge as a Model Context Protocol server.

- **Claude Code** picks it up from the repository’s `.mcp.json`; approve it on first use.
- **Other assistants:** add it to your MCP config:

```
{"mcpServers": {"aethersdr-automation": {
  "command": "python3", "args": ["tools/aether_mcp.py"]}}}
```

On Windows use `python` (or `py -3`).

It offers typed tools (`bridge_status`, `dump_tree`, `grab_widget`, `get_state`, `get_log`, `invoke`, `tune`, `slice`, `pan`, `record`, `menu`, `connect`, `capture_audio`, `wait_for`, `assert_state` and more), plus a raw `bridge_command` for everything else, a `validate_ui_change` prompt, and read-only state resources. The transmit verbs are deliberately reachable only through `bridge_command`.

7.6.3 Using transmit safety

- The bridge refuses to operate any control that keys the transmitter (MOX/PTT, TUNE, ATU, CWX send, packet/APRS send) through its generic “click this control” path, whoever is asking.
- The dedicated transmit verbs (`key`, `cwx`, `txtest`, `atu`, `transmit`, ...) work only when you have allowed TX: tick **Allow TX via MCP**, or launch with `AETHER_AUTOMATION_ALLOW_TX=1`.
- A force-unkey watchdog drops a bridge-started transmission that runs too long (`AETHER_AUTOMATION_TX_MAX_MS`, default 20 s). It does not affect your own, DAX or TCI transmissions.
- `AETHER_AUTOMATION_NO_TX=1` pins TX off for that process even if you allowed it before.

You remain responsible for every emission the bridge makes on your behalf. See [Before You Transmit](#) (p. 14).

7.6.4 Using observe-only mode

Tick **Observe only** to hand a client visibility without control. The bridge then answers only pure reads (ping, get, the widget tree, screenshots, logs and similar) and refuses everything else. It is enforced inside AetherSDR, so a client cannot switch it off, and it takes effect immediately on a running bridge. ping and whoami report it as readOnly. Launching with AETHER_AUTOMATION_READONLY=1 pins it on.

7.6.5 Checking which build you are talking to

ping needs no token and returns the app version plus the **build identity**: the git describe string, commit SHA, the release tag it is based on, commits since that tag, and whether the build had local changes. It is the same identity **Help** → **About** shows, so a test build can be told apart from a release with the same version number.

7.6.6 Reference

Environment variables

Variable	Effect
AETHER_AUTOMATION=1	Starts the bridge for this run; greys out the Enabled/Disabled button
AETHER_MCP_TOKEN	The access token for an MCP client; passed to AetherSDR itself, it overrides the key-chain
AETHER_AUTOMATION_ALLOW_TX=1	Allows the dedicated transmit verbs
AETHER_AUTOMATION_NO_TX=1	Pins TX off for the process
AETHER_AUTOMATION_TX_MAX_MS	Force-unkey watchdog limit for bridge-started transmissions (default 20 s)
AETHER_AUTOMATION_READONLY=1	Pins observe-only mode on
AETHER_AUTOMATION_SOCKET=<name>	Overrides the socket name at launch

7.6.7 Known issues

- The VOX button refuses every bridge action, including switching VOX off, while the vox_toggle shortcut can still switch it on (#6053¹⁹).
- key ptt off and key mox off report success without checking that the transmitter actually unkeyed (#5252²⁰).

7.6.8 Troubleshooting

Every verb except ping is refused

The client is not presenting the matching access token.

1. Copy the token with **Copy** in **Settings** → **Radio Setup...** → **Network** → **Advanced**.
2. Set AETHER_MCP_TOKEN in the environment of the shell that launches your assistant, then restart the assistant.
3. If the token was rotated, repeat with the new one.

Transmit verbs are refused

TX is not allowed for bridge clients.

1. Tick **Allow TX via MCP** and confirm, or launch with AETHER_AUTOMATION_ALLOW_TX=1.
2. Check that the process was not launched with AETHER_AUTOMATION_NO_TX=1, which pins TX off.

Only reads work

Observe-only mode is on.

1. Untick **Observe only** in **Settings** → **Radio Setup...** → **Network** → **Advanced**.
2. If it stays on, the process was launched with AETHER_AUTOMATION_READONLY=1; relaunch without it.

¹⁹<https://github.com/aethersdr/AetherSDR/issues/6053>

²⁰<https://github.com/aethersdr/AetherSDR/issues/5252>

The Enabled/Disabled button is greyed out

AetherSDR was launched with `AETHER_AUTOMATION=1`, and the environment variable wins.

1. Relaunch without `AETHER_AUTOMATION` to control the bridge from Radio Setup.

The client reaches a different AetherSDR instance

Each launch rewrites `aethersdr-automation.json` in the system temp folder, so with two instances running the discovery file points at the one started last.

1. Launch each instance with its own `AETHER_AUTOMATION_SOCKET=<name>`.
2. Point the client at that socket explicitly (for example `tools/automation_probe.py --socket <path>`).

7.6.9 See also

- AI-Assisted Development ([p. 275](#)): how contributors use the bridge to prove a change
- TCI Server ([p. 135](#)): the network protocol for logging, digital-mode and control-surface software
- USB Control Surfaces ([p. 189](#))
- StreamDeck ([p. 181](#))
- Before You Transmit ([p. 14](#))
- `docs/automation-bridge.md`²¹

²¹<https://github.com/aethersdr/AetherSDR/blob/main/docs/automation-bridge.md>

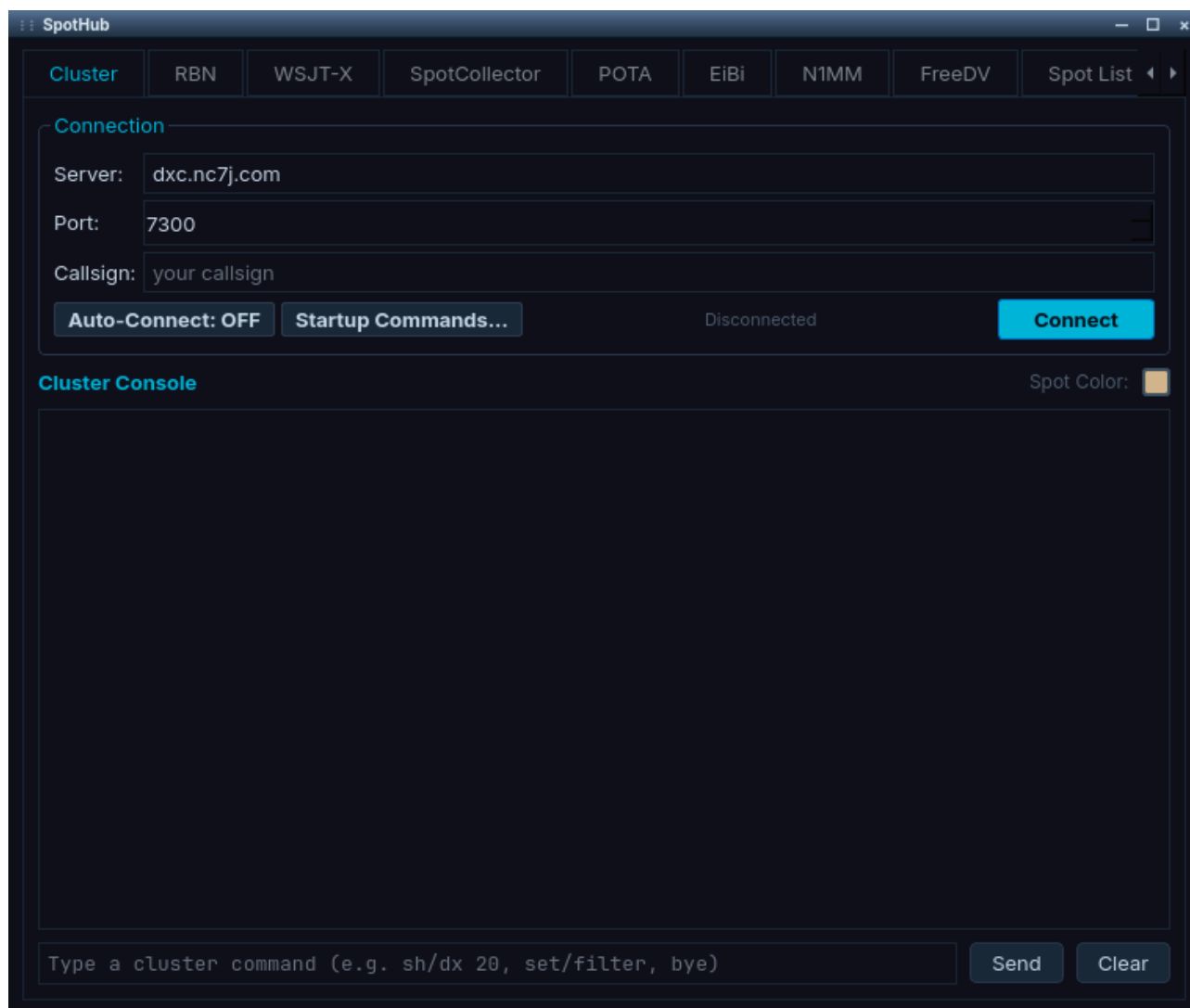
Spotting & Tools

8.1 SpotHub

Status: Supported · **Applies to:** FlexRadio, Hermes-Lite 2 (experimental), Networked Icom (early; IC-7300MK2 supported)

SpotHub is AetherSDR's spot manager. It gathers spots from DX clusters, the Reverse Beacon Network, WSJT-X, DXLab SpotCollector, POTA, N1MM+/DXLog, the EiBi shortwave schedule and FreeDV Reporter, and draws them on the panadapter as colour-coded callsign labels. It also has a sortable spot list with band filters, and a Display tab that controls how spots look. The **Kiwi DX** overlay and the panadapter-driven **Signal History** markers are switched on from the same Display tab.

All spot sources run on a worker thread, and spots are forwarded to the radio in batches once a second.



SpotHub, open on the Cluster tab. Each tab is one spot source.

8.1.1 Setup

Open SpotHub from **Settings** → **SpotHub...**

SpotHub has ten tabs: **Cluster**, **RBN**, **WSJT-X**, **SpotCollector**, **POTA**, **EiBi**, **N1MM**, **FreeDV**, **Spot List** and **Display**. The FreeDV tab is only present in builds with Qt WebSockets.

Every source tab has an **Auto-Start** (or **Auto-Connect**) toggle that starts it when the radio connects, a **Start / Stop** (or **Connect**) button, a status line and a console. Each console has a **Clear** button, which clears the console and deletes its stored log.

8.1.2 Using the spot sources

Cluster (DX Cluster)

Connects to a DX cluster node over telnet (DX Spider, AR-Cluster, CC Cluster).

- **Server / Port / Callsign** — default `dxc.nc7j.com:7300`
- **Auto-Connect** — connect when the radio connects
- **Startup Commands...** — cluster commands sent automatically after every login, one per line (for example `SET/NAME`, `SET/QTH`, `ACCEPT/SPOT`)

- **Cluster Console** — live output with a command line (sh/dx 20, set/filter, bye ...)
- **Spot Color** — default tan

RBN (Reverse Beacon Network)

Skimmer spots from the Reverse Beacon Network, over the same telnet protocol.

- **Server / Port** — default `telnet.reversebeacon.net:7000`
- **Callsign** — falls back to the cluster callsign when empty
- **Rate Limit** — maximum spots per second sent to the radio (default 10), so contests don't flood the display. Extra spots are queued for the next batch.
- **Startup Commands...**, **Auto-Connect**, **RBN Console** and **Spot Color** (default blue) as on the Cluster tab

WSJT-X (decode spotter)

Listens for WSJT-X decode messages over UDP and spots the stations you are hearing.

- **Address / Port** — default `224.0.0.1:2237` (multicast). Use `0.0.0.0` for unicast.
- **Spot Filter** — which decodes are spotted, each with its own colour:
 - **CQ** (default green)
 - **CQ POTA** (default cyan)
 - **Calling Me** (default red). Your callsign comes from the cluster login, or the station callsign if no cluster login is set.
 - **Default** colour (white) for anything else
- Label opacity follows signal strength: weak decodes are faint, strong ones solid.
- **Spot Life** — 30–300 s (default 120 s), separate from the cluster lifetime
- **WSJT-X Decodes** console — callsign, frequency, SNR and the decoded message

Each spot is placed on the band of the WSJT-X instance that reported it, so several instances on different bands don't mix. The decoded text (for example `CQ N0CALL DN18`) appears in the Spot List's Comment column.

Setup: in WSJT-X, **File** → **Settings** → **Reporting**, make sure the UDP server is on (port 2237). In AetherSDR, press **Start** on this tab (or turn on Auto-Start). See also WSJT-X Integration (p. 125).

SpotCollector

Receives DX spots pushed by DXLab **SpotCollector** over UDP.

- **UDP Port** — default 9999. In SpotCollector, enable UDP broadcast to this port.
- Alternatively, connect the Cluster tab to SpotCollector's telnet interface.

POTA (Parks on the Air)

Polls the POTA API (`api.pota.app`) for active activations. No account is needed.

- **Poll Interval** — 15–300 s, default **60 s**
- **POTA Activations** console — callsign, frequency, park reference, park name and mode
- **Spot Color** — default yellow
- Spots last as long as the activation's own expiry time. Only new spots are forwarded.

The Comment column shows the park reference and name, for example `US-4567 Yellowstone National Park SSB`.

EiBi (shortwave broadcast schedule)

Shows broadcast stations from the EiBi¹ shortwave schedule.

- The schedule is downloaded from `www.eibispace.de/dx/eibi.txt` and cached for 7 days. The tab shows **Cache File:** age and **Next Auto-Fetch:**.
- **Update Now** forces a fresh download.

¹<http://www.eibispace.de/>

- Only stations on the air **now** (by schedule day and UTC time) are shown, re-checked every minute.
- Hover a marker for the country, language or mode, target area and transmitter site.
- **Spot Color** — default #8aa8c0. Changing it repaints markers already on screen.

N1MM (contest bandmap)

Receives bandmap spots broadcast by **N1MM Logger+** or **DXLog**.

- **UDP Port** — default **12060**, the same as SmartSDR CAT's N1MM port, so existing broadcast setups work unchanged.
- **Spot Lifetime** — 0–1440 minutes, default 180. **0** = “**Never expire**”. This is a safety net in case the logger exits without deleting its spots.
- **Setup in N1MM+**: Config → Configure Ports... → **Broadcast Data**, tick **Spots**, and add this computer's address and port as a destination. “Contacts” (logged QSOs) is ignored.
- N1MM sends explicit add, update and delete messages, so spots follow the logger exactly.
- **Contest Status Colors** — eight swatches: Busted call, Dupe, Needed multiplier, CQ frequency, Busy (marked), QTC, New QSO (not mult) and No status. They follow your theme until you override them.

The Source column shows **N1MM**.

FreeDV (FreeDV Reporter)

Connects to the FreeDV QSO Reporter (qso.freedv.org) and spots active FreeDV stations in real time. When it connects, the server replays every active station, so spots appear at once. Spots show the mode and grid square in the Comment column.

- **Spot Color** — default dark orange #FF8C00; lifetime 120 s
- **Station Reporting** group:
 - **Enable FreeDV Reporter reporting when RADE is active** — reports your own station while you run RADE. Reporting needs a callsign and a grid square, because FreeDV Reporter is a public map.
 - **Callsign** (with **Use radio**), **Grid Square** (with **Use GPS**)
 - **Station Msg:** — the same message as the **Message** field in **Tools** → **FreeDV Reporter...**; the two stay in sync.

See also RADE Digital Voice ([p. 113](#)).

8.1.3 Using the Spot List

A sortable table of every spot from every source.

Column	Contents
Time	UTC time of the spot
Freq (kHz)	Spotted frequency
DX Call	Spotted station
Mode	Mode, when known
Comment	Mode, park reference, decoded message, ...
Spotter	Who spotted it
Band	Band
Source	Cluster, RBN, WSJT-X, SpotCollector, POTA, N1MM, FreeDV, ...

- **Bands:** checkboxes filter the list by band; they are saved.
- **Sorting:** click any column header. Click **Time** to go back to time order.
- **Column visibility:** right-click the header to show or hide Mode, Comment, Spotter, Band and Source. Time, Freq and DX Call are always shown. Your choice is saved.
- **Auto-scroll:** newest spots are at the **top** when sorted by Time (newest first). The list follows whichever end the newest spot is on; sorting by any other column stops auto-scroll so the view stays put.

- **Freeze** stops the list re-ordering so you can click a spot. Spots keep arriving in the background and appear as soon as you unfreeze.
- **Double-click** a row to tune the active slice.
- **Clear** empties the list and deletes the stored cluster/RBN logs. It does not send anything to the radio.

8.1.4 Using the Display tab

Controls how spots look on the panadapter. A row of toggle buttons sits at the top:

[Spots] [Passive] [Memories] [Kiwi DX] [Auto] [Signals] [QRM] [Clear All]

Toggle	What it does
Spots	Master switch for the spot overlay
Passive	Show the radio's spots without sending <code>spot add</code> commands back to it
Memories	Show the radio's memory channels as spot-like markers
Kiwi DX	Show the KiwiSDR DX Community database (off by default; see below)
Auto	Switch the slice mode when you click a spot that carries mode information
Signals	Gold markers for detected voice-width signals (Signal History, from the panadapter itself)
QRM	Red markers for persistent carriers and wideband interference (Signal History)
Clear All	Clears DX spots, the memories feed and Signal History / QRM markers in one click

General display settings

- **Levels** — how many rows of labels stack (1–10)
- **Position** — vertical starting position, % from the top of the spectrum
- **Font Size** — label size (8–32 px)
- **Spot Lifetime** — 10 seconds to 1 day (applies to cluster and RBN spots)
- **Override Colors** — draw every spot in one colour
- **Override Background** — label background colour, with an **Auto** contrast option
- **Background Opacity** — 0–100 %
- **Spot Lines** — vertical lines from each label down to the spectrum
- **Total Spots** — how many spots are on the panadapter

Kiwi DX overlay

The KiwiSDR DX Community database ships with AetherSDR, so no download is needed. Cyan diamond markers for utility, beacon, broadcast and DX stations sit on the **band-plan strip**, so the band plan must be visible. Hover a marker for its frequency, station, mode, passband and notes. Clicking one tunes the slice and sets the RX filter width; with **Auto** on it also sets the mode.

DXCC Coloring

Imports an ADIF log and colours spot labels by worked status: **New DXCC**, **New Band**, **New Mode** or **Worked**. The log file is watched, so colours update whenever your logger writes a QSO.

- **DXCC Colors** — Enabled / Disabled
- **Log File (ADIF)** — **Browse...**
- **Imported** — live QSO and entity counts
- **Colors** — four swatches

Signal History

Shapes the **Signals** and **QRM** markers (the toggles on the top row decide whether they are shown):

- **Marker Lifetime** — 15–300 s, default 60. How long an inactive marker stays.
- **QRM Gate** — 3–30 s, default 6. How long a narrow carrier or wideband signal must persist before it counts as QRM.

- **Colors** — Signals (default amber #FFC800) and QRM (default red)
- **Snap to Step** — clicking a marker tunes to the nearest multiple of the slice's step
- **Filter Opacity, Filter Delay, Filter Match Window** — settings for Smart Spot Filtering (below)

Smart Spot Filtering

View → **Smart Spot Filtering** (off by default) dims SSB and voice spots that have no detected signal nearby, so you can see at a glance which spotted stations are actually on the air. Spots on active frequencies stay bright. CW and digital spots, and memory bookmarks, are not affected. It needs Signal History to be running.

- **Filter Opacity** — 0–100, default 80. 100 hides unconfirmed spots completely; 80 leaves them about 20 % visible.
- **Filter Delay** — 0–120 s, default 30. How long to wait after turning Smart Spot Filtering on before dimming anything, so Signal History has time to fill.
- **Filter Match Window** — how close a signal must be to confirm a spot, 100–5000 Hz, default 1000 Hz. Double-click to reset.

8.1.5 Using spots on the panadapter

Spot density badges

When more spots overlap than **Levels** allows, they collapse into an amber +N badge. Click the badge for a list of the hidden callsigns; click one to tune.

Right-click a spot

Action	Description
Tune to <callsign>	Tunes the active slice to the spot
Copy Callsign	Copies the callsign to the clipboard
Lookup on QRZ	Opens the station's qrz.com page in your browser (see also Callsign Lookup (p. 163))
Remove Spot	Removes the spot

Adding a spot by hand

1. Right-click an empty part of the panadapter and choose **Add Spot at X.XXX MHz...**
2. Fill in **Frequency (MHz)** (pre-filled and snapped to the tuning step), **Callsign** (required), **Comment** (optional) and **Lifetime** (5 minutes to 2 hours).
3. Tick **Forward to DX Cluster** to send it to the connected cluster as a standard DX command. AetherSDR confirms when it forwards.

A spot you post is not shown twice when the cluster echoes it back.

Other radio families

On the Hermes-Lite 2 and networked Icom radios, which have no spot store of their own, AetherSDR draws cluster, RBN, WSJT-X, POTA and manual spots itself. Right-click → **Remove Spot** removes these too.

8.1.6 Reference

Deduplication

Spots drawn on the panadapter are deduplicated by callsign and frequency across all sources. The same callsign within 1 kHz is dropped while the first spot is still live; if the station moves, the old entry is replaced. N1MM spots are matched by callsign and band, because N1MM sends its own updates and deletes.

Where settings and logs are kept

SpotHub settings are stored with the rest of AetherSDR's settings in AetherSDR.db (see Settings and Backups (p. 220)). Source logs (dxcluster.log, rbn.log, wsjtx.log, pota.log, freedv.log, spotcollector.log) are in the spothub folder of AetherSDR's configuration directory (~/.config/AetherSDR/spothub/ on Linux).

Some useful setting keys, for the **Settings Browser** (**Settings** → **Settings Browser...**):

Key	Default	Description
DxClusterHost / DxClusterPort	dxc.nc7j.com / 7300	Cluster server
RbnHost / RbnPort	telnet.reversebeacon.net / 7000	RBN server
RbnRateLimit	10	Max RBN spots per second to the radio
WsjtxAddress / WsjtxPort	224.0.0.1 / 2237	WSJT-X listener
WsjtxSpotLifetime	120	WSJT-X spot lifetime (s)
SpotCollectorPort	9999	SpotCollector UDP port
PotaPollInterval	60	POTA poll interval (s)
N1MMSpotPort	12060	N1MM UDP port
N1MMSpotLifetimeSec	10800	N1MM spot lifetime (s); 0 = never expire
N1MMSpotAutoStart	False	Start N1MM listener on connect
EiBiAutoStart	False	Start EiBi feed on connect
EiBiSpotColor	#8aa8c0	EiBi marker colour
ShowKiwiDxSpots	False	Kiwi DX overlay
SmartSpotFilterOpacity	80	Smart Spot Filtering opacity
SmartSpotFilterDelayS	30	Smart Spot Filtering delay (s)
SmartSpotFilterMatchHz	1000	Smart Spot Filtering match window (Hz)
ManualSpotLifetime	1800	Default lifetime of hand-added spots (s)
SpotForwardToCluster	False	Forward hand-added spots to the cluster

8.1.7 Known issues

- On ANAN-G2 and RTL-SDR, cluster, RBN, WSJT-X, POTA and manual spots never appear on the panadapter (#6039²).
- The **SpotCollector** tab waits for UDP spots that DXLab SpotCollector does not send, so it receives nothing. Connect the **Cluster** tab to SpotCollector's telnet interface instead (#5131³).
- Repeated spots of the same station on the same frequency can stack into a block of overlapping labels (#5231⁴).

8.1.8 Troubleshooting

No spots appear on the panadapter

The spot overlay is off, or no source is running.

- On the **Display** tab, turn on **Spots**.
- On a source tab, press **Start** (or **Connect**) and watch its status line and console.
- If **Smart Spot Filtering** is on, voice spots with no signal nearby are dimmed. Turn it off under **View** → **Smart Spot Filtering** to check.

Kiwi DX markers don't appear

The markers sit on the band-plan strip.

- On the **Display** tab, turn on **Kiwi DX**.
- Make the band plan visible: **View** → **Band Plan** → **Small**, **Medium**, **Large** or **Huge** (anything but **Off**).

WSJT-X decodes are not spotted

WSJT-X is not sending UDP decode messages, or the WSJT-X tab is not listening.

- In WSJT-X, **File** → **Settings** → **Reporting**, turn on the UDP server (port 2237).

²<https://github.com/aethersdr/AetherSDR/issues/6039>

³<https://github.com/aethersdr/AetherSDR/issues/5131>

⁴<https://github.com/aethersdr/AetherSDR/issues/5231>

2. On the SpotHub **WSJT-X** tab, check the **Address / Port** match, then press **Start**.

The Spot List stops changing

Freeze is on. Spots keep arriving in the background.

1. Turn **Freeze** off; the held spots appear at once.

The Spot List no longer scrolls to new spots

The list is sorted by a column other than **Time**, which stops auto-scroll.

1. Click the **Time** header to go back to time order.

8.1.9 See also

- Panadapter Controls (p. 54)
- WSJT-X Integration (p. 125)
- PSK Reporter Map (p. 155)
- Callsign Lookup (p. 163)
- TCI Server (p. 135)
- RADE Digital Voice (p. 113)

8.2 PSK Reporter Map

The PSK Reporter map shows who is hearing your signal, and who you are hearing, using live reports from PSK Reporter⁵. Reports are drawn on an OpenStreetMap world map or a 3D globe, coloured by mode, with optional great-circle paths, a day/night terminator, NASA night lights and weather radar. The same window also has a one-shot **WSPR beacon**.

8.2.1 Setup

Open the map from **Tools** → **PSK Reporter...** Type your callsign in **Call:** to see who is hearing you and who you are hearing.

8.2.2 Using the map

Choosing reports

The **Reports** group at the top of the left sidebar chooses what is shown:

Control	What it does
Call:	A callsign whose sent and received reports you want to see (normally your own)
All Callsigns	Show every PSK Reporter report instead of one callsign
Band:	All, or one band
Mode:	All, FT8, FT4, WSPR, JS8, CW, PSK, RTTY, SSB or Other
Lookback:	15 min, 30 min, 1 hour, 2 hours, 4 hours or 8 hours
Active monitors	Also show stations that are monitoring and reporting right now
Globe	Switch between the flat map and an interactive 3D globe
Paths	Draw great-circle paths between the stations in each report
Day/night	Shade the part of the world that is in darkness

Reports arrive from PSK Reporter's live feed, backed up by periodic polling, and are cached so the map is not empty when you reopen it. Markers are coloured by mode; the legend is in the status bar at the bottom. Hover or click a marker for a card with the station's details.

When a band-conditions forecast is available, a **Band conditions** group appears below Reports.

⁵<https://pskreporter.info/>

Moving around the map

- **Drag** to pan. The flat map wraps continuously across the date line, so you can keep panning east or west; markers and paths repeat in every copy of the world.
- **Mouse wheel or trackpad scroll** zooms; equal steps in and out return to the same scale.
- **Pinch** to zoom on a trackpad.

The **Map** group has **Dark map** (a dark-tinted basemap) and **Map brightness**.

City lights

City lights overlays NASA's VIIRS night-lights image (a historical 2016 composite, not live data) on the map or globe. With **Day/night** on, the lights fade in between sunset and the end of civil twilight; with **Day/night** off they are shown worldwide.

Control	Default	Effect
Brightness	70 %	Overall intensity
Faint lights	50 %	Lifts dim settlements while keeping the background black. 0 restores the original image.
Warmth	0	0 keeps NASA's grayscale; higher values tint the lights gold

Some light sources are not cities. A status line at the bottom of the map reports loading and retries.

Weather

Tick **Weather overlay** to show precipitation on the map or globe. It is off by default and needs no account or API key. The **Weather precipitation** group chooses the sources:

Source	Coverage
Global primary · LibreWXR	Worldwide precipitation. This is a mix of ground radar, satellite estimates and model data, not worldwide radar.
US backup · NOAA	NOAA/NWS radar, United States
Canada backup · ECCC	Environment and Climate Change Canada radar
Europe backup · OPERA	EUMETNET OPERA radar, Europe

LibreWXR is the primary source. Each regional backup has its own checkbox and is used when LibreWXR is turned off or unavailable; turn LibreWXR off to see the regional radar feeds directly.

- **Radar coverage** shades the radar sites and their nominal range.
- **Intensity legend** shows the colour scale, with **Position at top** to move it.
- **Sources & licenses** explains each source and its terms.

Playback

Press the play button to loop through past observations.

- **History:** 1 h, 2 h or 4 h, limited to what the provider still holds (LibreWXR currently offers about two hours).
- **Speed:** 0.25× to 5×.
- Each frame shows its observation time in local time. In live mode the age shows as **Age unknown**, because live images carry no verified scan time.
- Playback shows original images only; nothing is interpolated between frames.

Radar can be delayed or missing. **This overlay is not a substitute for official weather warnings.**

If a weather or NASA server fails, AetherSDR backs off (60 seconds, then longer) and honours the server's **Retry-After**, so a busy map does not hammer a public service.

8.2.3 Using the WSPR beacon

The **WSPR beacon** group sends a single WSPR transmission.

Field	Meaning
TX call:	Callsign to encode
Grid:	Four-character Maidenhead locator, for example CN85
Band:	160 m to 6 m; default 20 m. Tunes the TX slice to the standard WSPR dial frequency in DIGU with a 1200–1800 Hz filter.
Reported:	Power to report, 0–60 dBm, default 30. This only goes into the message; it does not change your RF power.
Offset:	Audio tone, 1400–1600 Hz, default 1500
Level:	Generated audio level, –60 to –3 dBFS

Transmit once arms one transmission for the next even UTC minute; it does not repeat. The button becomes **Cancel** while armed. The status line shows progress, then **Complete**.

The beacon refuses to start when RADE is active, the radio cannot transmit, the transmitter is already in use, there is no TX slice or the TX slice is locked. It unkeys if the audio stream is lost, you close the window or you press Cancel.

8.2.4 Reference

Saved settings

All map choices (callsign, filters, layers, weather sources, playback history and speed, light settings, beacon fields) are saved with the PSK Reporter preferences and restored next time.

8.2.5 Troubleshooting

The WSPR beacon won't start

The beacon refuses to start when RADE is active, the radio cannot transmit, the transmitter is already in use, there is no TX slice or the TX slice is locked.

1. Turn RADE off if it is active.
2. Make sure a slice holds TX and is not locked.
3. Wait until nothing else is transmitting, then press **Transmit once** again.

The weather overlay is missing or out of date

Radar can be delayed or missing, and a failing server is retried with a growing delay.

1. Check the status line at the bottom of the map for loading and retry messages.
2. To use a regional radar feed directly, tick its backup source and turn **Global primary · LibreWXR** off.

8.2.6 See also

- SpotHub ([p. 148](#))
- WSJT-X Integration ([p. 125](#))
- AetherClock and GPS ([p. 161](#))
- PSK Reporter⁶
- Developer notes: weather radar⁷, regional radar and LibreWXR⁸, city lights⁹, provider retries¹⁰

⁶<https://pskreporter.info/>

⁷<https://github.com/aethersdr/AetherSDR/blob/main/docs/psk-reporter-weather-radar.md>

⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/weather-radar-regions.md>

⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/psk-reporter-city-lights.md>

¹⁰<https://github.com/aethersdr/AetherSDR/blob/main/docs/map-provider-retries.md>

8.3 AetherSweep

AetherSweep is AetherSDR's in-panadapter SWR sweep. It steps a 1 W tune carrier across the current TX band, reads SWR at each step, and draws the curve directly on the panadapter, with the resonance point and the usable bandwidth marked.

8.3.1 Requirements

- A connected radio, with split off.
- The slice on the band you want to sweep must be the **TX slice**, unlocked, with its panadapter visible.
- A supported amateur band other than **60 m**, which is channelized.
- Any amplifier in standby. AetherSweep refuses to start while a **Power Genius XL** is in OPERATE.

Other amplifiers are not checked. AetherSweep only knows about the PGXL. If you have an ACOM, SPE, VK3AMP, KPA1500 or any other amplifier in line, put it in standby or bypass yourself before sweeping. See Amplifiers (p. 170).

8.3.2 Using AetherSweep

Running a sweep

1. Make the slice on the band you want to sweep the **TX slice**, with its panadapter visible.
2. Open the panadapter overlay's **ANT** panel.
3. Optionally tick **Limit range** and set **From / To** (see What gets swept (p. 158)).
4. Click **Start Sweep**. The panadapter widens to show the whole band and the curve is drawn as the sweep runs. Press **Esc** to stop early.
5. When it finishes, the slice frequency, tune power and panadapter range are put back as they were.
6. **Save CSV** exports the result; **Clear Sweep** removes the overlay.

The sweep can also be started from **Tools → Start SWR Scan...**, which sweeps the whole band on the TX slice.

Licence confirmation

Before a sweep transmits, a modal “Antenna SWR Sweep — License Confirmation” explains that it sends a 1 W tune carrier at several frequencies across the band, and that you are responsible for checking the band is clear and for complying with your licence. Tick **Remember my answer** to skip it next time from the ANT panel. **Tools → Start SWR Scan...** always shows it.

What gets swept

- The current band, narrowed to the active regional band plan, minus a 5 kHz guard at each edge.
- Fixed **20 kHz** steps, up to 260 points.
- **Limit range** (off by default) confines the sweep to **From / To**. The fields start at the band edges, and the range can only narrow the sweep, never extend it past the band plan.

Reading the overlay

Visual	Meaning
Background shading	Green for SWR 1.0–1.5, amber for 1.5–2.0, red above 2.0, with grid lines at 1.5, 2, 3, 5 and 10
Amber curve and dots	Measured SWR at each step
Notches at each end	Sweep start and end
Dashed vertical line	The frequency being measured right now
Green caret and circle	Resonance — the lowest SWR measured
Bracket spans along the bottom	Bandwidth where SWR ≤ 1.5 (green) and ≤ 2.0 (amber), drawn once the sweep is complete

The corner readout shows:

- SWR 1.23:1 — the best SWR (with RUN while sweeping)

- Res 14.185 MHz — the resonant frequency, followed by the meter source (TGXL BYPASS or RADIO) when a TGXL is present
- BW 1.5 180 kHz 2.0 320 kHz — the bandwidth at $\text{SWR} \leq 1.5$ and ≤ 2.0 , after the sweep completes

Saving the result

Save CSV in the ANT panel exports the most recent completed sweep with the header `Frequency (Hz), SWR`. The suggested file is `swr_sweep_<band>_<YYYYMMDD>_<HHMMSS>.csv` in your Documents folder. It is refused while a sweep is running, and when there is no sweep to save.

Sweeping with a TGXL

If a 4O3A Tuner Genius XL is present and in OPERATE, AetherSweep puts it into **BYPASS** automatically before applying RF, waits for the relays to settle, and reads SWR from the radio — so you measure the antenna and feedline, not the tuner's match. The tuner is returned to its previous state after the sweep. If the tuner does not confirm bypass, the sweep stops before transmitting.

8.3.3 Reference

Limitations

- The sweep power is fixed at 1 W.
- Resolution is the 20 kHz step. Use **Limit range** to concentrate on part of the band.
- One slice at a time: the sweep runs on the TX slice.

Settings

Key	Description
SwrSweepLicenseConfirmed	Set when you accept the licence confirmation with Remember my answer

8.3.4 Troubleshooting

A sweep refuses to start

A message names the reason. Fix the one it gives:

1. **No radio connected, or split active:** connect, or turn split off.
2. **Not the TX slice, TX slice locked, or no visible panadapter:** make the slice the TX slice, unlock it, and show its panadapter.
3. **Already transmitting or tuning:** stop transmit or tune first.
4. **Power Genius XL in OPERATE:** put it in STANDBY.
5. **Outside a supported amateur band, or on 60 m:** tune to another amateur band. 60 m is channelized and can't be swept.
6. **Band too wide for one panadapter, more than 260 points, or a limited range too narrow to measure:** tick **Limit range** and set a **From / To** range that is narrower, or wider if it was too narrow.

The sweep stops before transmitting with a TGXL in line

The Tuner Genius XL did not confirm **BYPASS**, so AetherSweep stopped rather than measure through the tuner.

1. Check the TGXL is connected and responding (see TGXL Tuner Control (p. 173)).
2. Start the sweep again.

Save CSV is refused

A sweep is still running, or there is no completed sweep to save.

1. Wait for the sweep to finish, or run one, then press **Save CSV**.

8.3.5 See also

- Panadapter Controls (p. 54) — the overlay panels

- TGXL Tuner Control (p. 173) — the tuner
- Amplifiers (p. 170)

8.4 Net Scheduler

The Net Scheduler keeps a list of the nets you check into and reminds you when one is about to start, with a one-click button that tunes the radio to it.

The schedule is stored on your computer, not in the radio's memory channels, so it works with any radio and needs no radio connected to edit.

8.4.1 Setup

Open it from **Tools** → **Net Scheduler...**

8.4.2 Using the Net Scheduler

Adding a net

Click **Add...** and fill in (a new net starts as Weekly on today's weekday):

- **Name**
- **Repeats: Once (no repeat), Daily, Weekly or Monthly**, with **every N** days, weeks or months (1–52).
 - **Weekly:** tick the days of the week it meets.
 - **Monthly:** pick the week and day, for example “on the **second** Tuesday”. The choices are first, second, third, fourth, fifth and last.
 - **Once:** pick the date.
- **At:** the start time and its time zone. The list offers your system time zone and **Etc/UTC**; you can type another zone name.
- **Remind me: At start time**, or **5, 10, 15 or 30 minutes before**, or **1 hour before**.
- **Frequency**, with **Capture current VFO** to copy the active slice's frequency, mode and filter (it needs an open slice on a connected radio).
- **Mode and Filter**.
- **Notes:** net control, NetLogger name and anything else you want to keep with it.

A live **Next:** line under the form confirms when the net will next run, so you can check the recurrence before saving.

Reminders

At reminder time a banner appears in the main window with:

- **Tune Now** — tunes to the net's frequency, mode and filter, the same way recalling a memory channel does. It needs a connected radio with an open, unlocked slice; otherwise the status bar says what to do.
- **Dismiss** — closes the reminder.

AetherSDR also tries to show a desktop notification.

Import and export

Export... saves the schedule as a JSON file; **Import...** reads one back, which is handy for sharing a club's net list or moving to another computer. When an imported net matches one you already have, choose:

- **Skip existing** — keep yours.
- **Overwrite** — replace yours with the imported one.
- **Import as new** — keep both.

8.4.3 Reference

The schedule list

The window lists your nets with these columns:

Column	Meaning
On	Whether reminders are enabled for this net
Name	The net's name
Repeats	How often it recurs
Next	The next date and time it starts
Frequency	Where it meets
Mode	The mode to tune

Buttons: **Add...**, **Edit...**, **Remove**, **Enable** / **Disable**, **Import...** and **Export...**

8.4.4 Troubleshooting

Tune Now doesn't tune the radio

Tune Now needs a connected radio with an open, unlocked slice. The status bar says what is missing.

1. Connect to the radio.
2. Open a slice, or unlock the active one.
3. Press **Tune Now** again.

Capture current VFO does nothing

It copies from the active slice, so it needs an open slice on a connected radio.

1. Connect to the radio and open a slice.
2. Press **Capture current VFO** again, or type the frequency, mode and filter by hand.

A net doesn't remind you

Reminders are switched off for that net.

1. Select the net and press **Enable**. Its **On** column shows that reminders are on.

8.4.5 See also

- Memory Channels (p. 92) — for frequencies you want to recall at any time rather than on a schedule.

8.5 AetherClock and GPS

Two tools help you know where you are and what time it is. **AetherClock** decodes the WWV/WWVH and WWVB time signals off the air and shows how far your computer's clock is from them. The **GPS & Station Location** window shows what a GPS-equipped radio knows: position, satellites, frequency reference and time.

8.5.1 Requirements

- **AetherClock** needs a receiver tuned to WWV, WWVH or WWVB. On a FlexRadio it listens through a DAX channel; on radios without DAX (Hermes-Lite 2 and others) it uses the slice's own audio.
- **GPS & Station Location** needs a radio that reports GPS hardware, for example a FlexRadio fitted with the GPSDO option, or an IC-705. A radio without GPS shows no GPS window, indicators or controls.

8.5.2 Setup

Setting up AetherClock

AetherClock is an applet (applet-tray button **CLK**, title **AetherClock**, Station category). It is off by default.

1. Open the applet and its settings drawer.

2. Pick a **Station**: preset: **WWV 2.5 / 5 / 10 / 15 / 20 MHz** or **WWVB 60 kHz**.
3. Press **Tune slice** → **preset** to put the bound slice on the right frequency (USB, 1 kHz below the carrier). This works while the decoder is stopped.
4. On a FlexRadio, choose a DAX channel in the applet's DAX chooser (**DAX Off**, **DAX 1 ...**). On radios without DAX the chooser is hidden.
5. Press **Enable** (it turns green while running).

On a FlexRadio with a DAX channel assigned, AetherClock decodes that slice alone, regardless of your speaker volume or mute.

8.5.3 Using AetherClock

Item	Meaning
Lock LED	Green when locked
Decoded UTC	The time decoded from the signal
Offset	How far the computer's clock is from the decoded time
Station tag	WWV or WWVH , shown as “–” until it is sure
Quality · age	Decode confidence and how old the last good decode is
Alignment scope	The received signal against the expected pattern
Car, Tick, Frm, Dec, Vote	The five decoding stages, from carrier to a voted time; each turns green as it succeeds
Verdict line	A plain-language status, for example “Locked — all stages green”, “No carrier — check antenna / band / preset dial” or “Syncing frame... (marker search, up to ~2 min)”

A debug log in the settings drawer shows the decoder's internal steps, useful when reporting a problem.

8.5.4 Using the GPS dashboard

Open it from **Tools** → **GPS Dashboard...**, or click the GPS indicator in the status bar.

Header cards

GPS FIX, **SATELLITES**, **10 MHz REFERENCE** and **TELEMETRY** (the age of the latest GPS report).

Current Location

Position source, **Grid square**, **Latitude**, **Longitude**, the coordinates in the radio's own format, **Altitude**, **Speed** and **Course**, plus the nearest mapped address (an online OpenStreetMap lookup). Buttons: **Copy gridsquare**, **Copy address** and **Refresh address**.

Station Map

A map of the radio's position. Use the arrow keys to pan, plus and minus to zoom, and Home to re-centre.

Satellite Reception and Frequency Reference

Satellites **Tracked** and **Visible**, the tracking ratio, the GPS **Frequency error**, the reference **setting**, **actual** source and **lock**, and how long GPS has been locked.

Satellite Time

UTC, local time and time zone, the radio's GPS UTC, and **Clock agreement** between the radio's GPS time and your computer. The radio's GPS supplies the time of day but not the date, so dates come from your computer. Use Clock agreement as a rough guide when looking at FT8 DT, and keep the computer synchronised with NTP for precise work.

Radio Clock Synchronization

On radios that support it, this group shows and changes clock settings stored in the radio itself: **Enable NTP client**, the **NTP server** (with **Apply**), **Correct radio clock from GPS automatically**, and **Sync now**. On a local FLEX-8000 the window also shows a tip for using the radio as an NTP server on a trusted network.

Where your position is used

The GPS page in **Radio Setup** holds the radio's GPS settings. Your position also feeds the PSK Reporter Map (p. 155) home location, the APRS beacon in AetherModem Packet Radio (p. 118) and the distance and bearing on Callsign Lookup (p. 163) cards.

8.5.5 Troubleshooting

AetherClock shows “Bound slice has no DAX channel — no audio”

On a FlexRadio, AetherClock listens through DAX, and the bound slice has no DAX channel.

1. Choose a channel in the applet's DAX chooser (**DAX 1 ...**) instead of **DAX Off**.

AetherClock says “No carrier — check antenna / band / preset dial”

The slice is not hearing the time station.

1. Check the **Station**: preset matches a station you can receive.
2. Press **Tune slice** → **preset** to put the slice on the right frequency.
3. Check the antenna and try another WWV frequency.

AetherClock stays on “Syncing frame...”

The decoder is searching for the frame marker, which can take up to about two minutes.

1. Wait for the marker search to finish before changing anything.

There is no GPS window or indicator

The connected radio does not report GPS hardware. The GPS window only appears for a radio that has GPS, such as a FlexRadio with the GPSDO option or an IC-705.

8.5.6 See also

- Radio Setup (p. 214)
- DAX Virtual Audio (p. 128)
- PSK Reporter Map (p. 155)
- AetherModem Packet Radio (p. 118)
- Callsign Lookup (p. 163)

8.6 Callsign Lookup

AetherSDR can look up a station's name, location, grid square, licence class and photo on QRZ.com¹¹. Lookups appear in two places: the **Callsign Lookup** window, and a contact card that pops up in the CW decoder when it copies a station signing DE <call>.

