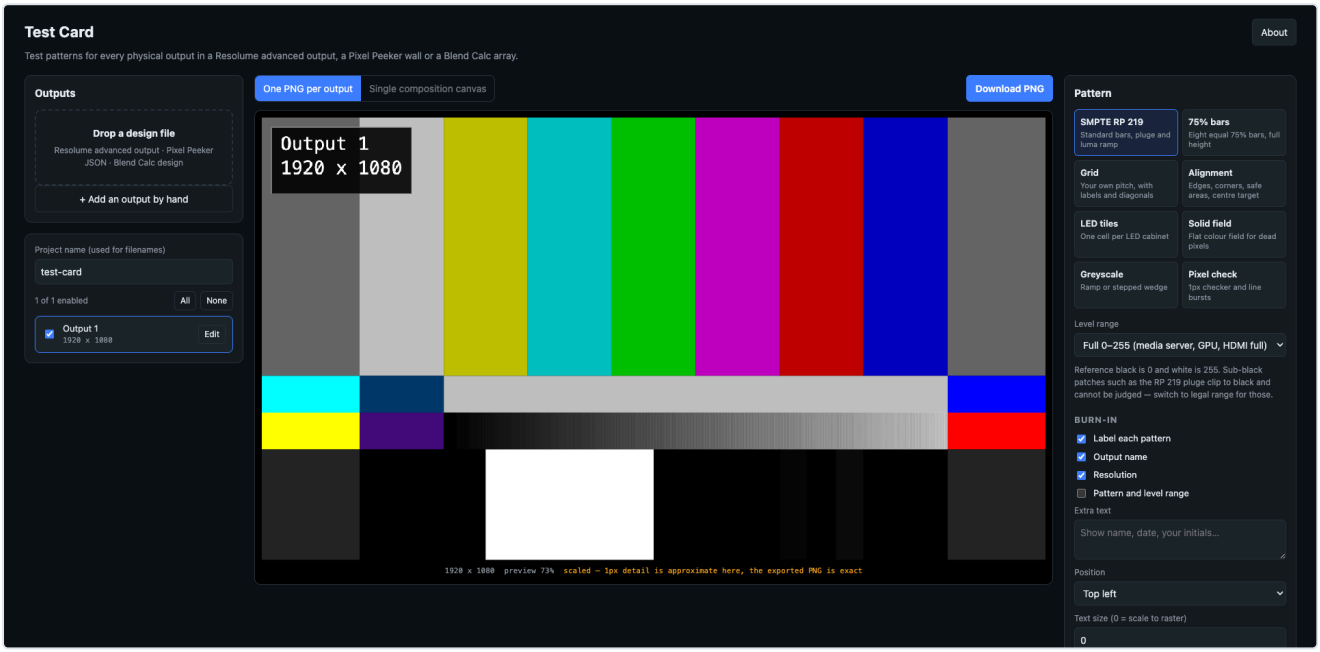
Test Card user guide

Test Card **generates test patterns for every physical output in a show, from the file that already describes the outputs.**



One 1920 × 1080 output added by hand, SMPTE RP 219 bars at full range with the output name and resolution burnt in. The preview is scaled; the exported PNG is pixel-exact.

Import a **Resolume advanced output**, a **Pixel Peeker** wall or a **Blend Calc** projector array, and get one PNG per physical output at that output's exact raster — or a single composition-sized image that proves the output map is what you think it is. Everything runs in the browser; nothing is uploaded.

> **Before you rely on this:** the RP 219 pattern is asserted against a reference **measured** from a real generator at four resolutions rather than transcribed from prose descriptions of the standard, and its fifteen colours are computed through Rec. 709 and land exactly on those measured values — which is itself the proof the measurement is sound. The importers are tested against **real files**, not hand-written fixtures, and the whole path has been driven end to end in a browser.
>
> **No image it produces has been played out to a real projector, LED processor or display and measured.** The patterns are correct as pixels; what a screen does with them is not something this can tell you.
>
> This codebase was created with AI assistance, directed and reviewed by a human author.

The patterns

Pattern	What it is for
SMPTE RP 219	Standard bars, +I/+Q, luma ramp and pluge.
75% bars	Eight equal full-height bars. Survives being squeezed onto a narrow LED strip where RP 219's four rows would be unreadable.
Grid	Your own pitch or division count, heavy lines every N, cell labels, diagonals, centre target.
Alignment	1px edge border, corner markers with pixel counts, safe areas, centre target and circles, 1px checker patches.
LED tiles	One cell per cabinet — either a uniform tile size you state, or the **real cabinet map** imported from Pixel Peeker, including irregular walls, chain order and per-port colouring.
Solid field	Flat colour at a chosen level, for dead pixels and uniformity.
Greyscale	Continuous ramp or stepped wedge, optionally split into R, G, B and neutral.
Pixel check	1px checkerboard and line bursts. **If it renders as flat grey on the real output, something in the chain is scaling.**

Every pattern can carry a burn-in — output name, resolution, pattern, plus your own text — and can draw the imported slice or port outlines on top.

Two ways to render, and when each is the right one

One PNG per output, each at that output's native raster. One output downloads as a bare PNG; several download as a ZIP with a `manifest.txt` recording which file is which output, the level range used, and **any raster the app had to infer**.

Single composition canvas — one image at composition size with every output's region drawn and labelled. Play it through Resolume and **each physical output should light up showing its own name.**

That second one is the fastest proof the advanced output map matches what you think it does, and it **needs no way of getting a file to each output individually** — which on a show day is often the real constraint.

Level range: the app asks because a PNG cannot say

SMPTE bars are defined in Y′CbCr. A PNG is RGB, and **a PNG cannot record which convention it was written in.** So the app asks, and writes the answer into the manifest.

- **Full 0–255** (default) — media server, GPU output, HDMI/DP set to full range. **The −2% pluge patch clips to black and cannot be judged.**
- **Legal 16–235** — studio swing, for an SDI chain. Sub-black and super-white survive, so the pluge is meaningful. **Looks washed out on a full-range display, correctly.**

Pick the one that matches the chain the file is going down, not the one that looks right on your laptop.

Imports, and what each file does not know

Resolume — one output per screen. The raster is read from the output device when it records one, and otherwise **inferred from slice bounds, which can come out too small.** Inferred rasters are badged in the UI and called out in the manifest — **correct them by hand.**

Pixel Peeker — use **Export → JSON**, not *Save project*. The project file stores cabinets as library references in millimetres and cannot be resolved to pixels here, so it is **refused with a message naming the right export** rather than half-imported.

Blend Calc — needs the projector resolution (not in the file) and, for designs using a fit mode, the **solved** overlap (also not in the file — Blend Calc computes it and never writes it back). Both are asked for. **Its Resolume XML export needs neither and is the better file to bring.**

You can also just add outputs by hand.

If something is wrong

Symptom	Cause
The pixel-check pattern is flat grey on the output	Something in the chain is scaling. That is exactly what the pattern is for.
An output's raster looks too small	It was inferred from slice bounds. It is badged; correct it by hand.
A Pixel Peeker file is refused	It is a project save, not the JSON export.
The pluge patch is invisible	Full-range levels clip it to black. Render at legal range for an SDI chain.
The bars look washed out	Legal-range content on a full-range display. Correct, not broken.
Blend Calc import asks for numbers	The projector resolution and the solved overlap are not in that file. Bring its Resolume XML instead.