# Companion — BlackMatrix user guide

This module drives a BlackMatrix crosspoint server from a Stream Deck or any other Companion surface: **a router panel for a fleet of Blackmagic ATEM switchers, where every bus that takes one source at a time is a destination.**

The README covers installing the module. This is how to build a panel with it, and where a press can mislead an operator.

> **Before you rely on this:** BlackMatrix has **no authentication** and binds every interface. Anyone who can reach its port can re-route program on a live switcher. Private production network only, and note that a press here re-points a live feed with no confirmation step.
>
> This module has never been loaded into Companion. It is verified by a harness driving its real source against a fake crosspoint server, which is a different claim from "it works on a rack".
>
> This module was built with AI assistance, directed and reviewed by a human author.

---

## Connecting

Two fields: the BlackMatrix server's host, and its port — **8533** by default. This is the crosspoint server's port, not an ATEM's.

The module holds a WebSocket for state and uses REST for commands, so a connection that is green but never updates is a different fault from one that will not take. Watch `$(blackmatrix:connection_status)` for the first and the module's log for the second.

---

## Build an X-Y panel, not a crosspoint matrix

This is the thing to get right before you lay out a page.

A fleet of switchers has thousands of crosspoints. One button per crosspoint is not a page, it is a spreadsheet. The generated presets are a **router panel** instead:

1. Press a **destination**. It turns blue. Nothing has been sent to any switcher yet.
2. Press a **source**. That crosspoint is made.

Two hundred buttons cover what tens of thousands would, and it is the interaction every router operator already has in their hands.

**The selection lives in this Companion**, not in the server. Two Companions on the same fleet each have their own armed destination and never fight over a half-finished take. The other side of that: a destination armed on the rack Companion is invisible on the one in the gallery.

Drop `Fleet → X-Y: what is selected` somewhere on the page. An armed destination that scrolled off the visible page is how a source press lands somewhere nobody expected.

### The panel reads as a monitor wall between takes

Destination buttons carry a live label of what they are currently carrying, so the panel is worth looking at even when nothing is armed. Source buttons light **amber** when the armed destination is already taking them — the route you are about to make is shown before you make it.

---

## Addressing, when you build buttons by hand

Destinations and sources are `deviceId:id`:

| Reference | What it is |
|---|---|
| `stage:aux.0` | Aux 1 on the switcher called `stage` |
| `stage:me.0.usk.1.fill` | ME 1 upstream keyer 2's fill |
| `stage:1` | Input 1 on `stage` |

Both dropdowns accept a custom value, so a variable can drive either end.

**A crosspoint lives on one switcher.** Routing `studio`'s source to `stage`'s aux is rejected by the module before a request is sent, because an ATEM cannot take another ATEM's source. If a button does nothing at all, check both halves name the same device.

---

## Salvos fire across the fleet, and partially

A salvo sets crosspoints on every switcher in it at once. Salvos are defined **in BlackMatrix**, not here — this module only fires them.

If part of a salvo is refused — a locked destination, or a source that switcher will not accept on that bus — **the rest still happens**, and the refusals are logged by name. That is the right behaviour for a live show and the wrong assumption for a button: a salvo button that lit up is not proof the whole salvo landed. Where a salvo has to be all-or-nothing, the check belongs in front of the button, not on it.

The server's refusals are worth reading rather than dismissing. "Aux 1 is not available on ME 1 Program" is a switcher doing its job, not a broken module.

## Locks are per IP address

BlackMatrix holds locks the way a Videohub does: by the **address** of whoever took them, not by a user or a session.

- A lock taken in BlackMatrix's browser UI on the machine running Companion is **the same owner** as one taken from these buttons. It will unlock without complaint.
- A lock taken from any other machine is somebody else's, and only **force** takes it.

Lock presets are generated for the **Outputs** (aux) section only, because that is the case worth a button. The lock *action* works on any destination, so build one by hand if a keyer needs protecting.

---

## Feedbacks, and which button they belong on

| Feedback | Put it on | Says |
|---|---|---|
| `crosspoint` | a direct-route button | this destination is taking this source |
| `sourceOnSelected` | source buttons | the armed destination already has this |
| `destinationSelected` | destination buttons | this one is armed |
| `locked` | destination buttons | locked, by anyone |
| `deviceOnline` | a per-switcher tile | that ATEM is connected to BlackMatrix |
| `panelAttached` | a per-switcher tile | a hardware Videohub panel is on that switcher |
| `serviceUp` | one tile on the page | BlackMatrix itself is answering |

`serviceUp` and `deviceOnline` answer different questions and fail separately. BlackMatrix can be perfectly healthy while the switcher it fronts has dropped off the network — the panel keeps taking presses and nothing reaches the ATEM. One tile of each, near each other, is the layout that sends people to the right fault.

`panelAttached` is the one people leave out and then want. A hardware Videohub panel on the same switcher routes the same destinations this page does, with no idea that this page exists.

---

## Variables worth knowing

| Variable | Value |
|---|---|
| `<switcher>_<destination>_source` | live label of what that bus is carrying |
| `selected_destination_label` | the armed destination's name |
| `selected_destination_source` | what the armed destination is currently taking |
| `<switcher>_videohub_port` | that switcher's Videohub port, `off` when it has none |
| `<switcher>_videohub_panels` | how many Videohub clients are attached to it |
| `locked_count`, `device_count`, `online_count`, `salvo_count` | fleet counts |

`$(blackmatrix:stage_aux_0_source)` on a button is how a monitor-wall button labels itself.

Destination ids are flattened for variable names — `stage:aux.0` becomes `stage_aux_0`.

## When a button does nothing

| Symptom | Where to look |
|---|---|
| Every button dead, no variables | Wrong host or port, or BlackMatrix is not running. `serviceUp` is the tile that says so. |
| Labels present but a take does nothing | The destination is locked, or the source is not legal on that bus. The module logs the server's reason. |
| Source press goes somewhere unexpected | A destination was still armed from earlier. Put the X-Y selection tile on the page. |
| Route silently refused | Source and destination name different switchers. A crosspoint cannot cross devices. |
| Panel disagrees with the rack | A hardware Videohub panel is routing the same destinations. `panelAttached` shows when one is on. |