Without a QRZ.com account, lookups still work at country level, using the prefix data AetherSDR ships for DXCC spot colouring.

8.6.1 Requirements

- For full lookups, a QRZ.com account. An XML Logbook Data subscription on QRZ.com returns full details; a free account returns limited fields.
- Without an account, you get country-level prefix data only.

8.6.2 Setup

Open **Radio Setup** → **QRZ & Callsigns** (in the **ONLINE & APPEARANCE** group).

QRZ.com Account

- **Enable QRZ callsign lookups**

¹¹<https://www.qrz.com/>

- **Username (callsign):** and **Password:**
- **Test Login** checks the credentials and reports “Login OK” or why it failed.

The password is stored in your operating system’s keychain, never in the settings file. On a build without keychain support it is kept for the current session only.

Lookup Cache

Looked-up callsigns are cached for **7 days**, together with station photos, so a busy net doesn’t ask QRZ twice for the same station. The group shows how many entries are cached, and **Clear Cache** empties it.

8.6.3 Using Callsign Lookup

The Callsign Lookup window

Open it from **Tools → Callsign Lookup...** or press **Ctrl+Shift+L**.

Type a callsign and press **Lookup**. The card shows the callsign, licence class, name, location, grid, county, QSL methods (LoTW, eQSL, QSL) and the distance and bearing from your station, plus the station photo. Click the callsign to open its QRZ.com page.

The status line says where the answer came from: “Fetched from QRZ.com”, or “From cache — fetched *n* day(s) ago”. **Refresh** fetches fresh data from QRZ.com even if the callsign is cached.

If QRZ.com is not configured or doesn’t answer, the card shows **prefix** data instead (country, continent and CQ zone) and says so.

Contact card in the CW decoder

When the CW Decoder (p. 104) copies DE <call>, a compact contact card for that station appears beside the decoded text. It shows the name, location, grid and distance; click the callsign to open its QRZ.com page, and ✕ hides the card.

Looking up a spot

Right-click any spot on the panadapter and choose **Lookup on QRZ** to open that station’s qrz.com page in your browser. See SpotHub (p. 148).

8.6.4 Reference

Distance and bearing

Distance and bearing are measured from your own position: the radio’s GPS fix when it has one, otherwise a grid square (the radio’s, or the one on your own QRZ.com record). See AetherClock and GPS (p. 161).

8.6.5 Known issues

- The CW decoder’s contact card has no setting to turn it off; ✕ only hides the current card (#5454¹²).

8.6.6 Troubleshooting

The card shows only country, continent and CQ zone

QRZ.com is not configured, or it didn’t answer, so AetherSDR fell back to prefix data.

1. Open **Radio Setup → QRZ & Callsigns**.
2. Tick **Enable QRZ callsign lookups** and enter your **Username (callsign):** and **Password:**.
3. Press **Test Login** and read the result.

The card is missing some details

A free QRZ.com account returns limited fields.

1. Full details need an XML Logbook Data subscription on QRZ.com.

¹²<https://github.com/aethersdr/AetherSDR/issues/5454>

A station's details are out of date

The answer came from the 7-day cache.

1. Press **Refresh** in the Callsign Lookup window to fetch fresh data for that callsign.
2. To empty the whole cache, press **Clear Cache** in **Radio Setup** → **QRZ & Callsigns**.

You have to enter the QRZ.com password every session

This build has no keychain support, so the password is kept for the current session only. Use a build with keychain support to have it remembered.

8.6.7 See also

- CW Decoder ([p. 104](#))
- SpotHub ([p. 148](#))
- Radio Setup ([p. 214](#))
- AetherClock and GPS ([p. 161](#))
- QRZ.com¹³

¹³<https://www.qrz.com/>

Station Accessories

9.1 Peripherals

Applies to: All radios · the TGXL and PGXL radio relay, and the PGXL applet, need a FlexRadio

Settings → **Radio Setup...** → **Peripherals** is where AetherSDR is told about the station accessories it talks to directly, over the network or a serial port, rather than through the radio. It covers the 403A Tuner Genius XL (TGXL), Power Genius XL (PGXL) and Antenna Genius (AG), the ShackSwitch, ACOM, SPE Expert, VK3AMP and Elecraft KPA1500 amplifiers, and the TelePost LP-100A wattmeter.

Each device has its own applet once it is connected. Those are described on TGXL Tuner Control ([p. 173](#)), Amplifiers ([p. 170](#)) and ShackSwitch ([p. 175](#)); the PGXL and Antenna Genius applets are covered below. The Green Heron Everywhere switch is set up from its own applet instead (see Green Heron Everywhere ([p. 176](#))).

9.1.1 Setup

Open **Settings** → **Radio Setup...** (on macOS, **AetherSDR** → **Preferences...**) and pick **Peripherals** under CONTROLLERS & HARDWARE.

Layout

The **Devices** list sits on the left and the selected device's settings on the right. Each row shows the device name, its address, and where its data is coming from:

Word	Meaning
● DIRECT	Connected directly to the device
● RADIO	Not connected directly, but the radio relays the device (TGXL and PGXL only)
● OFFLINE	No connection
Connecting...	A connection attempt is in progress

The TGXL, PGXL and Antenna Genius applets use the same words. A device that needs you (a rejected code, a failed connection, a keychain problem) also shows **Needs attention**. The detail page repeats the state in a status line that is always visible.

Connection Help at the bottom of the list explains local and remote (VPN, SmartLink) connections in a few lines.

Adding a device

1. Click **Add** and pick a device type. A type already in the list stays in the menu, greyed, marked “(already added)”.
2. Add only creates the entry and selects it. Nothing connects yet.
3. Enter the address (and port, or serial port) and click **Connect**.

The list persists across restarts. Disconnect before changing a device's address or port; the fields are locked while it is connected.

A TGXL or PGXL that the radio reports, and that is waiting for an access code, can appear as a temporary row so you can enter the code. Retrying it at its reported address does not save that address as a manual target; editing the address or port does.

Serial-capable devices have a **Connection Type** choice, an **Address/Serial Port** field, **TCP Port/Speed**, and **Refresh Serial Ports**. The ShackSwitch page has a  **Web UI** button that opens the device's own web interface. Connection defaults for every device type are under Reference (p. 168).

Connecting automatically

TGXL, PGXL, Antenna Genius and ShackSwitch each have a toggle, **on by default**:

- **Connect directly when available** (TGXL, PGXL). Off, the device is controlled through the radio's relay whenever the radio offers one, and its applet shows RADIO. A live direct session carries on until it drops.
- **Connect automatically** (Antenna Genius, ShackSwitch). These have no relay, so off leaves the device offline until you click **Connect**.

With the toggle off, nothing connects that device by itself: not startup, not discovery, not the radio reporting it, and not a reconnect after a drop. **Connect** always works and does not change the toggle. To stop a discovered device from connecting, keep its row and turn the toggle off.

Reconnect automatically on the same page retries every peripheral after a connection drops. (The VK3AMP reads this setting only at startup.)

Access codes (AUTH)

A TGXL, PGXL or Antenna Genius can require an access code before it accepts a direct connection. When the device's greeting asks for AUTH, AetherSDR sends the code from the **Auth. Code** field and holds all other commands until the device accepts it.

- An accepted code is saved in the system keychain. It is bound to how you named the device: a literal IP address binds it to that IP and port; a host name (for example a DDNS name) binds it to the lower-cased name and port, so a changing home IP behind the name does not lose it. Only save a code for a name you control.
- Leave the field blank to reuse the saved code for that address. A new code replaces the saved one only after the device accepts it.
- Builds without keychain support keep the code for the session only ("Code for this session only").
- The code is hidden unless you click **Show**, is concealed again when you change page, and is kept out of logs, the automation bridge and widget grabs. **Clear code** deletes the saved code.
- After repeated failures with a saved code, the device is blocked: "Authorization blocked; enter a code and click Connect to retry".
- If direct authentication fails, a TGXL or PGXL falls back to the radio relay. Its applet shows which path it is on.

Antenna Genius and ShackSwitch share one credential slot.

Removing a device

Remove asks for confirmation first. Confirming disconnects the device, clears its saved connection settings and its stored code, and returns its toggle to the default. Remove does not stop discovery: a device the radio or the network reports may connect again, unless you keep its row with the toggle off.

For a device with a saved code, Setup waits for the keychain to confirm the deletion and blocks every reconnect of that device meanwhile. If the keychain does not answer within 15 seconds, Setup reports the deletion as unconfirmed; close and reopen Setup to retry. If the keychain refuses, the row stays and says "Saved code remains in keychain; retry Remove".

The full outcome table is in docs/peripherals-settings-ui.md¹.

9.1.2 Using the PGXL applet (AMP)

The **AMP** applet (title “PGXL”, Amplifiers category) appears when the radio reports a Power Genius XL.

Docked in the applet panel it is a compact tile:

- **PWR**, **SWR** and **Id** (drain current, read from the radio’s ID meter) bar gauges, plus a drive readout from the radio’s DRV meter.
- **PA** and **HL** (Harmonic Load) heatsink temperatures. Click either to switch between °C and °F.
- **Vac** (mains) and **Vdd** (drain) voltages. Vdd reads 0.0 V most of the time the amplifier is idle: it keeps its drain rail down until it enters operate. A dash means there is no direct connection.
- One row with **MEffA**, the **Fan** pull-down (STANDARD / CONTEST / BROADCAST) and **OPERATE**, above a fixed 2×2 grid of PA/HL and Vac/Vdd.
- The source indicator (● DIRECT, ● RADIO, ● OFFLINE) at the end of the readout row.

Popped out, or placed on the Workspace Canvas (p. 98), the applet lays itself out like the amplifier’s front panel: larger meters, a status strip for each RF port showing its band, bias profile and the radio feeding it, a **STBY** key, and a fan key that cycles S / C / B. The port carrying transmit is outlined. In standby a single banner replaces both strips. A port with no band reads N/A, meaning nothing is driving it.

The fan control, MEffA, port detail and the voltages need the **direct** connection. Over the radio relay the applet still shows power, SWR, Id and the operate state. Meters update at about 60 Hz while transmitting and 4 Hz while receiving.

MEffA is the amplifier’s Maximum Efficiency Algorithm: lit while it is optimising, shown in a standby style while the PA idles in class AAB.

9.1.3 Using the Antenna Genius applet (AG)

The **AG** applet (Antennas & Switching) controls a 4O3A Antenna Genius.

- It discovers devices on the LAN by UDP broadcast and connects to the first one found. A device selector appears when there is more than one.
- An **IP address** field and **Connect** reach a device that discovery cannot see (remote, VPN). Remote devices usually need an access code; see Access codes (p. 167).
- **Port A** and **Port B** each show the current band and antenna and a grid of antenna buttons. Click to select; click the selected antenna again to deselect it. Port B is hidden on a single-port device.
- An antenna held by the other port is dimmed and cannot be selected. Antennas that are receive-only on the current band are marked RX.
- **AUTO** per port turns the device’s own auto mode on or off for that port.

9.1.4 Reference

Device types and defaults

Device	Connection	Default port
Tuner Genius XL	TCP, address reported by the radio or entered manually	9010
Power Genius XL	TCP, address reported by the radio or entered manually	9008
Antenna Genius	TCP; also found by UDP broadcast on the LAN	9007
ShackSwitch	TCP (Antenna Genius protocol); always port 9007	9007
ACOM Amplifier	Serial (9600 8N1) or a ser2net host in raw mode	7000
SPE Expert Amplifier	Serial (115200 8N1) or a ser2net host	7000

¹<https://github.com/aethersdr/AetherSDR/blob/main/docs/peripherals-settings-ui.md>

Device	Connection	Default port
VK3AMP Amplifier	TCP control, UDP telemetry	5005 (UDP 5010)
Elecraft KPA1500	TCP	1500
LP-100A Meter	Serial (115200 8N1) or a ser2net host in raw mode	2000

Settings

The TGXL, PGXL, Antenna Genius and ShackSwitch addresses keep their own keys (TGXL_ManualIp, PGXL_ManualIp, AG_ManualIp, SS_ManualIp and the matching port keys). The device list, **Reconnect automatically**, the per-device toggles (Peripherals.<id>.AutoConnect, ids tgxl, pgxl, ag, shackswitch) and the ACOM, SPE, VK3AMP, KPA1500 and LP-100A connection settings live in the Peripherals document in AetherSDR.db. Access codes live only in the OS keychain. See Settings and Backups (p. 220).

9.1.5 Known issues

- The AMP applet stays hidden for a PGXL connected directly by its IP address unless the radio also reports the amplifier (#6010²).
- With a PGXL and TGXL on the local network, the AMP applet can show no PWR or SWR while transmitting, although Id is shown (#4805³).
- A PGXL reached directly over the internet accepts its access code, shows Connected, then drops straight away (#6222⁴).
- TUNE can fail on a remote FLEX-8600 while the Genius peripherals are connected in AetherSDR (#6001⁵).
- MEffA is not restored when AetherSDR starts; turn it on again each session (#5956⁶).
- The Antenna Genius applet's own **Connect** always uses port 9007 and ignores the port set in Peripherals (#5997⁷).
- The Antenna Genius applet can show a band on Port A that the device did not report, taken from the active slice (#6151⁸).

9.1.6 Troubleshooting

A device shows “Authorization blocked”

Repeated failures with the saved access code have blocked the device.

1. Select the device in **Settings** → **Radio Setup...** → **Peripherals**.
2. Type the correct code in **Auth. Code**.
3. Click **Connect**. The new code replaces the saved one once the device accepts it.

A device keeps reconnecting after Remove

Remove does not stop discovery, so a device the radio or the network reports connects again.

1. Keep the device's row in the list (add it again if you removed it).
2. Turn off **Connect directly when available** (TGXL, PGXL) or **Connect automatically** (Antenna Genius, ShackSwitch).

Remove says “Saved code remains in keychain; retry Remove”

The system keychain refused to delete the stored code, so the row stays.

1. Make sure the keychain is unlocked.

²<https://github.com/aethersdr/AetherSDR/issues/6010>

³<https://github.com/aethersdr/AetherSDR/issues/4805>

⁴<https://github.com/aethersdr/AetherSDR/issues/6222>

⁵<https://github.com/aethersdr/AetherSDR/issues/6001>

⁶<https://github.com/aethersdr/AetherSDR/issues/5956>

⁷<https://github.com/aethersdr/AetherSDR/issues/5997>

⁸<https://github.com/aethersdr/AetherSDR/issues/6151>

2. Click **Remove** again.
3. If Setup reports the deletion as unconfirmed, close and reopen Setup, then retry.

The address and port fields cannot be edited

The fields are locked while the device is connected.

1. Disconnect the device.
2. Change the address or port, then click **Connect**.

9.1.7 See also

- Radio Setup ([p. 214](#))
- Amplifiers ([p. 170](#))
- TGXL Tuner Control ([p. 173](#))
- ShackSwitch ([p. 175](#))
- Green Heron Everywhere ([p. 176](#))
- AetherSweep ([p. 158](#))

9.2 Amplifiers

AetherSDR controls several linear amplifiers directly, independent of the radio: the ACOM S-series, the SPE Expert 1.3K-FA / 1.5K-FA / 2K-FA, the VK3AMP 600 / 1000 / 2000 W, and the Elecraft KPA1500. Each is configured in [Peripherals](#) ([p. 166](#)) and gets its own applet in the **Amplifiers** category. Any combination can be connected at once, alongside a PGXL.

The 4O3A Power Genius XL is relayed by the FlexRadio and has its own applet, described on [Peripherals](#) ([p. 166](#)) (PGXL applet). The TelePost LP-100A wattmeter is covered at the end of this page.

An amplifier's applet button stays hidden from the applet bar until that amplifier is configured and present; it then appears on its own.

SWR sweeps: AetherSweep ([p. 158](#)) only knows how to check a PGXL. Put any of the amplifiers on this page into standby or bypass yourself before sweeping.

9.2.1 Setup

1. Open **Settings** → **Radio Setup...** → **Peripherals**.
2. Click **Add** and choose the amplifier.
3. Fill in the connection (see [Reference](#) ([p. 172](#))) and click **Connect**.

Reconnect automatically on the [Peripherals](#) page retries after a dropped connection.

For serial-capable amplifiers, choose the **Connection Type**, then either pick the serial port (**Refresh Serial Ports** rescans) or enter the ser2net host and **TCP Port/Speed**.

9.2.2 Using the ACOM S-series

Supports the current S-series line: 500S, 600S, 700S, 1200S, 1400S and 2020S. The model is detected from the amplifier, so there is no model selector.

The **ACOM Amplifier** applet shows:

- A status pill with the amplifier mode (OPERATE, STANDBY, OFF).
- **PWR**, **REF** (reflected) and **SWR** gauges. They rescale with the detected model.
- An info grid with temperature (click to switch °C / °F), high voltage, drain current, band and the amplifier's total operating time.
- A fault banner while a fault stands, with **CLEAR** to clear it.
- **STANDBY**, **OPERATE** and **OFF** buttons.

9.2.3 Using the SPE Expert 1.3K-FA / 1.5K-FA / 2K-FA

The amplifier only answers when polled, so AetherSDR polls it every 100 ms. The model is identified from its status. If the amplifier goes silent behind a live ser2net link, the applet reports it as off.

ser2net: monitoring and control work with the proxy port in raw or telnet mode. Powering the amplifier **on** over the network drives the proxy's DTR/RTS lines, which needs an RFC 2217 telnet port (**accepter: telnet(rfc2217=true), <port>**). A sample configuration is in the tooltip on the network address field.

The **SPE Expert Amplifier** applet (button **SPE**):

- Gauges for output power (rescaled with the LOW / MID / HIGH level), antenna SWR, and the SWR seen before the ATU.
- Temperature, voltage and current readouts, and a warning / alarm banner.
- **ON** (a DTR/RTS power pulse; works over the network with RFC 2217), **OFF**, **OPER** / **STBY**, the power level (click to cycle LOW / MID / HIGH), **TUNE** (the ATU), **INPUT** 1/2, **ANT** (cycles the antenna for the current band), and drive up / down.

Popped out into its own window, the applet adds a live mirror of the amplifier's LCD and a **FRONT PANEL** key group: **BAND -** / **BAND +**, **SET**, and the manual ATU steps **L -** / **L +** / **C -** / **C +**. These keys work only while the LCD mirror is current; if its data goes stale they are dimmed, and the display stays visible.

9.2.4 Using the VK3AMP 600 / 1000 / 2000 W

A network amplifier controlled over TCP, with live telemetry on UDP. There is no serial option.

In Peripherals, pick the **Amplifier Model** (600 W, 1000 W or 2000 W; default 2000 W). It sets the forward-power gauge's scale.

The **VK3AMP Amplifier** applet:

- A status pill (**CONNECTED** / **—**).
- **PWR**, **REF** and **SWR** gauges.
- An info grid with temperature, voltage, current, band and antenna.
- A fault banner. Faults are shown as the amplifier's numeric code; there is no fault-name table.
- **BYPASS** / **AMP ON** (the label shows the current state) and **COOLING** (cooling override).
- **ANT** 1–3 antenna select.
- **RAIL LOW** / **HIGH** voltage-rail buttons. They are unavailable while the amplifier is bypassed.
- **RESET**, which asks for confirmation and then holds the reset command for about ten seconds.

The VK3AMP reads **Reconnect automatically** only at startup.

9.2.5 Using the Elecraft KPA1500

Not yet validated on real hardware. The KPA1500 support follows Elecraft's published programming reference but has not been run against a physical amplifier (#4097⁹). Reports from owners are welcome.

The **Elecraft KPA1500** applet:

- A status pill, **PWR**, **REF** and **SWR** gauges.
- An info grid with PA temperature, band and the ATU state.
- A fault banner showing the code in hex (for example **Fault B0**) and a **CLR FAULT** button.
- **OPERATE** / **STANDBY**. The label changes only when the amplifier confirms.
- The internal ATU: **ATU IN** / **ATU BYP**, and **TUNE**, which reads **TUNING** while a tune runs; press it again to cancel. A full-search tune needs RF, so key the radio's own **TUNE**: nothing in this applet keys the radio.
- **ANT 1** / **ANT 2**. An **ANT n** label appears when an external antenna switch reports an antenna number from 3 to 32.

⁹<https://github.com/aethersdr/AetherSDR/issues/4097>

There is no network PTT and no lost-link TX inhibit.

9.2.6 Using the LP-100A wattmeter

The TelePost LP-100A is a wattmeter, not an amplifier, but it is set up the same way: **Peripherals** → **Add** → **LP-100A Meter**, over a local serial port (115200 8N1) or a ser2net proxy in raw mode (default TCP 2000).

The **LP-100A Meter** applet is in the **Metering** category and is **read-only**: it does not change the meter's range, mode or alarm.

- **Power** and **SWR** gauges, plus dBm, return loss, impedance (Z), phase and the meter's power range.
- A status pill: **LIVE** (AetherSDR is polling the meter), **SHARED** (another program is already polling it; AetherSDR rides along instead of doubling the traffic), **NO DATA** (connected but the meter stopped answering) or **OFFLINE**.
- The meter reports which range it is on but not its full-scale value. Right click the applet to set each range's full scale, or **Reset to defaults**.

9.2.7 Reference

Amplifier	Connection	Defaults
ACOM Amplifier	Serial, or network through a ser2net proxy in raw mode	9600 8N1; TCP 7000
SPE Expert Amplifier	Serial, or network through ser2net (raw or telnet)	115200 8N1; TCP 7000
VK3AMP Amplifier	Network only: TCP control, UDP telemetry	TCP 5005; UDP 5010
Elecraft KPA1500	Network only: TCP	TCP 1500
LP-100A Meter	Serial, or network through a ser2net proxy in raw mode	115200 8N1; TCP 2000

9.2.8 Known issues

- An SPE amplifier set to connect at startup shows connected but no data until you disconnect and reconnect it (#4893¹⁰).

9.2.9 Troubleshooting

The SPE amplifier will not power on over the network

Powering on drives the ser2net proxy's DTR/RTS lines, which a plain raw or telnet port cannot do.

1. Configure the ser2net port as an RFC 2217 telnet port: `accepter: telnet(rfc2217=true), <port>`. The tooltip on the network address field has a sample configuration.
2. Reconnect the amplifier in **Peripherals** and press **ON**.

The VK3AMP takes a long time to connect on Windows

After the amplifier has been idle, the first connection can take 9–12 seconds, because the amplifier answers only broadcast ARP.

1. Wait: AetherSDR waits up to 18 seconds before trying again.

The SPE front-panel keys are dimmed

The popped-out **FRONT PANEL** keys work only while the LCD mirror is current. Its data has gone stale.

1. Check the amplifier's connection in **Peripherals**.
2. Wait for the LCD mirror to update; the keys come back with it.

9.2.10 See also

- Peripherals (p. 166)
- TGXL Tuner Control (p. 173)
- Meters (p. 72)

¹⁰<https://github.com/aethersdr/AetherSDR/issues/4893>

- AetherSweep ([p. 158](#))

9.3 TGXL Tuner Control

Applies to: All radios · the radio relay and OPERATE / STANDBY / BYPASS need a FlexRadio

AetherSDR controls the 403A Tuner Genius XL (TGXL) through the **TUN** applet, including manual adjustment of the tuner's Pi-network relays.

9.3.1 Setup

AetherSDR can reach the TGXL two ways, and the applet shows which one it is using with an indicator at the end of its readout row:

Indicator	Path
● DIRECT	A direct TCP connection to the tuner on port 9010
● RADIO	Through the FlexRadio, which relays the tuner's status and commands
● OFFLINE	Neither is available

When a TGXL is connected to your FlexRadio, the radio reports its address and AetherSDR opens the direct connection automatically. For a tuner the radio does not report (on another subnet, over a VPN, or with a non-Flex radio), add it in Peripherals ([p. 166](#)) (**Add** → **Tuner Genius XL**) with its address.

- **Connect directly when available** (on by default, in Peripherals) controls the automatic direct connection. Turn it off to use only the radio's relay.
- **Access code:** if the TGXL asks for authorization, enter its code in Peripherals. AetherSDR sends it before any other command and saves it in the system keychain once the tuner accepts it. See Peripherals ([p. 166](#)) for how codes are stored.
- If direct authorization fails, AetherSDR falls back to the radio relay and the indicator shows ● **RADIO**.

The direct connection is what adds port detail, the antenna buttons and manual relay control. Without it the applet still works from what the radio relays.

OPERATE, STANDBY and BYPASS go through the radio. A TGXL reached by IP only, with no FlexRadio relaying it, cannot be switched between them; pressing those keys raises a notice saying the control is not available rather than doing nothing silently.

9.3.2 Using the tuner applet

Docked in the applet panel, the TUN applet (title "TGXL") shows:

- **Fwd Power** gauge. It scales for barefoot, Aurora or PGXL power levels, and uses the tuner's own peak reading, so voice peaks read correctly.
- **SWR** gauge (1.0–3.0).
- **C1, L** and **C2** relay bars.
- **TUNE:** starts an auto-tune. While a tune runs it reads **STOP**; pressing it stops the tune. The tuner stays in operate.
- **OPERATE:** cycles OPERATE → BYPASS → STANDBY.
- **ANT 1 / ANT 2 / ANT 3** antenna buttons, shown with a direct connection to a TGXL that has an antenna switch.

Messages from the tuner appear across the applet for as long as the tuner shows them: after a successful tune it shows the SWR it settled on (Tuned SWR: 1.15:1), and a tune that could not run says why.

Meters update at about 60 Hz while transmitting and 4 Hz while receiving.

Popped out or on the canvas

Popped out into its own window, or placed on the Workspace Canvas (p. 98), the applet is laid out like the tuner's front panel: the two meters, a status strip for each RF port showing what is feeding it and where it is tuned, the C1/L/C2 relay positions as dials, and separate **STBY**, **BYP** and **TUNE** keys. A port shows its source only while it has a live reading.

9.3.3 Using manual relay control

With the direct connection active, the C1, L and C2 bars (and the dials in the popped-out layout) are adjustable:

- **Scroll up** over a relay to increase its position by 1.
- **Scroll down** to decrease it by 1.
- The cursor changes to $\updownarrow$ over an adjustable relay.

This is useful for:

- Tweaking an auto-tune result for lower SWR
- Experimenting with different matching configurations
- Matching antennas the auto-tuner struggles with

9.3.4 Reference

Relays

Relay	Component	Range
C1	Input capacitor	0–255
L	Inductor	0–255
C2	Output capacitor	0–255

Protocol

The TGXL uses a text protocol in the same format as the SmartSDR API:

Direction	Format
TGXL → Client (greeting)	V<version> or, when authorization is required, V<version> AUTH
Client → TGXL	C<seq> <command>\n
TGXL → Client	R<seq> <code> <body>\n
TGXL → Client (push)	S0 state key=val ... \n

Authorization: auth <code>. Manual relay command: tune relay=<0\|1\|2> move=<+1\|-1>.

9.3.5 Known issues

- TUNE can fail on a remote FLEX-8600 while the Genius peripherals are connected in AetherSDR (#6001¹¹).
- The TUN indicator in the status bar may not change the tuner's state, and the OPERATE cycle can disagree with the tuner's real state (#4946¹²).

9.3.6 Troubleshooting

A tune stops with “LOW RF POWER”

There was too little drive for the tuner to measure against.

1. Raise the drive.
2. Press **TUNE** again.

¹¹<https://github.com/aethersdr/AetherSDR/issues/6001>

¹²<https://github.com/aethersdr/AetherSDR/issues/4946>

Relay bars don't scroll

Relay control needs the direct connection.

1. Check the indicator reads ● DIRECT. ● RADIO means AetherSDR is using the radio's relay.
2. Check the TGXL's row in **Settings** → **Radio Setup...** → **Peripherals**. **Needs attention** means a rejected code or failed connection; enter the code and click **Connect**.
3. Make sure AetherSDR can reach TCP port 9010 on the tuner. Over SmartLink or a remote link, add the tuner's reachable address in Peripherals.

TGXL not detected

The radio is not reporting the tuner, or AetherSDR cannot reach it.

1. Verify the TGXL is connected to the radio and powered on.
2. Add it manually in Peripherals if the radio does not report it.
3. Check the logs in **Help** → **Support & Diagnostics...**

OPERATE, STANDBY or BYPASS says the control is not available

These go through the radio, and no FlexRadio is relaying this tuner.

1. Connect the TGXL to your FlexRadio, and connect AetherSDR to that radio, so the radio relays the tuner.

9.3.7 See also

- Peripherals ([p. 166](#))
- AetherSweep ([p. 158](#)) (automatic TGXL bypass during a sweep)
- Amplifiers ([p. 170](#))

9.4 ShackSwitch

ShackSwitch is an open-source antenna / accessory switch built by Nigel Fenton. It speaks the 4O3A Antenna Genius protocol on port 9007, so AetherSDR sets it up in Peripherals ([p. 166](#)) and gives it a dedicated **SS** applet for one-click antenna switching.

9.4.1 Setup

1. Open **Settings** → **Radio Setup...** → **Peripherals**.
2. Click **Add** → **ShackSwitch** and enter the device's IP address. Control always uses port 9007.
3. Click **Connect**. The **SS** applet appears in the applet panel (Antennas & Switching category).

Connect automatically is on by default, so AetherSDR reconnects the ShackSwitch at startup and after a drop. Turn it off to connect only when you click **Connect**. The  **Web UI** button on the Peripherals page opens the device's own web interface.

ShackSwitch shares the Antenna Genius connection and its credential slot, so one of the two is connected at a time. See Peripherals ([p. 166](#)) for access codes and removal.

9.4.2 Using the ShackSwitch applet

Area	What it shows
Status line	Connection state ("Not connected" until connected)
INPUT A	Band and antenna currently on input A
INPUT B	Band and antenna on input B (two-input devices only)
ANTENNA rows	One row per antenna, named from the device's own configuration (set in its web UI), with an A button and, on two-input devices, a B button
Dummy Load:	Which antenna port is your dummy load (None by default)
Settings 	Opens the device's web interface

Click **A** or **B** on a row to put that input on that antenna. The selected button is lit in that input's colour (cyan for A, orange for B).

Two-input (SO2R) mode

On a two-input device the applet shows both input cards and both button columns. A single-input device hides input B.

If both inputs land on the same antenna:

- With a **Dummy Load** set, AetherSDR moves input B to the dummy load and blinks the B button on the antenna B wanted, until you move B yourself.
- Without one, the B button blinks amber on the shared antenna as a warning.

9.4.3 Reference

Key	Description
SS_ManualIp	ShackSwitch address
SS_ControlPort	Control port (9007)
SS_DummyLoadAnt	Antenna chosen as the dummy load (-1 = none)
Peripherals.shackswitch.AutoConnect	Connect automatically toggle

These live in AetherSDR.db; see Settings and Backups (p. 220).

9.4.4 Troubleshooting

The applet doesn't appear

The ShackSwitch is not connected.

1. Check the ShackSwitch row in **Settings** → **Radio Setup...** → **Peripherals**. ● OFFLINE with **Connect automatically** off means it waits for **Connect**.
2. Confirm the address in the device's web UI, and that this PC can reach port 9007 on it.

Buttons are greyed out

The applet enables its buttons only after a successful connection.

1. Check the status word on the Peripherals page.
2. Click **Connect** if it reads ● OFFLINE.

9.4.5 See also

- Peripherals (p. 166)
- Green Heron Everyware (p. 176) for the Everyware switch
- USB Cable Management (p. 178) for radio-side BCD/Bit cables that switch accessories by band

9.5 Green Heron Everyware

The **GHE** applet selects the antenna on a Green Heron *Everyware* remote antenna switch and turns a rotator attached to the same station. It talks to the Everyware server directly over TCP, knows nothing about the radio, and works with any radio connected, or none.

The switch and the rotator arrive on one connection, which is why they share one applet.

9.5.1 Requirements

The address you enter is the Everyware **server**: a service running on a PC with the switch hardware attached to it over serial, not a switch. One server can present several switches (for example AS-84F-1 to AS-84F-4), each with the same set of antenna ports.

9.5.2 Setup

The GHE applet is in the **Antennas & Switching** category. It is configured in the applet itself, not in Peripherals (p. 166).

Field	Meaning
IP	Address of the Everyware server
Port	Its TCP port; 10000 unless the installation moved it
Switch	Which switch feeds this radio. Filled from the server's roster once connected, and remembered

Click **Connect**. The status line reads “Connected — waiting for roster”, then “Connected — waiting for switch state” until the server reports.

9.5.3 Using the antenna rows

A radio is wired to one switch, so the applet shows only that switch's antennas, one row per port:

Row	Meaning
Beam-20 · ON (lit)	The antenna the device reports this switch is on
Beam-15 · in use by AS-84F-3 (greyed)	Held by another switch; an antenna feeds one switch at a time, so the device would refuse it
EFHW-40	Free: click to select it

Nothing is assumed from a click. The row lights only when the device reports the change, which it does within about an eighth of a second. A relay that fails to move shows up as a row that does not light.

When the link drops, the last known state stays on screen marked stale (“Reconnecting — state is stale”) and the rows are disabled until it comes back.

9.5.4 Using the rotator

The rotator section appears under the antennas **only while the device is reporting a heading**:

```
Rotor      62.9° · asked 64.3° · Δ1.4°
[ 64.3      ] [ Turn ]
```

- Type a heading in degrees, then press **Turn** or Enter. Typing alone sends nothing.
- The readout is the **reported** heading. After a turn it adds what was asked and the difference. It never says “on target”: the measured sensor noise at rest is about $\pm 3.8^\circ$, larger than any useful arrival threshold.
- There is no stop or park command in the protocol, so a rotation that has started cannot be cancelled from AetherSDR. Use the rotator controller.
- If the server reports more than one rotator, a **Rotator** selector picks which one this applet drives.
- The section disappears when the rotator stops reporting (“The rotator has stopped reporting a heading”).

Popped out into its own window, the applet adds a compass rose under the Turn row. The needle is the reported heading; a dashed tick shows the asked heading. The rose is display-only: clicking it does nothing, by design.

9.5.5 Known issues

- If an Everyware server restarts and announces its switches in a different order, the applet can name the wrong switch as holding an antenna (#5250¹³).

9.5.6 Troubleshooting

Re-sending a heading does not correct a small error

The heading is inside the controller's deadband, so the same request does nothing.

¹³<https://github.com/aethersdr/AetherSDR/issues/5250>

1. Type a slightly different heading.
2. Press **Turn**.

The antenna rows are greyed out and marked stale

The link to the Everyware server has dropped. The applet shows “Reconnecting — state is stale” and keeps the last known state on screen.

1. Wait for the link to come back; the rows are enabled again when it does.
2. If it does not, check that the server is running and that this PC can reach its **IP** and **Port**.

9.5.7 See also

- ShackSwitch (p. 175)
- Peripherals (p. 166)
- MQTT Station Automation (p. 141)
- docs/green-heron-everyware.md¹⁴: more detail, including the protocol notes

9.6 USB Cable Management

Applies to: FlexRadio

AetherSDR can configure USB-serial adapters plugged into the FlexRadio’s rear-panel USB ports. These cables provide CAT control, band decoder output, per-bit switching, LDPA amplifier control and raw serial passthrough to external devices, all managed remotely over the network.

The radio detects USB-serial adapters on its rear USB ports and reports them in status messages. AetherSDR lists them on the **USB Cables** page of Radio Setup, where you set each cable’s type, serial parameters and behaviour.

These are the **radio’s** USB ports, not your computer’s. For USB-serial PTT/CW keying on your computer, see **Settings** → **Radio Setup...** → **Serial & Controllers** (USB Control Surfaces (p. 189)). For accessories AetherSDR talks to directly, see Peripherals (p. 166).

9.6.1 Setup

Open the **USB Cables** page:

- **Settings** → **USB Cables...** opens Radio Setup directly on the page.
- Or open **Settings** → **Radio Setup...** and pick **USB Cables** under CONTROLLERS & HARDWARE.

Cable list

The **Cables** list on the left shows every cable the radio reports, with its type in brackets (for example [BCD]) and “(unplugged)” when it is not plugged in:

Colour	Meaning
Green	Enabled and plugged in
Yellow	Enabled but unplugged
Grey	Disabled

Cables appear and disappear automatically as the radio reports them. Click a cable to edit it on the right. Every cable page starts with its name, an **Enabled** checkbox, its plugged-in status, and the **Cable Type** selector.

Choosing a cable type

Cable Type chooses the cable’s protocol: **CAT**, **Bit**, **BCD**, **LDPA** or **Passthrough**. Changing it reconfigures the cable on the radio, so the mouse wheel cannot change it by accident: click and choose.

¹⁴<https://github.com/aethersdr/AetherSDR/blob/main/docs/green-heron-everyware.md>

A freshly plugged cable with no type yet opens the **Unconfigured** page: “Select a cable type to configure this device.” Pick a type to configure it, or use **Remove This Cable** to remove it from the radio.

Each cable keeps its own serial settings. Cable configuration is stored in the radio, not in AetherSDR.

9.6.2 Using the cable types

- **Amplifier tracking:** a CAT cable sends frequency to your amplifier so it follows band changes.
- **Antenna switching:** a BCD cable drives a remote antenna switch.
- **Bandpass filters:** a Bit cable drives per-band relays on a filter bank.
- **VHF amplifier:** an LDPA cable sets a 2m/4m LDPA amplifier’s band and preamp.
- **Remote serial devices:** a Passthrough cable tunnels serial data to a device at the radio site.

9.6.3 Reference

CAT

Sends CAT (Computer Aided Transceiver) frequency and mode data over the serial port. Amplifiers, antenna tuners and logging software plugged into the radio’s USB port follow the radio.

- **Serial parameters:** Speed, Data Bits (7/8), Parity, Stop Bits (1/2), Flow (none, RTS/CTS, DTR/DSR, XON/XOFF)
- **Source:** which frequency to report: None, TX Pan, TX Slice, Active Slice, TX Ant, RX Ant or Ordinal Slice
- **Auto Report:** send updates automatically when frequency or mode changes

BCD

Outputs Binary Coded Decimal band data for external band decoders: antenna switches, bandpass filter banks and amplifier band switches.

- **BCD Type:** HF (bcd), VHF (vbcd) or HF+VHF (bcd_vbcd)
- **Polarity:** Active High or Active Low
- **Source:** which frequency determines the band output

Bit

Eight independent output bits (Bit 0–7) for per-band or per-frequency switching. Pick a bit in the **Bits** list and set it in **Bit Settings**:

- **Enabled**
- **Source**, and **Antenna/Slice** when the source needs one
- **Output:** band (matches a band) or `freq_range` (matches a frequency range)
- **Band** (for band), or **Low Freq (MHz)** and **High Freq (MHz)** (for `freq_range`)
- **Polarity:** High or Low
- **PTT Dependent:** the bit follows transmit
- **PTT Delay** and **TX Delay:** 0–10000 ms

LDPA

Controls an LDPA amplifier on 2m or 4m.

- **Band:** 2m or 4m
- **Preamp:** on or off
- **Source:** which frequency the amplifier follows

Passthrough

A raw serial tunnel: bytes sent to one end come out the other. Used to talk directly to a device plugged into the radio’s USB port, tunnelled over the network to your client.

- **Serial parameters:** Speed, Data Bits, Parity, Stop Bits, Flow

9.6.4 Known issues

- A CAT cable with **Source** set to TX Ant or RX Ant has no choice of which antenna port to follow; only Bit cables offer **Antenna/Slice** (#4816¹⁵).

9.6.5 Troubleshooting

A new cable shows the Unconfigured page

The radio has detected the adapter but it has no cable type yet.

1. Select the cable in the **Cables** list.
2. Pick a **Cable Type**, then set its serial parameters and source.

A cable is listed in yellow

The cable is enabled but not plugged in to the radio.

1. Check the adapter is plugged in to the radio's rear USB port, not your computer.
2. The entry turns green when the radio reports it plugged in.

9.6.6 See also

- Peripherals (p. 166)
- USB Control Surfaces (p. 189)
- ShackSwitch (p. 175)
- Radio Setup (p. 214)
- FlexRadio (p. 20)

¹⁵<https://github.com/aethersdr/AetherSDR/issues/4816>

Controllers

10.1 StreamDeck

AetherSDR drives the **Elgato Stream Deck+** directly over USB, with no Elgato software and no plugin. Its eight LCD keys show their assigned action, and its four dials tune, adjust RIT/XIT, volume, power and more. Other Stream Deck models are not driven natively; use them through the control-surface software of your choice talking to the TCI Server (p. 135) or the Automation Bridge and MCP (p. 144).

Upgrading from an older version: the Elgato Stream Deck and StreamController plugins are no longer shipped with AetherSDR (#5600¹). If you used them, set up a Stream Deck+ natively (below), or point your Stream Deck software at TCI or the automation bridge.

10.1.1 Requirements

- An Elgato Stream Deck+.
- A build with HID support (hidapi).

