amber user guide

amber makes **legacy Flash content playable in Resolume**. Resolume opens neither `.swf` nor `.flv`, so a decade and a half of VJ loops, web animation and motion tests is sitting in formats nothing on the machine will play.

There are two ways out, and they are good at different things:

- **The converter** turns `.swf` and `.flv` into DXV or Hap — the codecs Resolume decodes on the GPU — preserving frame rate, transparency and dimensions. Do this for a library.
- **The live plugin** plays a `.swf` on a layer with its timeline and ActionScript actually running. Do this when the movie needs to be interactive, or transparent.

> **Before you rely on this:** the converter has been run end to end against real 2003-era Flash and against real-world FLV in all three of its common codecs, with the output verified frame-accurate by decoding it again and comparing pixels. 39 tests cover it, and the codec-constraint tests **re-derive** the DXV and Hap dimension rules from the real encoder rather than asserting a table against itself.
>
> **The plugin is confirmed working in Resolume Arena on Apple Silicon.** Its Intel slice has never been run in an Intel Resolume, and the Windows build has never been loaded into a host.
>
> This codebase was created with AI assistance, directed and reviewed by a human author.

Installing

amber **vendors nothing**. It drives your own ffmpeg and your own build of Ruffle's exporter as subprocesses, which is what keeps it cleanly MIT with no linking question to answer. So there are two dependencies to satisfy before it can do anything.

```
brew install ffmpeg-full
```

`ffmpeg-full`, not plain `ffmpeg`. The plain formula has the DXV encoder but **not** the Hap encoder, so transparent content would have nowhere to go.

For `.swf` support you also need Ruffle's exporter, which Ruffle does not ship in its releases:

```
git clone --depth 1 https://github.com/ruffle-rs/ruffle ~/Documents/Amber/ruffle-src
cd ~/Documents/Amber/ruffle-src && cargo build --release --package=exporter
```

amber finds it there automatically. `.flv` conversion works without it.

Then run `amber doctor` before anything else. It reports which ffmpeg it found, whether that build has Hap, and whether the Ruffle exporter is present — which is the difference between "this file cannot be converted" and "this machine cannot convert that file yet".

Converting

```
amber doctor                          # what can this machine do?
amber info  ~/clips/badger.swf        # describe it without converting
amber convert ~/clips -o ~/converted  # a whole directory
```

```
badger.swf → badger.mov  544x400 @ 25fps  904 frames  15.2MB  [dxv]  [resized 550x400 → 544x400 (scale)]
```

The output profile is chosen from what the source needs, not from a default you have to remember:

Profile	When	
`dxv`	opaque content	Resolume's own codec, GPU-decoded
`hap_alpha`	the source has alpha	GPU-decoded, alpha byte-exact
`hap_q`	on request	46.8 dB against DXV's 45.0, worst-case error halved, ~80% larger
`prores4444`	alpha fallback where Hap is unavailable	CPU-decoded, much larger

---

Three things about DXV, before you convert a library

DXV cannot carry transparency, and does not say so. ffmpeg's DXV encoder offers exactly one format, whose own help text reads "Normal Quality, No Alpha". Handed an image with alpha it does not warn — it returns every pixel opaque. Since VP6-alpha exists precisely for transparent overlay loops, amber refuses to send alpha content to DXV and routes it to Hap Alpha instead. `--flatten` overrides that if you genuinely want the alpha gone.

DXV silently corrupts any width that is not a multiple of 16. The encoder returns success, writes a plausible file, records the right dimensions — and the picture inside is sheared diagonally. This is not a corner case: **550×400 is the default Flash stage size**, so a naive pipeline mangles a large fraction of all Flash ever made.

amber measures the constraint and works around it. For `.swf` that costs nothing — Flash is vector, so Ruffle rasterises straight to a legal size. For `.flv` the frames are raster and have to be resampled (`--fit scale` , the default) or padded (`--fit pad`).

DXV blocks up on smooth gradients. It is DXT1 at a fixed 4:1, so skies and soft falloffs show visible 4×4 blocking. Measured against a lossless reference on gradient-heavy 720p: DXV 45.0 dB with a worst-case error of 44; `hap_q` 46.8 dB with a worst-case error of 19, for about 80% more file. Reach for `--profile hap_q` when the content is gradient-heavy and the disk can take it. **Plain `--profile hap` is pointless** — identical quality to DXV and 27% bigger.

And plan disk space: 28 seconds of 3840×2160 came out at 1.4 GB. That is the codec working as intended, trading size for GPU-decodable playback.

The live plugin

Install `Amber.bundle` into `~/Library/Graphics/FreeFrame Plug-Ins` and it appears as a **source**, not an effect — it generates a layer rather than processing one.

Parameter	
Movie	the `.swf` to play
Run / **Restart**	transport
Speed	playback rate
Scaling	Fit / Fill / Stretch
Smoothing	filtering on the rasterised frame
Transparent	on by default

Transparent is what makes Flash usable as an overlay. Ruffle implements Flash's own `wmode=transparent`, so everything the movie does not draw comes through clear and the layers underneath show. Turn it off to get the movie's declared stage colour instead.

Only the plugin can do transparency — the converter cannot. The converter drives Ruffle's `exporter` binary, which has no background option and always renders opaque. So a `.swf` that needs a transparent background has to be played live. That is a gap in the exporter's command line rather than in Ruffle, and the converter tells you rather than quietly handing back an opaque clip.

Three limits worth knowing before a show

- **There is no audio.** FFGL provides no audio path at all, so a live Flash clip is silent no matter what it contains. The converter has the same limit.
- **Ruffle runs inside Resolume's own process.** Every call into it is guarded, which turns most bad content into a black layer rather than a crash — but a guard is not a process boundary, and content that hard-crashes Ruffle takes Resolume with it.
- **Only the Apple Silicon slice has been run in a host.** The bundle is universal and a Windows build exists; neither the Intel slice nor the Windows DLL has been loaded into a Resolume.

Live playback is `.swf` only. `.flv` goes through the converter.

If it goes wrong

`.swf` files are refused or skipped. Ruffle's exporter is missing — `amber doctor` says so. `.flv` does not need it.

Transparent content came back opaque. Either it went to DXV with `--flatten`, or it was a `.swf` put through the converter, which cannot preserve alpha at all. Play it live instead.

The picture is sheared diagonally. A width that is not a multiple of 16 reached DXV. amber guards against this, so seeing it means something went round the converter.

Everything looks blocky in the sky. DXV's fixed 4:1 compression. Convert that clip again with `--profile hap_q`.

Resolume disappeared while a Flash clip was playing. Ruffle hard-crashed inside Resolume's process. That content cannot be played live until an out-of-process helper exists; convert it instead.
