awj-surface user guide

awj-surface maps **MIDI and OSC controllers onto an Analog Way LivePremier (Aquilon) switcher**. Faders to layer opacity, encoders to size and position, buttons to select layers, apply sources, cycle crop modes and switch keying — **with the surface following the switcher**, not just driving it.

> **Before you rely on this:** the parameter model and every path in it were read from a live LivePremier, and the whole chain — a fader movement through to an opacity write, a TAKE, and the preset flip that follows it — has been **exercised end to end on a real Aquilon C**, with the frame captured beforehand and every one of 87 values verified restored afterwards.
>
> **But no physical control surface has ever been plugged into it.** The controller maps come from published documentation, not from hardware. **Treat a first run with a real APC40, X-Touch or MIDIcon as bring-up** — and see the Bring-up section below, which exists for exactly that.
>
> This codebase was created with AI assistance, directed and reviewed by a human author.

Running it

```
node hosts/node/server.js --device 192.168.2.140 --profile x-touch-mcu
```

Then open http://127.0.0.1:8532.

Run it with no `--device` and everything still works offline — the mapping editor, the on-screen surface, and writes logged instead of sent. That is how you build a show file before the frame arrives.

Read-only mode

Against show hardware you usually want to watch, not drive:

```
node hosts/node/server.js --device 192.168.2.142 --read-only
```

This is not a flag that skips writes. The client is built with **no reachable write path** — the write calls throw, so a change cannot reach the wire even by mistake.

That includes the **Subscriptions list**, which looks like a read and is not: **subscribing is itself a write.** Because of that there is no push in read-only mode, so it **polls** the mapped paths instead (`-poll <ms>`, default 2000).

Control movements are logged as `read-only, NOT sent:` rather than silently dropped, and the header shows **READ-ONLY in amber** — because "will this control reach the switcher?" must be answerable at a glance.

Bring-up: check a profile against real hardware first

Every shipped profile was transcribed from a manual, and **a MIDI implementation chart is exactly the kind of document that is quietly wrong** — a note number off by one, a channel that is 1-based in the manual and 0-based on the wire, an encoder that turns out to send notes rather than a CC.

The **Bring-up** tab turns "touch everything and see" into a checklist. Plug the surface in, sweep every control, and it reports:

- **Declared by the profile** — each control, and whether it has actually been seen. Anything still `never` is usually a wrong number in the transcription.
- **Sent, but not in the profile** — controls the manual left out, each with a guess at what it is.

That guess is the useful part. A sign-magnitude encoder read as a fader is the single most common way a transcribed profile is wrong, and **it does not look wrong — it looks like a parameter that runs away.** The classifier spots it because an encoder never sweeps: its values cluster either side of `0x40`.

```
Sent, but not in the profile
  cc:3:99   x10   fader              sweeps a wide range of values — absolute
  cc:0:80   x5    encoder (signed)   values cluster either side of 0x40 — relative
```

The three things that make this harder than a lookup table

A layer path contains a preset *letter*, and the letter moves. A screen holds three preset memories keyed A, B, C. **Nothing addresses "preview"** — you address a letter, and which letter is on air changes at every take. Bindings therefore say `PREVIEW` or `PROGRAM` and are resolved per event against live device state.

> **Get this wrong and a preview fader silently becomes a live one halfway through a show.**

Every write comes back. The device echoes changes to all clients, so naive feedback drives a motor fader into the hand that just moved it. Writes are attributed and the echo to the originating control is dropped — while **a change from anywhere else (the vendor UI, a second surface) still moves it.**

A fader that is not motorised lies. After a bank change it sits where the last layer left it, and the first touch would slam the new layer to that value. Non-motorised bindings use **pickup**: no write

until the control crosses the value it is steering. A fader that seems dead has not crossed yet.

Controllers

Profile	Surface	Feedback
`x-touch-mcu`	Behringer X-Touch, MC mode	motor faders, LED rings, scribble strips
`apc40`	Akai APC40	button LED colours, knob rings
`midicon-pro`	Elation MIDICON PRO	motor faders, button LEDs
`midicon-2`	Elation MIDICON-2	motor faders, button LEDs
`osc-default`	TouchOSC and similar	values returned on the same address
`generic-learn`	anything	as declared

Both MIDIcons send one note per rotary click rather than a relative CC, so each rotary is two controls in its profile.

Anything else

MIDI. Start on `generic-learn`, pick the input from the header (or `--midi`, which takes an exact name, a case-insensitive substring, or an index), then bind controls by touching them.

A profile with a name pattern finds its own port. A generic one has nothing to match on, so with a single input attached it just opens it — and **with several it refuses to guess** and lists them rather than picking the wrong controller.

OSC. Send from any layout — TouchOSC, a lighting desk, Companion. **The addresses are yours**: `osc-default` is only a starting layout, and learn binds whatever address arrives. Feedback goes back out on the same address, so a tablet fader tracks the switcher.

Both go through the same Bring-up view. OSC carries no hint of what a control *is*, so the value shape decides: only ever 0/1 is a button, anything that visits the middle is a fader.

If no MIDI binding is installed there is no hardware I/O at all. The header says so, and the on-screen surface still works.

The parameter catalogue

Nothing in it is hand-written. It is generated from a device, joining the real store tree (exact node and property names, but no ranges) with the device's own published attribute tables (min, max, default, type, read-only, enum members).

That is why `opacity` is **0–256** and not 0–255, `posH` is ±2,000,000, and the input-number enum has 482 members. **A store dump alone cannot tell you that opacity stops at 256** — which is exactly the kind of thing a hand-typed catalogue gets wrong.

If something is wrong

Symptom	Cause
A parameter runs away when I turn an encoder	The profile reads it as a fader. Run Bring-up; the classifier names it.
A control does nothing and Bring-up says `never`	Wrong number in the transcription. Learn it instead.
A motor fader fights my hand	Should not happen — the originating echo is dropped. If it does, report it.
A fader is dead after a bank change	Pickup. Sweep it through the layer's current value.
A preview fader turned out to be live	The preset letter moved. Bindings must say PREVIEW or PROGRAM, not a letter.
It opened the wrong MIDI port	With several inputs and a generic profile it refuses to guess — pick one with `--midi`.
Nothing reaches the switcher	Check for READ-ONLY in amber in the header.