10.1.2 Setup

1. Open **Settings** → **Radio Setup...** → **Serial & Controllers**.
2. In **USB Control Surfaces**, tick **Enable HID encoders / StreamDeck+ (RC-28, PowerMate, ShuttleXpress, ...)**.
3. **macOS:** allow AetherSDR under **System Settings** → **Privacy & Security** → **Input Monitoring** when asked.
4. **Linux:** install the udev rule from the source tree so your user can open the device:

```
sudo cp packaging/linux/60-hid-encoders.rules /etc/udev/rules.d/  
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Then unplug and replug the Stream Deck+.

10.1.3 Using the LCD keys

StreamDeck+ LCD Button Actions assigns an action to each of the 8 keys (**Key 1–8**). The key's image updates on the device to show its action. All keys start as **None**.

10.1.4 Using the dials

HID Encoder / StreamDeck+ Encoders sets what each dial does when turned, and **StreamDeck+ Encoder Push Actions** sets what pressing each dial does. Defaults and choices are under Reference (p. 182).

¹<https://github.com/aethersdr/AetherSDR/issues/5600>

Touch strip

The touch strip above the dials shows each dial's turn and push labels, with ON/OFF state for dials assigned to RIT or XIT.

10.1.5 Using other Stream Deck models

Other Stream Deck models (MK.2, XL, Mini, Pedal and so on) can still control AetherSDR, through software you choose. AetherSDR provides the protocol, not the button layer:

- **TCI Server** (p. 135): the network protocol many radio-control plugins already speak.
- **Automation Bridge and MCP** (p. 144): a local socket with verbs for reading state and operating controls.

10.1.6 Reference

Key actions

None, Toggle TX (MOX), Toggle Tune, Toggle RIT on/off, Toggle XIT on/off, Clear RIT offset, Clear XIT offset, Step Size Up, Step Size Down, Toggle Mute, Toggle Slice Lock, Toggle APF, Cycle AGC Mode, Toggle Band Zoom, Toggle Segment Zoom, Next Slice, Previous Slice, Volume Up (+5), Volume Down (-5), Toggle Split, Monitor TX Frequency.

Dial turn actions

Dial	Default
Encoder 1	Tune Slice
Encoder 2	RIT (Receive Incremental Tuning)
Encoder 3	XIT (Transmit Incremental Tuning)
Encoder 4	Master Volume

Other choices: Slice Audio Volume, Headphone Volume, AGC Threshold, APF Level, CW Speed, RF Power, None.

Dial push actions

Dial	Default
Encoder 1 push	Cycle Tuning Step
Encoder 2 push	Toggle RIT on/off
Encoder 3 push	Toggle XIT on/off
Encoder 4 push	None

Other choices: Toggle TX (MOX), Toggle Mute, Lock Slice.

10.1.7 Known issues

- Volume up/down actions start from the saved master volume, not the level of the audio path you are hearing, so the first step can jump (#6132²).

10.1.8 Troubleshooting

The Stream Deck+ does nothing

HID encoders are off by default.

1. Check that **Enable HID encoders / StreamDeck+** is ticked in **Settings → Radio Setup... → Serial & Controllers**. If the checkbox is missing, this build has no HID support.
2. Turn on the **Ext Devices** log category (see Support and Logging (p. 261)) to see HID connection events.

Linux: the device is not found

Your user cannot open the HID device.

²<https://github.com/aethersdr/AetherSDR/issues/6132>

1. Install the udev rule (see Setup (p. 181)).
2. Unplug and replug the Stream Deck+.

macOS: no response

AetherSDR does not have the Input Monitoring permission.

1. Allow AetherSDR under **System Settings** → **Privacy & Security** → **Input Monitoring**.

10.1.9 See also

- USB Control Surfaces (p. 189): RC-28, PowerMate, Contour Shuttle and serial keying
- Ulanzi Dial (p. 186)
- MIDI Controller Mapping (p. 192)
- TCI Server (p. 135)

10.2 FlexControl Tuning Knob

The FlexRadio FlexControl is a USB tuning knob with a weighted encoder, three Aux buttons and a push on the knob. AetherSDR detects it automatically and uses it for smooth tuning with acceleration. The buttons can also switch the knob to other jobs (RIT, XIT, volume, AGC-T, APF, CW speed, RF power) or run actions such as MOX, split or CWX macros.

The FlexControl is a host device: it plugs into your computer, so its settings stay available whichever radio you connect. Actions a radio cannot perform are refused with a notice.

AetherControl is an on-screen copy of the FlexControl that works with or without the physical knob (see Using AetherControl (p. 184) below).

10.2.1 Requirements

- FlexControl USB tuning knob (FlexRadio accessory)
- AetherSDR built with Qt serial-port support

The FlexControl is a USB serial device (VID 0x2192, PID 0x0010) running at 9600 baud 8N1. No special driver is needed.

10.2.2 Setup

Open **Settings** → **Radio Setup...** → **Serial & Controllers** and find the **FlexControl Tuning Knob** group. Shortcuts straight to it: **Settings** → **FlexControl Knob & Buttons...**, or the **Settings...** button in the AetherControl window.

Control	What it does
Status	“Not detected”, or the port it is connected on
Detect / Close	Look for the knob now and open it / release it
Button 1–3, Knob Button: Tap and Double	The action for a single and a double press of each button
Auto-detect on startup	Find and open the knob at launch (on by default)
Invert tuning direction	Reverse the knob

If you plug the knob in after launch, click **Detect**.

10.2.3 Using the knob

The encoder reports its own acceleration (1–6 steps per detent depending on how fast you turn). Each step is the current tuning step from the RX applet (10 Hz, 100 Hz, 1 kHz, ...), so change the step size to change the tuning resolution.

Wheel modes

Actions whose name starts with **Wheel** do not fire once: they change what the knob does. Tap a button bound to, say, `WheelRit`, and turning the knob now adjusts RIT; the button's LED shows which mode is active. Press the knob, or tap a button bound to an ordinary action, to return the knob to tuning.

Upgrading from an older version: the FlexControl volume action drives **Master Volume**, not the active slice's volume. The older `WheelMasterAf` action has been merged into it, and saved bindings that use it keep working as Master Volume. Use `WheelSliceAudio` for per-slice volume (#2925³, #2986⁴).

10.2.4 Using AetherControl (virtual FlexControl)

Settings → **AetherControl...** opens **AetherControl**, a “Virtual Tuning Controller” window laid out like the FlexControl. It shares its button assignments with the physical knob and mirrors it when one is connected.

- **Wheel:** double-click the knob to capture the mouse for circular tuning; double-click again to release (Esc also releases). A single fast mouse jump moves the wheel by at most 15°, and the wheel only coasts after a real flick.
- **Aux1–Aux3** and **PUSH**, each with a **Single Tap** and **Double Tap** action, from the action list below.
- **Wheel Tightness** (Tight ↔ Loose, default 45) and **Mouse Sensitivity** (Less ↔ More, default 50).
- **Auto Spin** animates the wheel when the slice frequency changes from elsewhere. **Reverse** reverses the tuning direction of both the virtual and the physical knob.
- **Compact** hides the configuration controls for a smaller window, which also suits short displays. The window is kept within the screen.
- **Settings...** opens the FlexControl Tuning Knob group in Radio Setup.

10.2.5 Reference

Default button actions

Button	Tap	Double
Button 1	StepUp	StepDown
Button 2	ToggleMox	ToggleTune
Button 3	ToggleMute	ToggleLock
Knob Button	StepUp	StepDown

Action list

Radio Setup lists actions by their internal names:

Action	What it does
None	Nothing
StepUp / StepDown	Next larger / smaller tuning step
ToggleMox	MOX on/off
ToggleTune	TUNE on/off
ToggleMute	Mute on/off
ToggleLock	Tune lock on/off
BandZoom / SegmentZoom	Band zoom / segment zoom
NextSlice / PrevSlice	Change the active slice
SplitActiveSlice	Split the active slice
SplitMonitorTx	Monitor the TX frequency
ToggleAgc	Cycle AGC mode

³<https://github.com/aethersdr/AetherSDR/issues/2925>

⁴<https://github.com/aethersdr/AetherSDR/issues/2986>

Action	What it does
ToggleApf	APF on/off
VolumeUp / VolumeDown	Slice audio up / down
ClearRit / ClearXit	Zero the RIT / XIT offset
CwxF1 ... CwxF12	CWX macro 1–12; follows the active slice's mode, so in voice modes it fires the matching DVK slot
WheelFrequency	Knob tunes the slice
WheelVolume	Knob sets Master Volume
WheelSliceAudio	Knob sets the active slice's audio volume
WheelHeadphoneVolume	Knob sets headphone volume
WheelRit / WheelXit	Knob adjusts RIT / XIT
WheelAgcT	Knob adjusts AGC threshold
WheelApf	Knob adjusts APF level
WheelCwSpeed	Knob adjusts keyer speed
WheelPower	Knob adjusts RF power

10.2.6 Known issues

- Pressing the knob always returns it to tuning, even when the knob button is bound to another action such as MOX (#2350⁵).
- On Windows the FlexControl is sometimes not detected until it is unplugged and plugged back in (#4276⁶).
- Volume up/down and the volume wheel start from the saved master volume, not the level of the audio path you are hearing, so the first step can jump (#6132⁷).

10.2.7 Troubleshooting

FlexControl not detected

The knob is not plugged in, or your user cannot open the serial port.

1. Check that it is plugged in: on Linux, `ls /dev/ttyUSB*` should show a device and `lsusb | grep 2192` should list the FlexControl.
2. On Linux your user must be in the serial-port group:

```
sudo usermod -aG uucp $USER      # Arch
sudo usermod -aG dialout $USER   # Debian/Ubuntu
```

Log out and back in for the change to take effect.

3. Click **Detect** in **Settings** → **Radio Setup...** → **Serial & Controllers**.

The knob stopped after a USB glitch

The knob's serial port dropped. AetherSDR notices this and re-detects the knob about every 2 seconds, so it comes back on its own, on Windows possibly on a different COM port.

1. Wait a few seconds for the knob to come back.
2. Turn on the **Ext Devices** log category (see Support and Logging (p. 261)) to see the serial error.

Tuning is too fast or too slow

The knob multiplies the tuning step by its acceleration (1–6×).

1. Change the step size in the RX applet.

⁵<https://github.com/aethersdr/AetherSDR/issues/2350>

⁶<https://github.com/aethersdr/AetherSDR/issues/4276>

⁷<https://github.com/aethersdr/AetherSDR/issues/6132>

Buttons don't work

The button actions may have been set to **None**, or the knob is in a wheel mode.

1. Check the button actions in **Settings** → **Radio Setup...** → **Serial & Controllers** → **FlexControl Tuning Knob**.
2. If the knob adjusts RIT or volume instead of tuning, it is in a wheel mode: press the knob to return it to tuning.

10.2.8 See also

- USB Control Surfaces (p. 189): RC-28, PowerMate, Contour Shuttle, Stream Deck+ and serial keying
- Ulanzi Dial (p. 186)
- MIDI Controller Mapping (p. 192)
- Keyboard Shortcuts (p. 224)

10.3 Ulanzi Dial

The Ulanzi Dial is a small desktop controller with a rotary knob, a push on the knob, three keys along the top and four keys on its sides. AetherSDR reads it directly on Linux, Windows and macOS: the knob tunes (or drives any other wheel action you pick) and every key can run any keyboard-shortcut action or MIDI button action, including press-and-hold transmit and CW keying.

No extra software is needed. The dial does not use the Ulanzi Studio app.

10.3.1 Setup

Open **Settings** → **Radio Setup...** → **Serial & Controllers** and look at the **USB Control Surfaces** group.

Setting	Default	What it does
Use a Ulanzi Dial when detected	On	AetherSDR claims a connected dial and uses it. Turn it off to leave the dial's media keys to the operating system.

The dial is detected before it is claimed, so nothing happens (and no permission prompt appears) until a dial is actually plugged in.

Upgrading from an older version: an install with no saved Ulanzi setting now claims a connected dial automatically. The built-in **MOX** and **TUNE** defaults on the two media-key pills were removed. If you used them, bind them again in the mapper (#5964⁸).

macOS: Input Monitoring

The first time AetherSDR claims a dial, macOS asks for **Input Monitoring** permission. Allow it in **System Settings** → **Privacy & Security** → **Input Monitoring**. The prompt only appears once a dial is attached.

Linux: Grant access

On Linux the dial's input device is normally readable only by root. If AetherSDR finds a dial it cannot open, the mapper shows “... **detected** — **needs permission**” and a **Grant access** button. Click it and approve the administrator prompt: AetherSDR installs a udev rule (`70-ulanzi-dial.rules`, via `pkexec`) that hands the device to the logged-in user, then connects. The offer appears again if a blocked dial is plugged back in.

If `pkexec` is missing, install `polkit`, or add your user to the group that owns the input device.

Windows

Windows cannot take exclusive control of the dial, so each key press also still reaches the operating system (for example as a media key) as well as AetherSDR.

⁸<https://github.com/aethersdr/AetherSDR/issues/5964>

10.3.2 Using the mapper

Open **Settings** → **Ulanzi Dial Mapping...** (“Ulanzi Dial — Control Mapping”). The dialog shows a picture of the dial with a drop-down “pill” on each physical control, plus:

- a status line (**Connected** — ..., **Disconnected**, or **Turned off in Radio Setup** → **Serial & Controllers**);
- a **Last event** readout that names the last key or rotation the dial sent and the action it ran;
- **Reset to Defaults**.

Mappings are saved with your settings and survive a restart; a mapping you clear stays cleared.

Keys and knob press

Each key pill, and the knob press (**Single tap:**), offers:

- **(unassigned)**;
- every keyboard-shortcut action, listed as [Category] Name: the same list as **Settings** → **Configure Shortcuts...** (see Keyboard Shortcuts (p. 224));
- MIDI button actions (toggles, triggers and press-and-hold “gate” actions), listed as [MIDI Category] Name, in builds with MIDI support.

No default keys the transmitter. MOX, TUNE, PTT and CW keying are only ever bound by you.

Press-and-hold transmit and CW keying

Keys bound to **PTT (Hold)**, the straight key, either paddle, or a held MIDI action work press-and-hold: press to key, release to un-key. Releases are honoured even when two keys are held together or arrive out of order, and if the dial is unplugged while it is holding the transmitter, everything it was holding is released immediately (#4611⁹).

10.3.3 Reference

Knob rotation (“Tuning:”)

Option	What the knob does
Frequency (p. 281) Tune Slice	Tunes the active slice (default)
Filter (p. 283) Filter Bandwidth	Steps the filter width; a faster turn moves further
Audio (p. 257) Slice Audio Volume	Active slice’s audio level
Audio (p. 257) Master Volume	Master volume
Audio (p. 257) Headphone Volume	Headphone level
Display (p. 258) Panadapter Zoom	Zooms the panadapter (1.25× per detent)
Display (p. 258) Band Zoom	Band zoom
Display (p. 258) Segment Zoom	Segment zoom
RIT/XIT (p. 284) RIT	RIT offset
RIT/XIT (p. 284) XIT	XIT offset
AGC (p. 66) AGC-T (Threshold)	AGC threshold
AGC (p. 66) RF Gain	RF gain
DSP (p. 62) APF (Audio Peaking Filter)	APF level
CW (p. 282) CW Speed	Keyer speed
TX (p. 194) RF Power	RF power

Zoom and other actions the connected radio cannot perform are refused rather than sent.

Default key actions

Control	Default action
Top Left	(unassigned)

⁹<https://github.com/aethersdr/AetherSDR/issues/4611>

Control	Default action
Top Middle	RIT Toggle
Top Right	(unassigned)
Left Top / Left Bottom	(unassigned)
Right Top	Next Slice
Right Bottom	(unassigned)
Dial Press	Mute Toggle

10.3.4 Known issues

- On macOS, a dial without the Input Monitoring permission shows only **Disconnected**, with no hint that permission is missing (#5247¹⁰).
- A Bluetooth-paired Ulanzi D100H stays **Disconnected** on Windows (#3485¹¹).

10.3.5 Troubleshooting

The dial does nothing

The dial is turned off in Radio Setup, or AetherSDR has not claimed it.

1. Check that **Use a Ulanzi Dial when detected** is ticked in **Settings** → **Radio Setup...** → **Serial & Controllers**.
2. Open **Settings** → **Ulanzi Dial Mapping...** and turn the knob: the **Last event** line should change.
3. Turn on the **Ext Devices** log category (see Support and Logging (p. 261)) to see dial detection and connection events.

Linux: the mapper says “needs permission”

Your user cannot open the dial’s input device.

1. Click **Grant access** in the mapper.
2. Approve the administrator prompt.
3. If `pkexec` is missing, install `polkit`, or add your user to the group that owns the input device.

macOS: the dial does not respond

AetherSDR does not have the Input Monitoring permission. macOS applies a new grant only after AetherSDR quits.

1. Open **System Settings** → **Privacy & Security** → **Input Monitoring** and allow AetherSDR.
2. Quit and reopen AetherSDR.

macOS: the trackpad or keyboard misbehaves

An older release grabbed other input devices as well as the dial (#5147¹²). Current releases claim only the dial.

1. Update to a current release.

10.3.6 See also

- USB Control Surfaces (p. 189): RC-28, PowerMate, Contour Shuttle, Stream Deck+ and serial keying interfaces
- FlexControl Tuning Knob (p. 183)
- MIDI Controller Mapping (p. 192)
- Keyboard Shortcuts (p. 224)

¹⁰<https://github.com/aethersdr/AetherSDR/issues/5247>

¹¹<https://github.com/aethersdr/AetherSDR/issues/3485>

¹²<https://github.com/aethersdr/AetherSDR/issues/5147>

10.4 USB Control Surfaces

AetherSDR works with hardware knobs, jog wheels, button decks and keying interfaces connected to the computer. Most of them are set up on one Radio Setup page, **Settings** → **Radio Setup...** → **Serial & Controllers**, which you can also reach from **Settings** → **FlexControl Knob & Buttons...**

These are host devices: they plug into your computer, not the radio, so the page and its controllers stay available on every radio family. Actions the connected radio cannot perform (for example band zoom on some radios) are refused with a notice rather than sent.

10.4.1 Requirements

- A build with Qt serial-port support: the **Serial & Controllers** page is only present in those builds.
- For the RC-28, PowerMate, Contour Shuttle, TMate 2 and Stream Deck+: a build with HID support (hidapi).

Tested devices

Device	Connection	Where it is set up
FlexRadio FlexControl	USB serial	FlexControl Tuning Knob (p. 183)
Icom RC-28 remote encoder	USB HID	Settings → Icom RC-28 Remote Encoder...
Griffin PowerMate	USB HID	Serial & Controllers → HID encoder groups
Contour ShuttleXpress / ShuttlePro v2	USB HID	Serial & Controllers → HID encoder groups
Elgato Stream Deck+	USB HID	StreamDeck (p. 181)
Ulanzi Dial	USB input device	Ulanzi Dial (p. 186)
MIDI controllers	USB MIDI	MIDI Controller Mapping (p. 192)
USB-serial PTT/CW interfaces (foot switches, straight keys, paddles, amplifier keying, sequencers)	USB serial	Serial & Controllers → Port Configuration / Pin Assignment

Other Stream Deck models and other control-surface software can drive AetherSDR through the TCI Server (p. 135) or the Automation Bridge and MCP (p. 144).

10.4.2 Setup

The **USB Control Surfaces** group at the top of **Serial & Controllers** has the switches:

Setting	Default	Notes
Use a Ulanzi Dial when detected	On	See Ulanzi Dial (p. 186).
Enable HID encoders / StreamDeck+ (RC-28, PowerMate, ShuttleXpress, ...)	Off	Turns on the RC-28, PowerMate, Contour Shuttle, TMate 2 and Stream Deck+. Only present in builds with HID support (hidapi).

HID encoders are off by default because, on macOS, the first time AetherSDR opens one the system asks for **Input Monitoring** permission. Leaving the switch off means that prompt never appears unless you own one of these devices. Allow it in **System Settings** → **Privacy & Security** → **Input Monitoring**.

Linux permissions

On Linux, HID devices are root-only until a udev rule grants access. AetherSDR ships one in the source tree at `packaging/linux/60-hid-encoders.rules` (RC-28, PowerMate, ShuttleXpress, ShuttlePro v2 and Stream Deck+):

```
sudo cp 60-hid-encoders.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Then unplug and replug the device.

10.4.3 Using encoder and button actions

Below the FlexControl group, Serial & Controllers has three groups shared by the HID devices:

- **HID Encoder / StreamDeck+ Encoders:** what each dial does when turned. Single-encoder devices (RC-28, PowerMate, ShuttleXpress) use **Encoder 1** only.
- **StreamDeck+ Encoder Push Actions:** what each dial does when pressed.
- **StreamDeck+ LCD Button Actions:** actions for **Key 1–8**.

The full action lists are under Reference (p. 191).

10.4.4 Using the Icom RC-28

The RC-28 is a weighted tuning knob with **F1**, **F2** and a **TX** bar. A community Arduino emulator (aether-pad) that speaks the same protocol is also recognised.

1. Tick **Enable HID encoders / StreamDeck+** (see Setup (p. 189)).
2. Open **Settings** → **Icom RC-28 Remote Encoder...** (“Icom RC-28 Button Mapping”).

The dialog has:

Group	Controls
Device	Connection status, VID:PID, path, serial and firmware
Transmit (PTT) Button	Momentary (hold to transmit, release to stop) or Latched (press to start, press again to stop)
Button Assignments	F1/F2 — short press and F1/F2 — long press
Encoder	Invert encoder direction ; Pulses per step (1–10, higher is less sensitive); Auto-snap to nearest 1 kHz after rotation stops
Activity	A log of recent button and encoder events

Defaults: F1 short = Step Up, F1 long = Fast Tuning; F2 short = Step Down, F2 long = Mode.

- **Short-press actions:** AGC, APF, Band, Band Zoom, Mode, Next/Previous Slice, Segment Zoom, Snap to Nearest 100 Hz / 500 Hz / 1 kHz / 100 kHz / 500 kHz, Step Up/Down, Volume Up/Down (±5).
- **Long-press actions:** Fine Tuning, Fast Tuning, Mode, Mute, RIT, Slice Lock, XIT.

Tuning speed follows how fast you turn the knob. Auto-snap rounds to the nearest 1 kHz about 600 ms after the knob stops. The TX bar is always PTT, and the RC-28’s TX LED follows the real transmit state.

10.4.5 Using the Griffin PowerMate and Contour Shuttle

The PowerMate (one knob with a push) and the Contour ShuttleXpress (5 buttons) and ShuttlePro v2 (15 buttons) work as single-encoder devices. Enable HID encoders, then choose the knob or jog wheel’s action as **Encoder 1**. Their buttons are numbered from 1 and use the **Key 1–8** slots in **StreamDeck+ LCD Button Actions**; the PowerMate’s push is Key 1.

10.4.6 Using the TMate 2

The TMate 2 (three encoders, six function keys and an LCD) has its own groups on the same page: **TMate 2 Key Actions** (F1–F6), **TMate 2 Encoder Actions**, **TMate 2 Encoder Push Actions** and **TMate 2 Display**. Its LCD shows the frequency, mode, S-meter and transmit state.

10.4.7 Using serial PTT, CW keys, paddles and foot switches

A USB-serial adapter can carry PTT and CW in both directions using its handshake pins. Set it up under **Settings** → **Radio Setup...** → **Serial & Controllers**:

Port Configuration: Port (with a custom **Path:** option), Baud, Data, Parity and Stop, plus **Open/Close**.

Pin Assignment:

Pin	Direction	Functions
DTR, RTS	Output from AetherSDR	None, PTT, CW Key, CW PTT: for keying an amplifier, sequencer or interface
CTS, DSR, DCD	Input to AetherSDR	None, PTT Input, CW Key Input, CW Dit Input, CW Dah Input: for a foot switch, straight key or paddles

Every pin has a **Polarity** (Active High / Active Low). **Paddle Swap (swap dit/dah)** reverses the paddles, and **Auto-open serial port on startup** reopens the port at launch.

The DCD input suits interfaces such as the HaliKey Serial, whose ring contact is wired to DSR and DCD. Iambic **Mode B** keeps its trailing element when both paddles are released together.

10.4.8 Reference

Encoder actions

Encoder action	What the dial does
Tune Slice	Tunes the active slice (Encoder 1 default)
RIT / XIT	RIT or XIT offset (Encoder 2 / 3 defaults)
Master Volume	Master volume (Encoder 4 default)
Slice Audio Volume	Active slice's audio level
Headphone Volume	Headphone level
AGC Threshold	AGC-T
APF Level	APF level
CW Speed	Keyer speed
RF Power	RF power
None	Nothing

Push and key actions

Push actions: Cycle Tuning Step, Toggle RIT on/off, Toggle XIT on/off, Toggle TX (MOX), Toggle Mute, Lock Slice, None.

Key actions: Toggle TX (MOX), Toggle Tune, Toggle RIT/XIT on/off, Clear RIT/XIT offset, Step Size Up/Down, Toggle Mute, Toggle Slice Lock, Toggle APF, Cycle AGC Mode, Toggle Band Zoom, Toggle Segment Zoom, Next/Previous Slice, Volume Up/Down (± 5), Toggle Split, Monitor TX Frequency.

10.4.9 Known issues

- On Windows, the RC-28 can take several seconds to move the tuned frequency, then jump (#6225¹³).
- After an upgrade, a Griffin PowerMate can stop working until **Enable HID encoders** / **StreamDeck+** is ticked (#3357¹⁴).
- Volume up/down and volume-wheel actions start from the saved master volume, not the level of the audio path you are hearing, so the first step can jump (#6132¹⁵).
- A USB-serial PTT switch can stop working after a while until the port is closed and opened again (#5124¹⁶).

10.4.10 Troubleshooting

An RC-28, PowerMate, Shuttle or Stream Deck+ does nothing

HID encoders are off by default.

1. Open **Settings** → **Radio Setup...** → **Serial & Controllers**.

¹³<https://github.com/aethersdr/AetherSDR/issues/6225>

¹⁴<https://github.com/aethersdr/AetherSDR/issues/3357>

¹⁵<https://github.com/aethersdr/AetherSDR/issues/6132>

¹⁶<https://github.com/aethersdr/AetherSDR/issues/5124>

2. Tick **Enable HID encoders / StreamDeck+ (RC-28, PowerMate, ShuttleXpress, ...)**. If the checkbox is missing, the build has no HID support.
3. Turn on the **Ext Devices** log category (see Support and Logging (p. 261)) to see HID connection events.

Linux: a HID device is not found

HID devices are root-only until a udev rule grants access.

1. Install `packaging/linux/60-hid-encoders.rules` as shown under Linux permissions (p. 189).
2. Unplug and replug the device.

macOS: a HID device does not respond

AetherSDR does not have the Input Monitoring permission.

1. Open **System Settings → Privacy & Security → Input Monitoring** and allow AetherSDR.

The Serial & Controllers page is missing

The build was made without Qt serial-port support. Use a build that includes it.

10.4.11 See also

- FlexControl Tuning Knob (p. 183)
- Ulanzi Dial (p. 186)
- StreamDeck (p. 181)
- MIDI Controller Mapping (p. 192)
- Keyboard Shortcuts (p. 224): the same actions from the keyboard
- CWX Panel (p. 102): CW keyer and macros

10.5 MIDI Controller Mapping

AetherSDR supports class-compliant USB MIDI controllers for hands-on control of the radio. Any standard controller — Behringer X-Touch Mini, Korg nanoKONTROL, Akai MIDImix, a DJ controller, or anything else that sends MIDI CC, Note or Pitch Bend messages — can map its knobs, faders, encoders and buttons to AetherSDR controls.

10.5.1 Requirements

- A class-compliant USB MIDI controller (no special drivers needed)
- MIDI support is built into AetherSDR on all platforms (RtMidi is bundled)

10.5.2 Setup

1. Plug in your MIDI controller.
2. Open **Settings → MIDI Mapping...**
3. Click **Refresh**, pick your controller from the **Port** list and click **Connect** — the status turns green.
4. Move a knob or fader — the activity line shows the MIDI message.
5. Choose a parameter (for example “AF Gain”).
6. Click **Learn**, then move the control you want to assign.
7. The binding appears in the table — that control now drives AF Gain.

10.5.3 Using the mapping dialog

MIDI Device

- **Port** with **Refresh** and **Connect / Disconnect**
- **Auto-connect on startup** — reconnect to the last device at launch
- **Activity line** — the last message received (channel, type, number, value)

Parameter Bindings

- Binding table: **Parameter**, **MIDI Source**, **Channel**, **Invert**, **Relative**, an edit (✎) button and a delete (×) button
- Category filter: All, RX, TX, Phone/CW, EQ, Global, Mode, Band, Filter, Slice, Display, Frequency
- **Learn** / **Manual...** to add a binding, **Clear All** to remove every binding
- **Profile** row: **Save**, **Load**, **Import...**, **Export...**

10.5.4 Adding bindings

MIDI Learn

1. Choose a parameter (use the category filter to narrow the list).
2. Click **Learn** — the button changes to **Cancel Learn**.
3. Move a knob, press a button or slide a fader.
4. AetherSDR captures the message and creates the binding. Bindings are saved automatically.

Manual entry

Click **Manual...** to type a binding instead of learning it: **Channel** (Any or 1–16), **Type** (Note On, Control Change (CC) or Pitch Bend), **Number** (note or CC number), plus **Invert value range** and **Relative (knob sends deltas)**.

The ✎ button on any row opens the same form filled in, so you can fix a binding that Learn captured wrongly without re-learning it. If two bindings share the same source, AetherSDR warns you — only the last one would work.

10.5.5 Using relative encoders

Endless encoders send small “turned up / turned down” deltas instead of an absolute position. Tick **Relative** on the binding. AetherSDR recognises both common relative encodings (two’s-complement 1/127 and centre-64 65/63) from the first detents and re-checks if the values stop making sense, so a mis-detected knob corrects itself. Fast turns are smoothed so a burst of deltas does not overshoot.

The **VFO Tune Knob** parameter is meant for an endless encoder bound as **Relative**; as an absolute control it treats 64 as centre.

10.5.6 Using profiles

Save your bindings as named profiles to switch between controllers or layouts.

- **Save** — type a name in the **Profile** box and click **Save**. The dialog reports whether it saved or overwrote a profile.
- **Load** — pick a profile and click **Load** to apply it to the current bindings.
- **Import...** — load a profile file into your profile list. It accepts AetherSDR profile XML or a **SmartSDR .map file** (the per-controller maps vendors publish for SmartSDR for iOS/Mac); the format is detected from the content, and SmartSDR band and mode selections are translated. A summary lists what was imported and anything skipped — controls with no AetherSDR equivalent, invalid values and duplicates — so nothing is dropped silently. A name that already exists gets a “ (2) ” suffix instead of being overwritten. Click **Load** to apply the imported profile.
- **Export...** — write the current bindings to an AetherSDR profile XML (suggested name `AetherSDR_MidiProfile_<date>_<time>_v<version>.xml`) that **Import** reads back.

10.5.7 Using MIDI with non-Flex radios

On Hermes-Lite 2, Icom, ANAN-G2 and RTL-SDR, bindings for features the radio does not have (for example CWL, NR/ANF on, split or CWX) refuse with a warning and a once-per-session notice instead of silently doing nothing. On these radios **NR Cycle** goes straight to NR2.

10.5.8 Reference

Supported MIDI messages

Message Type	Use	Example
Control Change (CC)	Knobs, faders, encoders	CC #7 → AF Gain (0–127 maps to 0–200)
Note On / Off	Buttons, pads	Note C3 → MOX
Pitch Bend	High-resolution controls	14-bit (0–16383)

Value mapping

- **Sliders:** CC value 0–127 maps linearly to the parameter’s range.
- **Toggles:** CC > 63 = on, ≤ 63 = off. Note On toggles the current state.
- **Triggers:** fire on Note On or CC > 63 (one shot, no repeat while held).
- **Gates:** on while the note or CC is held, off when released.
- **Pitch Bend:** 14-bit value (0–16383) for high-resolution control.
- **Invert** reverses the range (for faders mounted upside-down).

Mappable parameters

RX

Parameter	Type	Range
AF Gain	Slider	0–200
Squelch Level	Slider	0–100
AGC Threshold	Slider	0–100
Audio Pan	Slider	0–100
VFO Tune Knob	Slider	0–127 (relative, centre 64)
Noise Blanker, Adaptive RX Filter, Noise Reduction, Auto Notch	Toggle	on/off
NR2 (Spectral), RN2 (RNNoise), NR4 (Spectral Bleach), DFNR (DeepFilter)	Toggle	on/off
Squelch Enable, Audio Mute, Tune Lock, Center Lock	Toggle	on/off
RIT Enable, XIT Enable	Toggle	on/off
Step Size Up, Step Size Down	Trigger	momentary
NR Cycle, AGC Cycle	Trigger	momentary

TX

Parameter	Type	Range
RF Power	Slider	0–100
Tune Power	Slider	0–100
MOX	Toggle	on/off
TUNE	Toggle	on/off
ATU Start	Trigger	momentary
Two-Tone Tune	Trigger	momentary

Phone/CW

Parameter	Type	Range
Mic Level	Slider	0–100
Monitor Volume	Slider	0–100
VOX Level	Slider	0–100
AM Carrier	Slider	0–100
CW Speed	Slider	5–100 WPM

Parameter	Type	Range
CW Break-In Delay	Slider	0–2000 ms
Speech Processor, DAX, Monitor, VOX Enable	Toggle	on/off
CW Sidetone, CW Iambic, CW Iambic Mode (0=A, 1=B), CW Swap Paddles, CWL Frequency Offset, CW Break-In (QSK)	Toggle	on/off
PTT (hold)	Gate	held = transmit
Trigger straight key, Trigger CW Left Paddle, Trigger CW Right Paddle	Gate	held = key down

Gate parameters follow the button: press to key, release to un-key.

EQ

Parameter	Type	Range
TX EQ Enable, RX EQ Enable	Toggle	on/off
63 Hz ... 8 kHz (8 bands, TX EQ)	Slider	–10 to +10 dB

Global

Parameter	Type	Range
Master Volume	Slider	0–100
Headphone Volume	Slider	0–100
Master Mute, TX Button, TNF Global	Toggle	on/off
QSO Record, QSO Playback	Toggle	on/off
Band Up, Band Down	Trigger	momentary
Mode Up, Mode Down	Trigger	momentary
Next Slice, Previous Slice	Trigger	momentary

Mode, Band, Filter, Slice, Display, Frequency

Category	Parameters (all triggers)
Mode	Mode USB, LSB, CW, CWL, AM, SAM, FM, NFM, DFM, DSTR, DIGU, DIGL, RTTY
Band	Band 160m, 80m, 60m, 40m, 30m, 20m, 17m, 15m, 12m, 10m, 6m, 2m
Filter	Filter Widen, Filter Narrow
Slice	Split Toggle
Display	Band Zoom, Segment Zoom, Panadapter Zoom In, Panadapter Zoom Out, Open Memories
Frequency	Tune Up 1 MHz, Tune Down 1 MHz

Digital-voice modes that need a helper (such as DSTR) are only offered when that helper is available, and **Mode Up** / **Mode Down** skip DSTR while the D-STAR helper is not running.

Settings files

MIDI configuration lives in its own files in the AetherSDR configuration folder (~/.config/AetherSDR/ on Linux), separate from the main AetherSDR.db settings store:

```

midi.settings      – device preferences and active bindings
midi/<name>.xml    – saved profiles

```

10.5.9 Known issues

- A crash or a failed write while bindings are being saved can leave `midi.settings` partly written or empty, losing your live bindings without an error (#5160¹⁷). Export a profile as a backup.

¹⁷<https://github.com/aethersdr/AetherSDR/issues/5160>

10.5.10 Troubleshooting

No MIDI ports shown

The controller is not recognised, or the port list is out of date.

1. Check that your controller is plugged in and recognised (on Linux, `amidi -l` should list it).
2. Click **Refresh** to re-scan. Selections follow the device, not its position in the list, so a refresh does not swap your port.

A knob moves but the parameter doesn't change

The binding is missing or captured the wrong message, the radio is not connected, or the radio does not have that control.

1. Check the binding exists in the table and that the activity line shows the CC number you expect. If not, edit the binding with  or re-learn it.
2. Make sure the radio is connected.
3. On a non-Flex radio, look for a notice saying the radio does not support that control.

Values jump or are reversed

The control is upside down, or it is an endless encoder bound as absolute.

1. Tick **Invert** to reverse the direction.
2. If an endless encoder jumps between extremes, tick **Relative**.

Two bindings share a source and only one works

When two bindings use the same MIDI source, only the last one works. AetherSDR warns you when this happens.

1. Click  on one of the bindings and give it a different source, or delete it with .

10.5.11 See also

- FlexControl Tuning Knob ([p. 183](#))
- USB Control Surfaces ([p. 189](#))
- Ulanzi Dial ([p. 186](#))
- Keyboard Shortcuts ([p. 224](#))
- Settings and Backups ([p. 220](#))

10.6 CTR2 Proxy

Status: Early · **Applies to:** FlexRadio

The **CTR2 Proxy** applet connects a CTR2-Max controller to your FlexRadio through AetherSDR. The CTR2 talks to the radio exactly as it would directly; AetherSDR forwards every byte unchanged in both directions over its own network route to the radio, including a VPN or tailnet route the CTR2 could not reach by itself.

The CTR2 stays an independent radio client. The relay never passes through AetherSDR's own command or transmit paths.

10.6.1 Requirements

- AetherSDR connected to a radio that accepts more than one client, over the LAN or a VPN. The proxy is unavailable on SmartLink connections.
- **Wi-Fi (TCP) mode** works with current CTR2 firmware.
- **USB mode** needs CTR2 firmware with USB mode, which has not been released yet, and a build with USB HID support (hidapi).

10.6.2 Setup

The applet's button is **CTR2**, in the **Integration** category. It is not in the default applet bar; right-click any applet button to open the picker and add it.

The proxy is **off at every launch** and nothing about it is saved: mode, address and device are chosen again each session, and it only starts when you press **Start**.

The **Mode** selector offers **Wi-Fi (TCP)** and **USB**.

Check the mode every time. On builds with USB support, USB is the default and the choice is not saved. USB mode waits on a CTR2 USB firmware release, so with current CTR2 firmware pick **Wi-Fi (TCP)** at every launch.

10.6.3 Using Wi-Fi (TCP) mode

1. On the CTR2, set the radio IP address to **this PC's LAN address**.
2. In the applet, pick the **Listen** address: one of this PC's local IPv4 addresses. The listen port is 4992; it can be changed while stopped.
3. Press **Start**.

Each CTR2 connection gets a fresh connection to the radio. Only one CTR2 connection is active at a time; extra ones are closed and counted as rejected. Wi-Fi mode relays TCP only (no UDP discovery or UDP streams), so features of the CTR2 that rely on UDP may not work this way.

10.6.4 Using USB mode

The host side is complete: the CTR2's traffic travels in USB HID reports, and AetherSDR starts the link, so you only pick the device and press **Start**. It needs CTR2 firmware with USB mode, which has not been released yet.

- The **USB** device list shows candidate HID devices, with known CTR2 boards named and listed first. **Rescan** refreshes the list (and the local addresses).
- **Linux**: a udev rule for the CTR2 ships with the build. If the device cannot be opened, **Start** offers to install the rule (/etc/udev/rules.d/70-aethersdr-ctr2.rules) through pkexec, asking for your administrator password once. The rule only opens CTR2 controllers to whoever is logged in at the computer.

10.6.5 Reference

Status

While running, the applet shows:

- **State**: Stopped, Listening, Connecting, Relaying, Closing or Error.
- The endpoints: the listener, the connected CTR2, and the radio.
- Byte counters in each direction (↑ Radio, ↓ CTR2) with queued bytes, and in USB mode a UDP counter.
- The last error, kept apart from the relayed data.

Stop the proxy to change its settings.

Limits

- It always relays to the radio AetherSDR is connected to, captured at **Start**, and **stops when AetherSDR disconnects or switches radios**.
- It is unavailable on SmartLink connections and on radios without multi-client sessions.
- It is not a VPN or tunnel; the PC must be reachable from the CTR2 and the radio from the PC.

10.6.6 Troubleshooting

Start is unavailable: "Connect AetherSDR to a radio first"

The proxy relays to the radio AetherSDR is connected to, and there is none.

1. Connect AetherSDR to your radio.

2. Press **Start**.

“AetherSDR is connected through SmartLink”

The relay needs a direct LAN or VPN connection to the radio.

1. Disconnect from SmartLink.
2. Connect to the radio over the LAN or a VPN (see Tailscale Remote Access (p. 203)).
3. Press **Start**.

“This radio does not accept another client alongside AetherSDR”

The connected radio has no multi-client sessions, so the CTR2 cannot join as a second client. The proxy cannot be used with this radio.

USB mode is dimmed: “This build has no USB HID support (hidapi)”

The build was made without hidapi.

1. Pick **Wi-Fi (TCP)** mode instead.

The proxy stopped on its own

It stops when AetherSDR disconnects or switches radios.

1. Reconnect AetherSDR to the radio.
2. Check the mode and settings again, then press **Start**.

Settings are locked: “Stop the proxy to change its settings”

The proxy is running.

1. Press **Stop**.
2. Change the mode, address or device, then press **Start**.

10.6.7 See also

- Multi-Flex (p. 211)
- MIDI Controller Mapping (p. 192)
- Tailscale Remote Access (p. 203)
- docs/ctr2-tcp-proxy-design.md¹⁸: Wi-Fi relay and operator controls
- docs/ctr2-usb-relay-design.md¹⁹: USB link format and firmware status

¹⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/ctr2-tcp-proxy-design.md>

¹⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/ctr2-usb-relay-design.md>

Remote Operation

11.1 SmartLink Setup

Status: Supported · **Applies to:** FlexRadio

SmartLink lets you operate your FlexRadio over the internet. AetherSDR supports the full SmartLink sign-in and connection flow, including audio, panadapter and waterfall streaming.

Can't forward ports at the station (CGNAT, Starlink, cellular)? On a FLEX-8000 series or Aurora radio, see Tailscale Remote Access ([p. 203](#)). It needs no port forwarding and runs on Tailscale's free plan.

11.1.1 Requirements

- A FlexRadio registered with SmartLink
- Port forwarding on the router at the radio's location (TCP 4994, UDP 4993), or UPnP

11.1.2 Setup

At the station, using SmartSDR for Windows:

1. Open SmartSDR for Windows → **Settings** → **SmartLink Setup**.
2. Log in with your FlexRadio account.
3. Click **Register** to register the radio with SmartLink.
4. Under **Network Settings**, configure port forwarding:
 - **Forward TCP Port:** 4994 (external) → 4994 (internal)
 - **Forward UDP Port:** 4993 (external) → 4993 (internal)
 - Target: the radio's LAN IP address
5. Click **Test** — both indicators should be green.

11.1.3 Connecting with SmartLink

1. Open the connect panel (**File** → **Connect to Radio...**, or the + on the title bar's radio tabs).
2. Choose **Remote with SmartLink**.
3. In **SmartLink account**, enter your FlexRadio account email and password and click **Sign In**.
4. Your radio appears in the list of remote radios. Select it and click **Connect Remote Radio**.

You stay signed in across restarts (see Security ([p. 200](#))).

On a slow or metered connection, tick **Use low bandwidth mode** under *Connection options for slower links* before connecting, and consider Opus audio compression — see Low Bandwidth Connections ([p. 208](#)).

11.1.4 Certificate pinning

AetherSDR pins each SmartLink radio's TLS certificate on first use, to protect you from a man-in-the-middle.

- The first time you connect to a radio over SmartLink, AetherSDR stores the SHA-256 fingerprint of its certificate.
- On later connections the certificate must match. If it does not, the connection **pauses before authentication** and a warning shows the expected and presented fingerprints with two choices:
 - **Accept new certificate** — only if you know the certificate changed (for example after a firmware update or a radio replacement);
 - **Reject and disconnect** — the default.
- Nothing is sent that would authenticate you until you accept, so an impostor never gets a working session.

Managing pins

Settings → Radio Setup... → SmartLink → Pinned SmartLink Certificates lists each pinned radio with its **Host**, **SHA-256 fingerprint** and the date it was **Pinned**.

- **Forget selected** removes one pin.
- **Forget all** clears every pin after confirmation. Each radio is pinned again, silently, on its next connection.

Forget a pin only when you have deliberately changed a radio's certificate.

11.1.5 Reference

Security

- Sign-in uses FlexRadio's identity provider, and all SmartLink connections use TLS with certificate pinning (above).
- Your password is not stored. The SmartLink **refresh token** is saved in the operating system's keychain, so AetherSDR signs you in automatically on the next launch. Builds without keychain support keep it for the current session only. See Settings and Backups (p. 220).
- Logs redact email addresses, tokens, IP addresses and serial numbers — see Support and Logging (p. 261).

Limitations

- Port forwarding (or working UPnP) is required at the radio's location.
- Profile import/export (**Profiles → Import/Export Profiles...**) is disabled over SmartLink; use a direct LAN connection for it.

11.1.6 Known issues

- **TUNE** may not work on a remote radio over SmartLink while AetherSDR is connected to Genius peripherals (Antenna Genius, PGXL, TGXL) (#6001¹).
- On a remote link, receive audio can drop out briefly several times a minute when packets arrive out of order (#6051²).

11.1.7 Troubleshooting

Before reporting a SmartLink problem, turn on the **SmartLink**, **Audio** and **VITA-49** log categories in **Help → Support & Diagnostics...**

The radio is not listed after you sign in

The radio is not registered with SmartLink, or its port test does not pass.

1. At the station, open SmartSDR for Windows → **Settings → SmartLink Setup**.
2. Check that the radio is registered, and click **Test**: both indicators should be green.

¹<https://github.com/aethersdr/AetherSDR/issues/6001>

²<https://github.com/aethersdr/AetherSDR/issues/6051>

The connection drops after about 10 seconds

Port forwarding at the station is probably not set up correctly.

1. Check that TCP 4994 and UDP 4993 are forwarded to the radio's LAN IP address.
2. Run the SmartLink **Test** in SmartSDR again.

The radio is listed, but you cannot connect

TCP 4994 is probably not forwarded to the radio.

1. Check the TCP 4994 forward on the station's router.

A certificate mismatch warning appears

The radio presented a different certificate from the one AetherSDR pinned. See Certificate pinning ([p. 200](#)).

1. If you did not change anything at the station, choose **Reject and disconnect**.
2. If you know the certificate changed (for example after a firmware update or a radio replacement), choose **Accept new certificate**.

11.1.8 See also

- Tailscale Remote Access ([p. 203](#))
- Low Bandwidth Connections ([p. 208](#))
- Manual Connection ([p. 201](#))
- Support and Logging ([p. 261](#))
- FlexRadio ([p. 20](#))

11.2 Manual Connection

Connect to a radio by its IP address when automatic discovery cannot reach it, such as across a VPN or a routed network.

Use **Connect by IP** for:

- A radio on a different VLAN or subnet
- A radio behind a VPN, including a FLEX-8000/Aurora on your tailnet (see Tailscale Remote Access ([p. 203](#)))
- Remote access via a public IP with port forwarding (without SmartLink)
- A networked Icom, which is always reached this way (see Networked Icom ([p. 28](#)))

11.2.1 Setup

1. Open the **Connect to Radio** window: **File** → **Connect to Radio...**, or the + button beside the title bar's radio tabs → **Connect manually...**
2. Choose the **Connect by IP** card ("Best for VPN or routed station access when you already know the radio IP").

The **Radio IP address** group starts with two rows:

Row	What to enter
Radio type:	The radio family to look for: FlexRadio , Hermes-Lite 2 , ANAN-G2 , Icom (network) or RTL-SDR (USB) . Only this family is probed.
Radio IP:	The IP address or host name, for example 10.0.0.25.

RTL-SDR (USB) appears only in builds that include the RTL-SDR backend. Choosing **Icom (network)** or **ANAN-G2** adds that family's own rows; see Networked Icom ([p. 28](#)) and ANAN-G2 ([p. 32](#)). For every family's status, see Supported Radios ([p. 18](#)).

11.2.2 Using Connect by IP

Connecting to a FlexRadio

1. Set **Radio type:** to **FlexRadio**.
2. Enter the radio's IP address in **Radio IP:**.
3. Click **Connect by IP**. AetherSDR probes the radio on TCP 4992.
4. If the probe fails, the panel shows the reason. **Network Diagnostics**, beside the button, helps you check the path.

If your VPN creates more than one active network adapter, open **Advanced: choose the VPN source path** and pick the **Source path:**. Most users can leave this on Auto.

Port forwarding (FlexRadio remote access)

For FlexRadio remote access without SmartLink, forward these ports on your router:

- **TCP 4992** → radio's LAN IP (command channel)
- **UDP 4991** → radio's LAN IP (VITA-49 audio/spectrum)

Then enter your public IP in **Radio IP:**.

