LivePremier Plus user guide

LivePremier Plus is a **local app that puts extra panels inside an Analog Way LivePremier Web RCS session**, drawn in Web RCS's own design language so they read as part of the product rather than as a bolt-on.

- **VPU Map** — the device's mixing-resource allocation, drawn as a budget. Which units are fitted, who holds them, what is spare, and what a staged preconfig would change.
- **Console** — a lighting-desk command grammar for a video switcher.
- **Timeline** — a theatre-style cue stack that advances on one GO, with per-cue fade, delay and follow times.
- **MIDI Mapping** — a control surface driving the switcher, from the page itself.
- **Arithmetic in the vendor's own numeric fields** — type `1080-80` into a layer width and get 1000.

It rides the vendor app's own WebSocket. **No second connection to the device, no replacement UI, and nothing to install in the browser.**

> **Before you rely on this:** the panels render inside a real Web RCS session and the device store mirrors live — verified through this proxy against **LivePremier Simulator 6.2.73**, along with cue-stack persistence and the whole setup flow. The VPU map has been **read from a live Aquilon C** and is tested against that capture, including Optimized mode, interleaved output links, and a staged preconfig differing from the running one.
>
> **Nothing has ever been written to that device**, the timeline has only ever fired at a simulator, and **no panel has been driven live against physical hardware in a browser.** That is the first thing to try.
>
> Built with AI assistance, directed and reviewed by a human author.

Running it

Needs **Node 20 or newer** and has no dependencies to install.

```
npm start
```

Then open `http://127.0.0.1:8535/` and give it the switcher's address. That address is remembered.

```
npm start -- --device 192.168.2.142
```

flag	meaning
`--device <host[:port]>`	the switcher. Port defaults to 80 (a simulator is usually `:3000`).
`--port <n>`	local port to listen on (default 8535)
`--host <addr>`	local address to bind (default `127.0.0.1`)
`--data <dir>`	where cue stacks are kept

There is a desktop app too — a tray launcher with an interface and port picker.

> **On binding wide.** The default is loopback for a reason: **this proxy is an unauthenticated route to a switcher's entire control surface.** `--host 0.0.0.0` hands that to everyone on the network. Do it deliberately, not by habit.

Open it through the proxy, not at the switcher

If you open the switcher's own address directly, the panels are not there and MIDI will not work. Web MIDI is a secure-context API; `http://127.0.0.1:<port>` counts as one and a plain-HTTP LAN address does not. The panel says so rather than failing silently.

Console

Verb first, then objects, innermost scope last. Every keyword abbreviates to any unambiguous prefix; **Enter executes and nothing is sent before it.**

```
Recall Screen 1 Memory 5
R Sc 1 Th 4 Me 5 Pre          the same command
Store Master 12
Take Screen 1
```

The line **parses as you type and shows what it will do** — `Recall 3 → Screen 1 Preview` — before anything reaches the device. Tab completes; ↑ recalls history.

This panel owns no grammar. Every token, rule and device path comes from mynah's own build output — the same artefact its Companion module vendors. **If a command means the wrong thing, the fix is in mynah.**

That is worth more than tidiness: mynah's compiler and this repo's command builder were arrived at independently, and they emit **byte-identical** store paths for the commands both know. Two independent derivations agreeing is the strongest evidence either is right, and it stays true only while nobody re-types the grammar here.

MIDI Mapping

Under **Virtual RC400T**. Pick an input, an output for feedback, and a controller profile; press Start. Faders, encoders and buttons then drive the selected layer.

The engine is awj-surface's, vendored whole along with its stock profiles — X-Touch/Mackie, APC40, MIDIcon 2 and Pro, plus a generic learn profile.

Soft pickup is the behaviour worth knowing. A non-motorised fader will not move a live value until it has swept *through* that value, so picking up a fader mid-show cannot jump a layer's opacity. **The panel shows the hold-off rather than looking broken** — if a fader appears dead, that is what you are seeing.

Arithmetic in numeric fields

Type an expression into one of Web RCS's own numeric fields and it is evaluated when you commit — on Enter, or on leaving the field:

typed	becomes
`1080-80`	`1000`
`(1920-40)/2`	`940`
`1920*2`	`3840`

`+ - * / ( )` and decimals, with the usual precedence. **The result is clamped to the field's own declared range and rounded to its step**, so a width of `9000+1000` lands on 8192 rather than being refused.

Nothing else about the field changes: the vendor still validates it, still decides what to send, and still drags the other axis along if the aspect lock is on.

Where it applies is narrow on purpose. Only fields the vendor itself marks as numeric by putting `min`, `max` and `step` on them. That excludes everything that must not be touched — labels have no `step`, and neither does the transition-time field, whose `00:01.000` would be mangled by anything treating `:` as arithmetic.

Opacity and zoom are deliberately excluded. They are real number inputs, and a number input *discards* anything it cannot parse — after typing `1080-80` the browser reports an empty value, and the expression is visible on screen but unreadable from script. Winning arithmetic there would mean mutating a vendor element's type on every focus, in a UI driving a live show. Not a good trade.

There is **no** `eval` anywhere in this path; anything the parser does not fully understand is refused rather than guessed at.

The demo environment

```
npm run demo
```

brings the whole thing up against a **LivePremier Simulator**, with a real Aquilon C's resource map folded in and an example cue stack loaded, so every panel has something real to show without a switcher in the room.

The simulator has **no VPU at all**, so the panel against a bare one correctly reports there is nothing to draw; the demo supplies that from the recorded capture — 32 of 64 mixers fitted, 26 allocated, one screen in Optimized mode, and a staged preconfig differing in 26 mixers. Everything outside that stays the simulator's own live state, **so cues fired from the timeline really do go on the wire.**

> **It refuses to run against anything that is not loopback.** The demo splices a store subtree and seeds a cue stack, and "I thought it was the simulator" is exactly the mistake worth making impossible.

The demo's cue stack lives in a temporary directory and is deleted on exit; your own is never touched.

If something is wrong

Symptom	Cause
No panels in Web RCS	You opened the switcher's address directly. Go through the proxy.
MIDI does nothing	Same cause — Web MIDI needs a secure context, which loopback is and a LAN address is not.
A fader does not move anything	Soft pickup. Sweep it through the current value.
A console command means the wrong thing	The grammar is mynah's; the fix is there.
Arithmetic does not work in a field	It is not one the vendor marks numeric — opacity and zoom are deliberately excluded.
The VPU panel says there is nothing to draw	A simulator has no VPU. That is correct, not a failure.
The demo refuses to start	It only runs against loopback, on purpose.