For a networked Icom behind NAT, see the **Icom ports:** row on Networked Icom ([p. 28](#)).

Slower links

Use **low bandwidth mode** and **Enable adaptive frame-rate throttle** are in the same window, under “Connection options for slower links”. See Low Bandwidth Connections ([p. 208](#)).

Auto-reconnect

Connect by IP remembers the radio type you chose and the addresses you have used. With **Connect to last radio on start up** checked, AetherSDR reconnects to the last radio on the next launch. If startup auto-connect gives up, the Connect to Radio window opens again.

11.2.3 Reference

Connect by IP compared with SmartLink

This comparison applies to FlexRadio. Other radio families do not use SmartLink.

Feature	Connect by IP	SmartLink
Authentication	None	FlexRadio account
Port forwarding	TCP 4992 + UDP 4991	TCP 4994 + UDP 4993
TLS encryption	No	Yes
NAT traversal	No (needs a static IP or port forward)	Built-in
FlexRadio account	Not required	Required (SmartSDR+)

See SmartLink Setup ([p. 199](#)).

11.2.4 Troubleshooting

The probe fails

AetherSDR could not reach the radio on TCP 4992. The panel shows the reason.

1. Check that **Radio type:** matches the radio; only that family is probed.
2. Click **Network Diagnostics** beside the button to check the path.
3. For access from outside your network, check that TCP 4992 and UDP 4991 are forwarded to the radio's LAN IP.

The connection fails over a VPN with several network adapters

AetherSDR may be using the wrong adapter to reach the radio.

1. Open **Advanced**: choose the VPN source path.
2. Pick the VPN's adapter in **Source path**: and connect again.

Audio works over a VPN, but the spectrum and waterfall stay empty

Spectrum and waterfall packets are larger than the tunnel allows, so they are dropped while the smaller audio packets get through.

1. Lower **Network MTU**; see Low Bandwidth Connections (p. 208).

11.2.5 See also

- Tailscale Remote Access (p. 203)
- SmartLink Setup (p. 199)
- Low Bandwidth Connections (p. 208)
- First Connection (p. 11)
- Supported Radios (p. 18)

11.3 Tailscale Remote Access

Operate a FLEX-8000 series or Aurora radio from anywhere: a small container installed **inside the radio** joins your private Tailscale network. It works behind carrier-grade NAT (CGNAT), Starlink, cellular home internet or any router whose ports you can't forward, and it needs no SmartLink account.

AetherSDR installs and sets up the container for you from the **Waveforms** window. After that you connect from anywhere with **Connect by IP**, as if the radio were on your desk.

11.3.1 It costs nothing

Tailscale's **Personal plan is free forever**, for non-commercial use (which amateur radio is). It covers up to **6 users** and **unlimited devices**, and includes the **subnet routing** used here to share station accessories. A typical ham station (the radio plus your laptop, phone and tablet) fits easily, and you never need a paid plan for this. The container and AetherSDR are free and open source too.

Sign up with a personal login (Gmail/Google, Apple, a personal Microsoft or GitHub account) to land on the free plan. A login on your own or your employer's domain is treated as a business and starts a business trial instead.

Radios: FLEX-8000 series and Aurora, which can run Docker waveform containers. It was developed and measured on a FLEX-8600 (firmware 4.2.20); the FLEX-8400 and Aurora have not been tested yet. FLEX-6000 series radios can't run containers.

11.3.2 What Tailscale is

Tailscale³ is a private network service built on WireGuard⁴, a modern encrypted VPN protocol. Every device you add (your laptop, your phone, and here your radio) joins one private network called a **tailnet** and gets a fixed address in the 100.x.y.z range. The address stays the same wherever the device is.

Some things to know:

- **No port forwarding.** Devices find each other through Tailscale's coordination servers and then connect **directly**, punching through NAT where they can. When no direct path exists (two strict NATs, for example), traffic goes through Tailscale's **DERP relay servers**. It still works, just with more delay.
- **End-to-end encrypted.** Traffic is encrypted between your devices with WireGuard. A relay passes encrypted packets along and can't read them.

³<https://tailscale.com>

⁴<https://www.wireguard.com>

- **Private by default.** Only devices signed in to *your* tailnet can reach the radio. Nothing is opened to the internet.
- **Free for personal use.** The **Personal** plan is free forever for non-commercial use: up to **6 users, unlimited devices**, and subnet routers (used here to share station devices) all included. Check <https://tailscale.com/pricing> for the current terms.

Tailscale or SmartLink?

	Tailscale (this page)	SmartLink Setup (p. 199)
Port forwarding at the station	Not needed; works behind CGNAT	Required (TCP 4994, UDP 4993) or UPnP
Account	A free Tailscale Personal account	A FlexRadio SmartLink account
Radios	FLEX-8000 series, Aurora	All FlexRadio models with SmartLink
Station accessories (Antenna Genius, PGXL, TGXL)	Reachable remotely	Not reachable
What the remote computer needs	The Tailscale app	Nothing extra

11.3.3 Part 1 — Create a free Tailscale account

Tailscale has no passwords of its own. You sign in with an account you already have. Sign in with a **personal** account and you're on the **free Personal plan** automatically, with nothing to cancel later.

1. Go to <https://tailscale.com> and choose **Get Started** (or **Sign up**).
2. Pick a sign-in method: **Google**, **Microsoft**, **GitHub**, **Apple**, a **passkey**, or another single sign-on provider (Okta, OneLogin, or a custom OpenID Connect provider). Email-and-password sign-up isn't offered. **Use a personal account** (a Gmail, Apple, personal Microsoft or personal GitHub login): Tailscale treats a login on a custom domain, such as `you@yourcall.com` or a work address, as a business and puts it on a trial rather than the free plan.
3. When asked, choose **personal** use. That's the free plan. This creates your tailnet, and you become its owner. Skip any business or trial offers; you don't need them.
4. Tailscale offers to install the app on your first device. Do that on the computer you'll operate from (next part).

The **admin console** at <https://login.tailscale.com/admin> is where you manage the tailnet: the **Machines** page lists every device, and **Settings** → **Keys** creates auth keys.

11.3.4 Part 2 — Put your operating computer on the tailnet

Install Tailscale on every computer, phone or tablet you'll run AetherSDR from:

1. Download it from <https://tailscale.com/download> (Windows, macOS, Linux, iOS and Android are all available).
2. Install it, open it and **log in** with the same account as Part 1.
3. The device appears on **Machines** in the admin console with its own `100.x.y.z` address.

Linux: to reach shared station devices (Part 5), also run `sudo tailscale set --accept-routes` once. Windows, macOS, iOS and Android accept shared routes automatically.

11.3.5 Part 3 — Install the container on the radio

Do this **at the station**, with AetherSDR connected to the radio over its local network.

1. Open **File** → **Waveforms...**
2. Click **Install...** and choose **Remote Access (Tailscale)**.
3. AetherSDR downloads the container image (about 8 MB) and checks its size and SHA-256 against the values built into AetherSDR. Any other file is refused, and nothing is installed.
4. The image uploads to the radio and appears in the installed list as **flex-tailnet-shim**, with a **Configure...** button.

The container starts with the radio from then on, about 20 seconds after power-on.

Without internet on the AetherSDR computer, download `flex-tailnet-shim-<version>.tar.gz` from the AetherSDR GitHub release tagged `flex-tailnet-shim-v<version>`. Then use **Install... → Docker Waveform Image...** and pick that file.

The container can run alongside other waveform containers, such as FreeDV. AetherSDR has its own RADE digital voice, so it doesn't need the FreeDV container (see RADE Digital Voice (p. 113)).

11.3.6 Part 4 — Join the radio to your tailnet

Create an auth key

An **auth key** lets a device join your tailnet without anyone signing in on it. The radio has no screen, so this is how it signs in.

1. In the admin console open **Settings → Keys**, then **Generate auth key**.
2. Leave **Reusable** off: the key is used once.
3. Leave **Ephemeral** off, so the radio stays in your tailnet when it is switched off.
4. Turn on **Pre-approved** if your tailnet requires device approval.
5. Optionally add a **tag** (for example `tag:radio`) if you manage access with tags.
6. Click **Generate key** and copy it. It starts `tskey-auth-` and is shown only once.

Keys expire after at most 90 days, but this one only needs to work once. After the radio has joined, the tailnet remembers it, including across reboots.

Configure the container

1. In **Waveforms**, click **Configure...** on the **flex-tailnet-shim** row. The **Remote Access** window opens.
2. Paste the key into **AUTH KEY**. It is sent to the radio once and kept nowhere: not in AetherSDR, not in its settings, not in its logs.
3. **MACHINE NAME** is how the radio appears in your tailnet. It's filled in from the radio's nickname; change it if you like (lowercase letters, digits and dashes).
4. **WHO MAY CONNECT** (optional, see Security (p. 207)): leave it empty to allow every device in your tailnet, or list tailnet logins and tags, for example `you@example.com, tag:operators`.
5. Click **Join Tailnet**. Joining takes a few seconds, up to about a minute.

When the status reads **On your tailnet**, the **Tailnet** card shows:

Field	Meaning
TAILNET NAME	The radio's name on your tailnet
TAILNET ADDRESS	The <code>100.x.y.z</code> address to connect to. Copy puts it on the clipboard.
REMOTE SESSIONS	Clients connected through the tailnet right now
CONTAINER VERSION	The container's version

Write the tailnet address down: it's what you connect to when you're away.

The admin token

The first successful join returns an **admin token** that AetherSDR stores in your system keychain (Windows Credential Manager, macOS Keychain, or the Secret Service / GNOME Keyring on Linux), keyed to this radio's serial number. From then on the container accepts changes only from a computer that holds it: changing the key, who may connect, device sharing, or signing out.

- The first setup is open to anyone on the radio's local network, the same trust the radio's own API uses there.
- If the keychain can't store the token, the window says so and the token is kept until AetherSDR quits.
- If the token is lost, remove and reinstall the container (Part 7) to start over.

11.3.7 Part 5 — Share station devices (optional)

The container listens on the radio's network for **Antenna Genius**, **Power Genius XL** and **Tuner Genius XL** units and offers them to your tailnet, so their applets work remotely too.

1. In **Remote Access**, the **Station devices** card lists what it found. **Share the devices found here over the tailnet** is on by default.
2. To share other devices on the station network (for example an amplifier with a web page), list their LAN addresses in **OTHER DEVICES**, separated by commas, then click **Save Sharing**.
3. **Approve the routes once** in the admin console: **Machines** → the radio → **Subnets** → **Edit** → **Edit route settings**, tick the routes and **Save**. The window shows **Offered, not yet confirmed** until a remote connection actually travels through a route; after that it shows **Confirmed in use over the tailnet**.
4. Remotely, set each device's applet to its **LAN address** (for example 192.168.1.40), exactly as at home. See Amplifiers (p. 170) and TGXL Tuner Control (p. 173).

Remember --accept-routes on Linux computers (Part 2).

11.3.8 Part 6 — Connect from anywhere

1. Make sure Tailscale is running and logged in on the remote computer.
2. In AetherSDR open **File** → **Connect to Radio...** and choose **Connect by IP** (see Manual Connection (p. 201)).
3. Set **Radio type**: to **FlexRadio**, enter the radio's **tailnet address**, and click **Connect by IP**.

The container passes the session through to the radio: control, audio, panadapter, waterfall, meters, DAX, and the extra connections the radio opens on demand (file transfers, for example). Each remote client gets its own session, and the radio sees an ordinary client.

Audio and bandwidth

- **Opus by default.** A radio reached at a tailnet address gets **Opus** audio compression unless you have chosen otherwise, as SmartLink does under **Auto**. **Radio Setup** shows which setting applies; an explicit choice always wins. See Audio Settings (p. 217).
- **MTU.** The container sets the radio's network MTU to 1200 bytes for its sessions, so VITA-49 packets fit inside the WireGuard tunnel without fragmenting.
- **Measured on a relayed path** (both ends forced through a DERP relay): connection in under a second, a median round trip of 67 ms, and about 3.5 Mbit/s uncompressed or 1.9 Mbit/s with Opus. A direct path is faster. For slow links, see Low Bandwidth Connections (p. 208).

Other clients

The container relays the standard SmartSDR protocol, so other clients that can **connect by IP** from a computer running Tailscale also work through it. SmartSDR for Windows has been used this way. A device that can't run Tailscale itself can't reach the tailnet.

11.3.9 Network Diagnostics

While you're connected over the tailnet, **Network Diagnostics** adds the radio side's view of the link, reported by the container:

- **Overview** → **Latency** card: a second line such as *Tunnel 42 ms, DERP sea*.
- **Connection Details** → **Remote Link (Tailscale)**:
 - **Path:** **Direct** (with the address), **Peer relay**, or **Relayed (DERP region)**. Relayed is slower; allowing outbound UDP on both ends' firewalls often lets Tailscale find a direct path.
 - **On This Path:** how long the current path has held, and how many times it changed.
 - **Radio-Side RTT:** the container's own round trip to your computer.
 - **To / From This Client:** throughput through the tunnel.
 - **Breaks Before Tunnel:** VITA-49 sequence breaks that happened inside the radio.

- **Breaks Added by Tunnel:** breaks between the radio and your computer. These two separate radio-side loss from internet loss.
- **Send Failures:** packets the container couldn't hand to the tunnel.
- **Container:** version, CPU, memory, and the MTU clamp.
- **Trends:** a **Tunnel RTT** trace on Latency & Jitter, and **Added by tunnel** on Packet Loss.

The container answers these requests only on the tailnet, and only with the asking computer's own session.

11.3.10 Part 7 — Change, sign out or remove

- **Change Key:** join again with a new auth key, for example to move the radio to another tailnet. It needs the admin token.
- **Save Access List:** change **WHO MAY CONNECT** without leaving the tailnet.
- **Sign Out of Tailnet:** ends remote sessions, logs the radio out, and deletes its tailnet state and token. You'll need a new auth key to set it up again. If the container can't delete its saved settings, it changes nothing and says so, and AetherSDR keeps the token.
- **Remove the container** from the **Waveforms** list to uninstall it. That also wipes its tailnet state. Then delete the radio from **Machines** in the admin console.

Settings can only be changed **from the radio's local network**. The window shows **Unavailable** when AetherSDR is connected to the radio over the tailnet itself. Remotely you can operate, but not reconfigure, the container.

11.3.11 Security

- **Anyone allowed to connect can operate the radio, including transmitting.** An empty **WHO MAY CONNECT** means everyone in your tailnet. If you share the tailnet (family, club members), list the people or tags who may operate.
- The container checks every connection with Tailscale's identity service (WhoIs) against that list before passing it to the radio.
- On the tailnet the container exposes only the radio's own ports, its forwarded side channels, shared station devices, and the diagnostics report. Its setup interface listens only on the radio's local network.
- The auth key and the admin token travel over the station LAN unencrypted, to the radio's local address. The key is single-use and spent within seconds; the container stores only a hash of the token.
- Over the internet, everything travels inside WireGuard and is encrypted end to end.

11.3.12 Ports and resources

Where	Port	Use
Tailnet	TCP 4992	Radio control (SmartSDR API)
Tailnet	UDP	VITA-49 audio, panadapter, waterfall, meters, DAX (per session)
Tailnet	TCP, as the radio opens them	Side channels such as file transfer
Tailnet	TCP 48993	Network Diagnostics report (GET /v1/session)
Tailnet, via shared routes	UDP 9007 / 9008 / 9010 discovery, device TCP ports	Antenna Genius / PGXL / TGXL
Radio LAN only	TCP 48992	Setup interface used by Remote Access

The container runs on the radio's own processor. On a relayed link it used about a fifth of one core and around 34 MB of memory.

11.3.13 Troubleshooting

- **Remote Access says "Unavailable":** connect to the radio over its local network, not over the tailnet, to change settings.

- **“Doesn’t look like a Tailscale auth key”**: paste the whole key, starting `tskey-auth-`.
- **“Could not join the tailnet”**: the key is used up, expired or revoked. Generate a new one. If your tailnet requires device approval, approve the radio on **Machines**, or use a **Pre-approved** key.
- **“The container rejected this computer’s admin token”**: this computer doesn’t hold the token from the first setup. Use the computer that set it up, or remove and reinstall the container.
- **Can’t connect remotely**:
 - check that Tailscale is running on the remote computer, that the radio shows as connected on **Machines**, and that your login or tag is in **WHO MAY CONNECT**;
 - right after a radio reboot, give the container about 20 seconds.
- **Spectrum or audio stutters**:
 - check **Network Diagnostics** → **Remote Link** for a relayed path;
 - keep Opus on, and see Low Bandwidth Connections (p. 208).
- **Station device applets can’t reach the device**:
 - approve its route in the admin console;
 - on Linux, run `sudo tailscale set --accept-routes`;
 - set the applet to the device’s LAN address.
- Before reporting a problem, turn on the relevant log categories in **Help** → **Support & Diagnostics...** (see Support and Logging (p. 261)).

11.3.14 Licensing

The container (`tools/flex-tailnet-shim` in the AetherSDR repository) is part of AetherSDR and licensed under the GNU GPL v3. It includes Tailscale’s open-source client library and other Go modules under their own BSD, MIT, ISC and Apache-2.0 licenses, listed in the `LICENSES` file inside the image and in `THIRD_PARTY_LICENSES`. The image is built reproducibly, so anyone can rebuild it from the source at its release tag and get the exact bytes AetherSDR checks for.

Tailscale is a trademark of Tailscale Inc. AetherSDR is not affiliated with Tailscale.

11.4 Low Bandwidth Connections

Status: Supported · **Applies to:** FlexRadio

Reduce network traffic and smooth out problems when connecting over metered, congested or bandwidth-limited links such as LTE, satellite, hotel Wi-Fi, SmartLink, or VPN tunnels (ZeroTier, WireGuard, Tailscale).

A normal connection to a FlexRadio carries several UDP streams:

Stream	Format	Approx. Bandwidth
RX Audio	uncompressed stereo, 24 kHz	~1.5 Mbit/s
FFT Spectrum	uint16 bins	~0.5–1.0 Mbit/s
Waterfall Tiles	uint16 bins	~0.3–0.5 Mbit/s
Meters	uint16 pairs	negligible

On a home LAN this is fine. Over a cellular or VPN link the combined 2–3 Mbit/s can cause dropouts, latency and data charges. AetherSDR has several independent controls. Use whichever your link needs.

11.4.1 Using the Low-Bandwidth Options

Audio compression (Opus)

Location: **Settings** → **Radio Setup...** → **Audio** → **Audio Compression (SmartLink / tailnet)**

Setting	Behaviour
Uncompressed (default)	Uncompressed on every connection
Auto	Opus on SmartLink and on a radio reached over a tailnet, uncompressed on the local network
Opus	Opus on every connection

Until you choose, a radio reached over a tailnet gets Opus.

Opus reduces the audio stream from about 1.5 Mbit/s to about 64–128 kbit/s. It applies to both receive audio and PC-microphone transmit audio. On a LAN connection the radio may ignore the request and send uncompressed audio anyway.

When to use: choose **Auto** if you use SmartLink, or **Opus** on any metered or limited link.

Low bandwidth mode

Location: connect panel → *Connection options for slower links* → **Use low bandwidth mode**

The *Connection options for slower links* box appears when you choose **Remote with SmartLink** or **Connect by IP**. With the box ticked, AetherSDR asks the radio for a low-bandwidth session (`client_low_bw_connect`) as it connects, and the radio reduces the FFT, waterfall and meter data it sends.

- It takes effect when you connect. To change it, disconnect and reconnect.
- The choice is remembered for SmartLink and Connect-by-IP connections. Connecting to a FlexRadio found **On This Network** always uses full bandwidth.
- On a Hermes-Lite 2 (p. 23), the same option caps the panadapter span at 96 kHz (about 6 Mbit/s).

AetherSDR exposes this as a manual choice because overlay networks such as ZeroTier, WireGuard and Tailscale can make a radio look local even when the real path is cellular or satellite.

Adaptive frame-rate throttle

Location: connect panel → **Enable adaptive frame-rate throttle** (off by default)

When enabled, AetherSDR automatically lowers the FFT and waterfall frame rate when network quality degrades, reducing the chance of a disconnect on a congested link. It applies to every connection type.

- Toggling it while connected takes effect at the next quality update. To lift a cap that has already been applied, reconnect.
- The **Stream Rates** page of **Tools** → **Network Diagnostics...** charts the frame-rate cap — see Runtime Monitor (p. 266).

Network MTU

Location: **Settings** → **Radio Setup...** → **Network** → **Advanced** → **Network MTU**

Network MTU sets the largest VITA-49 UDP packet the radio sends. AetherSDR's default is **1450 bytes** (the radio default is 1500), which suits most VPN and SD-WAN tunnels. The range is 576–9000.

VITA-49 FFT and waterfall packets are about 1436 bytes at an MTU of 1500. Tunnels add overhead that reduces the effective MTU (see Reference (p. 210)). A packet larger than the tunnel MTU is fragmented (adding latency) or silently dropped (losing spectrum and waterfall). Audio packets are smaller and usually get through — which is why you can hear audio but see no spectrum over a VPN.

AetherSDR sends `client set enforce_network_mtu=1 network_mtu=<value>` to the radio on each connect. It applies to this client's session only and does not permanently change the radio or affect other clients.

VITA-49 RX buffer

Location: **Settings** → **Radio Setup...** → **Network** → **Advanced** → **VITA-49 RX buffer**

The operating-system receive buffer for the radio's data stream. The slider snaps to 256 KB, 512 KB, 1 MB, 2 MB or **4 MB (default)**. A larger buffer absorbs bursts of panadapter and waterfall data so they are not dropped — dropped bursts show up as dips in the network statistics and can trigger the adaptive throttle.

The **granted** label shows what the operating system actually allowed. On Linux the size is capped by `net.core.rmem_max`.

Audio smoothing

Two audio settings in **Settings → Radio Setup... → Audio** help on lossy links even when bandwidth is not the problem:

- **Smooth packet loss** (on by default) fades over lost audio packets instead of producing a click.
- **Audio Buffer** (default 100 ms, range 50–1000 ms) — increase it if audio stutters because of jitter.

See Audio Settings (p. 217).

Reducing panadapter traffic

You can reduce traffic further from the panadapter's **Display** panel (right-click the spectrum):

- **Lower FFT FPS** — PANADAPTER section, **FFT FPS** (5–60, default 25). Try 5–10 on a slow link.
- **Slow the waterfall** — WATERFALL section, **WtrFall Rate**.
- **Zoom in** — a narrower span sends fewer FFT bins.
- **Shrink the waterfall** — if you only need the spectrum trace, a minimal waterfall height reduces tile traffic.

Diagnosing a link

Tools → Network Diagnostics... shows latency, jitter, packet loss by stream, stream rates and audio-buffer health over time, with symptom search. See Runtime Monitor (p. 266).

11.4.2 Reference

Recommended settings by connection type

Connection	Audio Compression	Low bandwidth mode	Adaptive throttle
Home LAN (Ethernet/Wi-Fi)	Uncompressed	—	Off
SmartLink over broadband	Auto	Optional	Optional
SmartLink over LTE / hotel Wi-Fi	Auto	On	On
VPN over broadband	Opus	Optional	Optional
VPN over LTE/cellular	Opus	On	On
VPN over satellite	Opus	On	On

Typical tunnel MTUs

Tunnel Type	Typical Effective MTU
WireGuard	1420 bytes
OpenVPN (UDP)	1400 bytes
Cisco SD-WAN	1400–1450 bytes
ZeroTier	~1400 bytes
Tailscale	1280–1420 bytes

Recommended MTU settings

Connection	MTU Setting
Home LAN	1500 (or leave at 1450)
VPN / SD-WAN	1450 (default)
Satellite / high-overhead tunnels	1300–1400

11.4.3 Known issues

- On a remote link, receive audio can drop out briefly several times a minute when packets arrive out of order, although none are lost (#6051⁵).
- The network-quality indicator in the status bar is averaged, so it can stay green while the link is bad; use **Tools** → **Network Diagnostics...** for live conditions (#2825⁶).

11.4.4 Troubleshooting

Audio works, but the spectrum and waterfall stay empty over a VPN

Spectrum and waterfall packets are larger than the tunnel's MTU, so they are dropped while the smaller audio packets get through.

1. Open **Settings** → **Radio Setup...** → **Network** → **Advanced**.
2. Lower **Network MTU** to suit your tunnel (see Recommended MTU settings (p. 210)).
3. Reconnect; AetherSDR sends the MTU to the radio on each connect.

The VITA-49 RX buffer says it was capped

On Linux, the operating system limits the buffer to `net.core.rmem_max`.

1. Raise `net.core.rmem_max` to use a larger buffer.
2. Check the **granted** label again.

Audio stutters or clicks on a remote link

Network jitter or lost packets are reaching the audio buffer.

1. Raise **Audio Buffer** in **Settings** → **Radio Setup...** → **Audio** a little at a time.
2. Leave **Smooth packet loss** on.
3. Use Opus compression and low bandwidth mode.

Low bandwidth mode has no effect

It takes effect only when you connect, and never on a FlexRadio found **On This Network**.

1. Disconnect.
2. Connect again with **Remote with SmartLink** or **Connect by IP**, with **Use low bandwidth mode** ticked.

11.4.5 See also

- SmartLink Setup (p. 199)
- Tailscale Remote Access (p. 203)
- Manual Connection (p. 201)
- Audio Settings (p. 217)
- Runtime Monitor (p. 266)

11.5 Multi-Flex

Status: Supported · **Applies to:** FlexRadio

Multi-Flex lets AetherSDR operate alongside SmartSDR, Maestro, or another AetherSDR instance, each as an independent client on the same radio.

Multi-Flex is a FlexRadio feature. Other radio families allow one client at a time, so the multiFLEX controls are not offered on them (see Supported Radios (p. 18)).

⁵<https://github.com/aethersdr/AetherSDR/issues/6051>

⁶<https://github.com/aethersdr/AetherSDR/issues/2825>

11.5.1 Requirements

- The radio must have multiFLEX enabled (`mf_enable=1`).
- Enable it in SmartSDR (Settings → Radio Setup → Radio → multiFLEX → Enabled), or in AetherSDR with the **multiFLEX**: toggle on Settings → Radio Setup... → Radio, or the dashboard's **Enabled / Disabled** button.

11.5.2 How Multi-Flex Works

When several clients connect to the same FlexRadio, each gets its own:

- Client handle (assigned by the radio)
- Slice(s) and panadapter
- Independent tuning, mode and filter settings

AetherSDR filters status messages and VITA-49 packets by `client_handle`, so you only see your own slices and panadapter data.

AetherSDR handles this automatically:

- **On connect**: AetherSDR checks whether existing slices belong to another client. If so, it creates its own slice and panadapter.
- **Ownership**: panadapter and slice ownership is taken from the radio's status, not from the order packets arrive in, so joining a radio that already has clients attaches you to the right objects.
- **Joining**: the connection watchdog allows a grace period while a second client joins, so the join does not trigger a false disconnect.
- **Status filtering**: only slice, pan and waterfall updates that match your `client_handle` are processed. Spectrum and waterfall packets from other clients' panadapters are dropped.
- **Shared radios** show "Shared radio on your network via multiFLEX" in the Connect to Radio list.

11.5.3 Using the multiFLEX Dashboard

Settings → multiFLEX... opens the **multiFLEX Dashboard**, a live list of every connected station:

Column	Meaning
LOCAL PTT	A check mark shows which station holds PTT authority
STATION	The client's program and station name
TX ANT	Transmit antenna
TX FREQ (MHz)	Transmit frequency
(button)	Disconnect removes that client from the radio

Below the table:

- **Enabled / Disabled** turns multiFLEX on or off on the radio.
- **Enable** (Local PTT) requests PTT for your station when another client holds it.

A green **multiFLEX** button appears in the title bar while other clients are connected. Hover it for the list of stations; click it to open the dashboard.

11.5.4 Connected Stations

When multiFLEX is **disabled** on the radio and another client is already connected, the **Connected Stations** dialog appears automatically when you connect. It lists the radio and each connected station. Select a station and click **Disconnect Station** to free the radio, or **Cancel**.

11.5.5 Reference

Slice Letter Display

Settings → **Radio Setup...** → **Appearance & Behavior** → **Slice Letter Display** chooses how slice letters appear in badges, faders and applet labels:

- **Global slot index** (A=0, B=1, ...), the default.
- **Radio-assigned letter with global subscript** (A_2), which matches SmartSDR's letters in multi-client sessions and still shows which physical slot you are on.

11.5.6 Troubleshooting

The Connected Stations dialog appears when you connect

multiFLEX is disabled on the radio, and another client is already connected.

1. To take over the radio, select the other station and click **Disconnect Station**.
2. To share the radio instead, click **Cancel**, enable multiFLEX on the radio (see Requirements (p. 212)) and connect again.

You cannot transmit while another station is connected

Another client holds PTT authority. The **LOCAL PTT** column of the dashboard shows which one.

1. Open **Settings** → **multiFLEX...**
2. Click **Enable** (Local PTT) to request PTT for your station.

11.5.7 See also

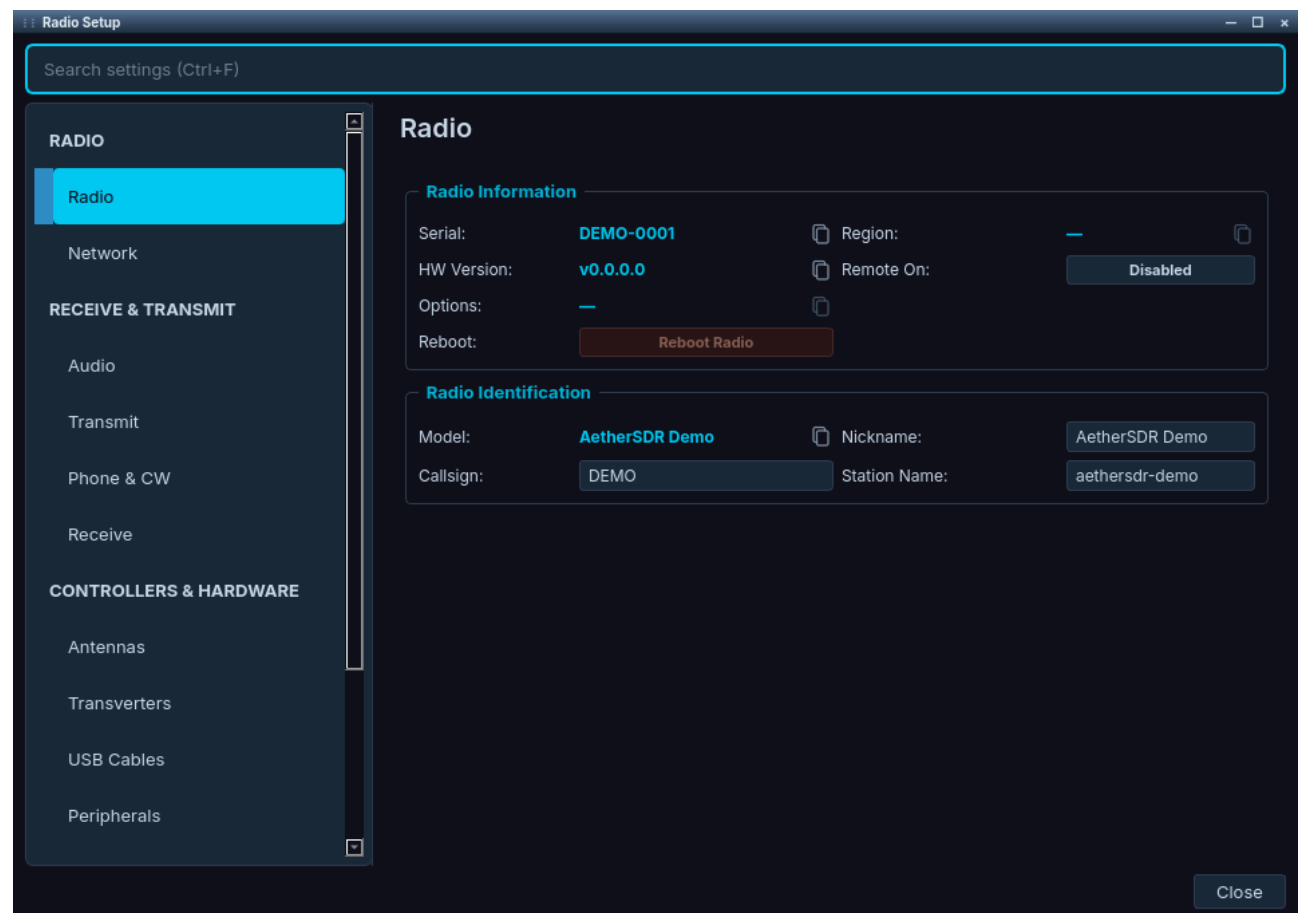
- FlexRadio (p. 20)
- Multi-Slice Operation (p. 83)
- First Connection (p. 11)
- Troubleshooting (p. 256)

Settings

12.1 Radio Setup

Radio Setup holds the radio's configuration and most of AetherSDR's own preferences in one window.

Which pages appear depends on the connected radio: pages for hardware the radio does not have are hidden.



Radio Setup, open on the Radio page. The settings tree on the left groups the pages; the search box at the top finds any setting.

12.1.1 Setup

Open Radio Setup from the menu:

- **Windows and Linux: Settings → Radio Setup...**
- **macOS: AetherSDR → Preferences...**

Several menu items open Radio Setup straight at the right page:

- **Settings → FlexControl Knob & Buttons...** → Serial & Controllers
- **Settings → USB Cables...** → USB Cables
- **Tools → Configure KiwiSDR...** → Antennas

12.1.2 Finding a setting

Radio Setup is a searchable browser: a categorised tree of pages on the left and the selected page on the right.

- **Search settings** (Ctrl+F, ⌘F on macOS) filters the tree as you type. It matches page names and also common symptom words. Try *latency*, *ptt*, *calibration* or *certificate*. Press Enter to jump to the first match.
- Pages for hardware the connected radio does not have are hidden. Individual controls a radio cannot use stay visible but dimmed, with the reason in the tooltip.

12.1.3 Reference

RADIO

Page	What it covers
Radio	Radio information (serial, hardware version, region, options), identification (model, nickname, callsign), license info and Firmware Update (p. 232)
Network	The radio's IP address, mask, MAC and gateway; DHCP/static IP configuration; Advanced : Enforce Private IP Connections, Agent Automation (MCP) (see Automation Bridge and MCP (p. 144)), Network MTU and the VITA-49 RX buffer (see Low Bandwidth Connections (p. 208))
GPS	Whether a GPS is installed, position, grid, altitude, satellites and lock status. The full dashboard is Tools → GPS Dashboard.... See AetherClock and GPS (p. 161). Hidden on radios without GPS hardware.
Calibration	Host-side frequency calibration in ppb, for radios that cannot calibrate themselves (Hermes-Lite 2 (p. 23)). Hidden on FlexRadio, which calibrates on the Receive page.
RTL Receiver	ppm correction, IQ DC suppression and serial for an RTL-SDR (p. 34) dongle. Only when one is connected.
HL2 Hardware	Board variant, dither, companion filter board, CL1 external 10 MHz reference and tuner for the Hermes-Lite 2 (p. 23). Only when an HL2 is connected.
Droop Correction	Panadapter edge-droop correction for the ANAN-G2 (p. 32). Only when one is connected.

RECEIVE & TRANSMIT

Page	What it covers
Audio	The radio's Line Out, Headphone and Front Speaker outputs; SmartLink audio compression; packet-loss smoothing; PC audio devices, Audio Boost and Audio Buffer; recording. See Audio Settings (p. 217).
Transmit	TX timings (ACC TX, TX delay, timeout), TX REQ interlock polarity (RCA, Accessory), Max Power, Show TX in Waterfall, Slice/TX Follow, and TX Band Settings (per-band RF/tune power, PTT inhibit and interlocks). See TX Controls (p. 68).
Phone & CW	Microphone bias and +20 dB boost; CW iambic mode, paddle swap, sideband and CWX sync, and CW decoder RX/TX decode; Digital (RTTY mark default). See CW Decoder (p. 104) and CWX Panel (p. 102).
Receive	Frequency offset (GPSDO detection, manual calibration) and the 10 MHz reference source
Filters	Voice, CW and Digital filter sharpness (Low Latency ↔ Sharp Filters) with Auto buttons, and Use Low Latency Filters for Digital Modes

CONTROLLERS & HARDWARE

Page	What it covers
Antennas	Local display names for antenna ports, and the KiwiSDR RX Antennas list. See KiwiSDR and Web-888 (p. 36).

Page	What it covers
Transverters	One tab per transverter plus + to add one. See XVTR (Transverters) (p. 97).
APD	Adaptive predistortion external sampler per TX antenna. Only on radios that report APD support (FLEX-8000 series).
USB Cables	The radio's USB cables, each with a Cable Type of CAT, Bit, BCD, LDPA or Passthrough. See USB Cable Management (p. 178).
Peripherals	Station devices AetherSDR talks to directly: tuners, amplifiers, antenna switches and wattmeters. See Peripherals (p. 166).
Serial & Controllers	USB control surfaces (Ulanzi Dial, Stream Deck+ and HID encoders), serial port and pin assignment for PTT/CW keying, the FlexControl knob, and TMate 2. See FlexControl Tuning Knob (p. 183), USB Control Surfaces (p. 189) and Ulanzi Dial (p. 186). Present only in builds with serial-port support.

ONLINE & APPEARANCE

Page	What it covers
Appearance & Behavior	Slice Letter Display (see Multi-Flex (p. 211)), slice colours (see Slice Colors (p. 230)), Single-click delay and mouse-wheel behaviour. Themes are under View → Theme . See Themes and Theme Editor (p. 228).
SmartLink	Pinned SmartLink certificates. See SmartLink Setup (p. 199).
QRZ & Callsigns	QRZ.com account and lookup cache. See Callsign Lookup (p. 163).

12.1.4 Known issues

- On macOS, opening the **Audio** page for the first time can send the Preferences window behind the main window (#6277¹).

12.1.5 Troubleshooting

A page you expect is missing

Radio Setup hides pages for hardware the connected radio does not have. For example, **Calibration** is hidden on a FlexRadio, and **HL2 Hardware**, **RTL Receiver** and **Droop Correction** appear only when that radio is connected.

1. Check which radio you are connected to.
2. Use **Search settings** (Ctrl+F, ⌘F on macOS) to find where the setting lives on this radio. For example, a FlexRadio calibrates on the **Receive** page.

A control is greyed out

The connected radio or this build cannot use that control.

1. Hover the control. Its tooltip gives the reason.

You can't find where to connect to a radio

Radio connection is not part of Radio Setup.

1. Use **File** → **Connect to Radio...**, or the + button on the title bar's radio tabs.
2. Tick **Connect to last radio on start up** in the connect panel if you want AetherSDR to connect automatically at launch.

See First Connection (p. 11) and Manual Connection (p. 201).

12.1.6 See also

- Audio Settings (p. 217)
- Settings and Backups (p. 220)

¹<https://github.com/aethersdr/AetherSDR/issues/6277>

- [First Connection](#) (p. 11)
- [Menu Reference](#) (p. 235)

12.2 Audio Settings

Audio settings control the radio's own audio outputs, the computer's speaker and microphone (PC Audio), SmartLink audio compression and QSO recording.

12.2.1 Setup

Audio settings live on the **Audio** page of Radio Setup:

- **Windows and Linux:** **Settings** → **Radio Setup...** → **RECEIVE & TRANSMIT** → **Audio**
- **macOS:** **AetherSDR** → **Preferences...**, then **Audio**

See Radio Setup (p. 214).

12.2.2 Choosing PC audio devices

Input and Output

Choose the computer's microphone (**Input**) and speaker (**Output**) devices. The selection is saved and restored at startup, and you can switch devices while connected.

AetherSDR handles sample-rate conversion itself. The microphone is captured at whatever rate the device offers, processed at 48 kHz, and converted once to the radio's transport rate. The PC microphone is used for transmit when the radio's mic source is **PC** (see Choosing the mic source (p. 218)).

When a device is plugged in or removed

When the operating system adds or removes an audio device, an **Audio Device Detected** dialog offers to switch PC audio to the new device or keep the current one, with a **Don't ask me again** option. It does not appear while your selected device is still available. Turn it on or off with **Prompt on Audio Device Changes** (on by default).

Audio Boost

Audio Boost (Enabled / Disabled, off by default) applies a fixed software gain boost to PC speaker audio, for setups where AGC-controlled audio is quieter than you would like.

Audio Buffer

Audio Buffer sets how much received audio AetherSDR holds before playing it. The default is **100 ms**, adjustable from 50 to 1000 ms. Increase it if audio stutters over a VPN or SmartLink link with a lot of jitter; lower values reduce latency on a clean LAN. The **Audio Health** page in **Tools** → **Network Diagnostics...** shows the buffer level live. See Runtime Monitor (p. 266).

12.2.3 Volume and mute

- **AF slider** (VFO flag or RX applet): the slice's audio level, 0–100.
- **Mute button**: single-click mutes or unmutes this slice; double-click mutes or unmutes all of your slices. **Radio Setup** → **Appearance & Behavior** → **Single-click delay** sets how long AetherSDR waits to tell a single click from a double click (0 makes single clicks instant and turns off the double-click action).
- Slice mute belongs to the radio, so it is not restored by AetherSDR on reconnect.
- **Title-bar speaker icon and master slider** always show the path you are actually hearing. With **PC Audio** on, they show and control the computer's mute and level. With PC Audio off, they show the radio's Line Out mute and level, including changes made by another Multi-Flex client.
- The title-bar headphone controls follow the radio's headphone output, and are dimmed on radios that have none.

12.2.4 Recording QSOs

The **Recording** group controls QSO recording from the record and play buttons on the VFO flag and in AetherRX/AetherTX.

- Client-side recordings include the CW you send, so a QSO that switches between voice and CW stays in one file.
- Each recording is a 24 kHz or 48 kHz WAV, depending on the source, and never overwrites an existing file.
- Client-side recording needs PC Audio. With PC Audio off, AetherSDR refuses with “PC Audio Required for Client-Side Recording” rather than writing an empty file.
- On radios without a radio-side recorder, **Radio Side** is dimmed with the reason, and the record buttons record on this computer instead.

The settings are listed under Recording settings (p. 219).

12.2.5 Choosing the mic source

On a FlexRadio the microphone source is chosen in the P/CW applet. The sources are listed under Mic sources (p. 219).

AetherTX (the client-side transmit processing chain) works only on audio from the computer, so it needs **PC** selected. If another source is selected, the Aetherial Audio applet says so and shows how to fix it. See Aetherial Audio (p. 74).

12.2.6 Reference

Radio audio outputs

These control the physical audio outputs on the radio itself, for speakers or headphones plugged into the radio.

Output	Control	Command
Line Out	Gain slider (0–100) + Mute	mixer lineout gain N/mixer lineout mute 0 1
Headphone	Gain slider (0–100) + Mute	mixer headphone gain N/mixer headphone mute 0 1
Front Speaker	Mute only (models with a built-in speaker, the “M” models)	mixer front_speaker mute 0 1

On radios without one of these outputs, the matching controls are dimmed with the reason in the tooltip.

Audio Compression (SmartLink)

Over SmartLink the radio’s audio can be Opus-compressed to save bandwidth.

Setting	LAN	SmartLink (WAN)
Uncompressed (default)	Uncompressed	Uncompressed
Auto	Uncompressed	Opus
Opus	Opus	Opus

- Opus must be chosen explicitly; the default is Uncompressed.
- The setting affects both receive audio and PC-microphone transmit audio.
- On a LAN connection the radio may ignore an Opus request and send uncompressed audio anyway.
- See Low Bandwidth Connections (p. 208) for when to use it.

Smooth packet loss

Smooth packet loss (conceal dropped audio packets) is **on by default**. When a UDP audio packet from the radio is lost, AetherSDR fades the gap to silence (uncompressed audio) or uses Opus’s own concealment (Opus

audio) instead of splicing the next packet straight in. This removes the high-pitched click you otherwise hear on lossy Wi-Fi, VPN or SmartLink links. Concealment covers up to about 80 ms; a longer gap drops to clean silence.

Prevent system sleep

Prevent system sleep while connected (off by default) stops the computer idling into sleep while a radio is connected, so long sessions keep their network and audio streams alive.

Recording settings

Setting	Meaning
Record Mode	Client Side (default) records on this computer; Radio Side uses the radio's own recorder
Save to	Folder for client-side recordings
Auto-record on TX	Start recording automatically when you transmit
Idle timeout	Stop recording after this many seconds of inactivity (10–3600 s, default 120 s)

Mic sources

Source	Description
MIC	Front-panel microphone jack
BAL	Rear balanced input
LINE	Rear line input
ACC	Accessory connector
PC	The computer microphone selected above, streamed to the radio

12.2.7 Known issues

- A saved PC audio device that is missing when AetherSDR starts, such as Bluetooth headphones that connect later, is forgotten at the next device change, so you have to select it again (#6040²).
- On remote links that deliver packets out of order, received audio has short dropouts because late packets are treated as lost (#6051³).
- Client-side recordings can sound noisier than the audio you hear when client-side noise reduction is on (#6246⁴).

12.2.8 Troubleshooting

No audio from the speaker

The panadapter and waterfall run but nothing comes out of the computer's speakers. PC Audio is off, the wrong output device is selected, or something is muted.

1. Check that **PC Audio** is on in the title bar.
2. Check the **Output** device on the **Audio** page.
3. Check that the slice is not muted, and that the title-bar speaker and master slider are not muted or at zero.
4. Turn on the **Audio** log category in **Help** → **Support & Diagnostics...** and look for device-open errors. See Support and Logging (p. 261).

More causes are listed in Troubleshooting (p. 256).

“PC Audio Required for Client-Side Recording”

Record Mode is **Client Side**, which records the computer's audio, but PC Audio is off.

1. Turn **PC Audio** on in the title bar, or
2. Set **Record Mode** to **Radio Side** on radios that have a radio-side recorder.

²<https://github.com/aethersdr/AetherSDR/issues/6040>

³<https://github.com/aethersdr/AetherSDR/issues/6051>

⁴<https://github.com/aethersdr/AetherSDR/issues/6246>

The Aetherial Audio applet says the mic source is not PC

AetherTX processes only audio from the computer, and the radio's mic source is set to MIC, BAL, LINE or ACC.

1. Select **PC** as the mic source in the P/CW applet.
2. Check that the **Input** device on the **Audio** page is your microphone.

Audio stutters or clicks on a remote link

The link has jitter or packet loss, and the audio buffer runs dry or packets go missing.

1. Raise **Audio Buffer** a little at a time from its 100 ms default.
2. Leave **Smooth packet loss** on.
3. Use Opus compression. See Low Bandwidth Connections (p. 208).
4. Watch **Tools** → **Network Diagnostics...** → **Audio Health** to see whether the buffer still empties.

12.2.9 See also

- Radio Setup (p. 214)
- DAX Virtual Audio (p. 128): audio to and from digital-mode programs
- Aetherial Audio (p. 74): AetherRX and AetherTX processing chains
- Low Bandwidth Connections (p. 208)
- Troubleshooting (p. 256)

12.3 Settings and Backups

AetherSDR keeps its own client-side settings (window layout, applet choices, display preferences, spot sources, controller mappings and so on) in a single SQLite database on your computer. Settings that belong to the radio (profiles, memories, TX settings, antenna choices on a FlexRadio) stay on the radio and are not touched by anything on this page.

This page covers where the database lives, how it is protected, how to inspect or repair it, and how to reset it properly.

12.3.1 Where settings live

The settings store is a file called `AetherSDR.db` in AetherSDR's configuration folder:

Platform	Configuration folder
Linux	<code>~/.config/AetherSDR/</code>
macOS	<code>~/Library/Preferences/AetherSDR/</code>
Windows	<code>%LOCALAPPDATA%\AetherSDR\</code>

The same folder also holds:

Item	What it is
<code>AetherSDR.db</code> (plus <code>-wal</code> / <code>-shm</code> while running)	The settings database
<code>AetherSDR.settings</code>	The old XML settings file, kept as a frozen snapshot after the upgrade (see below)
<code>settings-backups/</code>	Verified automatic backups of the database
<code>settings-quarantine/</code>	Where a damaged database is moved if one is ever found
<code>logs/</code>	Application logs and support bundles. See Support and Logging (p. 261).

To print the exact database path on your machine, run `AetherSDR --config path` (see Using the command line (p. 222)).

MIDI controller mappings are kept in their own `midi.settings` file in the same folder.

12.3.2 Passwords and tokens are kept in the OS keychain

Credentials are **never** written to the settings database. AetherSDR stores them in the operating system's credential store (Keychain on macOS, Credential Manager on Windows, the desktop keyring on Linux):

- SmartLink sign-in (refresh token)
- MQTT broker password
- Agent automation (MCP) access token
- Copy Assist remote-server API key
- KiwiSDR / Web-888 receiver passwords
- Networked Icom user password
- QRZ.com password
- Access codes for 403A peripherals (TGXL, PGXL, Antenna Genius)

The Linux AppImage and the Windows builds include keychain support. On a build without it, these credentials are kept for the current session only and must be re-entered after each restart; nothing falls back to plain text.

12.3.3 How the store protects itself

- **Saves are transactional.** Only the settings you changed are written, in one transaction. A crash mid-save cannot leave a half-written file.
- **Integrity check at startup.** The database is checked every time AetherSDR starts.
- **Verified automatic backups** go into `settings-backups/`. Each backup is checked before it counts:
 - one right after the upgrade from the old XML file (`*-postmigration.db`);
 - at most one a week (`*-auto.db`, the last 5 kept);
 - one immediately before **Reset Settings** removes anything (`*-prereset.db`, the last 3 kept).
- **Automatic recovery.** If the database is ever found to be genuinely corrupt, it is moved to `settings-quarantine/` with a timestamp, the newest verified backup is restored, and AetherSDR shows a notice naming the backup and how old it is. If no usable backup exists, the frozen XML snapshot is imported instead.
- **Permission and lock problems are not treated as corruption.** If the file cannot be opened because of permissions, a lock held by another process, read-only storage or a disk error, AetherSDR leaves the file exactly as it is, reports the problem and refuses to save for that session. A later launch can then retry without anything having been rolled back.
- **Newer stores open read-only.** If you run an older AetherSDR against a database written by a newer version, the older version can read it but will not save to it.

Bookmarks and band-stack entries are saved the moment you make them, so a crash cannot lose them.

Radio-specific state for radios that do not remember it themselves (for example the Hermes-Lite 2 (p. 23)'s per-band drive and gain, nicknames and band stacks) is stored as one document per radio and feature inside the same database.

Upgrading from an older version: releases before the SQLite store kept settings in an XML file, `AetherSDR.settings`. On the first launch of a newer version:

- The XML file is imported automatically and verified key by key, then **left in place, untouched**, as a frozen snapshot of your settings on upgrade day.
- The MQTT password, the automation-bridge token and the Copy Assist remote API key are moved out of the settings and into the OS keychain.
- If you roll back to an older release, it finds your settings as they were on upgrade day. Changes made in the newer version stay with the newer version.
- If an older release then writes to the XML file, the next launch of the newer version shows a one-time notice that those changes were not carried forward. Settings are never merged in either direction.

12.3.4 Using the Settings Browser

Settings → **Settings Browser...** lets you look through and edit the whole store from inside the app.

- The left tree lists the scopes: **App Settings**, the **Station** section, and under **Radios** each radio's stored feature documents.
- **Filter keys and values...** narrows the table as you type (case-insensitive).
- **Add Key...**, **Delete**, **Refresh** and **Export Sanitized...** sit below the table.
- Values stored as `True/False` get a `True/False` picker instead of free text. Feature documents open in a viewer; **Edit Raw JSON...** validates the JSON before saving.
- Credential-shaped values are masked and read-only.
- **Export Sanitized...** writes a secret-redacted dump for diagnosis. It is not a restorable backup.

Edits in the Settings Browser apply immediately and bypass each feature's own validation. Prefer the feature's normal controls when one exists.

12.3.5 Using the command line (**AetherSDR --config**)

AetherSDR --config <command> reads or changes the settings store without starting the GUI. It is the escape hatch when a bad stored value (for example a window placed off-screen) stops AetherSDR from starting normally. The commands are listed under Reference (p. 223).

- Run it from a terminal, using the same program you normally launch, for example `./AetherSDR-*.AppImage --config path` on Linux or `/Applications/AetherSDR.app/Contents/MacOS/AetherSDR --config path` on macOS.
- `set` and `unset` confirm the change by reading it back from the file, and report an error if it did not stick (for example when the store is read-only).
- Credentials cannot be created with `set`. Enter them in the app, which stores them in the keychain.
- An AetherSDR that is already running keeps its own copy of the settings in memory and will not see command-line changes until it is restarted. Quit AetherSDR before editing.
- Advanced: the `AETHER_SETTINGS_DIR` environment variable points AetherSDR (and `--config`) at a different configuration folder.

12.3.6 Resetting settings

Use **Settings** → **Reset Settings...** to return AetherSDR to a first-run state.

1. A confirmation lists exactly which files will be removed and where the backup will be written.
2. A verified `*-prereset.db` backup is written to `settings-backups/`.
3. AetherSDR removes the settings database, the frozen XML snapshot and its leftovers, the NR2 FFTW wisdom cache, the rolling automatic backups and anything in quarantine. The pre-reset backups are kept.
4. AetherSDR then quits, so nothing is written back. Start it again to begin fresh.

Settings stored on the radio are not affected.

Deleting `AetherSDR.db` by hand is **not** a reset. See Troubleshooting (p. 223).

12.3.7 Restoring a backup

The files in `settings-backups/` are complete SQLite databases. To go back to one by hand:

1. Quit AetherSDR.
2. Keep a copy of the current `AetherSDR.db` somewhere safe.
3. Copy the backup you want over `AetherSDR.db`.
4. Delete any `AetherSDR.db-wal` / `AetherSDR.db-shm` files beside it.

12.3.8 Client settings vs radio profiles

The settings store holds **this computer's** preferences. FlexRadio global, TX and microphone profiles, memories and other radio-side configuration are stored on the radio. To back those up or move them between SmartSDR and AetherSDR, use **Profiles → Import/Export Profiles....** See Profile Management (p. 94).

12.3.9 Reference

AetherSDR --config commands:

Command	What it does
list [prefix]	Print every setting as key<TAB>value, optionally only keys starting with prefix
get <key>	Print one value (exit code 2 if the key does not exist)
set <key> <value>	Create or change a setting
unset <key>	Delete a setting
export	Sanitized dump of the whole store, secrets redacted (diagnostic output, not a backup)
features [family]	List radio-scoped feature documents (sanitized)
path	Print the database path

12.3.10 Known issues

- Running two copies of AetherSDR on one computer, which share one settings store, can make the second copy wrongly report that it last closed unexpectedly while popping out a panadapter, and restore its panadapters docked (#4901⁵).

12.3.11 Troubleshooting

Your settings came back after you deleted AetherSDR.db

The next launch found the frozen AetherSDR.settings snapshot and imported it again, restoring your settings as they were on upgrade day.

- For a real factory reset, use **Settings → Reset Settings....**
- To remove a single troublesome value, quit AetherSDR and run `AetherSDR --config unset <key>`.

A stored setting stops AetherSDR from starting

A bad stored value, such as a window placed off-screen, is applied at startup.

- Run `AetherSDR --config list` to find the key.
- Run `AetherSDR --config unset <key>` to remove it, or `AetherSDR --config set <key> <value>` to correct it.
- Start AetherSDR normally.

AetherSDR reports that it cannot save settings

The settings file could not be opened because of permissions, a lock held by another process, read-only storage or a disk error. AetherSDR leaves the file untouched and does not save for that session.

- Quit AetherSDR, and close any other process using the configuration folder.
- Check that the folder and AetherSDR.db are writable by your user.
- Start AetherSDR again. It retries the file on every launch.

An older version of AetherSDR doesn't save your changes

A newer version wrote the settings database, and older versions open newer stores read-only.

- Run the newer version, or
- Restore an older backup from `settings-backups/` as described in Restoring a backup (p. 222).

⁵<https://github.com/aethersdr/AetherSDR/issues/4901>

Passwords must be re-entered after every restart

This build of AetherSDR has no keychain support, so credentials are kept for the current session only.

1. Use the Linux AppImage or the Windows builds, which include keychain support.

Changes made with `--config` don't show up

AetherSDR was running when you edited the store, and a running copy keeps its own settings in memory.

1. Quit AetherSDR.
2. Make the change with `AetherSDR --config` again.
3. Start AetherSDR.

12.3.12 See also

- Support and Logging ([p. 261](#)): logs, the support bundle (which includes a sanitized copy of your settings) and File an Issue
- Profile Management ([p. 94](#))
- Troubleshooting ([p. 256](#))

12.4 Keyboard Shortcuts

AetherSDR has two kinds of keyboard shortcut:

- **Operating shortcuts** drive the radio: tuning, MOX, push-to-talk, AF gain, band and mode changes and so on. They are **off by default**.
- **Application shortcuts** belong to menus and windows (Minimal Mode, UI scale, Full Screen). They always work.

Frequency Band Mode TX CW Audio Slice Filter Tuning DSP AGC EQ Display RIT/XIT

Key: **(none)** Action: **(none)** Clear Reset to Default

Filter: Search actions... Category: All

Action	Category	Current Key	Default Key
Tune Up (1 step)	Frequency	Right	Right
Tune Down (1 step)	Frequency	Left	Left
Tune Up (10 steps)	Frequency	Shift+Right	Shift+Right
Tune Down (10 steps)	Frequency	Shift+Left	Shift+Left
Tune Up 1 MHz	Frequency		
Tune Down 1 MHz	Frequency		
Go to Frequency	Frequency	G	G
160m	Band		

Import... Export... Reset All to Defaults Close

The keyboard shortcuts editor. Bound keys are coloured by category on the keyboard map; the table below lists every action.

12.4.1 Setup

Settings → **Keyboard Shortcuts** (a checkable item) turns operating shortcuts on or off. It is **off by default**: if the keys below do nothing, check this first.

While shortcuts are off, keys go to whatever control has focus, so Space, the arrow keys and other bound keys work normally in sliders, lists and text fields. A bound key never reaches a transmit-keying button by accident. The first time in a session that you press an operating-shortcut key that nothing else uses, the status bar shows **“Keyboard shortcuts are off”** for 10 seconds (also announced to screen readers).

With shortcuts on, AethersDR still backs off while you type in a text field, does nothing to the radio while disconnected, and won’t tune a locked slice.

12.4.2 Using keyboard shortcuts

Changing shortcuts

Settings → **Configure Shortcuts...** opens the editor:

1. Click a key on the on-screen keyboard map.
2. Pick the action for that key. If the key is already in use, AethersDR asks before reassigning it.
3. **Clear** removes the selected key’s action; **Reset to Default** restores one shortcut; **Reset All to Defaults** starts over.

The editor works one key at a time, so it is easiest for single-key bindings. Cleared bindings stay cleared after a restart.

Backing up and copying shortcuts

Export... saves your whole shortcut setup as a CSV file (AetherSDR_Shortcuts_<timestamp>_v<version>.csv); **Import...** applies one, for example on another computer.

- An older release imports the actions it knows and reports newer ones as skipped. Keep the original file: the skipped rows are still there when you go back to a newer release.
- Actions you never customized adopt the importing release's defaults; customized and deliberately cleared bindings come across exactly.
- If an imported shortcut takes a key you had bound to a different action, your local binding is cleared and reported. Use **Reset to Default** to restore it.
- A malformed file is rejected before anything changes.

Minimal Mode

Minimal Mode (Ctrl+Shift+M, or **View** → **Minimal Mode**) shrinks the main window to a narrow strip holding the applet panel. The panadapters are hidden and their rendering paused (floating panadapters stay open), the status bar and menus are hidden, and the title bar keeps only the active radio tab and status badges. Press Ctrl+Shift+M again, or the title bar's maximize button, to return. Minimal Mode is also a bindable action for controllers. The Mini-Pan applet (see Panadapter Controls (p. 54)) is useful here because it shows a narrow spectrum without a full panadapter.

Upgrading from an older version: Minimal Mode was Ctrl+M on Windows. It is now Ctrl+Shift+M everywhere, so Cmd+M is free for Minimize on macOS.

12.4.3 Reference

Default operating shortcuts

Key	Action
Right / Left	Tune up / down 1 step
Shift+Right / Shift+Left	Tune up / down 10 steps
G	Go to frequency entry on the active slice
T	Toggle MOX
Space	Push-to-talk while held
Up / Down	AF gain up / down
M	Mute toggle
] / [	Step size up / down
L	Tune lock toggle

Push-to-talk hold

PTT (Hold) keys the transmitter while its key is held and un-keys when you let go. It follows whatever key you bind it to (Space by default). Releasing the key always un-keys, and the transmitter also un-keys if AetherSDR loses keyboard focus or a modifier is released mid-hold.

Actions you can assign

The editor lists many more actions that ship **unbound**. They are grouped by category:

Category	Examples
Frequency	Tune up/down 1 MHz, Go to Frequency
Tuning	Step size, Tune Lock, Center Lock Active Slice

Category	Examples
Slice	Next / Previous Slice, Cycle TX Slice, Split Toggle, Split Up 1 kHz / 5 kHz / 10 kHz
Bands and modes	Band jumps (160 m to 2 m); USB, LSB, CW, CWL, AM, SAM, FM, NFM, DFM, DIGU, DIGL, RTTY
Audio	Master Volume Up / Down (± 5 , auto-repeat), Master Mute, Mute All Slices, Squelch
AGC	AGC mode cycle, AGC-T up/down, RF Gain up/down
DSP	NB, ANF, NR2, NR4, RN2, DFNr, NR Cycle, TNF global toggle
Filter	Filter Widen / Narrow (± 100 Hz, in the right direction for the mode)
RIT/XIT	RIT and XIT toggles
CW	Speed ± 5 WPM, iambic on/off and A/B, break-in, sidetone, swap paddles, CWL offset
TX	TUNE, Two-Tone Tune, VOX, speech processor, TX monitor, DAX TX
EQ	RX EQ and TX EQ toggles
Display	Minimal Mode, Panadapter Zoom In/Out, Band Zoom, Segment Zoom, Open Memories

Monitor TX (Hold) listens on a split's transmit frequency while held. It is unbound by default; see Split Operation (p. 85).

Band and segment zoom do nothing on radios that can't do them. CWX and DVK F1–F12 keys follow the active slice's mode, so only the matching panel responds (see CWX Panel (p. 102) and DVK Panel (p. 107)). The same actions are available to MIDI, FlexControl and other controllers; see MIDI Controller Mapping (p. 192) and USB Control Surfaces (p. 189).

Application shortcuts

These work whether or not operating shortcuts are on:

Key	Action
Ctrl+Shift+M	Toggle Minimal Mode (every platform)
Ctrl+= / Ctrl+- / Ctrl+0	UI scale up / down / reset to 100 %
Ctrl+F	FPS Meters
Ctrl+Shift+F	Frameless Window
Ctrl+Shift+S	Pop out the applet panel
Ctrl+Shift+L	Callsign Lookup
F11 (Windows/Linux), Cmd+Ctrl+F (macOS)	Full Screen
Cmd+M (macOS)	Minimize; no default elsewhere

The Full Screen and Minimize keys can be rebound in Configure Shortcuts, and they also work in secondary windows and while a text field or slider has focus. On macOS, Ctrl in these shortcuts is the Command key.

12.4.4 Known issues

- **Master Volume Up / Down** steps from the last saved master level rather than the level of the path you are hearing, so after switching PC Audio on or off the first press can jump the volume (#6132⁶).

12.4.5 Troubleshooting

The shortcut keys do nothing

Operating shortcuts are off, which is the default. The status bar says “**Keyboard shortcuts are off**” the first time you press one.

1. Turn on **Settings** → **Keyboard Shortcuts**.
2. Press the key again.

⁶<https://github.com/aethersdr/AetherSDR/issues/6132>

A key works in one place but not another

A text field has focus, so AetherSDR leaves the key to it.

1. Click the panadapter (or anywhere outside the text field) and press the key again.

The tuning keys don't move the slice

The slice is tune-locked, or no radio is connected.

1. Press **L** (Tune lock toggle), or click the  button beside the VFO flag, to unlock the slice.
2. Check that a radio is connected; operating shortcuts do nothing to the radio while disconnected.

An import took one of my keys

The imported file bound that key to a different action, so your local binding was cleared and reported.

1. Open **Settings** → **Configure Shortcuts...**
2. Select the action and choose **Reset to Default**, or bind it to another key.

12.4.6 See also

- Menu Reference (p. 235)
- Accessibility (p. 231)
- MIDI Controller Mapping (p. 192)
- USB Control Surfaces (p. 189)
- Split Operation (p. 85)

12.5 Themes and Theme Editor

Every colour, font and size in AetherSDR comes from the active **theme**. You can switch themes live, with no restart, and create your own in the built-in Theme Editor.

12.5.1 Choosing a theme

View → **Theme** lists every installed theme, with a check mark on the active one:

- **Default Dark**: the standard look, and the default.
- **Default Light**: a light theme for bright rooms.
- Any themes you have created or imported.

The change applies immediately across the whole application.

12.5.2 Using the Theme Editor

View → **Theme Editor...** opens the editor for the active theme. It is a normal window, so you can keep it open while you look at the result. The controls are listed under Reference (p. 229).

Edits are saved as you make them.

Built-in themes are protected

Default Dark and Default Light can't be changed, renamed or deleted. When you edit a built-in theme, the editor asks you to **Save As** a copy (suggested name "My <theme name>") and your changes go there. Deleting the theme you are using switches back to Default Dark.

Sharing theme files

- Exported themes use the `.aethertheme` extension (a JSON file). **Import from file...** accepts `.aethertheme` and `.json`, and you can also drag a theme file onto the editor window.
- Your own themes are stored in a `themes` folder inside the AetherSDR configuration folder (on Linux, `~/.config/AetherSDR/themes/`). A user theme with the same name as a built-in one is ignored, so use **Save As** with a new name.

12.5.3 What a theme controls

Themes cover the whole interface, including:

- **Slice colours:** the default colours for slices A–H (see Slice Colors (p. 230)).
- **Waterfall colour schemes:** the palettes chosen under the panadapter's **Display** → **Scheme** (see Panadapter Controls (p. 54)). Switching theme or scheme recolours the waterfall already on screen, including scrollback.
- The 7- and 14-segment display fonts used for frequency readouts (the DSEG fonts, under the SIL Open Font License).

Analog meter faces have their own **Face theme** choice on each meter (see Meters (p. 72)).

12.5.4 Reference

Control	What it does
Scope:	Which part of the interface you are editing. "(root)" is the whole application; a nested scope (for example one applet) applies your change only inside that container.
◦ Inspect	Click it, then point at any part of the main window to find the tokens that paint it. Esc cancels.
Filter tokens...	Narrows the token list (for example <i>accent</i> , <i>slice</i> , <i>meter</i>).
Token editors	Change colours (with transparency), gradients, fonts and sizes. Changes show live.
Clear override	Puts one token back to the theme's value.
Theme actions	Rename theme..., Delete theme..., Export to file..., Import from file....
Save As...	Saves the current edits as a new theme.

Theme authors can find the token names and their meanings in the repository:

- docs/theming⁷: the token reference
- docs/style/theme-style-guide.md⁸: how the interface uses tokens

Contributors writing new interface code take every colour from theme tokens rather than hard-coded values; see Contributing Guide (p. 269).

12.5.5 Known issues

- The Theme Editor's token list can draw light text on white rows, so tokens such as `font.family.freq` are hard to read (#5934⁹).

12.5.6 Troubleshooting

Import says the theme matches a built-in theme name

The theme file's "name" field is Default Dark or Default Light. Built-in names are reserved, and a user theme with one of them is ignored.

1. Open the theme file in a text editor and change its "name" field to a new name.
2. Import it again with **Theme actions** → **Import from file....**

Your edits to Default Dark or Default Light went into a new theme

Built-in themes are protected, so the editor saved your changes to a copy named "My <theme name>".

1. Pick the copy from **View** → **Theme** to use it.
2. Rename it with **Theme actions** → **Rename theme...** if you like.

12.5.7 See also

- Slice Colors (p. 230)

⁷<https://github.com/aethersdr/AetherSDR/tree/main/docs/theming>

⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/style/theme-style-guide.md>

⁹<https://github.com/aethersdr/AetherSDR/issues/5934>

- Panadapter Controls (p. 54)
- Meters (p. 72)
- Accessibility (p. 231)
- docs/theming¹⁰

12.6 Slice Colors

Each slice (A–H) has its own colour, used for its frequency marker, filter passband and badges on the panadapter, so you can tell slices apart at a glance. You can keep the theme’s colours or choose your own.

12.6.1 Using custom slice colours

Open **Settings** → **Radio Setup...** → **Appearance & Behavior** and find the **Slice Colors** group:

1. Choose **Custom colors** (instead of **Use Aether defaults**).
2. Click a slice button (**A–H**) to open a colour picker, and choose a colour.
3. The change applies immediately everywhere that slice is drawn.

The dimmed version of a custom colour is worked out automatically (about 40 % of its brightness). **Reset All to Defaults** puts every slot back to the theme’s colours. Switching back to **Use Aether defaults** keeps your custom choices stored, so you can switch between the two.

Custom colours are saved in the settings database (AetherSDR.db) under `SliceColorsUseCustom` and `SliceColor0–SliceColor7`. See **Settings and Backups** (p. 220).

12.6.2 Reference

Default colours

By default the slice colours come from the active theme (see **Themes and Theme Editor** (p. 228)). In **Default Dark**:

Slice	Colour
A	Cyan
B	Magenta
C	Green
D	Yellow
E	Orange
F	Teal
G	Pink
H	Purple

Default Light uses darker shades of the same set so they stay readable on a light background. Each colour also has a dimmed variant for inactive slices.

Where the colour shows up

The slice colour is used for the slice marker and filter band on the spectrum, the slice’s VFO flag and badges, and the slice indicators in the RX applet.

Theme authors

The defaults are the theme tokens `color.slice.a ... color.slice.h` and `color.slice.dim.a ... color.slice.dim.h`, so a custom theme can supply its own palette. Custom colours chosen in **Radio Setup** override the theme.

¹⁰<https://github.com/aethersdr/AetherSDR/tree/main/docs/theming>

12.6.3 Troubleshooting

Changing the theme doesn't change the slice colours

Custom colours chosen in Radio Setup override the theme.

1. Open **Settings** → **Radio Setup...** → **Appearance & Behavior**.
2. In **Slice Colors**, choose **Use Aether defaults**. Your custom choices stay stored, so you can switch back later.
3. To clear your custom choices instead, click **Reset All to Defaults**.

12.6.4 See also

- Themes and Theme Editor ([p. 228](#))
- VFO Widget ([p. 61](#))
- Multi-Slice Operation ([p. 83](#))
- Settings and Backups ([p. 220](#))

12.7 Accessibility

AetherSDR is built to be usable with a screen reader and from the keyboard. This page summarises what works today and how to get help when something doesn't.

12.7.1 Requirements

AetherSDR uses the platform accessibility interfaces, so it works with the screen reader you already use:

Platform	Screen readers
macOS	VoiceOver
Windows	NVDA, Narrator, JAWS
Linux	Orca

12.7.2 Using a screen reader

What you can expect:

- **Every interactive control has a name** (and, where useful, a description), so a screen reader announces “Master volume” or “Dock applet panel left” rather than just “button”.
- **Live values are announced.** The S-meter and the VFO frequency announce their values as they change. Announcements wait for a value to settle, so tuning across a band doesn't flood you with interim readings.
- **Custom-drawn meters report their values.** Meters and indicators that AetherSDR paints itself expose their current reading to the screen reader.
- **Status is given in words, not colour alone.** For example, each radio tab in the title bar states its connection state (“link lost” and so on) as text as well as with the coloured dot.
- **Unavailable controls say why.** A control your radio or build can't use is dimmed rather than hidden, and the reason is announced along with it (for example, a headphone control on a radio with no headphone output).
- **Notices are announced.** For example, pressing a shortcut key while keyboard shortcuts are off gives a “Keyboard shortcuts are off” notice that is announced as well as shown.

12.7.3 Using the keyboard

- You can reach controls with Tab and Shift+Tab, including the tabs on the VFO flag (Audio, DSP, mode, X/RIT, DAX).
- The title bar's radio list (“+”) is keyboard reachable, including its search box and per-radio actions.
- Operating shortcuts (tuning, MOX, PTT hold, band and mode changes) are off by default. Turn them on with **Settings** → **Keyboard Shortcuts**; see Keyboard Shortcuts ([p. 224](#)). With them off, keys go to the focused control.

- On Windows and Linux, Alt+letter opens each menu from the title bar's ≡ button. See Menu Reference (p. 235).

12.7.4 Reduced motion

Animated elements, such as those in the About window, hold still when your operating system asks for reduced motion (macOS *Reduce motion*, Windows animation effects, and the GNOME/KDE equivalents).

12.7.5 Reporting a problem

Please report accessibility gaps. If something is silent, mislabelled or unreachable from the keyboard, use **Help** → **File an Issue...** and say which screen reader and platform you use. See Support and Logging (p. 261). Ongoing screen-reader work, including JAWS, is tracked in issue #4896¹¹.

12.7.6 Known issues

- The WAVE waveform scope announces only its name, not its peak, RMS or clip readout (#3959¹²).

12.7.7 Troubleshooting

Space, the arrow keys or other operating shortcuts do nothing

Operating keyboard shortcuts are off by default, so keys go to the focused control instead.

1. Open **Settings** → **Keyboard Shortcuts** and turn them on.
2. See Keyboard Shortcuts (p. 224) for the full list.

Animations still move

Your operating system is not asking for reduced motion.

1. Turn on macOS *Reduce motion*, turn off Windows animation effects, or turn on the GNOME/KDE equivalent.

12.7.8 See also

- Keyboard Shortcuts (p. 224)
- Menu Reference (p. 235)
- Contributing Guide (p. 269)
- docs/a11y.md¹³: accessibility patterns for contributors (accessible names, live value announcements, announced reasons for disabled controls, contrast)

12.8 Firmware Update

Status: Experimental · **Applies to:** FlexRadio

AetherSDR can upload new firmware to a FlexRadio on Linux, macOS and Windows, using the SmartSDR installer you download from FlexRadio.

Caution: firmware update from AetherSDR is a highly experimental feature. Use it at your own risk. Updating with the SmartSDR for Windows application is still the recommended route.

12.8.1 Requirements

- A connected FlexRadio. Firmware update is a FlexRadio feature: the group is hidden while you are connected to any other radio family (see Supported Radios (p. 18)).
- The SmartSDR installer from flexradio.com¹⁴, or an extracted `.ssdr` firmware file.

¹¹<https://github.com/aethersdr/AetherSDR/issues/4896>

¹²<https://github.com/aethersdr/AetherSDR/issues/3959>

¹³<https://github.com/aethersdr/AetherSDR/blob/main/docs/a11y.md>

¹⁴<https://www.flexradio.com>

12.8.2 Setup

Open **Settings** → **Radio Setup...** (on macOS: **AetherSDR** → **Preferences...**), choose the **Radio** page, and find the **Firmware Update** group. It shows the radio's current **FW Version** and three buttons: **Check for Update**, **Select Installer...** and **Upload Firmware**.

12.8.3 Updating the Firmware

Check for updates

1. Click **Check for Update**.
2. AetherSDR asks the FlexRadio website for the latest version and compares it with the radio's firmware.
3. The status line reads either "Firmware is up to date (vX.Y.Z)" or "Update available: vX.Y.Z", with a reminder to download the SmartSDR installer from flexradio.com.

AetherSDR does not download the installer for you.

Stage the firmware

1. Download the SmartSDR installer from flexradio.com¹⁵.
2. Click **Select Installer...** and choose one of:
 - the **.msi** installer (SmartSDR v4.2 and later);
 - the **.exe** installer (v4.1.x and earlier);
 - an already extracted **.ssdr** firmware file.
3. AetherSDR extracts the firmware for your radio's model family and shows its progress. When staging finishes, **Upload Firmware** becomes available.

Upload

1. Click **Upload Firmware**.
2. A confirmation names the file and the radio, and warns that the radio reboots after the update. Do not disconnect during the upload.
3. The progress bar and status line follow the transfer.
4. The upload ends in one of three outcomes:

Outcome	What it means	What to do
Succeeded (green)	The radio confirmed the update.	Wait for the radio to reboot, then reconnect.
Unconfirmed (amber)	The bytes were sent, but the radio did not confirm the installation. It is probably rebooting to apply the image.	Reconnect and check the firmware version. A retry is blocked until you reconnect.
Failed (red)	The upload did not complete, or the radio rejected it.	Read the status line, then try again.

Closing Radio Setup during an upload asks for confirmation first. Closing it stops the transfer; once the bytes have been sent, it does not undo the update.

12.8.4 Reference

Where to find .ssdr files

- Windows, after installing SmartSDR: C:\ProgramData\FlexRadio Systems\SmartSDR\Updates\
- Or let **Select Installer...** extract the file from the installer for you.

Firmware files

File	Models
FLEX-6x00_vX.Y.Z.ssdr	All consumer radios (6000, 8000, Aurora series)

¹⁵<https://www.flexradio.com>

File	Models
FLEX-9600_vX.Y.Z.ssd	Government only (FLEX-9600 / DragonFire)

All consumer FlexRadios (Microburst, DeepEddy, BigBend, Aurora platforms) use the same FLEX-6x00 firmware.

Firmware compatibility

The active test target is FLEX-8600 firmware 4.2.18 (SmartSDR protocol v1.4.0.0). Earlier 4.x firmware works; v3.x is unsupported.

12.8.5 Known issues

- The transfer reaches 100%, but the radio does not apply the update or reboot, and can report that the installation failed (#5572¹⁶).

12.8.6 Troubleshooting

The Firmware Update group is missing

You are connected to a radio that is not a FlexRadio, so the group is hidden.

1. Connect to a FlexRadio.
2. Open **Settings** → **Radio Setup...** → **Radio** again.

Upload Firmware stays unavailable after an Unconfirmed upload

A retry is blocked until you reconnect, because the radio is probably rebooting to apply the image.

1. Wait for the radio to come back, then reconnect.
2. Check **FW Version** in the **Firmware Update** group.

The upload Failed

The upload did not complete, or the radio rejected it.

1. Read the status line for the reason.
2. Try the upload again.

12.8.7 See also

- FlexRadio (p. 20)
- Supported Radios (p. 18)
- Radio Setup (p. 214)
- FlexRadio downloads¹⁷

¹⁶<https://github.com/aethersdr/AetherSDR/issues/5572>

¹⁷<https://www.flexradio.com>

Reference

13.1 Menu Reference

Every menu item in AetherSDR, where it lives, and what it opens, followed by a tour of the title bar that sits above the panadapters. Menu items that need a connected radio, or a feature your radio or build does not have, are **dimmed** rather than hidden, and hovering one shows the reason.

The top-level menus are **File · Settings · Profiles · Tools · View · Window · Help**.

- **Windows and Linux:** the menus sit behind the ≡ button at the far left of the title bar. Alt+letter mnemonics (Alt+F, Alt+S, ...) still open each menu under the button, and every menu shortcut keeps working.
- **macOS:** the menus are in the system menu bar at the top of the screen. **Radio Setup...** appears as **AetherSDR → Preferences...**



The title bar: the menu button, radio tabs, PC Audio and volume controls, panel layout buttons and window controls.

13.1.1 File

Item	What it does
Connect to Radio...	Opens the connection panel (LAN radios, SmartLink, manual IP). See First Connection (p. 11) and Manual Connection (p. 201).
Disconnect	Ends the radio session without closing AetherSDR.
Quit	Closes the application.

13.1.2 Settings

Item	What it does
Radio Setup...	The main multi-page configuration dialog. See Radio Setup (p. 214). On macOS: AetherSDR → Preferences...
AetherControl...	Opens the AetherControl controller window.
FlexControl Knob & Buttons...	Jumps to the FlexControl section of Radio Setup → Serial & Controllers. See FlexControl Tuning Knob (p. 183).
Receive Sync ▸	Timing alignment between the radio and a KiwiSDR receiver: Sync Kiwi Audio & Display, Manual Offset / Auto Assist , Delay KiwiSDR 50 ms, Delay Flex 50 ms, Reset Offset, and Latency ▸ Normal (360 ms) / More Stable (520 ms) / High Jitter (1000 ms). See KiwiSDR and Web-888 (p. 36).

Item	What it does
MQTT...	Broker, subscriptions and publish buttons. See MQTT Station Automation (p. 141).
MIDI Mapping...	See MIDI Controller Mapping (p. 192).
Icom RC-28 Remote Encoder...	See USB Control Surfaces (p. 189).
Ulanzi Dial Mapping...	See Ulanzi Dial (p. 186).
USB Cables...	See USB Cable Management (p. 178).
Keyboard Shortcuts	Checkable. Turns operating shortcuts on or off; off by default . See Keyboard Shortcuts (p. 224).
Configure Shortcuts...	The shortcut editor.
SpotHub...	See SpotHub (p. 148).
multiFLEX...	The multi-client dashboard. See Multi-Flex (p. 211).
TX Band Settings...	Per-band RF power, tune power and inhibit choices.
Inhibit during TUNE ▶	Picks which integrations are held while the radio tunes.
AetherRX...	The receive audio chain. See Aetherial Audio (p. 74).
Settings Browser...	Browse and edit stored settings. See Settings and Back-ups (p. 220).
Autostart CAT with AetherSDR / Autostart TCI... / Autostart DAX...	Start those services automatically.
Reset Settings...	Restores application settings after confirmation. A backup is written first.

Items that depend on optional build features (MQTT, MIDI, serial controllers) only appear in builds that include them.

13.1.3 Profiles

Item	What it does
Profile Manager...	The main profile dialog.
Import/Export Profiles... (profile names)	SmartSDR-compatible <code>.ssdr_cfg</code> import and export. Below the separator, the radio's global profiles; pick one to load it.

See Profile Management (p. 94).

13.1.4 Tools

Item	What it does
Add Panadapter...	Opens the layout picker and adds a pan if the radio has capacity.
AetherTX...	Checkable. Shows the transmit audio chain. See Aetherial Audio (p. 74).
CW Keyer	Checkable. Shows the CW keyer panel. See CWX Panel (p. 102).
Copy Assist	Checkable. Speech-to-text panel (builds with speech-to-text only). See Copy Assist (p. 121).
AetherModem...	Packet and HF modem workspace. See AetherModem Packet Radio (p. 118).
Configure KiwiSDR...	Opens Radio Setup → Antennas.
Start SWR Scan...	Transmits . Runs an SWR sweep. See AetherSweep (p. 158).
Pre-tune ATU Bands...	Transmits . Steps the ATU through your bands.
Clear ATU Memories...	Clears the radio's ATU memories after confirmation.
Calibrate AGC-T...	Noise-floor AGC-T calibration for the active slice. It listens only; it does not transmit. The same calibration is on the AGC-T slider's right-click menu.
Callsign Lookup...	QRZ lookup (Ctrl+Shift+L). See Callsign Lookup (p. 163).
PSK Reporter...	See PSK Reporter Map (p. 155).

Item	What it does
FreeDV Reporter...	FreeDV Reporter station list.
Net Scheduler...	See Net Scheduler (p. 160).
Memory...	See Memory Channels (p. 92).
Waveforms...	Install and manage radio waveforms (radios with installable waveforms only).
Wideband Bandscope...	The radio's converter view before tuning; dimmed on radios without one. See Hermes-Lite 2 (p. 23).
Radio Health...	Reachability, other clients holding the radio, PA temperature — works even with no session.
GPS Dashboard...	See AetherClock and GPS (p. 161).
Network Diagnostics...	Link quality, trends and logs. See Troubleshooting (p. 256).
Runtime Monitor...	CPU, memory, threads and logs for the app itself. See Runtime Monitor (p. 266).

13.1.5 View

Item	What it does
Workspace Canvas ▶	The experimental canvas shell. See Workspace Canvas (p. 98).
Band Plan ▶	Size (Off / Small / Medium / Large / Huge), Show Spots , and the region plan. See Panadapter Controls (p. 54).
Theme ▶	Pick a theme. See Themes and Theme Editor (p. 228).
Theme Editor...	Edit or create a theme.
VFO Marker Size ▶	Default VFO marker: Off, 1 px or 3 px (default 3 px). See VFO Widget (p. 61).
VFO Filter Edge ▶	Default filter-edge lines: Show (default) or Hide.
Single-Click to Tune	Tune with one click instead of a double-click. Off by default.
Pan Follows VFO	Keep the active slice in view while tuning. On by default.
UI Scale ▶	75 % to 200 %, plus Zoom In (Ctrl+=), Zoom Out (Ctrl+-) and Reset (100 %) (Ctrl+0). Takes effect after a restart.
Reset Applet Order	Restores the default applet order.
Minimal Mode	Ctrl+Shift+M. Collapses the window to a narrow applet strip. See Keyboard Shortcuts (p. 224).
Frameless Window	Ctrl+Shift+F. On by default. Turn it off to return the main window to the operating system's own title bar and borders.
Propagation Conditions	Shows the propagation forecast on the panadapters. Off by default.
Smart Spot Filtering	Dims SSB spots with no detected voice signal nearby. Off by default; needs SpotHub Signal History.
FPS Meters	Ctrl+F. Shows spectrum and waterfall frame rates on each pan. Off by default.
Blink Status Indicator	Pulses the radio-link dot on the radio tab. On by default.

13.1.6 Window

Lists every open AetherSDR top-level window (pick one to bring it forward), plus:

Item	What it does
Minimize	Cmd+M on macOS; no default key elsewhere.
Zoom (macOS) / Maximize or Restore (Windows, Linux)	Toggles the window's maximized state.
Enter Full Screen / Exit Full Screen	F11 on Windows/Linux, Cmd+Ctrl+F on macOS.
Bring All to Front	Raises every AetherSDR window.

The Minimize and Full Screen keys can be rebound in **Settings → Configure Shortcuts...**, and they work even with Keyboard Shortcuts turned off.

Upgrading from an older version: Minimal Mode is **Ctrl+Shift+M on every platform** (it was Ctrl+M), so Cmd+M can be Minimize on macOS.

13.1.7 Help

Item	What it does
Getting Started..., AetherSDR Help..., What's New...	Built-in guides, each in its own window.
Understanding Noise Cancellation..., Configuring AetherSDR Controls..., Configuring Data Modes...	Topic guides.
AetherSDR Website, Donate to AetherSDR	Open the web pages.
Submit your Idea...	The AI-assisted feature-request helper.
File an Issue...	Opens a GitHub issue pre-filled with a redacted log tail.
Contributing to AetherSDR...	See Contributing Guide (p. 269).
Support & Diagnostics...	Logging categories and support bundles. See Support and Logging (p. 261).
Slice Troubleshooting...	A diagnostics window for slice, audio and display state.
Check for Updates...	Checks for a newer release now.
About AetherSDR	The About window (below).

13.1.8 The Title Bar

The main window has one 52 px title bar. From left to right:

Part	What it is
☰	The application menu (Windows and Linux only).
AetherSDR brand mark	Drag the bar here to move the window.
Radio tabs	One tab per known radio (LAN, SmartLink or routed). See below.
EXPERIMENTAL badge	Shown when the connected radio family's support is experimental.
multiFLEX	Appears when other clients share the radio; hover lists them, click opens the dashboard.
TX station	Red badge when another client is transmitting.
Transmit timer	Elapsed key-down time (below).
PC Audio	Routes receive audio and PC-mic transmit through this computer. Hover shows the input and output devices.
Speaker + master slider	Mute and level for the audio path you are hearing (below).
Headphone + slider	Headphone mute and level. Dimmed, with the reason, on radios that have no headphone output.
Dock left / Dock right / Pop out	Move the applet panel to the left or right of the panadapters (click the active side again to hide it), or float it in its own window. Ctrl+Shift+S also pops it out.
Window controls	Minimize, maximize and close (see "Per platform").

Radio tabs

- Line one is the radio's name; line two is its state in words (for example "link lost"), so colour is never the only signal.
- The dot on the active tab is the **radio-link indicator**. It pulses once per discovery heartbeat, turns amber while discovering and red after three missed heartbeats (blinking, or solid if **View** → **Blink Status Indicator** is off). The alarm stays on the radio that dropped until you disconnect on purpose, start a new session or remove the tab. A deliberate disconnect never raises it. Right-click the tabs to toggle blinking.
- When the tabs no longer fit, scroll them with the mouse wheel or drag them sideways.
- "+" opens the radio list: search by name, model, address or status; a per-radio **Actions** menu (Disconnect, Rename..., Radio setup..., Remove from tabs / Add to tabs); **Connect manually...**; **Rescan radios**. Removing a tab only hides it; it does not forget the radio or its credentials. Selecting a radio opens the connection panel on it; nothing connects without your confirmation.

Transmit timer

A green outlined M:SS readout (H:MM:SS past an hour) just left of **PC Audio**. It appears when you key up, starts from 0:00 for each transmission, holds the final time for 15 seconds after you unkey, then fades out. It counts operator transmissions only (MOX, PTT, footswitch, VOX, CW, tune) and never TCI or DAX transmissions from other programs.

Speaker and master slider

The speaker icon and master slider show the path you are actually hearing:

PC Audio	Speaker icon shows	Master slider shows
On	PC mute	PC playback level
Off	The radio's Line Out mute	The radio's Line Out level

With PC Audio off, a Line Out change made by another client (or in Radio Setup) shows up here too. See Audio Settings (p. 217).

Per platform

Platform	Window frame and controls
macOS	Native window with the standard traffic-light buttons, tiling menu, Stage Manager and native full screen.
Windows	Qt-drawn frame with the system shadow and resize borders; AetherSDR draws its own minimize / maximize / close buttons. Snap Layouts on hover over maximize does not appear yet.
Linux	Frameless window with a 6 px resize band on the sides and bottom (no resize along the top edge under the bar).

If your window manager or compositor misbehaves, turn off **View → Frameless Window** to go back to the system title bar on any platform.

In **Minimal Mode** the bar shrinks to the active radio tab and status badges; the ≡ button, audio controls and dock icons are hidden, and the maximize button leaves Minimal Mode.

13.1.9 About AetherSDR

Help → About AetherSDR opens a rounded, borderless window showing:

- the version and commit;
- a **build details** card (Qt version, compile date, and the spectrum renderer in use). The text is selectable, so you can paste it into a bug report;
- a **Contributor Logbook** ↗ button that opens contributors.aethersdr.com¹.

Drag anywhere in the window to move it. **Esc** or **Ctrl+W** (Cmd+W) closes it, and choosing About again raises the open window. Its animations stop when the operating system asks for reduced motion.

13.1.10 Known issues

- On Windows 10 with **Frameless Window** on, the ≡ menu and the panadapter's right-click menu flash and close straight away, and click-to-tune does nothing (#6272²).
- On Windows 10, a thin white strip shows along the left, right and bottom edges of the main window (#6266³).
- **View → Pan Follows VFO** has no visible effect for most tuning; it does not keep the slice centred the way Center Lock does (#5622⁴).

¹<https://contributors.aethersdr.com/#all-time>

²<https://github.com/aethersdr/AetherSDR/issues/6272>

³<https://github.com/aethersdr/AetherSDR/issues/6266>

⁴<https://github.com/aethersdr/AetherSDR/issues/5622>

13.1.11 Troubleshooting

A menu item is dimmed

The item needs a connected radio, or a feature your radio or build does not have.

1. Hover over the item to read the reason.
2. If it says no radio is connected, connect with **File** → **Connect to Radio...** and open the menu again.

The ≡ menu or the window frame misbehaves

Your window manager or compositor does not get on with AetherSDR's own frame.

1. Turn off **View** → **Frameless Window** (Ctrl+Shift+F). The main window goes back to the operating system's title bar and borders.
2. If the ≡ menu itself won't stay open, press **Ctrl+Shift+F** instead of using the menu.

The window is a narrow strip with no panadapters

Minimal Mode is on.

1. Press **Ctrl+Shift+M**, or click the title bar's maximize button, to leave Minimal Mode.

13.1.12 See also

- Keyboard Shortcuts ([p. 224](#))
- Radio Setup ([p. 214](#))
- Panadapter Controls ([p. 54](#))
- Workspace Canvas ([p. 98](#))
- Support and Logging ([p. 261](#))

Keyboard Shortcuts

This page is printed in full in another chapter: Keyboard Shortcuts ([p. 224](#)).

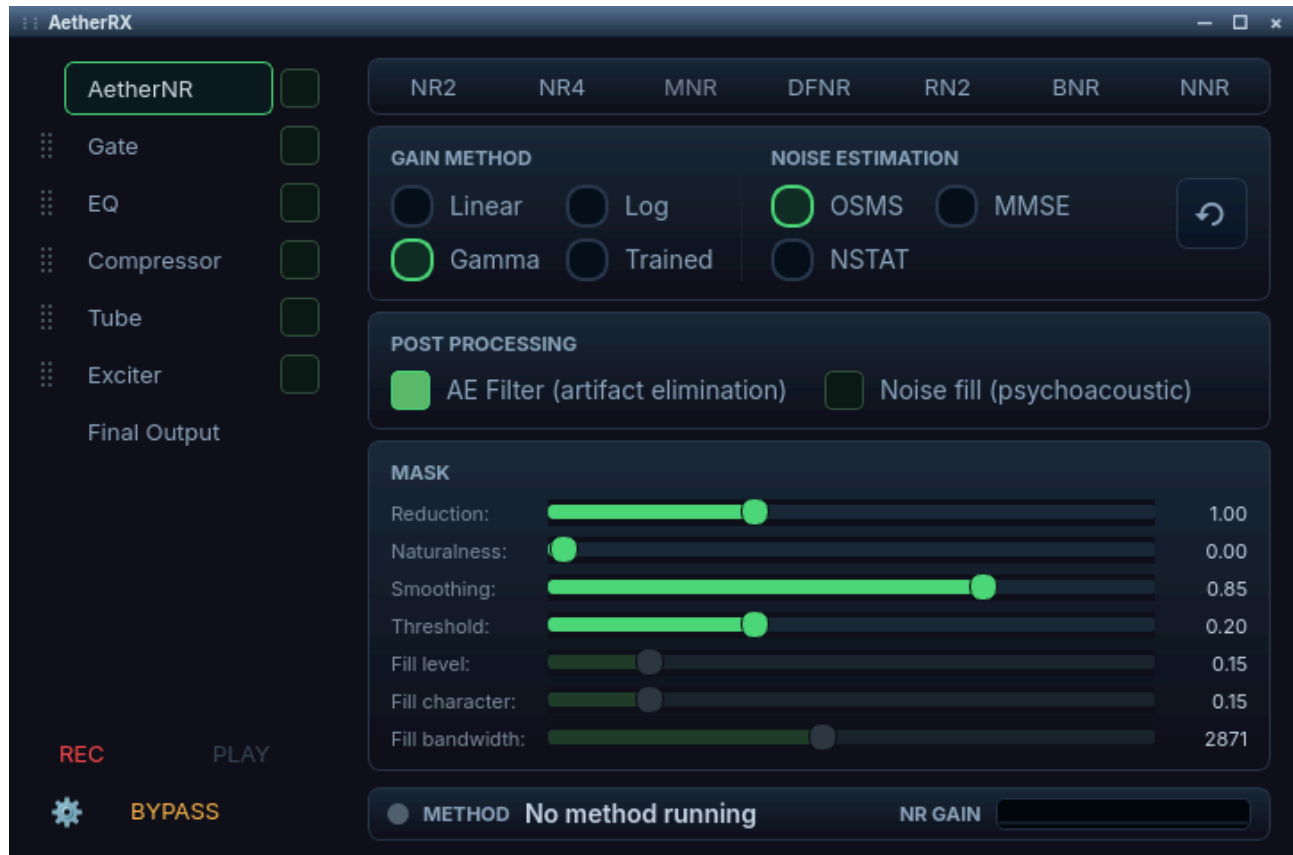
Understanding

14.1 DSP Noise Mitigation

Status: Supported · **Applies to:** All radios (radio-side filters: FlexRadio)

AetherSDR gives you two layers of noise mitigation. The **radio-side** filters run inside your FlexRadio. The **client-side** methods run on your computer after the audio arrives: AetherSDR has seven of them — **NR2** (spectral), **RN2** (RNNoise), **NR4** (libspecbleach), **NNR** (WDSP Neural Noise Reduction), **DFNR** (DeepFilterNet3), **BNR** (NVIDIA Maxine, on a local GPU) and **MNR** (macOS). Every client-side method denoises the left and right channels independently, so pans and diversity reception survive it.

The in-app guide **Help** → **Understanding Noise Cancellation...** covers the same ground for newcomers.



The AetherRX window: the receive processing chain on the left, the selected stage's settings on the right.

14.1.1 Using the radio-side filters (FPGA)

These run on the radio's FPGA — zero CPU cost to your PC. All use preset, optimized parameters with no user adjustments required — just toggle on/off. We did, however, add “amount” sliders in the DSP slide bar menu if you're feeling adventurous.

The radio-side buttons are on the VFO flag's **DSP** tab: NR, NB, ANF and APF on the first row; NRL, NRS, RNN and NRF on the second.

NB — Noise Blanker

Blanks short impulse noise (ignition interference, switching power supplies). Has an adjustable level (0-100) in the DSP side panel.

NR — Noise Reduction

General-purpose noise reduction. Adjustable level (0-100). Good starting point for noisy bands.

ANF — Auto Notch Filter

Removes a single persistent tone from the passband. Adjustable level (0-100). Best for a dominant interfering carrier.

NRL — Noise Reduction Leaky

Purpose: Reduce random band noise and background hiss.

Adaptive time-domain “Leaky LMS” filter that preserves correlated signals (voice, CW) while removing uncorrelated noise. Best for daily operation when signals are buried in static. Avoid when signals are already clean — may introduce slight “watery” audio if unnecessary.

Available on all Flex Radios.

ANFL — Adaptive Notch Filter (Leaky)

Purpose: Remove steady tones or power-line hum.

An adaptive filter that detects and cancels correlated tone interference. Best for hum, steady carriers, or single persistent tones. Avoid when no hum/tone is present — could partially notch the desired signal.

Available on all Flex Radios.

ANFT — Adaptive Notch Filter (FFT)

Purpose: Suppress up to five narrowband tones in the passband (≥ -110 dB).

FFT-based spectral notch filter that identifies and removes stable tone noise. Best for power supply hum, transformer buzz, grounding-related tones, or spurs.

Available on all Flex Radios.

NRF — Noise Reduction Filter

Purpose: Reduce steady background noise using frequency-domain processing.

Spectral subtraction estimates and removes broadband noise from each frequency bin. Best for constant hiss, fan noise, or environmental hum.

Available on FLEX-8000 (BigBend) and Aurora series only.

NRS — Noise Reduction Speech

Purpose: Improve speech clarity by lowering noise most aggressively between words.

Spectral subtraction combined with Voice Activity Detection (VAD). Best for SSB/AM voice operation in consistent noise environments.

Available on FLEX-8000 (BigBend) and Aurora series only.

RNN — Recurrent Neural Network Suppression

Purpose: AI-based removal of complex or varying noise while keeping speech natural.

Deep-learning model that separates speech from noise across time and frequency. Best for mixed or changing noise sources — fans, equipment, electrical noise, band noise.

Available on FLEX-8000 (BigBend) and Aurora series only.

APF — Audio Peaking Filter

Narrow bandpass filter centered on the CW pitch frequency. Enhances CW signals by attenuating everything outside the peak. Adjustable level (0-100). Only available in CW mode; the level slider is inactive until APF is engaged.

WNB — Wideband Noise Blanker

Wideband impulse noise blanker. Controlled via the ANT sub-menu on the left sidebar. Has an adjustable level (0-100) and appears as a “WNB” indicator on the FFT display when active. WNB belongs to the radio: AetherSDR uses whatever state the radio reports on connect and does not replay a saved state of its own.

14.1.2 Using the client-side methods

Important: the seven client-side methods process audio **on your PC**, not on the radio. They only affect audio played through your computer’s speakers or headphones. If you listen through the radio’s physical outputs (line out, headphone jack or front speaker), you will **not** hear client-side processing — only the radio’s built-in filters (NR, NRS, NRL, RNN and so on) apply there. To benefit from client-side processing, switch to PC audio.

Where the controls are

All client-side methods live in **AetherRX**, on its **AetherNR** tab. Open it with **Settings** → **AetherRX...** or the **AetherRX** button on the VFO flag’s DSP tab. That button lights up whenever a client NR method is running.

- The **method row** picks one of NR2, NR4, MNR, DFNR, RN2, BNR or NNR; the page below it holds that method’s settings. Methods that are not available on your computer or in your build are disabled with a tooltip explaining why.
- A status strip shows the active method and its key setting.
- **BYPASS**, at the foot of the AetherRX window beside the settings gear, suppresses every receive stage at once, AetherNR included. Click it again to restore what was on. Use it to A/B processed and unprocessed audio.
- The **NR Cycle** action (Off → NR → NR2 → NR4 → DFNR) can be bound to a key under **Settings** → **Configure Shortcuts...** or to a MIDI control. On radios other than FlexRadio it skips the radio-side NR and goes straight to NR2.
- AetherRX profiles save the active NR method together with the rest of the receive chain. See Aetherial Audio (p. 74).

Rules that apply to all seven

- **One at a time.** Selecting any client method turns off the one that was running. Any client method can be stacked with any radio-side filter.
- **Voice modes only.** The client methods switch themselves off in CW, CWL, RTTY and DIGU/DIGL, where they would suppress CW tones or corrupt data. Switch back to a voice mode and they become available again. For noise help in CW, use the radio’s NR and ANF.
- **Stereo.** Each method runs one instance per audio channel, so a hard pan lands instantly and diversity keeps each antenna in its own ear.

NR2 — Spectral Noise Reduction

Purpose: Advanced noise reduction using the Ephraim-Malah MMSE Log-Spectral Amplitude estimator, derived from Warren Pratt NR0V’s WDSP library.

Contributed by @EI6JGB

NR2 estimates the noise floor in each frequency bin and applies a spectral gain that suppresses noise while preserving speech. It excels at steady noise — constant hiss or hum — and gives you the most knobs of any method.

Settings (AetherNR → NR2):

- **Gain method** — Linear, Log, **Gamma** (default) or Trained
- **Noise estimation** — **OSMS** (default), MMSE or NSTAT (for noise that changes quickly)
- **Reduction** (0.50–2.00), **Naturalness** (residual noise floor, 0–0.15), **Smoothing** (0.50–0.98) and **Threshold** (speech detector, 0.05–0.50)
- **AE Filter (artifact elimination)** — smooths ringing and musical artifacts
- **Noise fill (psychoacoustic)** — off by default. Mixes a little shaped noise back into the gaps between syllables so the result sounds less “dead”; **Fill level**, **Fill character** and **Fill bandwidth** shape it.

NR2 keeps its noise estimate across a transmission, so it does not have to re-converge every time you unkey.

The first time you enable NR2, AetherSDR optimizes its FFT plans. See NR2 Noise Reduction (p. 250).

NR2 — Neural Noise Suppression (RNNoise)

Purpose: AI-based removal of complex or varying noise using the Xiph RNNoise recurrent neural network.

Unlike the radio’s RNN (which runs on BigBend/Aurora FPGAs), NR2 runs on your PC’s CPU and works with any radio. It handles non-stationary noise that statistical methods struggle with — fans cycling, changing band noise — and costs almost no CPU.

Settings:

- **Noise Floor** — 0–50 %, **default 0 %**. How much of the original signal RN2 leaves under the denoised audio. At 0 % RN2 suppresses fully and goes silent between phrases; 10–20 % leaves a steady, quiet noise floor so the band still sounds live.

When to use: quick cleanup with no setup; SSB voice in noisy conditions; FLEX-6000 radios without radio-side RNN. If it clips very weak signals, try DFNR or NR2.

NR2 is also available on the transmit side, in AetherTX’s Gate toolbar; the Noise Floor setting affects receive only.

NR4 — SpecBleach Spectral Denoiser

Purpose: Advanced spectral noise reduction with psychoacoustic masking and a learned noise profile.

NR4 is powered by Luciano Dato’s open-source `libspectbleach` library. It spends about one second learning a noise profile (passing audio through untouched meanwhile), then uses spectral masking to suppress the noise without the “musical” artefacts classical spectral subtraction can produce.

Settings: Reduction Amount (0–40 dB), Smoothing, Whitening, Adaptive Noise, Noise Estimation method (SPP-MMSE, Brandt or Martin), Masking Depth and Suppression Strength.

When to use: a high, constant noise floor (switch-mode supplies, plasma TVs, broadband RFI). A moderate Reduction Amount (15–20 dB) often sounds more natural than NR2 at the same depth.

Build requirement: NR4 needs `libspectbleach` at build time (on Windows that means `clang-cl` from LLVM). If your build doesn’t include it, NR4 is disabled with a tooltip explaining why.

NNR — WDSP Neural Noise Reduction

Purpose: Neural noise reduction trained specifically on off-air HF — over a hundred noise recordings from real receivers, with speech passed through an SSB transmit chain before mixing.

NNR comes from the WDSP library. It is meant for voice: it treats a steady carrier as noise and removes it.

Settings:

- **Strength** — how far NNR may attenuate each frequency bin. The marker on the slider shows the default; lower settings leave more real band noise in place, which often sounds more natural.

- **Model — Standard** (the default, about 13 % of one CPU core) or **Premium** (measurably better at poor signal-to-noise, about twice the CPU; not suitable for a Raspberry Pi).
- **Advanced** — Alpha, Knee, Tau, Max gain, Attack and Release. WDSP leaves these undocumented; the defaults are a good place to stay.

Tips:

- NNR runs after the AGC. Set the AGC threshold (AGC-T) as far above the noise floor as is practical — an AGC riding the noise floor moves the level faster than NNR’s own level tracker can follow, and the result can sound worse than no noise reduction at all.
- NNR adds about 192 ms of latency.
- It also works on KiwiSDR and Web-888 (p. 36) receive audio.
- NNR uses about 8.8 MB of memory per receive channel, whether or not it is switched on.

DFNR — DeepFilterNet3 AI Noise Reduction

Purpose: Best all-around AI noise reduction, running entirely on your CPU.

DFNR uses Hendrik Schröter’s DeepFilterNet3 model, designed for real-time speech enhancement. It adds about 10 ms of latency and needs no special hardware; the model is bundled.

Settings:

- **Attenuation Limit** — 0–100 dB. 0 is passthrough. Try 20–30 dB for weak signals, 40–60 dB for casual listening, 80–100 dB for strong signals in heavy noise.
- **Post-Filter Beta** — 0–0.30. Leave at 0 for the cleanest sound; 0.05–0.15 for a little extra cleanup.

When to use: the best default if you don’t have an NVIDIA RTX GPU, and on macOS. Start at Attenuation Limit 40 dB and Post-Filter Beta 0.

BNR — NVIDIA Maxine Background Noise Removal

Purpose: Large-scale AI noise removal on your local NVIDIA GPU.

BNR runs the NVIDIA Maxine Audio Effects denoiser **in-process** on an NVIDIA GeForce RTX 40-series or later GPU under **Linux or Windows** — no Docker, no container. The NVIDIA runtime (about 1 GB) downloads once on first use, and a one-time NVIDIA license dialog appears the first time you enable it.

Settings: **Intensity** 0–100 (0 is passthrough).

Not available on macOS or without a supported NVIDIA GPU — use DFNR there. See BNR GPU Noise Removal (p. 248) for setup and troubleshooting.

MNR — Apple MMSE-Wiener (macOS only)

Purpose: macOS-native spectral noise reduction using Apple’s vDSP / Accelerate framework.

MNR tracks steady background noise with smoothed minimum statistics and applies a Wiener mask. It costs almost nothing on Apple Silicon. At full Strength it attenuates stationary noise by about 24.4 dB.

Settings: **Strength** 0–100 % (default 100 %). Selecting MNR in the method row is the only on/off switch; there is no separate enable checkbox.

On non-macOS builds MNR is disabled with an explanatory tooltip.

14.1.3 Choosing a method

- Only enable a filter when its target noise type is present
- Some filters can be used together, but excessive stacking may change audio tone or introduce artifacts
- As a rule of thumb:
 - **NRL** → Random noise, background hiss
 - **ANFL / ANFT** → Tones, hum, spurs
 - **NRF / NRS / RNN** → Voice clarity enhancement (BigBend/Aurora only)

- **NR2** → Steady hiss on a quiet band, fine-grained tunables
- **RN2** → Voice in noise with zero setup
- **NR4** → Stubborn broadband noise (RFI, switch-mode supplies); needs a moment of pure noise to learn
- **NNR** → HF band noise on voice, with the AGC threshold set well above the noise floor
- **DFNR** → Best all-around AI denoising on CPU, no GPU needed (recommended default)
- **BNR** → Maximum noise removal on an NVIDIA RTX GPU (Linux/Windows)
- **MNR** → macOS users wanting a native one-knob option
- **NB / WNB** → Impulse noise (ignition, switching)
- **APF** → CW signal peaking (CW mode only)
- If you stack radio-side NR with a client method, keep the radio-side level light and let the client method do the heavy lifting.

14.1.4 Reference

Quick reference

Feature	Full Name	Target Noise	Runs On
NB	Noise Blanker	Impulse noise (ignition, switching)	All radios
NR	Noise Reduction	General broadband noise	All radios
NR2	Spectral Noise Reduction	Steady noise, weak signals	Client (CPU)
RN2	Neural Noise Suppression (RNNNoise)	Complex/varying noise, zero setup	Client (CPU)
NR4	SpecBleach Spectral Denoiser	Stubborn broadband noise	Client (CPU)
NNR	WDSP Neural Noise Reduction	HF band noise (trained on off-air recordings)	Client (CPU)
DFNR	DeepFilterNet3 AI	Best all-around AI denoising	Client (CPU)
BNR	NVIDIA Maxine background noise removal	All noise types (large-scale AI)	Client (NVIDIA GPU, Linux/Windows)
MNR	Apple MMSE-Wiener	Steady background noise	Client (CPU, macOS only)
ANF	Auto Notch Filter	Single persistent tone	All radios
NRL	Noise Reduction Leaky	Random band noise, background hiss	All radios
NRS	Noise Reduction Speech	Background noise during voice	BigBend + Aurora
RNN	Recurrent Neural Network	Complex/varying noise (AI-based)	BigBend + Aurora
NRF	Noise Reduction Filter	Steady broadband noise	BigBend + Aurora
ANFL	Adaptive Notch Filter (Leaky)	Steady tones, power-line hum	All radios
ANFT	Adaptive Notch Filter (FFT)	Multiple narrowband tones (up to 5)	All radios
APF	Audio Peaking Filter	CW signal enhancement	All radios (CW mode only)
WNB	Wideband Noise Blanker	Wideband impulse noise	All radios

“All radios”, “BigBend” and “Aurora” in this table refer to FlexRadio models.

Mode-dependent availability

The seven client-side methods are voice-only: they switch themselves off in CW, CWL, RTTY and DIGU/DIGL. The radio-side filters shown in each mode are:

Mode	Radio-side filters shown	Hidden
USB/LSB	NR, NB, ANF, NRL, NRS, RNN, NRF, ANFL, ANFT	APF
CW	NR, NB, APF, NRL, NRS, NRF	ANF, RNN, ANFL, ANFT
RTTY	NR, NB, NRL, NRS, RNN, NRF	ANF, ANFL, ANFT
DIGU/DIGL	NR, NB, NRL, NRS, RNN, NRF	ANF, ANFL, ANFT
FM/NFM	Hidden (OPT tab replaces DSP)	All

14.1.5 Known issues

- NR2 can sound scratchy or produce clicks and artefacts on voice (#632¹).
- NNR's **Premium** model makes receive audio choppy and broken up; the **Standard** model does not (#5989²).
- DFNR can switch itself back on in every profile after you turn it off and save the profile (#4018³).
- The client-side methods stay on in the digital-voice and data modes DSTR, FDV, FDVU and DRM, where they can corrupt the decoder's input (#5701⁴).

14.1.6 Troubleshooting

Client-side noise reduction makes no difference

You are listening through the radio's own outputs (line out, headphone jack or front speaker). Client-side methods only process audio played on your computer.

1. Switch to PC audio.
2. If you must listen on the radio's outputs, use the radio-side filters (NR, NRS, NRL, RNN and so on) instead.

A client-side method turns itself off, or its button is unavailable

The client-side methods are voice-only and switch off in CW, CWL, RTTY and DIGU/DIGL. A method can also be disabled because your computer or build doesn't support it; its tooltip says why.

1. Switch back to a voice mode to make the methods available again.
2. In CW, use the radio's NR and ANF instead.
3. If the method is disabled in a voice mode, hover over it in the AetherNR method row and read the tooltip. For BNR see BNR GPU Noise Removal (p. 248); NR4 needs a build with `libspeccbleach`; MNR is macOS only.

NNR sounds worse than no noise reduction

The AGC is riding the noise floor. NNR runs after the AGC, and an AGC on the noise floor moves the level faster than NNR's level tracker can follow.

1. Raise the AGC threshold (AGC-T) as far above the noise floor as is practical.
2. Lower NNR's **Strength** if the result still sounds unnatural.

Audio sounds watery or changes tone

Too many filters are stacked, or a filter is on with no matching noise present.

1. Turn off filters whose target noise isn't present (see Choosing a method (p. 245)).
2. If you stack radio-side NR with a client method, keep the radio-side level light.
3. Use **BYPASS** in AetherRX to compare processed and unprocessed audio.

14.1.7 See also

- NR2 Noise Reduction (p. 250)
- BNR GPU Noise Removal (p. 248)
- Aetherial Audio (p. 74)
- RX Controls (p. 64)
- Copy Assist (p. 121) transcribes the audio after client noise reduction, so cleaner audio also means a cleaner transcript.

¹<https://github.com/aethersdr/AetherSDR/issues/632>

²<https://github.com/aethersdr/AetherSDR/issues/5989>

³<https://github.com/aethersdr/AetherSDR/issues/4018>

⁴<https://github.com/aethersdr/AetherSDR/issues/5701>

14.2 BNR GPU Noise Removal

Status: Supported · **Applies to:** All radios · **Platforms:** Linux, Windows

BNR is AetherSDR’s NVIDIA-powered receive noise remover. It runs the NVIDIA Maxine **Audio Effects (AFX)** denoiser **inside AetherSDR, on your local NVIDIA GPU** — no Docker, no container, no NGC account and no network service. It is one of the seven client-side noise-reduction methods described in DSP Noise Mitigation (p. 241).

Upgrading from an older version: earlier releases ran BNR in a Docker/NIM container reached over gRPC. That backend has been removed. You can stop and delete any old Maxine BNR container; it is no longer used.

14.2.1 Requirements

Requirement	Detail
GPU	NVIDIA GeForce RTX 40-series or later (Ada or newer)
Operating system	Linux or Windows. BNR is not available on macOS.
Download	A one-time download of about 1 GB (the NVIDIA runtime and the denoiser model for your GPU)

The Linux and Windows release builds include BNR. On a Mac, or a PC without a supported NVIDIA GPU, use **DFNR** instead — it runs on any CPU.

If no runtime pack has been published for your GPU’s architecture, BNR disables itself and says so (“No BNR pack for your GPU ... — use DFNR”) instead of showing an error.

14.2.2 Setup

1. Open **Settings** → **AetherRX...** (or click the **AetherRX** button on the VFO flag’s DSP tab) and go to the **AetherNR** tab.
2. Select **BNR** in the method row. The BNR page shows the **NVIDIA AFX** panel.
3. Click **Download (~1 GB)**. AetherSDR detects your GPU’s architecture and fetches the matching pack, verifying every file by SHA-256. The **Installed components** list shows progress, then the installed versions.
4. The first time you turn BNR on, a one-time **NVIDIA Software License** dialog appears. Click **Accept** to use BNR; **Decline** leaves BNR off.

The download is cached, so later launches start straight away:

- **Linux:** `~/ .local/share/AetherSDR/AetherSDR/nvidia-afx/current/`
- **Windows:** `%LOCALAPPDATA%\AetherSDR\AetherSDR\nvidia-afx\current\`

The doubled AetherSDR folder is expected.

On Linux the CUDA libraries come straight from NVIDIA’s own PyPI distribution; the rest of the pack is a small AetherSDR release asset. On Windows the pack is a single self-contained zip.

When a newer pack is available the button reads **Update**; otherwise it offers **Re-download**.

14.2.3 Using BNR

- **One client NR method at a time.** Turning BNR on turns off NR2, RN2, NR4, NNR, DFNR or MNR, and vice versa. BNR can be stacked with any radio-side filter (NR, NB, ANF and so on).
- **Voice only.** BNR switches itself off in CW, CWL, RTTY and DIGU/DIGL, where it would suppress CW tones or corrupt data.
- **Stereo.** BNR denoises the left and right channels independently, so pans and diversity reception keep each antenna in its own ear.
- The **AetherRX** launcher on the VFO flag lights up while BNR (or any client NR method) is running.

- Like every client-side method, BNR only affects audio played through your computer. It does not affect the radio's own headphone or speaker outputs.

14.2.4 Reference

Controls

Control	Range	What it does
Status	—	Whether the runtime is installed, downloading, failed, or not available in this build
Intensity	0–100	Denosing strength. 0 passes audio through untouched; 100 is maximum. Remembered across restarts.
Download / Update / Re-download	—	Fetches or refreshes the runtime pack

Licensing

BNR uses NVIDIA software and an NVIDIA denoiser model, governed by the NVIDIA Software License Agreement, the Product-Specific Terms for NVIDIA AI Products and the NVIDIA Community Model License. They are licensed for use on NVIDIA RTX / GeForce RTX GPUs on a single-user PC or workstation. The full license texts are included in the downloaded pack (licenses/).

14.2.5 Troubleshooting

BNR is greyed out: “BNR requires an NVIDIA RTX 40-series or later GPU”

Your computer has no NVIDIA GPU, or the GPU is older than the RTX 40-series.

1. Use **DFNR** instead; it runs on any CPU.

BNR is greyed out: “No BNR pack for your GPU ... yet”

Your GPU is new enough, but no runtime pack has been published for its architecture.

1. Use **DFNR** instead.

No BNR option, or “Not available in this build”

You are on macOS, or using a build compiled without BNR (-DENABLE_NVIDIA_AFX=OFF).

1. Use **DFNR** instead, or use a Linux or Windows release build.

The download fails

The connection dropped, or there isn't enough free disk space for the pack.

1. Check your internet connection.
2. Make sure about 1 GB of disk space is free.
3. Click the **Download** button again.

BNR turns off when you change mode

This is expected. BNR is voice-only and switches off in CW, CWL, RTTY and DIGU/DIGL.

1. Switch back to a voice mode.

Audio sounds hollow or robotic

BNR is denosing too hard, or a radio-side NR stacked with it is adding its own artefacts.

1. Lower **Intensity**.
2. Lighten any radio-side NR stacked with BNR.

14.2.6 See also

- DSP Noise Mitigation ([p. 241](#)) — all noise-reduction methods compared
- Aetherial Audio ([p. 74](#))

- Developer reference: docs/nvidia-bnr.md⁵

14.3 NR2 Noise Reduction

NR2 is a client-side spectral noise reduction method that complements the radio's built-in NR. It uses the Ephraim-Malah MMSE Log-Spectral Amplitude estimator, derived from Warren Pratt NR0V's WDSP library, and runs on your computer's CPU — not on the radio.

Contributed by @EI6JGB

For how NR2 compares with the other six client-side methods, see DSP Noise Mitigation (p. 241).

14.3.1 Requirements

FFTW3 is required to build AethersDR (the bundled WDSP library depends on it). Release builds include it. When building from source:

- Arch: `sudo pacman -S fftw`
- Ubuntu/Debian: `sudo apt install libfftw3-dev`
- macOS: `brew install fftw`
- Windows: run `scripts/setup/setup-fftw.ps1`

See Building from Source (p. 277).

14.3.2 How it works

NR2 processes the demodulated receive audio on your CPU. It estimates the noise floor in each frequency bin and applies a spectral gain that suppresses noise while preserving speech. Like every client-side method, it processes the left and right channels independently, and it keeps its noise estimate across a transmission so it does not have to re-converge every time you unkey.

14.3.3 Using NR2

Turning NR2 on

- **AetherRX**: open **Settings** → **AetherRX...** (or the **AetherRX** button on the VFO flag's DSP tab), go to the **AetherNR** tab and select **NR2** in the method row. NR2's settings appear below it.
- **Shortcut**: bind **NR Cycle (Off/NR/NR2/NR4/DFNR)** under **Settings** → **Configure Shortcuts...**, or to a MIDI control.

NR2 is a voice method: it switches itself off in CW, CWL, RTTY and DIGU/DIGL.

When to use NR2

- **Steady noise** — constant hiss or hum on a relatively quiet band, where you want to dial in exactly the right amount of suppression
- **Weak signals** where the radio's NR isn't enough
- **Stacking with radio NR** — NR and NR2 use different algorithms and can complement each other (enable NR on the radio, NR2 in AethersDR)

NR2 is less effective against impulse noise (clicks, pops, lightning crashes); use the radio's NB or WNB for those.

FFTW wisdom (first use)

The first time you enable NR2, AethersDR optimizes the FFT plans it uses (FFTW “wisdom”). An **AethersDR — FFTW Wisdom** window appears while this runs:

- It shows progress, a **Working** activity indicator and an **Elapsed** time counter. Planning can take several minutes on some systems; if the progress pauses, the window says it is still working.

⁵<https://github.com/aethersdr/AetherSDR/blob/main/docs/nvidia-bnr.md>

- The window is modeless — keep operating while it runs. On macOS it stays visible when AetherSDR is not the frontmost app.
- Clicking NR2 again while it runs brings the same window forward instead of starting a second run.
- **Cancel** stops after the FFT plan currently being computed. NR2 is then not enabled, and your audio is left unchanged.
- NR2 switches on when planning finishes, and the window closes itself.

The result is saved **as soon as planning finishes**, so a crash or force-quit afterwards does not lose it, and later launches skip the wait. The same cache also speeds up the WDSP receive setup on the Hermes-Lite 2 (p. 23).

The wisdom file is stored at:

Platform	Path
Linux / macOS	~/.config/AetherSDR/aethersdr_fftw_wisdom
Windows	%APPDATA%\AetherSDR\aethersdr_fftw_wisdom

To force a fresh optimization (for example after upgrading FFTW), quit AetherSDR, delete that file, and enable NR2 again.

Upgrading from an older version: older builds kept the file in a double-nested ~/.config/AetherSDR/AetherSDR/ folder. Current builds use the path above, so NR2 optimizes its plans once more after upgrading. You can delete the old file.

14.3.4 Reference

Settings

Setting	Range / options	What it does
Gain method	Linear, Log, Gamma , Trained	The speech model used to compute the suppression gain. Gamma is the default.
Noise estimation	OSMS , MMSE, NSTAT	How the noise floor is tracked. Try NSTAT if the noise changes quickly.
Reduction	0.50–2.00	How aggressively noise is suppressed.
Naturalness	0–0.15	Leaves a small residual noise floor, trading depth for a more natural sound.
Smoothing	0.50–0.98	Smooths the gain over time to reduce musical artifacts. Higher is smoother but slower.
Threshold	0.05–0.50	Speech-detector sensitivity. Lower keeps quiet voices but lets more noise through.
AE Filter (artifact elimination)	On / off	Reduces ringing and musical artifacts.
Noise fill (psychoacoustic)	Off by default	Mixes a little shaped noise back into the gaps between syllables. Fill level , Fill character and Fill bandwidth shape it.

Start with the defaults. See Troubleshooting (p. 251) if you hear musical warbling or weak voices disappear.

14.3.5 Known issues

- NR2 can sound scratchy or produce clicks and artefacts on voice (#632⁶).

14.3.6 Troubleshooting

Musical warbling in the background

The gain is changing too quickly from frame to frame.

⁶<https://github.com/aethersdr/AetherSDR/issues/632>

1. Raise **Smoothing**.
2. Turn on **AE Filter (artifact elimination)** if it is off.

Weak voices disappear

The speech detector is missing quiet speech and treating it as noise.

1. Lower **Threshold**.
2. If that is not enough, lower **Reduction** or raise **Naturalness**.

The FFTW Wisdom window takes a long time

AetherSDR is optimizing its FFT plans the first time you enable NR2. This can take several minutes on some systems.

1. Keep operating; the window is modeless.
2. Wait for it to finish. NR2 switches on and the window closes itself, and later launches skip the wait.
3. If you click **Cancel**, NR2 stays off; enable it again to restart planning.

NR2 turns off when you change mode

This is expected. NR2 is a voice method and switches off in CW, CWL, RTTY and DIGU/DIGL.

1. Switch back to a voice mode.
2. In CW, use the radio's NR and ANF instead.

NR2 does nothing against clicks and crashes

NR2 targets steady noise, not impulse noise.

1. Use the radio's **NB** or **WNB** for clicks, pops and lightning crashes.

14.3.7 See also

- DSP Noise Mitigation (p. 241) — how NR2 compares with the other client-side methods
- Aetherial Audio (p. 74)
- Building from Source (p. 277)
- Hermes-Lite 2 (p. 23)

14.4 GPU Rendering

Status: Supported · **Applies to:** All radios · **Platforms:** Windows, Linux, macOS (Apple Silicon)

AetherSDR draws the spectrum and waterfall on the graphics card through Qt's QRhi layer, using each platform's native graphics API. This keeps the display smooth at high frame rates while leaving the CPU free for audio and DSP. No setup is needed: the right backend is chosen at startup.

14.4.1 Requirements

- Every AetherSDR build requires **Qt 6.12**, which always includes GPU rendering support.
- Whether a particular binary uses the GPU is decided **when it is built**: the **AETHER_GPU_SPECTRUM** build option (on by default) selects the GPU renderer or the CPU **QPainter** renderer. The CPU renderer is a build-time alternative, **not** an automatic fallback at runtime. Source builds also need Qt's private GUI headers; without them CMake turns the option off.

For build dependencies see Building from Source (p. 277).

14.4.2 Using GPU rendering

Choosing the GPU

On computers with more than one graphics adapter, the panadapter **Display** panel has a **SYSTEM** → **GPU**: selector.

- The choice takes effect on the next launch (“Restart to apply”).
- On Windows hybrid-graphics laptops the discrete GPU is used by default. The integrated GPU is listed but greyed out (“disabled (#1921)”) because it crashes when a panadapter is popped out or docked.
- Adapters whose selection path hasn’t been tested on real hardware are marked “(experimental)”.
- Selecting the discrete GPU on a hybrid laptop also works under Wayland on Linux.

Heat Map mode

Display → Heat Map (on by default) colours the spectrum trace and fill by signal strength, so strong signals stand out. Turn it off for a single solid colour chosen with the **FFT Fill** colour button, with the fill slider setting its opacity.

Forcing software rendering (AETHER_NO_GPU)

If your GPU or driver draws the spectrum wrongly, force software OpenGL without rebuilding:

```
AETHER_NO_GPU=1 ./AetherSDR-*.AppImage
```

This is worth trying first on a Raspberry Pi or another system whose Mesa driver is newer than its hardware. On Windows, AETHER_NO_GPU=1 (or QT_OPENGL=software) switches every panadapter to software OpenGL instead of Direct3D 11.

Wayland and XWayland (Linux)

On a Wayland session with a display connected, AetherSDR uses native Wayland when it can and XWayland otherwise. On a headless Wayland session (VNC or wayvnc with no monitor) it starts on XWayland automatically. To override:

```
QT_QPA_PLATFORM=xcb ./AetherSDR-*.AppImage # force XWayland
QT_QPA_PLATFORM='wayland;xcb' ./AetherSDR-*.AppImage # force native Wayland
```

A value you set yourself always wins.

14.4.3 Reference

Platform backends

Platform	Backend	Notes
Windows	Direct3D 11	AETHER_NO_GPU switches to software OpenGL (below).
macOS (Apple Silicon)	Metal	
macOS (Intel)	—	The Intel DMG is deliberately built with the CPU renderer, because the GPU path misbehaves on older Metal/OpenGL hardware.
Linux x86_64	OpenGL	Mesa or vendor drivers.
Linux aarch64 (Raspberry Pi and other ARM)	OpenGL	The aarch64 AppImage renders on the GPU and needs glibc 2.38 or newer (Raspberry Pi OS Trixie; Bookworm can’t run it). The 3D view works on the Raspberry Pi 5.

Help → About AetherSDR shows which renderer is in use. If the “GPU” is really a software rasteriser (llvmpipe, WARP, SwiftShader and similar), the renderer line says CPU QRhi (...) instead of GPU QRhi (...); the display is correct, just slower.

How it works

- **Waterfall** — a GPU texture used as a ring buffer. Each new row uploads one scanline; scrolling is done in the shader, with no memory copies.
- **Spectrum trace** — drawn per pixel on the GPU from a single column texture of FFT data, so there is no per-frame vertex rebuild on the CPU. Trace width is the **FFT Line** setting (default 1.0 px).
- **Overlays** — grid, scales, band plan, passbands, slice markers, notches and spots are painted once into a cached texture and only repainted when something changes (retune, zoom, resize, mode change).

- Repaints are coalesced to at most one per 16 ms, so the display tops out at about **60 fps**. The spectrum rate itself is **Display → FFT FPS** (5–60, default 25).
- The **3D Stacked Trace** view (see Panadapter Controls (p. 54)) also runs on the GPU.

14.4.4 Known issues

- On Linux under native Wayland, the GPU spectrum can freeze the whole window at startup; it happens on some launches and not others (#4704⁷).
- On some Linux Wayland desktops the panadapter and waterfall update very slowly (#4725⁸).
- On some Linux systems (reported with an NVIDIA RTX 5080 on an X11 session) the spectrum and waterfall area stays empty and the connection window does not appear (#4998⁹).
- With the CPU renderer on a slow computer such as a Raspberry Pi 5, the **Heat Map** fill can use almost all of the main thread and leave the window unresponsive (#5192¹⁰).
- On Windows, popping out a panadapter has crashed AethersDR with an older AMD Radeon HD 7000 series graphics card (#5990¹¹).

14.4.5 Troubleshooting

A “Spectrum renderer unavailable” card covers the panadapter

The GPU renderer failed to start. The card names the reason, and the failure is also written to the log.

1. Quit AethersDR and launch it with `AETHER_NO_GPU=1` (see Forcing software rendering (p. 253)).
2. On macOS the card suggests rebuilding with `-DAETHER_GPU_SPECTRUM=OFF` instead.

The spectrum is blank but controls and audio work

QRhi could not start on this system: there is no usable OpenGL, Direct3D or Metal, as on some virtual machines and headless hosts.

1. Launch with `AETHER_NO_GPU=1`.
2. To see what Qt tried, launch with `QT_LOGGING_RULES="qt.rhi.*=true"` and look for the QRhi initialisation lines in the log.

The panadapter is black under remote desktop on Linux

The session is running on native Wayland.

1. Launch with `QT_QPA_PLATFORM=xcb` to force XWayland (see Wayland and XWayland (p. 253)).

The display is correct but slow, and About says CPU QRhi

The graphics adapter AethersDR found is a software rasteriser (llvmpipe, WARP, SwiftShader and similar), not a real GPU.

1. If your computer has more than one adapter, open **Display** and pick the real GPU under **SYSTEM → GPU**.
2. Restart AethersDR; the choice takes effect on the next launch.

14.4.6 See also

- Panadapter Controls (p. 54)
- Troubleshooting (p. 256)
- Building from Source (p. 277)
- Linux (p. 43)
- docs/BUILDING.md: GPU spectrum rendering¹²

⁷<https://github.com/aethersdr/AethersDR/issues/4704>

⁸<https://github.com/aethersdr/AethersDR/issues/4725>

⁹<https://github.com/aethersdr/AethersDR/issues/4998>

¹⁰<https://github.com/aethersdr/AethersDR/issues/5192>

¹¹<https://github.com/aethersdr/AethersDR/issues/5990>

¹²<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md#gpu-spectrum-rendering>

Help

15.1 Troubleshooting

This page is the symptom index for AetherSDR. Find the symptom you see below; if it belongs to one feature, the By feature (p. 259) list links that feature's own Troubleshooting section. To report a problem that isn't solved here, see Filing a bug (p. 259).

Start with the built-in tools:

- **Help** → **Support & Diagnostics...** for logs. See Support and Logging (p. 261).
- **Tools** → **Network Diagnostics...** for link problems, and **Tools** → **Runtime Monitor...** for performance. See Runtime Monitor (p. 266).
- **Help** → **Slice Troubleshooting...** walks through common slice problems.
- The Log Analyzer¹ checks a log or support bundle for known problems and links the fix. It runs in your browser; the file is never uploaded.

15.1.1 Connecting

No radio appears under On This Network

Discovery broadcasts are not reaching AetherSDR. The radio may be on a different subnet (broadcasts do not cross routers or most VPNs), a firewall may be blocking UDP port 4992, or the radio may enforce private-IP connections while you are on a different network.

1. Check the radio is reachable: `ping <radio-ip>`.
2. Allow discovery through the firewall, for example `sudo ufw allow 4992/udp`.
3. Use **Connect by IP**: choose the **Radio type**, enter the radio's address and click **Connect by IP**. See Manual Connection (p. 201).

The radio refuses the connection

The radio rejected AetherSDR's GUI-client registration (for example with error F3000001). AetherSDR stops, closes the connection, does not retry on its own, and returns to the connect panel showing the radio's error and what to do about it.

1. Read the message in the connect panel and follow it.
2. Typically another client must disconnect first, or the radio needs a restart.

A Connected Stations dialog appears when you connect

multiFLEX is off on the radio and other clients are already connected.

1. Disconnect one of the stations listed in the **Connected Stations** dialog.

¹<https://docs.aethersdr.com/log-analyzer>

2. See Multi-Flex (p. 211).

SmartLink: signed in, but no radio is listed

The radio is not registered with SmartLink, or port forwarding is not set up.

1. Register the radio with SmartLink.
2. Forward TCP 4994 and UDP 4993 to the radio.
3. See SmartLink Setup (p. 199).

SmartLink stops with a certificate mismatch warning

The radio's certificate does not match the one AetherSDR pinned.

1. Read the certificate pinning (p. 200) section of SmartLink Setup (p. 199) before accepting the new certificate.

15.1.2 Audio

No audio from the speakers

The panadapter and waterfall run but nothing comes out of the speakers. PC Audio is off, the wrong device is selected, something is muted, or the audio device failed to open.

1. Check that **PC Audio** is on in the title bar and that the right **Output** device is selected under **Radio Setup** → **Audio**. See Audio Settings (p. 217).
2. Check the slice is not muted, and that the title-bar speaker and master slider are not muted or at zero.
3. Turn on the **Audio** log category in **Help** → **Support & Diagnostics...** and look for device-open errors.
4. **Linux with PipeWire**: if the PipeWire library is installed but PipeWire's client configuration is not, AetherSDR keeps Qt's audio off PipeWire at startup rather than crashing. Install your distribution's `pipewire` (Arch) or `pipewire-bin` (Debian/Ubuntu) package.

The audio device lists show only “Dummy Output”

This affects source builds on Ubuntu and Linux Mint. The AppImage does not need this.

1. Follow the distro notes in docs/BUILDING.md².

Audio stutters or clicks on a remote link

The link has jitter or packet loss.

1. Raise **Radio Setup** → **Audio** → **Audio Buffer** (default 100 ms) a little at a time.
2. Leave **Smooth packet loss** on.
3. Use Opus compression and low bandwidth mode. See Low Bandwidth Connections (p. 208).
4. Watch latency, jitter, packet loss and the audio buffer over time in **Tools** → **Network Diagnostics....**

15.1.3 Keyboard

Space bar or arrow keys do nothing

Operating keyboard shortcuts (Space for PTT, arrows to tune, and so on) are **off by default**. While they are off, the first key press that nothing accepts shows “Keyboard shortcuts are off” in the status bar.

1. Turn them on with **Settings** → **Keyboard Shortcuts**.
2. See Keyboard Shortcuts (p. 224).

15.1.4 Starting and crashing

error while loading shared libraries: libOpenGL.so.0

Ubuntu 26.04 no longer installs `libopengl0` by default on the desktop image. AetherSDR's GPU spectrum and waterfall link against `libOpenGL.so.0`, so it refuses to start without it.

²<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md#distro-notes>

1. Run `sudo apt install libopengl0`.

This is the modern GLVND desktop OpenGL runtime, not the legacy `libgl1`; both can be installed side by side.

The AppImage does not run on Raspberry Pi OS Bookworm

The ARM AppImage needs glibc 2.38 or newer, and Bookworm has glibc 2.36.

1. Use Raspberry Pi OS Trixie. See Installation (p. 8).

AetherSDR crashes opening a dialog under XWayland (GLX BadAccess)

Some compositors produce this crash under XWayland.

1. Run natively on Wayland: `QT_QPA_PLATFORM='wayland;xcb' ./AetherSDR-*.AppImage`.

Problems after a crash while popping out a panadapter

AetherSDR saves a floating panadapter only after the pop-out succeeds, so a crash during a pop-out does not repeat on the next launch. A stale floating layout can still be left behind.

1. Use **View** → **Workspace Canvas** → **Workspaces** → **Import pop-outs onto canvas** to recover it. See Workspace Canvas (p. 98).

A stored setting stops AetherSDR from starting

A bad stored value, such as a window placed off-screen, is applied at startup.

1. Use `AetherSDR --config` to inspect or remove the value. See Settings and Backups (p. 220).

AetherSDR crashes on exit with MangoHud

This is not an AetherSDR bug.

1. Disable MangoHud: `MANGOHUD=0 ./AetherSDR`.

15.1.5 Display

Panadapter is black, blank or drawn wrongly

The renderer cannot draw on this display or driver.

1. **Headless Linux (VNC, no monitor):** AetherSDR detects a Wayland session with no connected display and starts on XWayland automatically. If you have set `QT_QPA_PLATFORM` yourself, your setting wins: remove it or set `QT_QPA_PLATFORM=xcb`.
2. **Driver problems:** force software OpenGL with `AETHER_NO_GPU=1` (no rebuild needed). This is worth trying first on a Raspberry Pi.
3. **Help** → **About AetherSDR** shows which renderer is in use. See GPU Rendering (p. 252).

15.1.6 Settings

Your settings came back after you deleted AetherSDR.db

Deleting `AetherSDR.db` by hand is **not** a reset: the next launch re-imports the frozen settings snapshot from your upgrade.

1. Use **Settings** → **Reset Settings....** It writes a backup first, removes AetherSDR's local settings, then quits. Settings stored on the radio are not affected.
2. See Settings and Backups (p. 220).

15.1.7 Reporting a crash

Every release has matching debug-symbol archives, so a crash report from your computer is enough for developers to find the cause. When you file the issue (**Help** → **File an Issue...**), say which download you ran (AppImage, DMG, Windows installer or portable) and attach:

- **Linux:** run `coredumpctl list AetherSDR`, then `coredumpctl info <PID> > aethersdr-crash.txt` and attach the text file. Do **not** attach the core file itself. It is a copy of the program's memory and can contain passwords and tokens.
- **macOS:** the newest `AetherSDR-*.ips` from **Console** → **Crash Reports** or `~/Library/Logs/DiagnosticReports/`. It names your user folder in file paths; edit that out if you prefer.
- **Windows:** Windows keeps a crash dump only if local dumps are switched on. In an administrator PowerShell:

```
$k = 'HKLM:\SOFTWARE\Microsoft\Windows\Windows Error
Reporting\LocalDumps\AetherSDR.exe'
New-Item -Force $k | Out-Null
New-ItemProperty -Force $k -Name DumpType -PropertyType DWord -Value 1 | Out-Null
```

After the next crash, attach the dump from `%LOCALAPPDATA%\CrashDumps\ (AetherSDR.exe.<pid>.dmp)`. The full guide is `docs/debugging-crashes.md`³.

15.1.8 Filing a bug

If nothing here or on the feature's own page fixes the problem, use **Help** → **File an Issue...** It builds a support bundle and opens a pre-filled GitHub bug report. See [Support and Logging \(p. 261\)](#) for which log categories to turn on first.

15.1.9 Reference

File locations

	Linux	macOS	Windows
Settings (AetherSDR.db)	<code>~/.config/AetherSDR/</code>	<code>~/Library/Preferences/AetherSDR/</code>	<code>%LOCALAPPDATA%\AetherSDR\</code>
Logs	<code>~/.config/AetherSDR/logs/</code>	<code>~/Library/Preferences/AetherSDR/logs/</code>	<code>%LOCALAPPDATA%\AetherSDR\logs\</code>

Each launch writes a new timestamped log; attach the newest one. See [Support and Logging \(p. 261\)](#) and [Settings and Backups \(p. 220\)](#).

15.1.10 By feature

These pages have their own Troubleshooting section:

- [Accessibility \(p. 231\)](#)
- [AetherClock and GPS \(p. 161\)](#)
- [Aetherial Audio \(p. 74\)](#)
- [AetherModem Packet Radio \(p. 118\)](#)
- [AetherSweep \(p. 158\)](#)
- [Amplifiers \(p. 170\)](#)
- [ANAN-G2 \(p. 32\)](#)
- [Audio Settings \(p. 217\)](#)
- [Automation Bridge and MCP \(p. 144\)](#)
- [Before You Transmit \(p. 14\)](#)
- [BNR GPU Noise Removal \(p. 248\)](#)
- [Building from Source \(p. 277\)](#)
- [Callsign Lookup \(p. 163\)](#)
- [CAT Control \(p. 132\)](#)
- [Copy Assist \(p. 121\)](#)

³<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

- [CTR2 Proxy \(p. 196\)](#)
- [CW Decoder \(p. 104\)](#)
- [CWX Panel \(p. 102\)](#)
- [D-STAR \(ThumbDV\) \(p. 115\)](#)
- [DAX IQ Streaming \(p. 130\)](#)
- [DAX Virtual Audio \(p. 128\)](#)
- [Demo Mode \(p. 39\)](#)
- [Diversity and ESC \(p. 87\)](#)
- [DSP Noise Mitigation \(p. 241\)](#)
- [DVK Panel \(p. 107\)](#)
- [Firmware Update \(p. 232\)](#)
- [First Connection \(p. 11\)](#)
- [FlexControl Tuning Knob \(p. 183\)](#)
- [FlexRadio \(p. 20\)](#)
- [GPU Rendering \(p. 252\)](#)
- [Green Heron Everywhere \(p. 176\)](#)
- [Hermes-Lite 2 \(p. 23\)](#)
- [Installation \(p. 8\)](#)
- [Keyboard Shortcuts \(p. 224\)](#)
- [KiwiSDR and Web-888 \(p. 36\)](#)
- [Linux \(p. 43\)](#)
- [Low Bandwidth Connections \(p. 208\)](#)
- [macOS \(p. 47\)](#)
- [Manual Connection \(p. 201\)](#)
- [Memory Channels \(p. 92\)](#)
- [Menu Reference \(p. 235\)](#)
- [Meters \(p. 72\)](#)
- [MIDI Controller Mapping \(p. 192\)](#)
- [MQTT Station Automation \(p. 141\)](#)
- [Multi-Flex \(p. 211\)](#)
- [Multi-Slice Operation \(p. 83\)](#)
- [Net Scheduler \(p. 160\)](#)
- [Networked Icom \(p. 28\)](#)
- [NR2 Noise Reduction \(p. 250\)](#)
- [Panadapter Controls \(p. 54\)](#)
- [Peripherals \(p. 166\)](#)
- [Profile Management \(p. 94\)](#)
- [PSK Reporter Map \(p. 155\)](#)
- [RADE Digital Voice \(p. 113\)](#)
- [Radio Setup \(p. 214\)](#)
- [RTL-SDR \(p. 34\)](#)
- [RTTY Operation \(p. 110\)](#)
- [Runtime Monitor \(p. 266\)](#)
- [RX Controls \(p. 64\)](#)
- [Settings and Backups \(p. 220\)](#)
- [ShackSwitch \(p. 175\)](#)
- [Slice Colors \(p. 230\)](#)

- SmartLink Setup (p. 199)
- Split Operation (p. 85)
- SpotHub (p. 148)
- StreamDeck (p. 181)
- Support and Logging (p. 261)
- Tailscale Remote Access (p. 203)
- TCI Server (p. 135)
- TGXL Tuner Control (p. 173)
- Themes and Theme Editor (p. 228)
- TNF (Tracking Notch Filters) (p. 90)
- TX Controls (p. 68)
- Ulanzi Dial (p. 186)
- USB Cable Management (p. 178)
- USB Control Surfaces (p. 189)
- VFO Widget (p. 61)
- Windows (p. 50)
- Workspace Canvas (p. 98)
- WSJT-X Integration (p. 125)

15.1.11 See also

- Support and Logging (p. 261)
- Runtime Monitor (p. 266)
- Settings and Backups (p. 220)
- Installation (p. 8)
- docs/debugging-crashes.md⁴

15.2 Support and Logging

When something isn't working, AetherSDR has several built-in tools for capturing what happened and turning it into a useful report:

- **Help** → **Support & Diagnostics...**: log categories and a live log viewer
- **Help** → **File an Issue...**: builds a support bundle and opens a pre-filled GitHub bug report
- **Help** → **Submit your Idea...**: an AI-assisted reporter for feature requests and conceptual bugs
- **Help** → **Slice Troubleshooting...**: a guided check for slice problems
- **Tools** → **Runtime Monitor...**, **Tools** → **Network Diagnostics...** and **Tools** → **Radio Health...**: see Runtime Monitor (p. 266)

Before you file a report, drop your log or support bundle on the Log Analyzer⁵. It checks for known problems and links the fix for each one it finds. It runs in your browser; the file is never uploaded.

The Help menu also has **Getting Started...**, **AetherSDR Help...**, **What's New...**, three how-to guides (**Understanding Noise Cancellation...**, **Configuring AetherSDR Controls...**, **Configuring Data Modes...**), **AetherSDR Website**, **Donate to AetherSDR**, **Contributing to AetherSDR...**, **Check for Updates...** and **About AetherSDR**. On Windows and Linux the menus are behind the ≡ button at the left of the title bar; on macOS they are in the system menu bar.

⁴<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

⁵<https://docs.aethersdr.com/log-analyzer>

15.2.1 Using Support & Diagnostics

Help → **Support & Diagnostics...** has two parts.

Diagnostic Logging

A grid of checkboxes, one per log category. Hover a box for what it covers. Turn on the categories relevant to your problem, then reproduce it. The dialog advises restarting AetherSDR after changing categories so the new settings are in effect from startup.

Enable All / Disable All toggle every category at once. The categories, and which are on by default, are listed under Log categories ([p. 263](#)).

Log viewer

Below the grid, the dialog shows the path of the current log file, its size, and the last part of the log. **Refresh** re-reads it, **Clear Log** empties it (handy just before reproducing a bug), and **Open Log Folder** opens the log directory in your file manager.

15.2.2 Filing a bug report

Help → **File an Issue...**:

1. **Builds a support bundle:** a timestamped zip, support-bundle-<date>-<time>.zip, in the logs/support/ folder. Its contents are listed under Support bundle contents ([p. 265](#)).
2. **Copies an AI prompt** to your clipboard, pre-filled with your AetherSDR version, Qt version, OS and radio, and opens a dialog with buttons for **Claude**, **ChatGPT**, **Gemini**, **Grok** and **Perplexity**. Paste the prompt into the AI, describe what went wrong, and it writes a structured bug report.
3. **Submit Bug Report** opens GitHub's bug-report form, pre-filled with your system and radio details and a redacted tail of the recent log. If the log tail is too long for the link, it is put on your clipboard instead.
4. Drag the support bundle into the GitHub form.

Reporting a crash

For a crash, also attach the crash report your operating system kept: coredumpctl info on Linux, the .ips file on macOS, or a minidump on Windows. Every release has matching debug-symbol archives, so this is enough for a developer to see exactly where it crashed. Step-by-step instructions are in Troubleshooting ([p. 256](#)) and docs/debugging-crashes.md⁶.

15.2.3 Submitting an idea

Help → **Submit your Idea...** opens the **AI-Assisted Issue Reporter**. It is for **feature requests and conceptual bug reports** rather than runtime diagnostics. No support bundle is built.

1. If a newer release exists, AetherSDR first says so and suggests updating, since your issue may already be fixed.
2. The **AI-Assisted Issue Reporter** offers Claude, ChatGPT, Gemini, Grok and Perplexity. Click one: a structured prompt is copied and that AI opens in your browser.
3. Paste the prompt and replace the bracketed section with your idea or bug. The prompt tells the AI to check the open issue list for duplicates first, then write a complete issue in the project's format.
4. Click **Submit Your Idea** (feature request) or **Report a Bug** to open the matching GitHub issue template, and paste the AI's output.

15.2.4 Finding your version and build details

Help → **About AetherSDR** shows the version and commit, and a build-details card (Qt version, build date and the active renderer) whose text you can select and copy into a bug report. **Contributor Logbook** ↗ opens contributors.aethersdr.com⁷. Esc or Ctrl/Cmd+W closes it.

⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

⁷<https://contributors.aethersdr.com/#all-time>

15.2.5 Before you share a log

- Logs, the GitHub log tail and the support bundle are redacted as described under What is redacted (p. 265), but **skim before you post publicly**.
- `radio-info.json` carries your callsign in clear and the radio's serial and IP only in redacted form.
- Passwords and tokens (SmartLink, MQTT, automation token, Copy Assist API key and others) are kept in the OS keychain, never in the settings, so they cannot appear in `settings.txt`.
- On Linux, never attach a core file. Attach the `coredumpctl info` text instead.

15.2.6 Reference

When to use which

Situation	Use
Crash or wrong behaviour while operating	Enable the relevant categories → reproduce → Help → File an Issue...
Audio glitch, spectrum freeze, DAX problem	Support & Diagnostics... with <i>Audio</i> , <i>VITA-49</i> or <i>DAX</i> ; Tools → Network Diagnostics...
App feels slow or freezes	Tools → Runtime Monitor... ; <i>Performance</i> and <i>Render</i> categories
Connection or discovery problem	<i>Discovery</i> + <i>Connection</i> / <i>Commands</i>
SmartLink problem	<i>SmartLink</i> + <i>Audio</i> + <i>VITA-49</i>
WSJT-X / TCI / CAT keying the wrong slice	<i>TCI</i> / <i>CAT</i> / <i>rigctld</i>
Feature request or design idea	Help → Submit your Idea...

Log categories

Label	Category	Covers
Discovery	<code>aether.discovery</code>	UDP radio discovery broadcasts
Connection / Commands	<code>aether.connection</code>	Raw TCP command channel: commands sent, responses, socket state
Protocol / Status	<code>aether.protocol</code>	Parsed SmartSDR protocol handling and status updates
Audio	<code>aether.audio</code>	RX/TX audio, device negotiation, volume
Audio Summary	<code>aether.audio.summary</code>	Audio routing and device summaries for support logs
VITA-49	<code>aether.vita49</code>	UDP packet routing: FFT, waterfall, meters, DAX
DSP	<code>aether.dsp</code>	Client noise reduction and CW decoder processing
RADE	<code>aether.rade</code>	FreeDV RADE digital voice
SmartLink	<code>aether.smartlink</code>	SmartLink sign-in, TLS tunnel, WAN streaming
TCI / CAT / rigctld	<code>aether.cat</code>	TCI server (slice and DAX arming, TX audio summary), rigctld TCP servers, PTY virtual serial ports
DAX	<code>aether.dax</code>	Virtual audio bridge (PipeWire / CoreAudio)
Meters	<code>aether.meters</code>	Meter definitions and value conversion
Transmit	<code>aether.transmit</code>	TX state, ATU, profiles, power control
Firmware	<code>aether.firmware</code>	Firmware download, staging, upload
Tuner/AGM	<code>aether.tuner</code>	TGXL tuner, Antenna Genius state
Peripheral Authorization	<code>aether.peripheral.auth</code>	Peripheral access-code storage (code values are never logged)

Label	Category	Covers
GUI	aether.gui	Windows, applets, dialogs
DX Cluster	aether.dxcluster	Cluster telnet connection and spot parsing
MQTT	aether.mqtt	MQTT client connection and messages
RBN	aether.rbn	Reverse Beacon Network connection and spots
Ext Devices	aether.devices	Serial ports, FlexControl, MIDI, HID encoders
Performance	aether.perf	Render timing and CPU profiling
Render	aether.render	GPU/QRhi path selection and fallback, paint stalls
Propagation	aether.propforecast	Solar and propagation forecast updates
CW / netCW	aether.cw	CW keying, MIDI paddle, iambic and netCW timing
S History	aether.shistory	Signal History voice detection
AetherModem / AX.25 Link	aether.ax25, aether.ax25.link	Packet modem, and connected-mode link timing and retransmits
Waveform	aether.waveform	Waveform install upload and the local waveform helper (D-STAR)
KiwiSDR / KiwiSDR Audio/DSP	aether.kiwisdr, aether.kiwisdr.audio	KiwiSDR receivers; the audio category is high-rate
Automation Bridge	aether.automation	The agent automation bridge
QRZ Lookup	aether.qrz	QRZ.com lookups and cache
AetherClock	aether.clock	WWV/WWVB time-signal decoder
Hermes-Lite 2 / Hermes-Lite 2 TX	aether.hl2, aether.hl2.tx	HL2 backend; the TX category is high-rate transmit telemetry
ANAN Protocol 2	aether.anan.p2	ANAN-G2 Protocol 2 session
Icom Session / Streams / CI-V / Scope / Link / Credentials	aether.icom.*	Networked Icom backend; Icom CI-V logs every frame and is high-rate
System Info	aether.sysinfo	Startup hardware inventory: OS, CPU and SIMD features, RAM, GPU

On by default: Discovery, Connection / Commands, Protocol / Status, Audio Summary, KiwiSDR and System Info. Everything else is off until you turn it on. A category that is off still records its warnings and errors.

Log files

Platform	Log folder
Linux	~/.config/AetherSDR/logs/
macOS	~/Library/Preferences/AetherSDR/logs/
Windows	%LOCALAPPDATA%\AetherSDR\logs\

- Every launch starts a new file named `aethersdr-<date>-<time>.log`. Attach the newest one. On Linux and macOS, `aethersdr.log` in the same folder is a link to it; on Windows it is a `.lnk` shortcut.
- A log that grows past 100 MB is rotated to a new file.
- Old logs are pruned at startup after 7 days or once they total more than 500 MB; the two most recent are always kept.
- Logging is buffered and written in the background, so it does not slow the app down.

What is redacted

Every line is scrubbed before it is written: IP addresses (all but the last part), MAC addresses, radio serial numbers (all but the last group), your home-directory path, email addresses, tokens and passwords, personal names, GPS coordinates and grid squares. Your **callsign**, radio model, firmware version, frequencies and modes are deliberately kept, because they are what makes a log useful. Skim a log before sharing it anyway.

Driving categories from the command line

The categories are Qt logging categories, so QT_LOGGING_RULES can raise one before the GUI opens. For example, to log one line for every TCI transmit-routing decision (useful when TCI keys the wrong slice):

```
QT_LOGGING_RULES="aether.cat.info=true" ./AetherSDR-*.AppImage
```

For anything else, prefer the checkboxes in **Help → Support & Diagnostics....**

Support bundle contents

File	Contents
aethersdr.log (plus up to two older logs)	Recent logs, re-scrubbed on the way out
system-info.json	AetherSDR version, Qt version, OS, kernel, CPU model with SIMD features, architecture, RAM, GPU, build date
radio-info.json	Radio model, firmware, protocol version and callsign, with serial and IP redacted
settings.txt	A sanitized dump of your AetherSDR settings with every password and token removed, plus any settings-store notice. See Settings and Backups (p. 220).
enabled-categories.txt	Which log categories were on

15.2.7 Known issues

- With WNB on, a FLEX-8600 fills the default-on Connection / Commands log category with a `wnb Updating` status line every few seconds (#6069⁸).

15.2.8 Troubleshooting

The log has no detail about your problem

The relevant log category is off, so only its warnings and errors were recorded.

1. Open **Help → Support & Diagnostics...** and tick the categories for your problem. When to use which (p. 263) suggests a set.
2. Restart AetherSDR so the categories are on from startup.
3. Click **Clear Log**, reproduce the problem, then use **Help → File an Issue....**

The GitHub bug report form has no log tail

The log tail was too long to fit in the link, so AetherSDR put it on your clipboard instead.

1. Paste the clipboard into the GitHub form.
2. Drag the support bundle from `logs/support/` into the form as well.

You can't find the support bundle

The bundle is written to the `support/` folder inside the log folder listed under Log files (p. 264).

1. Open **Help → Support & Diagnostics...** and click **Open Log Folder**.
2. Open the support folder and take the newest `support-bundle-<date>-<time>.zip`.

15.2.9 See also

- Troubleshooting (p. 256): symptom index and crash-report instructions
- Runtime Monitor (p. 266)

⁸<https://github.com/aethersdr/AetherSDR/issues/6069>

- Settings and Backups (p. 220): **Settings** → **Reset Settings...** and **Settings** → **Settings Browser...**
- Contributing Guide (p. 269)
- docs/debugging-crashes.md⁹

15.3 Runtime Monitor

Tools → **Runtime Monitor...** shows how AetherSDR itself is behaving on your computer: CPU use, the busiest threads, memory and how responsive the user interface is. Use it when the app feels sluggish, audio stutters, or the panadapter stalls, to see whether the cause is on the computer rather than the radio or the network.

The same group at the bottom of the **Tools** menu has two related diagnostic windows, **Radio Health...** and **Network Diagnostics...**, described further down.

| **Upgrading from an older version:** this window was called **System Info**.

15.3.1 Using the Runtime Monitor

A **Timeframe** selector in the window header (1 minute, 5 minutes (default), 15 minutes or 1 hour) applies to every chart. It is hidden on the Threads and Logs tabs, which have no adjustable time axis.

Overview

Four cards across the top, each with a chart below:

Card	Meaning
CPU Total	AetherSDR's share of the whole machine
Max Thread	The busiest single thread's share of one core, and its name
Memory	Resident memory the OS currently holds for AetherSDR
GUI Tick Lag	How late the user-interface thread's 50 ms heartbeat fired, at worst, during the last sample. Zero is on time; a large value means the interface was busy for that long.

The charts show CPU (process and busiest thread), memory (resident and peak), the top five threads, and GUI tick lag (worst and mean) over the chosen timeframe.

Why Max Thread matters: AetherSDR's typical stall is one thread pinned at 100 % of one core while the others idle. The machine-wide CPU figure (and the status-bar CPU readout) can look low while that happens. Max Thread shows it directly.

Threads

A live table of every AetherSDR thread with **Thread**, **TID**, **State**, **CPU %** (share of one core), **Peak 60 s**, **Total CPU (s)** and a **Last 60 s** sparkline. The summary line above the table turns red when any thread crosses 90 % of a core.

Windows does not report per-thread state, so the State column shows a dash there.

Memory

Resident, peak resident, private memory and virtual address space, with a chart of resident, private and peak memory over the timeframe. Virtual address space is reserved space, not memory in use, so a large figure there is normal. Click a legend entry to show one series alone.

Logs

A live tail of the current log file, filtered to the three performance-related categories: **Performance**, **Render** and **Audio**. The file path is shown above the viewer.

- **Live** follows new lines as they arrive. Scrolling up pauses following so you can read around a stall; press **Live** again to resume.

⁹<https://github.com/aethersdr/AetherSDR/blob/main/docs/debugging-crashes.md>

- A category that is switched off in **Help** → **Support & Diagnostics...** is shown dimmed. Only its warnings and errors reach the log until you turn it on there.

To capture more detail, turn on the **Performance** and **Render** categories in **Help** → **Support & Diagnostics...** before reproducing the problem. See Support and Logging (p. 261).

15.3.2 Using Radio Health

Tools → **Radio Health...** lists the health and status registers that the connected radio itself reports, refreshed every 500 ms. A dash means the radio has not reported that value, which is not the same as zero. **Copy** puts the whole table on the clipboard as text for a bug report.

What appears depends on the radio:

- **FlexRadio:** Flex radios do not publish health registers through this window, so it reads “This radio reports no health registers.” Use the **MTR** (Radio Vitals) applet for PA temperature, fan speed and supply voltage. See Meters (p. 72).
- **Hermes-Lite 2** (p. 23): link, temperature, supply, PA current and gateway, a transmit voice-chain section (mic level and gain, ALC, forward power) and converter (ADC) rows. Some rows, such as whether another client is holding the radio, whether it is reachable and its PA temperature, are available even before you connect.
- **Networked Icom** (p. 28): includes the DATA OFF MOD and DATA MOD input selections and the MIC/USB/ACC/WLAN/LAN levels.
- **RTL-SDR** (p. 34): the capture window (centre, rate and edges) and receive-pipeline counters such as queue drops and late frames.

Radio Health is a different thing from the **HLTH** (Antenna Health) applet, which trends SWR and forward power for your antenna system.

15.3.3 Using Network Diagnostics

Tools → **Network Diagnostics...** shows the health of the link between AetherSDR and the radio. A **Connect by IP** connection also has a **Network Diagnostics** button beside the connect button.

Pages are grouped in a tree on the left:

Group	Pages
STATUS	Overview (status, latency, packet loss and audio-buffer cards with summary charts), Connection Details
TRENDS	Latency & Jitter , Stream Rates (including the <i>Adaptive Throttle — FPS Cap</i> chart when the throttle is on), Packet Loss , Audio Health (RX audio buffer and timing, per-stream delivery table), Digital Voice (when a digital-voice waveform is active)
SUPPORT	Application Logs (category filters, Live/Paused), TCI Clients (when the TCI server is running: connected clients, their likely role, and a TCI message monitor you can pause, clear, filter and save)

- The chart **Timeframe** offers 1 minute, 5 minutes, 15 minutes, 1 hour, 1 day and 1 week.
- **Search diagnostics** (Ctrl+F, ⌘F on macOS) filters the pages by symptom, stream, protocol or measurement. Try *jitter*, *underrun*, *UDP*, *packet gaps*, *bitrate* or *WSJT-X*.

For what to do about a poor link, see Low Bandwidth Connections (p. 208).

15.3.4 Troubleshooting

AetherSDR is sluggish but CPU use looks low

One thread is pinned at 100 % of one core while the others idle, so the machine-wide figure stays low.

1. Open **Tools** → **Runtime Monitor...** and read the **Max Thread** card, which names the busiest thread.
2. Check **GUI Tick Lag**. A large value means the interface thread was busy for that long.
3. Turn on the **Performance** and **Render** log categories in **Help** → **Support & Diagnostics...**, reproduce the stall, and attach the log to your report. See Support and Logging (p. 261).

Radio Health says “This radio reports no health registers”

You are connected to a FlexRadio, which does not publish health registers through this window.

1. Open the **MTR** (Radio Vitals) applet for PA temperature, fan speed and supply voltage. See Meters ([p. 72](#)).

A category in the Logs tab is dimmed

That log category is switched off, so only its warnings and errors reach the log.

1. Open **Help** → **Support & Diagnostics...** and tick the category.
2. Restart AetherSDR so the category is on from startup, then reproduce the problem.

15.3.5 See also

- Support and Logging ([p. 261](#)): log categories, the support bundle and File an Issue
- Low Bandwidth Connections ([p. 208](#))
- GPU Rendering ([p. 252](#)): renderer details also appear in **Help** → **About AetherSDR**
- Troubleshooting ([p. 256](#))

Contributing

16.1 Contributing Guide

AetherSDR welcomes bug reports, feature ideas, documentation and code. The rules live in the repository; this page is a short orientation.

16.1.1 Requirements

Read these before you contribute:

Document	What it is for
CONTRIBUTING.md (https://github.com/aethersdr/AetherSDR/blob/main/CONTRIBUTING.md)	Contribution policy: what is accepted, who reviews what
CONSTITUTION.md (https://github.com/aethersdr/AetherSDR/blob/main/CONSTITUTION.md)	The project's governing principles. Read it before writing or reviewing code.
GOVERNANCE.md (https://github.com/aethersdr/AetherSDR/blob/main/GOVERNANCE.md)	Roles, decisions and the RFC process
AGENTS.md (https://github.com/aethersdr/AetherSDR/blob/main/AGENTS.md)	The canonical project guide: architecture, conventions and must-knows, read by every contributor and AI tool
docs/DEVELOPER-GUIDE.md (https://github.com/aethersdr/AetherSDR/blob/main/docs/DEVELOPER-GUIDE.md)	Architecture, threading, protocol reference, coding conventions
docs/first-contribution-cheatsheet.md (https://github.com/aethersdr/AetherSDR/blob/main/docs/first-contribution-cheatsheet.md)	A beginner's on-ramp
docs/COMMIT-SIGNING.md (https://github.com/aethersdr/AetherSDR/blob/main/docs/COMMIT-SIGNING.md)	Setting up commit signing on every platform

Where AGENTS.md or the developer guide appears to conflict with CONSTITUTION.md or GOVERNANCE.md, the constitution and governance win.

16.1.2 Contributing code

1. Browse open issues¹. Issues labelled `good first issue` are good starting points.
2. Fork the repository and create a feature branch from `main`.
3. Build it. **Qt 6.12 is required**; `scripts/setup/setup-qt.sh` installs it. See Building from Source (p. 277).
4. Implement the fix or feature (one issue per PR).

¹<https://github.com/aethersdr/AetherSDR/issues>

5. **Sign your commits.** `main` requires signed commits. SSH signing is the simple path; see [docs/COMMIT-SIGNING.md](#)².
6. Open a pull request against `main` referencing the issue (Fixes #42).

Every PR needs green CI and review from a code owner other than yourself.

Most AetherSDR development is done with AI coding tools. See AI-Assisted Development ([p. 275](#)).

Key rules

- **Use `AppSettings`, never `QSettings`.** Settings live in the SQLite store; credentials go to the OS keychain, never into settings.
- **Every colour is a theme token.** Take colours from `ThemeManager` tokens, never hard-coded hex values, so the Default Light theme and user themes work. See [docs/theming/](#) and [docs/style/](#) in the repository.
- **Dim, don't hide.** A control the connected radio cannot use stays visible but dimmed, with the reason in its tooltip.
- **Accessible names.** New widgets need accessible names so screen readers can use them (see [docs/ally.md](#)).
- **No feedback loops.** Use `QSignalBlocker` when updating the UI from model signals.
- **Multi-Flex safety.** Filter radio data by `client_handle`.
- **The radio is authoritative** for its own live state; `FlexLib` is the reference for protocol behaviour.
- **Cite the principle** your change honours at the end of the commit subject, for example `Principle V`.
- **One PR per issue.** Keep changes focused, and don't reformat code you are not changing.
- **Test the RX chain.** Discovery → connect → FFT → audio must still work.
- **Open an [RFC] issue first** for any UX, visual or architecture change.
- **Native only.** No Wine or CrossOver workarounds.

16.1.3 Writing documentation

These docs live in the AetherSDR repository under `docs/user/docs/`³, one Markdown file per page, and change through an ordinary pull request. Every page follows a standard skeleton, status front matter and set of writing rules, described in the Docs Style Guide ([p. 271](#)). Read it before adding or restructuring a page.

- **Fixing a page:** use the **Edit this page** link at the bottom of it, or edit the file in your fork, and open a pull request against `main`.
- **Changing behaviour:** a pull request that changes what users see or do updates the matching page in the same pull request.
- **Checking your change:** `npm run build` in `docs/user` fails on any link to a page or heading that does not exist. `docs/user/README.md`⁴ covers previewing, screenshots and the generated reference pages.
- **Translating:** see [Translating the Docs](#) ([p. 274](#)).

16.1.4 Reporting bugs

The fastest path is **Help → File an Issue...** inside AetherSDR. It builds a support bundle, copies an AI prompt to your clipboard, and opens GitHub's bug-report form pre-filled with your system details and a redacted log tail. See [Support and Logging](#) ([p. 261](#)) for log categories and privacy notes, and [Troubleshooting](#) ([p. 256](#)) for how to attach a crash report.

If you'd rather file by hand, use the bug report template⁵ and include:

- what happened vs. what you expected
- radio model and firmware version
- OS and AetherSDR version (**Help → About AetherSDR** shows both and lets you copy the build details)

²<https://github.com/aethersdr/AetherSDR/blob/main/docs/COMMIT-SIGNING.md>

³<https://github.com/aethersdr/AetherSDR/tree/main/docs/user/docs>

⁴<https://github.com/aethersdr/AetherSDR/blob/main/docs/user/README.md>

⁵https://github.com/aethersdr/AetherSDR/issues/new?template=bug_report.yml

- the log: in `~/.config/AetherSDR/logs/` on Linux, `~/Library/Preferences/AetherSDR/logs/` on macOS, `%LOCALAPPDATA%\AetherSDR\logs\` on Windows

16.1.5 Requesting features

Use **Help** → **Submit your Idea...** . It copies a structured prompt into Claude, ChatGPT, Gemini, Grok or Perplexity, so you can describe your idea in plain English and have the AI write the issue body, then opens the right GitHub template.

To file by hand, use the feature request template⁶. Describe the problem you are solving, reference SmartSDR behaviour where it helps (screenshots are welcome), and keep to one feature per issue.

16.1.6 See also

- AI-Assisted Development (p. 275)
- Building from Source (p. 277)
- Docs Style Guide (p. 271)
- Translating the Docs (p. 274)
- Support and Logging (p. 261)
- CONTRIBUTING.md⁷

16.2 Docs Style Guide

How AetherSDR documentation pages are laid out and written. Every page follows the same skeleton, so readers know where to look, and contributors (human or AI) know what to write. The approach borrows from the ArchWiki style guide⁸, the Home Assistant documentation templates⁹ and the Diátaxis¹⁰ framework.

16.2.1 Page Types

Each page is mainly one of four kinds. Don't mix them on one page; link instead.

Type	Answers	Examples
Tutorial	"Teach me" — a guided first experience with a guaranteed result	Your First Session (p. 5), Before You Transmit (p. 14)
How-to / feature guide	"Help me do X" — setup and use of one feature	CAT Control (p. 132), Split Operation (p. 85), Hermes-Lite 2 (p. 23)
Reference	"Look it up" — complete, dry, accurate tables	Menu Reference (p. 235), Keyboard Shortcuts (p. 224)
Explanation	"Help me understand" — background and trade-offs	DSP Noise Mitigation (p. 241), GPU Rendering (p. 252)

Landing pages (for example FlexRadio (p. 20), Linux (p. 43), macOS (p. 47), Windows (p. 50), Supported Radios (p. 18)) are a second way in: they are organized by the reader's situation and link across all four types.

16.2.2 Page Skeleton

Each page is one Markdown file in `docs/user/docs/`¹¹, named after its slug (`cat-control.md` is served at `/cat-control`). Sections appear in this order. Omit a section that has nothing in it; never leave an empty heading.

⁶https://github.com/aethersdr/AetherSDR/issues/new?template=feature_request.yml

⁷<https://github.com/aethersdr/AetherSDR/blob/main/CONTRIBUTING.md>

⁸<https://wiki.archlinux.org/title/Help:Style>

⁹<https://developers.home-assistant.io/docs/documenting/create-page>

¹⁰<https://diataxis.fr/>

¹¹<https://github.com/aethersdr/AetherSDR/tree/main/docs/user/docs>


```

---
title: "Page Title"
slug: "/page-title"
description: "One sentence: what this is. Search results and link previews show it."
status: "Experimental"
applies_to: ["Hermes-Lite 2"]
platforms: ["Linux", "macOS", "Windows"]
---

:::info[Status]

**Status:** Experimental · **Applies to:** Hermes-Lite 2 · **Platforms:** Linux, macOS,
Windows

:::

One or two untitled introductory paragraphs: what this is and why you'd use it.

## Requirements
## Setup
## Using <the feature>          (one or more task-oriented sections)
## Reference                    (tables of controls, settings, ranges, defaults)
## Known issues
## Troubleshooting
## Making a change

These docs are Markdown in the AetherSDR repository and change through an ordinary pull
request:

1. Edit or add the page under `docs/user/docs/` (and `docs/user/sidebars.js` for a new
page). The **Edit this page** link at the bottom of every page opens the file on GitHub.
2. Preview it with `npm start` in `docs/user`, or run `npm run build` to get the full
link check. [`docs/user/README.md`](https://github.com/aethersdr/AetherSDR/blob/main/docs/user/README.md) has the details.
3. Open a pull request against `main`. A change to user-visible behaviour updates the
matching page in the same pull request as the code.

The pages under `generated/` are written by `tools/docs/gen_reference.py` from the
source; change the source and re-run the script rather than editing them.

## See also

```

The page title comes from the `title` front matter; don't repeat it as a `#` heading. Every new page also needs an entry in `docs/user/sidebars.js`¹², or readers can only reach it through links.

Status

Include the status fields and the matching `:::info[Status]` callout when support differs by radio or platform, or when the feature is not fully supported. Leave both out of pages that apply everywhere. The front matter is the machine-readable copy and the callout is what readers see, so keep them in step.

- **status** is one of: **Supported** · **Early** · **Experimental** · **Experimental, receive-only** · **Off by default**. These match the README's hardware tiers.

¹²<https://github.com/aethersdr/AetherSDR/blob/main/docs/user/sidebars.js>

- **Mixed support:** when a page covers families at different tiers, **status** gives the tier of the best-supported family (usually FlexRadio), and **applies_to** marks every other family with its own tier in brackets, for example ["FlexRadio", "Hermes-Lite 2 (experimental)", "Networked Icom (early; IC-7300MK2 supported)"].
- **applies_to** lists the radio families the page covers: FlexRadio, Hermes-Lite 2, Networked Icom, ANAN-G2, RTL-SDR, KiwiSDR/Web-888, Demo. Use ["All radios"] when it has no limits.
- **platforms** is included only when something is platform-limited. For example, MNR is macOS only, and BNR is Linux and Windows.

Known issues vs Troubleshooting

These are separate sections and must not be merged.

- **Known issues** lists unsolved problems and current limitations. **Every entry links its open GitHub issue** in this form: Short description of the symptom ([#1234](https://github.com/aethersdr/AetherSDR/issues/1234)). When the issue closes, the entry is deleted. Don't write dates or version numbers into an entry; the linked issue carries that history.
- **Troubleshooting** covers problems that have a fix. Write each one as a ### heading naming the **symptom** as the user experiences it, followed by the likely cause and numbered steps to fix it:

No audio from the speaker

The PC Audio path is off, or the wrong output device is selected.

1. Check the title-bar speaker icon: ...
2. Open ****Settings → Radio Setup... → Audio**** and ...

See also

See also comes last. It holds links to other pages of these docs first, then external links. Never call it “External links” or “More resources”.

16.2.3 Writing Rules

- **Describe current behaviour.** No “previously”, no “as of v26.x”, no history. A one-time behaviour change that existing users will notice goes in a callout that starts > ****Upgrading from an older version:****.
- **Use exact UI names in bold**, with menu paths written as **Tools → PSK Reporter...** and settings as **Settings → Radio Setup... → Peripherals**. Check them against `src/gui/MainWindow_Menus.cpp`.
- **Links between pages** are relative file links with the page title as the text: [CAT Control](./cat-control.md), or [pinning](./smartlink-setup.md#certificate-pinning) for a heading. The site build fails on a link to a page or heading that does not exist. Link source docs with full GitHub URLs, such as <https://github.com/aethersdr/AetherSDR/blob/main/docs/HERMES.md>.
- **Placeholders** go in code: `<name>`. A bare `<name>` in prose is read as an HTML tag and disappears.
- **Spelling and voice:** British spelling (colour, behaviour), second person, short sentences.
- **Never invent a fact.** If you can't verify it against the source, leave it out.
- **Settings** live in AetherSDR.db. See Settings and Backups (p. 220).

16.2.4 See also

- Contributing Guide (p. 269)
- AI-Assisted Development (p. 275)
- Translating the Docs (p. 274)
- Diátaxis¹³

¹³<https://diataxis.fr/>

- ArchWiki: Help:Style¹⁴

16.3 Translating the Docs

How to add and maintain a translation of the AetherSDR documentation. The site is built with Docusaurus¹⁵, which keeps each language in its own folder next to the English pages, so a translation is a set of Markdown and JSON files that change through ordinary pull requests.

English is the source. Today it is the only language; the site is set up so that adding one is a pull request, not a rebuild.

Only these docs can be translated. The AetherSDR application itself is not translated yet: its menus, dialogs and messages are English only.

16.3.1 Requirements

- **At least two contributors per language.** A new language starts only when two people who read and write it commit to it: one translates, the other reviews, and they swap roles. A language with one contributor cannot be reviewed, so it is not added.
- **Nobody approves their own translation.** Every translation pull request is reviewed and approved by someone other than its author, and for the wording, by another speaker of that language.
- **Normal pull requests.** Translations follow the same rules as any other change: a fork, a branch, signed commits and review. See the Contributing Guide (p. 269).
- Node.js 20 or newer, to preview the site (see `docs/user/README.md`¹⁶).

Open an issue proposing the language first, naming both contributors. The pull request that adds it can follow once a maintainer agrees.

16.3.2 Adding a language

The examples use French (`fr`); use your language's locale code¹⁷.

1. Add the locale to `il8n.locales` in `docs/user/docusaurus.config.js`, for example `locales: ['en', 'fr']`. The commented example there shows the optional `localeConfigs` entry for the language's name in the menu.
2. Generate the files for the site's own text (navbar, footer, sidebar category names, version labels):

```
cd docs/user
npm ci
npm run write-translations -- --locale fr
```

This writes JSON files under `il8n/fr/`. Translate the `message` values and leave the keys alone.

3. Copy the pages you are translating into the locale's docs folder, keeping the file names, and translate them:

```
mkdir -p il8n/fr/docusaurus-plugin-content-docs/current
cp docs/first-connection.md il8n/fr/docusaurus-plugin-content-docs/current/
```

4. Preview the language: `npm start -- --locale fr`. Run `npm run build` before opening the pull request; it builds every locale and fails on a broken link or heading in any of them.

A page you have not translated yet falls back to the English one, so a language can start with a few pages and grow.

¹⁴<https://wiki.archlinux.org/title/Help:Style>

¹⁵<https://docusaurus.io/>

¹⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/user/README.md>

¹⁷https://en.wikipedia.org/wiki/IETF_language_tag

16.3.3 The i18n/ layout

```
docs/user/i18n/fr/
├── code.json                                # text in the site's own components
├── docusaurus-theme-classic/
│   ├── navbar.json
│   └── footer.json
├── docusaurus-plugin-content-docs/
│   ├── current.json                       # sidebar labels, Next (main)
│   ├── current/                           # translated pages, Next (main)
│   │   └── first-connection.md
│   ├── version-stable.json                # sidebar labels, Stable
│   └── version-stable/                    # translated pages, Stable
└── ...
```

Translate in `current/`, which follows `main`, the same as the English pages in `docs/user/docs/`. The `version-stable/` folder is the translation of the stable snapshot; it is refreshed from `current/` when a release is prepared, the same way the English snapshot is.

Keep each translated file's front matter `slug` and every heading's meaning: links from other pages point at the slug and at the heading anchors. To keep an anchor stable when the heading text changes in translation, give the heading an explicit id, for example `## Première connexion {#first-connection}`.

16.3.4 Keeping a translation in sync

The English pages change with the application, and a translation does not change with them. To find what has moved on since you last updated a page, list the English page's history after the date of your last translation commit:

```
git log --oneline --since=2026-10-01 -- docs/user/docs/first-connection.md
git diff <commit>..main -- docs/user/docs/first-connection.md
```

Bring the translation up to date in a pull request that names the English commit it now matches. When an English page is renamed or removed, rename or remove the translated file in the same way, or the build fails on its links. Run `npm run write-translations -- --locale fr` again after sidebar or navbar changes: it adds new strings and keeps the ones you have already translated.

A translated page that is badly out of date is worse than the English fallback. If a language has no one keeping it current, its maintainers may remove the stale pages so readers get the English ones.

16.3.5 See also

- Contributing Guide (p. 269)
- Docs Style Guide (p. 271)
- Docusaurus: i18n tutorial¹⁸
- Kubernetes: Localizing Kubernetes documentation¹⁹

16.4 AI-Assisted Development

AetherSDR is developed through an AI-augmented open-source workflow. The project lead (Jeremy, KK7GWY) and the core contributors work mainly through Claude Code²⁰; other contributors use Codex, Copilot, Cursor,

¹⁸<https://docusaurus.io/docs/i18n/tutorial>

¹⁹<https://kubernetes.io/docs/contribute/localization/>

²⁰<https://claude.com/claude-code>

²¹<https://github.com/aethersdr/aetherclaude>

Gemini, Aider and more; and the AetherClaude²¹ bot triages issues, drafts implementation plans and opens PRs for issues labelled `aetherclaude-eligible`.

Every change passes the same gate whichever tool (or person) produced it: signed commits, green CI and human code-owner review before anything reaches `main`.

You don't need to be a C++ developer to contribute.

16.4.1 The guide every AI tool reads

AGENTS.md²² is the canonical project guide: architecture, conventions, the must-knows, and path-scoped sub-guides under `docs/agents/` (settings, GUI, backends, Flex protocol, tests and CI). Each tool's own instruction file (`CLAUDE.md`, `GEMINI.md`, `.github/copilot-instructions.md`) is just a pointer to it.

The Constitution²³ and **GOVERNANCE.md**²⁴ outrank **AGENTS.md**. **GOVERNANCE.md** also sets the limits on what an AI agent may change without a person deciding.

16.4.2 Setup

1. Install an AI coding tool. The project recommends **Claude Code** for consistency with the core team.
2. Fork the repository: <https://github.com/aethersdr/AetherSDR/fork>
3. Clone your fork: `git clone https://github.com/YOUR_USERNAME/AetherSDR.git`
4. Install Qt 6.12 with `scripts/setup/setup-qt.sh` and build. See Building from Source (p. 277).
5. Set up commit signing. Tell your assistant *“read docs/COMMIT-SIGNING.md and help me set up commit signing”*; the top of that file is written for AI assistants to follow.

16.4.3 Starting a feature

Start with a prompt like:

Read **AGENTS.md**, **CONTRIBUTING.md** and **CONSTITUTION.md**. I want to implement [feature] for issue #[number].
Plan the implementation before writing code.

Tips

- **Be specific.** “Add a button that does X” works better than “improve the UI”.
- **Test incrementally.** Build and run after each change.
- **Point at existing code**, for example “look at how `RxApplet` handles filters”.
- **One issue per PR.**
- **Ask for an RFC** before any visual, UX or architecture change. AI agents must not decide those on their own.

16.4.4 Rules your assistant must follow

AGENTS.md has the full list. The ones that catch people most often:

1. Use `AppSettings`, never `QSettings`.
2. Every colour is a `ThemeManager` token, with no hard-coded hex.
3. A control the radio cannot use is dimmed with a reason, never hidden.
4. Use `QSignalBlocker` when updating the UI from model signals.
5. Filter radio data by `client_handle` for Multi-Flex safety.
6. Sign every commit, and end the commit subject with the principle it honours (Principle <N>.).
7. The core RX path (discovery → connect → FFT → audio) must still work.

²²<https://github.com/aethersdr/AetherSDR/blob/main/AGENTS.md>

²³<https://github.com/aethersdr/AetherSDR/blob/main/CONSTITUTION.md>

²⁴<https://github.com/aethersdr/AetherSDR/blob/main/GOVERNANCE.md>

16.4.5 Letting an AI drive AetherSDR

AetherSDR has a built-in **agent automation bridge** with an MCP server, so an AI assistant can observe and operate a running AetherSDR. This is useful for reproducing a bug and proving a fix. It is off by default; turn it on under **Radio Setup** → **Network** → **Advanced** → **Agent Automation (MCP)**. Transmit control is a separate opt-in. See Automation Bridge and MCP (p. 144).

16.4.6 AI-to-AI debugging

If your agent needs to coordinate with the maintainers:

1. Open a GitHub issue with your AI's analysis, code references and proposed fix.
2. Include relevant log output. See Support and Logging (p. 261) for where logs live and which categories to enable.
3. The maintainers' agents monitor issues and will respond.

16.4.7 See also

- Contributing Guide (p. 269)
- Building from Source (p. 277)
- Automation Bridge and MCP (p. 144)
- Docs Style Guide (p. 271)
- CONTRIBUTING.md²⁵
- AGENTS.md²⁶

16.5 Building from Source

Most operators should use the pre-built downloads on Installation (p. 8). Build from source if you want to test a branch, develop AetherSDR, or package it for a distribution.

The authoritative instructions are in the repository and are kept current with every change. This page is a summary:

- README: Building from Source²⁷: dependency lines and the quick build
- docs/BUILDING.md²⁸: Windows 11 and macOS step-by-step, what each dependency enables, distro notes, GPU rendering and Wayland
- BUILD-OPTIONS.md²⁹: every compile-time switch, its default and its prerequisites

16.5.1 Requirements

Qt 6.12

Every source build needs **Qt 6.12**, the same Qt every release is built with. Few distributions package it yet, so the repository provides a script that installs exactly the release Qt plus qtkeychain into a per-user cache that CMake finds on its own:

- Linux and macOS: `scripts/setup/setup-qt.sh`
- Windows: `scripts\setup\setup-qt.ps1`

The script checks your machine before downloading (about 2 GB): glibc 2.34+ on x86_64 or 2.38+ on aarch64, Xcode 16+ on macOS, a working python3 -m venv (on Debian, Ubuntu and Raspberry Pi OS: `sudo apt install python3-venv`) and enough free disk space. Re-running it is a no-op once Qt is installed. A distribution Qt that is already 6.12 or newer also works.

²⁵<https://github.com/aethersdr/AetherSDR/blob/main/CONTRIBUTING.md>

²⁶<https://github.com/aethersdr/AetherSDR/blob/main/AGENTS.md>

²⁷<https://github.com/aethersdr/AetherSDR#building-from-source>

²⁸<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md>

²⁹<https://github.com/aethersdr/AetherSDR/blob/main/BUILD-OPTIONS.md>

Machines the release Qt cannot support cannot build AetherSDR from source: Linux below those glibc versions, CPU architectures Qt publishes no binaries for, and Macs below macOS 14.5 (Xcode 16 needs 14.5).

Dependencies

Everything except Qt and qtkeychain comes from the system. Copy the current line for your distribution from the README³⁰. It covers Arch / CachyOS / Manjaro, Debian / Ubuntu / Linux Mint, Fedora and macOS (Homebrew), and includes FFTW, RTL-SDR, PortAudio, hidapi, PipeWire and the X11/xcb libraries Qt's binaries need.

Optional libraries enable optional features; the build succeeds without them and the corresponding feature is left out. docs/BUILDING.md³¹ has the dependency-to-feature table.

On macOS, do not also install Homebrew's qt. Two Qt installations visible to CMake at once break the build.

Windows toolchain

You need Visual Studio 2022 **17.14 or newer** with the C++ workload, CMake 3.25+, Ninja, Git and Python 3.

16.5.2 Setup

Build on Linux and macOS

```
git clone https://github.com/aethersdr/AetherSDR.git
cd AetherSDR
scripts/setup/setup-qt.sh          # Qt 6.12 (cached per user) + qtkeychain
cmake -B build -G Ninja -DCMAKE_BUILD_TYPE=RelWithDebInfo
cmake --build build -j$(nproc)
./build/AetherSDR
```

Build on Windows 11

From an MSVC developer prompt:

```
powershell -File scripts\setup\setup-qt.ps1
powershell -File scripts\setup\setup-fftw.ps1
cmake -B build -G Ninja -DCMAKE_BUILD_TYPE=RelWithDebInfo
cmake --build build --target AetherSDR
```

See docs/BUILDING.md § Windows 11³² for the full sequence, including the optional dependency scripts.

Install (optional, Linux)

```
sudo cmake --install build
```

This installs the binary, a desktop entry and the icon.

16.5.3 Developing AetherSDR

AetherSDR is C++20 and Qt 6. Key source directories:

- src/core/: protocol, audio, networking and the radio backends
- src/models/: radio, slice, meter and transmit models
- src/gui/: widgets, applets and dialogs

Start with AGENTS.md³³, the canonical project guide, and docs/DEVELOPER-GUIDE.md³⁴ for architecture and conventions. See Contributing Guide (p. 269) before opening a pull request.

³⁰<https://github.com/aethersdr/AetherSDR#dependencies>

³¹<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md#what-each-dependency-enables>

³²<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md#windows-11>

³³<https://github.com/aethersdr/AetherSDR/blob/main/AGENTS.md>

³⁴<https://github.com/aethersdr/AetherSDR/blob/main/docs/DEVELOPER-GUIDE.md>

16.5.4 Reference

Build options

BUILD-OPTIONS.md³⁵ lists every switch and is checked against the build files, so it stays accurate. A few you may want:

Option	Default	Purpose
CMAKE_BUILD_TYPE	—	RelWithDebInfo for normal use, Debug for development
AETHER_GPU_SPECTRUM	ON	QRhi GPU spectrum and waterfall. Needs Qt's private GUI headers; without them the CPU renderer is built.
ENABLE_RTL	ON	Experimental receive-only RTL-SDR backend, when librtlsdr and single-precision FFTW are found
ENABLE_ASX	ON	Copy Assist on-device speech-to-text
ENABLE_NVIDIA_AFX	ON	BNR NVIDIA GPU noise removal (x86-64 Linux and Windows)
AETHER_FETCH_QT	OFF	Run the Qt setup script during configure if the pinned Qt is missing
USE_SYSTEM_*	OFF	For distribution packagers: use system zlib, libmspack, libmosquitto, RtMidi, libwhisper or SQLite instead of the bundled copies
LOWER_CASE_BINARY_NAME	OFF	Lower-case executable name on Linux

Builds honour SOURCE_DATE_EPOCH for reproducible output.

16.5.5 Troubleshooting

The build still uses an old Qt after you installed Qt 6.12

An existing build directory remembers the Qt it first found.

1. Reconfigure from scratch: `cmake --fresh -B build`.
2. Build again.

setup-qt.sh stops because python3 -m venv does not work

Debian, Ubuntu and Raspberry Pi OS ship Python without the venv module.

1. Run `sudo apt install python3-venv`.
2. Run `scripts/setup/setup-qt.sh` again.

The macOS build fails with two Qt installations

Homebrew's qt is installed alongside the Qt from `setup-qt.sh`, and CMake sees both.

1. Remove Homebrew's Qt: `brew uninstall qt`.
2. Reconfigure with `cmake --fresh -B build` and build again.

16.5.6 See also

- Installation (p. 8)
- Contributing Guide (p. 269)
- AI-Assisted Development (p. 275)
- GPU Rendering (p. 252)
- docs/BUILDING.md³⁶
- BUILD-OPTIONS.md³⁷

³⁵<https://github.com/aethersdr/AetherSDR/blob/main/BUILD-OPTIONS.md>

³⁶<https://github.com/aethersdr/AetherSDR/blob/main/docs/BUILDING.md>

³⁷<https://github.com/aethersdr/AetherSDR/blob/main/BUILD-OPTIONS.md>

Generated Reference

17.1 Generated Reference

The pages in this section are generated from the AetherSDR source code, not written by hand, so they always match the version of the app they were built with.

- Default Keyboard Shortcuts (p. 280): assignable keyboard actions and their default keys.
- MIDI Controller Actions (p. 284): every control you can map to a MIDI controller.
- FlexControl Actions (p. 288): FlexControl button actions and defaults.
- Ulanzi Dial Actions (p. 289): Ulanzi Dial wheel actions and button defaults.
- Stream Deck+ and HID Controller Actions (p. 290): Stream Deck+, HID encoder and TMate 2 actions and defaults.
- Log Categories (p. 293): logging categories for support logs.
- TCI Commands (p. 294): command names the TCI server recognises.

17.1.1 Refreshing these pages

Run this from the top of the source tree after changing any of the source files a page names:

```
python3 tools/docs/gen_reference.py
```

`python3 tools/docs/gen_reference.py --check` changes nothing and exits with an error, printing the difference, when a committed page no longer matches the source. CI runs it, so a change that alters one of these tables has to carry the regenerated page with it.

17.2 Default Keyboard Shortcuts

Generated page

This page is generated from `src/gui/MainWindow_Shortcuts.cpp`, `src/core/ShortcutManager.cpp`, `src/gui/MainWindowHelpers.h`, `src/core/DigitalVoiceFeature.h`, `src/core/DigitalVoiceModeRegistry.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

These are the actions you can bind in **Settings** → **Configure Shortcuts...**, with the key each one has before you change it. An action with no default key does nothing from the keyboard until you bind one.

- The **ID** is the stable name used in a shortcuts backup file and by the automation bridge's `shortcut` verb.
- On macOS, Qt reads **Ctrl** as the Command (⌘) key and **Meta** as the Control key.
- Menu accelerators and fixed application shortcuts are not in this list; they are not assignable.

17.2.1 Frequency

Action	ID	Default key	Notes
Tune Up (1 step)	tune_up_1	Right	Repeats while held
Tune Down (1 step)	tune_down_1	Left	Repeats while held
Tune Up (10 steps)	tune_up_10	Shift+Right	Repeats while held
Tune Down (10 steps)	tune_down_10	Shift+Left	Repeats while held
Tune Up 1 MHz	tune_up_1mhz		
Tune Down 1 MHz	tune_down_1mhz		
Go to Frequency	go_to_freq	G	

17.2.2 Band

Action	ID	Default key	Notes
160m	band_160m		
80m	band_80m		
60m	band_60m		
40m	band_40m		
30m	band_30m		
20m	band_20m		
17m	band_17m		
15m	band_15m		
12m	band_12m		
10m	band_10m		
6m	band_6m		
2m	band_2m		

17.2.3 Mode

Action	ID	Default key	Notes
USB	mode_usb		
LSB	mode_lsb		
CW	mode_cw		
CWL	mode_cwl		
AM	mode_am		
SAM	mode_sam		
FM	mode_fm		
NFM	mode_nfm		
DFM	mode_dfm		
DSTR	mode_dstr		Only in builds with AETHER_ENABLE_DIGITAL_VOICE_HELPER
DIGU	mode_digu		
DIGL	mode_digl		
RTTY	mode_rtty		
FDVU	mode_fdvu		
FDVL	mode_fdvl		

17.2.4 TX

Action	ID	Default key	Notes
MOX Toggle	mox_toggle	T	Keys the transmitter
PTT (Hold)	ptt_hold	Space	Keys the transmitter
ATU Start	atu_start		Keys the transmitter
TUNE Toggle	tune_toggle		Keys the transmitter
Two-Tone Tune	two_tone_tune		Keys the transmitter
VOX Toggle	vox_toggle		
Speech Processor Toggle	speech_proc_toggle		
DAX TX Toggle	dax_toggle		
TX Monitor Toggle	tx_monitor_toggle		

17.2.5 CW

Action	ID	Default key	Notes
CW Speed Up (+5 WPM)	cw_speed_up		
CW Speed Down (-5 WPM)	cw_speed_down		
CW Sidetone Toggle	cw_sidetone_toggle		
CW Iambic Toggle	cw_iambic_toggle		
CW Iambic Mode Toggle (A/B)	cw_iambic_mode_toggle		
CW Swap Paddles Toggle	cw_swap_paddles_toggle		
CWL Frequency Offset Toggle	cwl_toggle		
CW Break-In (QSK) Toggle	cw_breakin_toggle		
Trigger straight key	cwkey		Keys the transmitter
Trigger CW Left Paddle	cwdit		Keys the transmitter
Trigger CW Right Paddle	cwdah		Keys the transmitter

17.2.6 Audio

Action	ID	Default key	Notes
AF Gain Up	af_gain_up	Up	
AF Gain Down	af_gain_down	Down	
Mute Toggle	mute_toggle	M	
Mute All Slices	mute_all_slices_toggle		
Master Mute Toggle	master_mute_toggle		
Master Volume Up	master_volume_up		Repeats while held
Master Volume Down	master_volume_down		Repeats while held
Squelch Toggle	squelch_toggle		

17.2.7 Slice

Action	ID	Default key	Notes
Next Slice	next_slice		
Prev Slice	prev_slice		
Split Toggle	split_toggle		
Monitor TX (Hold)	split_monitor_tx		
Split Up 1 kHz	split_up_1		

Action	ID	Default key	Notes
Split Up 5 kHz	split_up_5		
Split Up 10 kHz	split_up_10		
Cycle TX Slice	cycle_tx_slice		

17.2.8 Filter

Action	ID	Default key	Notes
Filter Widen	filter_widen		
Filter Narrow	filter_narrow		

17.2.9 Tuning

Action	ID	Default key	Notes
Step Size Up	step_up	]	
Step Size Down	step_down	[	
Tune Lock Toggle	lock_toggle	L	
Center Lock Active Slice	center_lock_toggle		

17.2.10 DSP

Action	ID	Default key	Notes
NB Toggle	nb_toggle		
NR2 Toggle	nr2_toggle		
RN2 (RNNoise) Toggle	rn2_toggle		
NR4 Toggle	nr4_toggle		
DFNR Toggle	dfnr_toggle		
TNF Global Toggle	tnf_toggle		
NR Cycle (Off/NR/NR2/NR4/DFNR)	nr_cycle		
ANF Toggle	anf_toggle		

17.2.11 AGC

Action	ID	Default key	Notes
AGC Mode Cycle	agc_cycle		
RF Gain Up	rf_gain_up		Repeats while held
RF Gain Down	rf_gain_down		Repeats while held
AGC-T Up	agct_up		Repeats while held
AGC-T Down	agct_down		Repeats while held

17.2.12 EQ

Action	ID	Default key	Notes
TX EQ Toggle	tx_eq_toggle		
RX EQ Toggle	rx_eq_toggle		

17.2.13 Display

Action	ID	Default key	Notes
Band Zoom	band_zoom		
Segment Zoom	segment_zoom		
Panadapter Zoom In	pan_zoom_in	=	
Panadapter Zoom Out	pan_zoom_out	-	
Open Memories Dialog	open_memories	/	
Minimize Active Window	window_minimize	macOS: Ctrl+M; other: none	Window management: works with keyboard shortcuts switched off
Toggle Active Window Full Screen	window_fullscreen	macOS: Ctrl+Meta+F; other: F11	Window management: works with keyboard shortcuts switched off
Minimal Mode Toggle	minimal_mode		

17.2.14 RIT/XIT

Action	ID	Default key	Notes
RIT Toggle	rit_toggle		
XIT Toggle	xit_toggle		

17.3 MIDI Controller Actions

Generated page

This page is generated from `src/gui/MainWindow_Controllers.cpp`, `src/gui/MainWindowHelpers.h`, `src/core/DigitalVoiceFeature.h`, `src/core/DigitalVoiceModeRegistry.cpp`, `src/core/MidiControlManager.h` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

These are the controls you can map to a MIDI controller in **Settings** → **MIDI Mapping**.... The same list feeds the Ulanzi Dial button menus.

- **Slider**: a continuous control (knob, fader, pitch bend), scaled to the range shown.
- **Toggle**: on/off. A CC above 63 is on; a note toggles.
- **Trigger**: fires once on a note or CC.
- **Gate (held)**: on while the note is held, off when it is released.

MIDI control exists only in builds with `HAVE_MIDI`.

17.3.1 RX

Control	ID	Type	Range	Notes
AF Gain	rx.afGain	Slider	0 to 200	
Squelch Level	rx.squelch	Slider	0 to 100	
AGC Threshold	rx.agcThreshold	Slider	0 to 100	
Audio Pan	rx.audioPan	Slider	0 to 100	
Noise Blanker	rx.nbEnable	Toggle		
Adaptive RX Filter	rx.adaptiveFilter	Toggle		
Noise Reduction	rx.nrEnable	Toggle		
Auto Notch	rx.anfEnable	Toggle		

Control	ID	Type	Range	Notes
Squelch Enable	rx.squelchEnable	Toggle		
Audio Mute	rx.mute	Toggle		
Tune Lock	rx.tuneLock	Toggle		
Center Lock	rx.centerLock	Toggle		
RIT Enable	rx.ritEnable	Toggle		
XIT Enable	rx.xitEnable	Toggle		
NR2 (Spectral)	rx.nr2Enable	Toggle		
RN2 (RNNNoise)	rx.rn2Enable	Toggle		
NR4 (Spectral Bleach)	rx.nr4Enable	Toggle		
DFNR (DeepFilter)	rx.dfnrEnable	Toggle		
Step Size Up	rx.stepUp	Trigger		
Step Size Down	rx.stepDown	Trigger		
VFO Tune Knob	rx.tuneKnob	Slider	0 to 127	
NR Cycle	global.nrCycle	Trigger		
AGC Cycle	global.agcCycle	Trigger		

17.3.2 TX

Control	ID	Type	Range	Notes
RF Power	tx.rfPower	Slider	0 to 100	
Tune Power	tx.tunePower	Slider	0 to 100	
MOX	tx.mox	Toggle		
TUNE	tx.tune	Toggle		
ATU Start	tx.atuStart	Trigger		
Two-Tone Tune	global.twoToneTune	Trigger		

17.3.3 Phone/CW

Control	ID	Type	Range	Notes
Mic Level	phone.micLevel	Slider	0 to 100	
Monitor Volume	phone.monGain	Slider	0 to 100	
Speech Processor	phone.procEnable	Toggle		
DAX	phone.daxEnable	Toggle		
Monitor	phone.monEnable	Toggle		
VOX Enable	phone.voxEnable	Toggle		
VOX Level	phone.voxLevel	Slider	0 to 100	
AM Carrier	phone.amCarrier	Slider	0 to 100	
CW Speed	cw.speed	Slider	5 to 100	
CW Break-In Delay	cw.delayMs	Slider	0 to 2000	
CW Sidetone	cw.sidetoneEnable	Toggle		
CW Iambic	cw.iambicEnable	Toggle		
CW Iambic Mode (0=A, 1=B)	cw.iambicMode	Toggle		
CW Swap Paddles	cw.swapPaddles	Toggle		
CWL Frequency Offset	cw.cwlEnable	Toggle		
CW Break-In (QSK)	cw.breakInEnable	Toggle		
Trigger straight key	cwkey	Gate (held)		

Control	ID	Type	Range	Notes
Trigger CW Left Paddle	<code>cwdit</code>	Gate (held)		
Trigger CW Right Paddle	<code>cwdah</code>	Gate (held)		
PTT (hold)	<code>cw.ptt</code>	Gate (held)		

17.3.4 EQ

Control	ID	Type	Range	Notes
TX EQ Enable	<code>eq.txEnable</code>	Toggle		
RX EQ Enable	<code>eq.rxEnable</code>	Toggle		
63 Hz	<code>eq.band63</code>	Slider	−10 to 10	
125 Hz	<code>eq.band125</code>	Slider	−10 to 10	
250 Hz	<code>eq.band250</code>	Slider	−10 to 10	
500 Hz	<code>eq.band500</code>	Slider	−10 to 10	
1 kHz	<code>eq.band1000</code>	Slider	−10 to 10	
2 kHz	<code>eq.band2000</code>	Slider	−10 to 10	
4 kHz	<code>eq.band4000</code>	Slider	−10 to 10	
8 kHz	<code>eq.band8000</code>	Slider	−10 to 10	

17.3.5 Global

Control	ID	Type	Range	Notes
Master Volume	<code>global.masterVolume</code>	Slider	0 to 100	
Headphone Volume	<code>global.hpVolume</code>	Slider	0 to 100	
Master Mute	<code>global.masterMute</code>	Toggle		
TX Button	<code>global.txButton</code>	Toggle		
RADE Modem	<code>global.rade</code>	Toggle		Only in builds with HAVE_RADE
TNF Global	<code>global.tnfEnable</code>	Toggle		
Band Up	<code>global.bandUp</code>	Trigger		
Band Down	<code>global.bandDown</code>	Trigger		
Mode Up	<code>global.modeUp</code>	Trigger		
Mode Down	<code>global.modeDown</code>	Trigger		
Next Slice	<code>global.nextSlice</code>	Trigger		
Previous Slice	<code>global.prevSlice</code>	Trigger		
QSO Record	<code>global.qsoRecord</code>	Toggle		
QSO Playback	<code>global.qsoPlay</code>	Toggle		

17.3.6 Mode

Control	ID	Type	Range	Notes
Mode USB	<code>global.modeUSB</code>	Trigger		
Mode LSB	<code>global.modeLSB</code>	Trigger		
Mode CW	<code>global.modeCW</code>	Trigger		
Mode CWL	<code>global.modeCWL</code>	Trigger		
Mode AM	<code>global.modeAM</code>	Trigger		
Mode SAM	<code>global.modeSAM</code>	Trigger		
Mode FM	<code>global.modeFM</code>	Trigger		
Mode NFM	<code>global.modeNFM</code>	Trigger		

Control	ID	Type	Range	Notes
Mode DFM	global.modeDFM	Trigger		
Mode DSTR	global.modeDSTR	Trigger		Only in builds with AETHER_ENABLE_DIGITAL_VOICE_HELPER
Mode DIGU	global.modeDIGU	Trigger		
Mode DIGL	global.modeDIGL	Trigger		
Mode RTTY	global.modeRTTY	Trigger		
Mode FDVU	global.modeFDVU	Trigger		
Mode FDVL	global.modeFDVL	Trigger		

17.3.7 Band

Control	ID	Type	Range	Notes
Band 160m	global.band160m	Trigger		
Band 80m	global.band80m	Trigger		
Band 60m	global.band60m	Trigger		
Band 40m	global.band40m	Trigger		
Band 30m	global.band30m	Trigger		
Band 20m	global.band20m	Trigger		
Band 17m	global.band17m	Trigger		
Band 15m	global.band15m	Trigger		
Band 12m	global.band12m	Trigger		
Band 10m	global.band10m	Trigger		
Band 6m	global.band6m	Trigger		
Band 2m	global.band2m	Trigger		

17.3.8 Slice

Control	ID	Type	Range	Notes
Split Toggle	global.splitToggle	Trigger		

17.3.9 Filter

Control	ID	Type	Range	Notes
Filter Widen	global.filterWiden	Trigger		
Filter Narrow	global.filterNarrow	Trigger		

17.3.10 Frequency

Control	ID	Type	Range	Notes
Tune Up 1 MHz	global.tuneUp1mhz	Trigger		
Tune Down 1 MHz	global.tuneDown1mhz	Trigger		

17.3.11 Display

Control	ID	Type	Range	Notes
Band Zoom	global.bandZoom	Trigger		
Segment Zoom	global.segmentZoom	Trigger		
Panadapter Zoom In	global.panZoomIn	Trigger		
Panadapter Zoom Out	global.panZoomOut	Trigger		

Control	ID	Type	Range	Notes
Open Memories	<code>global.openMemories</code>	Trigger		

17.4 FlexControl Actions

Generated page

This page is generated from `src/gui/FlexControlDialog.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

The FlexControl knob's three Aux buttons can each run one action on a single tap and another on a double tap. Set them in **Settings** → **FlexControl Knob & Buttons...**

17.4.1 Default button actions

Button	Single tap	Double tap
Aux1	Step Up	Step Down
Aux2	MOX	Toggle Tune
Aux3	Toggle Mute	Toggle Lock

17.4.2 Actions

Action	ID
Tune Slice	<code>WheelFrequency</code>
Band Zoom	<code>BandZoom</code>
Segment Zoom	<code>SegmentZoom</code>
RIT (Receive Incremental Tuning)	<code>WheelRit</code>
XIT (Transmit Incremental Tuning)	<code>WheelXit</code>
Master Volume	<code>WheelVolume</code>
Slice Audio Volume	<code>WheelSliceAudio</code>
Headphone Volume	<code>WheelHeadphoneVolume</code>
AGCT (Automatic Gain Control Threshold)	<code>WheelAgcT</code>
APF (Audio Peaking Filter)	<code>WheelApf</code>
Clear RIT	<code>ClearRit</code>
Clear XIT	<code>ClearXit</code>
Toggle APF	<code>ToggleApf</code>
Change Active Slice	<code>NextSlice</code>
Split Active Slice	<code>SplitActiveSlice</code>
Monitor TX Frequency	<code>SplitMonitorTx</code>
MOX	<code>ToggleMox</code>
RF Power	<code>WheelPower</code>
CW Speed	<code>WheelCwSpeed</code>
CWX Macro 1	<code>CwxF1</code>
CWX Macro 2	<code>CwxF2</code>
CWX Macro 3	<code>CwxF3</code>
CWX Macro 4	<code>CwxF4</code>
CWX Macro 5	<code>CwxF5</code>
CWX Macro 6	<code>CwxF6</code>
CWX Macro 7	<code>CwxF7</code>

Action	ID
CWX Macro 8	CwxF8
CWX Macro 9	CwxF9
CWX Macro 10	CwxF10
CWX Macro 11	CwxF11
CWX Macro 12	CwxF12
Step Up	StepUp
Step Down	StepDown
Toggle Tune	ToggleTune
Toggle Mute	ToggleMute
Toggle Lock	ToggleLock
Previous Slice	PrevSlice
Toggle AGC	ToggleAgc
Slice AF Up	VolumeUp
Slice AF Down	VolumeDown
None	None

17.5 Ulanzi Dial Actions

Generated page

This page is generated from `src/gui/UlanziDialMapperDialog.cpp`, `src/core/UlanziDialMappings.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

Set the Ulanzi Dial up in **Settings** → **Ulanzi Dial Mapping....**

- The **wheel** runs one of the actions below.
- Each **button** can run any action from Default Keyboard Shortcuts ([p. 280](#)), or any Toggle, Trigger or Gate control from MIDI Controller Actions ([p. 284](#)).

17.5.1 Wheel actions

Category	Action	ID	Default
Frequency	Tune Slice	WheelFrequency	Default
Filter	Filter Bandwidth	FilterWidth	
Audio	Slice Audio Volume	WheelSliceAudio	
Audio	Master Volume	WheelVolume	
Audio	Headphone Volume	WheelHeadphoneVolume	
Display	Panadapter Zoom	PanadapterZoom	
Display	Band Zoom	BandZoom	
Display	Segment Zoom	SegmentZoom	
RIT/XIT	RIT (Receive Incremental Tuning)	WheelRit	
RIT/XIT	XIT (Transmit Incremental Tuning)	WheelXit	
AGC	AGC-T (Threshold)	WheelAgcT	
AGC	RF Gain	WheelRfGain	
DSP	APF (Audio Peaking Filter)	WheelApf	
CW	CW Speed	WheelCwSpeed	
TX	RF Power	WheelPower	

17.5.2 Button defaults

Button	Button ID	Default action	Stored as
Top Left	top_left	Unassigned	
Top Middle	top_middle	[RIT/XIT] RIT Toggle	shortcut:rit_toggle
Top Right	top_right	Unassigned	
Left Top	side_lt	Unassigned	
Left Bottom	side_lb	Unassigned	
Right Top	side_rt	[Slice] Next Slice	shortcut:next_slice
Right Bot	side_rb	Unassigned	
Dial Press	dial_press	[Audio] Mute Toggle	shortcut:mute_toggle

17.6 Stream Deck+ and HID Controller Actions

Generated page

This page is generated from `src/gui/RadioSetupDialog.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

Set these up in **Settings** → **Radio Setup...** → **Serial & Controllers**. The encoder tables also apply to single-encoder HID devices such as the RC-28, PowerMate and ShuttleXpress, which use Encoder 1 only.

These controls exist only in builds with `HAVE_SERIALPORT` and `HAVE_HIDAPI`.

17.6.1 Stream Deck+ and HID encoders

LCD key actions

Action	ID
None	None
Toggle TX (MOX)	ToggleMox
Toggle Tune	ToggleTune
Toggle RIT on/off	ToggleRit
Toggle XIT on/off	ToggleXit
Clear RIT offset	ClearRit
Clear XIT offset	ClearXit
Step Size Up	StepUp
Step Size Down	StepDown
Toggle Mute	ToggleMute
Toggle Slice Lock	ToggleLock
Toggle APF	ToggleApf
Cycle AGC Mode	ToggleAgc
Toggle Band Zoom	BandZoom
Toggle Segment Zoom	SegmentZoom
Next Slice	NextSlice
Previous Slice	PrevSlice
Volume Up (+5)	VolumeUp
Volume Down (-5)	VolumeDown
Toggle Split	SplitActiveSlice
Monitor TX Frequency	SplitMonitorTx

Encoder turn actions

Encoder	Default
Encoder 1	Tune Slice
Encoder 2	RIT (Receive Incremental Tuning)
Encoder 3	XIT (Transmit Incremental Tuning)
Encoder 4	Master Volume

Action	ID
Tune Slice	WheelFrequency
RIT (Receive Incremental Tuning)	WheelRit
XIT (Transmit Incremental Tuning)	WheelXit
Master Volume	WheelVolume
Slice Audio Volume	WheelSliceAudio
Headphone Volume	WheelHeadphoneVolume
AGC Threshold	WheelAgcT
APF Level	WheelApf
CW Speed	WheelCwSpeed
RF Power	WheelPower
None	None

Encoder push actions

Encoder	Default
Encoder 1	Cycle Tuning Step
Encoder 2	Toggle RIT on/off
Encoder 3	Toggle XIT on/off
Encoder 4	None

Action	ID
Cycle Tuning Step	StepCycle
Toggle RIT on/off	ToggleRit
Toggle XIT on/off	ToggleXit
Toggle TX (MOX)	ToggleMox
Toggle Mute	ToggleMute
Lock Slice	ToggleLock
None	None

17.6.2 TMatе 2

Function key actions

Key	Default
F 1	Toggle TX (MOX)
F 2	Cycle AGC Mode
F 3	Toggle Band Zoom
F 4	Toggle APF
F 5	Toggle Mute
F 6	Toggle RIT on/off

Action	ID
None	None
Toggle TX (MOX)	ToggleMox
Toggle Tune	ToggleTune
Toggle RIT on/off	ToggleRit
Toggle XIT on/off	ToggleXit
Clear RIT offset	ClearRit
Clear XIT offset	ClearXit
Step Size Up	StepUp
Step Size Down	StepDown
Toggle Mute	ToggleMute
Toggle Slice Lock	ToggleLock
Toggle APF	ToggleApf
Cycle AGC Mode	ToggleAgc
Toggle Band Zoom	BandZoom
Toggle Segment Zoom	SegmentZoom
Next Slice	NextSlice
Previous Slice	PrevSlice
Volume Up (+5)	VolumeUp
Volume Down (-5)	VolumeDown
Toggle Split	SplitActiveSlice
Monitor TX Frequency	SplitMonitorTx

Encoder turn actions

Encoder	Default
Encoder 1	Tune Slice
Encoder 2	RIT (Receive Incremental Tuning)
Encoder 3	XIT (Transmit Incremental Tuning)

Action	ID
Tune Slice	WheelFrequency
RIT (Receive Incremental Tuning)	WheelRit
XIT (Transmit Incremental Tuning)	WheelXit
Master Volume	WheelVolume
Slice Audio Volume	WheelSliceAudio
Headphone Volume	WheelHeadphoneVolume
AGC Threshold	WheelAgcT
APF Level	WheelApf
CW Speed	WheelCwSpeed
RF Power	WheelPower
None	None

Encoder push actions

Encoder	Default
Encoder 1	Cycle Tuning Step
Encoder 2	Toggle RIT on/off

Encoder	Default
Encoder 3	Toggle XIT on/off

Action	ID
Cycle Tuning Step	StepCycle
Toggle RIT on/off	ToggleRit
Toggle XIT on/off	ToggleXit
Toggle TX (MOX)	ToggleMox
Toggle Mute	ToggleMute
Lock Slice	ToggleLock
None	None

17.7 Log Categories

Generated page

This page is generated from `src/core/LogManager.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

These are the logging categories listed in **Help** → **Support & Diagnostics...**. Turn one on there to add its detail to the support log. The **Category** is the name the automation bridge's `log set` verb takes.

Label	Category	What it logs
Discovery	<code>aether.discovery</code>	UDP radio discovery broadcasts
Connection / Commands	<code>aether.connection</code>	Raw TCP command channel lines: TX commands, RX responses, and socket state
Protocol / Status	<code>aether.protocol</code>	Parsed SmartSDR protocol handling and model status updates
Audio	<code>aether.audio</code>	RX/TX audio, device negotiation, volume
Audio Summary	<code>aether.audio.summary</code>	Default support log summaries for audio routing and sink/source negotiation
VITA-49	<code>aether.vita49</code>	UDP packet routing: FFT, waterfall, meters, DAX
DSP	<code>aether.dsp</code>	NR2, RN2, CW decoder processing
RADE	<code>aether.rade</code>	FreeDV Radio Autoencoder digital voice
SmartLink	<code>aether.smartlink</code>	Auth0 login, TLS tunnel, WAN streaming
TCI / CAT / rigctld	<code>aether.cat</code>	TCI server (slice and DAX arming, TX audio summary, TX_CHRONO stall summary at debug: maxGap/latePolls/catch-up), rigctld TCP servers, PTY virtual serial ports
DAX	<code>aether.dax</code>	Virtual audio bridge (PipeWire/CoreAudio)
Meters	<code>aether.meters</code>	Meter definitions and value conversion
Transmit	<code>aether.transmit</code>	TX state, ATU, profiles, power control
Firmware	<code>aether.firmware</code>	Firmware download, staging, upload
Tuner/AGM	<code>aether.tuner</code>	TGXL tuner, Antenna Genius state
Peripheral Authorization	<code>aether.peripheral.auth</code>	Credential store availability and save failures; code values are never logged
GUI	<code>aether.gui</code>	Window, applets, dialogs
DX Cluster	<code>aether.dxcluster</code>	DX cluster telnet connection and spot parsing
MQTT	<code>aether.mqtt</code>	MQTT telemetry client connection and messages
RBN	<code>aether.rbn</code>	Reverse Beacon Network connection and spots
Ext Devices	<code>aether.devices</code>	Serial port, FlexControl, MIDI, HID encoder

Label	Category	What it logs
Performance	<code>aether.perf</code>	Render timing and CPU profiling data
Render	<code>aether.render</code>	Render pipeline: RHI/GPU path selection and fallback, paint stalls, texture upload churn
Propagation	<code>aether.propforecast</code>	Solar and propagation forecast updates
CW / netCW	<code>aether.cw</code>	CW keying, MIDI paddle, iambic, and netCW timing
S History	<code>aether.shistory</code>	Past-Signals voice detection: noise floor, region width, band-plan filter
AetherModem	<code>aether.ax25</code>	AX.25 modem lifecycle, RX/TX audio, demod, framing, and packet diagnostics
AX.25 Link	<code>aether.ax25.link</code>	Connected-mode data link: session open/close, measured round-trip vs configured T1, retransmits, idle-link polls
Waveform	<code>aether.waveform</code>	Docker waveform image install upload and local waveform helper lifecycle
KiwiSDR	<code>aether.kiwisdr</code>	KiwiSDR remote RX antennas: connect, handshake, audio/waterfall negotiation, reconnect, profile lifecycle
ANAN Protocol 2	<code>aether.anan.p2</code>	ANAN/Saturn Protocol 2 wire session: DDC sequence gaps, speaker-audio FIFO level and underflow reports, unexpected sender ports
KiwiSDR Audio/DSP	<code>aether.kiwisdr.audio</code>	Verbose KiwiSDR receive audio: frame decode, resampler, jitter/FIFO under/overflow, mixing (high-rate; off by default)
Automation Bridge	<code>aether.automation</code>	Agent-drivable test bridge (#3646): QLocalServer verbs, widget snapshots, captures (AETHER_AUTOMATION only)
QRZ Lookup	<code>aether.qrz</code>	QRZ.com callsign lookups: session, cache, CW callsign spotting, photos
AetherClock	<code>aether.clock</code>	WWV/WWVB time-signal decoder: state transitions, per-second alignment, frame decodes, voter verdicts
Hermes-Lite 2	<code>aether.hl2</code>	HL2 backend: band changes, J16 companion-filter selection, LNA gain, and radio health telemetry
Hermes-Lite 2 TX	<code>aether.hl2.tx</code>	HL2 transmit telemetry — TWO different instruments on one toggle, and the difference is the point. The RADIO's DSIQ FIFO depth with its underflow/overflow flags and the forward/reflected power counts; AND the HOST queue's own starvation lines, which are NOT that FIFO and which that FIFO cannot see. Separate toggle — ticking "Hermes-Lite 2" does NOT enable this (high-rate)
Icom Session	<code>aether.icom.session</code>	Icom RS-BA1 session: handshake, token/auth, capabilities, keepalive
Icom Streams	<code>aether.icom.stream</code>	Icom UDP stream lifecycle: control/serial/audio handshakes and ports
Icom CI-V	<code>aether.icom.civ</code>	Every CI-V frame in and out, decoded — command, subcommand and payload bytes. The only way to tell 'the radio never sent it' from 'we sent it and dropped the reply' (high-rate)
Icom Scope	<code>aether.icom.pan</code>	Icom spectrum scope: sweep frames, division reassembly, bounds
Icom Link	<code>aether.icom.link</code>	Icom backend link state: connect/disconnect, model resolution, capability publication
Icom Credentials	<code>aether.icom.cred</code>	Icom credential storage and retrieval (no secret values are logged)
System Info	<code>aether.sysinfo</code>	Startup hardware/capability inventory: OS, CPU model + SIMD features, RAM, and the speech-engine ISA baseline check (#4986). A few lines once per launch

17.8 TCI Commands

Generated page

This page is generated from `src/core/TciProtocol.cpp`, `src/core/TciServer.cpp` by `tools/docs/gen_reference.py`. Do not edit it by hand: change the source, then run `python3 tools/docs/gen_reference.py`.

These are the command names the TCI server recognises from a client. Names are case-insensitive. A command the server does not recognise is ignored, as the TCI specification requires. Recognising a command does not mean every radio supports every value of it; see the TCI Server page for behaviour.

17.8.1 Audio, IQ and session commands

Handled per client connection: audio and IQ streams, stream format, and sensor reports.

Command
audio_start
audio_stop
audio_samplerate
audio_stream_sample_type
rx_sensors_enable
tx_sensors_enable
iq_samplerate
iq_start
iq_stop
spectrum_event
audio_stream_samples
tx_stream_audio_buffering
line_out_start
line_out_stop
line_out_recorder
audio_stream_channels

17.8.2 Radio commands

Command
start
stop
tx_enable
cw_msg
cw_macros
cw_macros_stop
spot
spot_delete
spot_clear
keyer
cw_keyer_speed
cw_macros_delay
cw_terminal
dds
if
rx_channel_enable

Command
rx_volume
rx_mute
rx_balance
mon_enable
mon_volume
rx_nb_param
rx_bin_enable
rx_anc_enable
rx_dse_enable
rx_nf_enable
digl_offset
digu_offset
set_in_focus
tx_frequency
vfo_limits
if_limits
vfo_lock
vfo
modulation
trx
tune
drive
tune_drive
mic_level
rit_enable
xit_enable
rit_offset
xit_offset
split_enable
rx_filter_band
cw_macros_speed
lock
sql_enable
sql_level
volume
mute
agc_mode
agc_gain
rx_nb_enable
rx_nr_enable
rx_anf_enable
rx_apf_enable

17.8.3 AetherSDR extensions

Not part of the TCI specification.

Command`rx_record``rx_play``tx_gain``active_slice`
