# OFX Plug-ins in Hope Creative

A guide for OpenFX plug-in developers: what the Hope Creative host does, how it differs from other hosts, and what to check so your plug-in runs at its best.

Host version 1.0 (pre-release) · Windows x64 (macOS planned) · October 2026

---

## Contents


1. About Hope Creative
2. Quick facts
3. Installation and discovery
4. Identifying the host
5. Images, coordinates and time
6. Region of Definition
7. Temporal clip access
8. GPU rendering
9. Parameters
10. Extra input clips
11. Overlays (on-screen controls)
12. Messages
13. Threading, instances and stability
14. Resolve compatibility extensions
15. Not supported yet
16. Testing checklist
17. Further reading


## 1. About Hope Creative

Hope Creative is a layer-based motion graphics and compositing application. It hosts OpenFX image effect plug-ins through a general OFX host written against the official OpenFX headers. There is no host-specific shortcut for any one plug-in: if a plug-in follows the OFX standard, it should run.

The most important difference to an editing or node-based host: an OFX effect is applied to a **layer**. The picture your plug-in receives is that layer's own picture (a video, a text, a shape, a solid…) before the layer is transformed into the composition. Effects on adjustment layers receive the composite below them.

Hope Creative also has its own native plug-in format ( `.hpx` ) with GPU effects on WebGPU, CPU effects and modifiers; see the *Native Plug-in Guide* if you want to build for Hope Creative directly.

## 2. Quick facts

| Item | Value |
|---|---|
| Host name ( `kOfxPropName` ) | `com.hopecreative.HopeCreative` |
| Host label ( `kOfxPropLabel` ) | `Hope Creative` |
| API version | 1.4, plus selected 1.5 features (StrChoice parameters, dynamic choice menus, Draw Suite overlays) |
| Platform | Windows x64 ( `Contents/Win64` ); macOS is planned |
| Contexts | Filter (preferred), General, Generator |

| Item | Value |
| --- | --- |
| Pixel format | RGBA, 32-bit float, premultiplied; one depth for all clips |
| Multi-resolution | Yes (render scale follows the viewer resolution) |
| Tiles | No: the render window is always the whole image |
| Temporal clip access | Yes |
| GPU | CUDA (with stream) and OpenCL (buffers) · no Metal or OpenGL rendering yet |
| Overlays | Yes: OpenGL overlay interacts (V1) and the OFX 1.5 Draw Suite (V2), pen and key events |
| Native origin | Bottom left |

## 3. Installation and discovery

Hope Creative looks for plug-ins where every OFX host looks:

- `%CommonProgramFiles%\OFX\Plugins` (usually `C:\Program Files\Common Files\OFX\Plugins`)
- every folder listed in the `OFX_PLUGIN_PATH` environment variable

Only folders named `*.ofx.bundle` are plug-ins (searched in sub-folders too). The binary is `<name>.ofx.bundle/Contents/Win64/<name>.ofx`; the bundle's `Win64` folder is on the DLL search path while the binary loads, so your own DLLs can sit next to it.

### How plug-ins are described

- Each new or changed binary is loaded and described in a **separate child process** (`HopeCreative.exe --ofx-describe`), several at a time, with a 40 second time-out. A plug-in that crashes or hangs while being described only takes that child process along; it is listed as "can't be used" with the reason.
- The child process runs `Load`, `Describe`, `DescribeInContext`, then makes **one instance** (`CreateInstance` / `DestroyInstance`) to read menus that are only filled at run time (see *Parameters*), and `Unload`.
- The result is cached per binary (file size and modification time). After an update of your plug-in it is described again automatically.
- Users can switch every plug-in on or off in **Developer ▸ OFX Plug-ins…**. Usable plug-ins appear in the **Effects** panel under the category **OFX**, by their label.

> **Please keep `Load` and `Describe` fast and free of side effects** (no dialogs, no blocking network calls). If your licence check needs time, do it lazily, at `CreateInstance` or the first render, and report problems through the Message suite (section 12).

## 4. Identifying the host

The host always identifies itself honestly as `com.hopecreative.HopeCreative`; it never presents itself as another host and never touches a plug-in's licensing. If your licensing or feature set depends on the host, check `kOfxPropName` on the host property set.

## 5. Images, coordinates and time

- **Pixels:** RGBA float, premultiplied by alpha. Rows run bottom-up (OFX convention); image bounds start at (0, 0); `kOfxImagePropRowBytes` = width × 16.
- **Colour:** the image is display-encoded sRGB (not scene-linear), the same encoding the application's own effects work in. Plug-ins that offer their own input colour space choice (for example a "source" or "input" menu) keep full control over it.
- **The project is the layer:** project size / extent = the layer's picture in canonical coordinates at render scale 1 (plus any margin from your Region of Definition, section 6). Pixel aspect ratio is 1.0; no fields.
- **Render scale** follows the preview resolution (Full, Half, Third, Quarter); exports always render at 1.0.
- **Time** is in frames of the layer's own timeline (frame 0 = the layer's start, where its source begins: trimming doesn't shift it), at the composition's frame rate. Motion blur in an export renders in-between, fractional times.
- **Point parameters** (XY, XYAbsolute, normalised variants) are shown as handles in the viewer and converted to and from canonical coordinates of the image you receive.

## 6. Region of Definition

Hope Creative calls `kOfxImageEffectActionGetRegionOfDefinition`. If your result reaches beyond the input (glows, shadows, blurs, distortions), return the larger rectangle: the layer's working picture grows by that reach (up to 2048 pixels per side), so nothing is cut at the layer's edge. Without it, the result is clipped to the layer, which is very visible on text and shape layers.

## 7. Temporal clip access

- Set `kOfxImageEffectPropTemporalClipAccess` and fetch other frames with `clipGetImage` at other times.
- Implement `kOfxImageEffectActionGetFramesNeeded` where you can: announced frames are provided with the first render.
- Frames fetched without announcing them also work: the host notes them, marks that frame as provisional and renders it again (the preview cache and exports only keep the correct result). Announcing simply saves that second pass.
- Other frames are the layer's input at those times, as it sits now (the content plus the effects above yours).
- Not available on 3D layers and adjustment layers yet.

## 8. GPU rendering

| Path | What your plug-in gets |
|---|---|
| CUDA | `kOfxImageEffectPropCudaEnabled` = 1, image data = CUDA device pointers (linear RGBA float), `kOfxImageEffectPropCudaStream` set. The device's primary context (device 0) is current during the render, so the CUDA runtime and driver API both work. |
| OpenCL | `kOfxImageEffectPropOpenCLEnabled` = 1, image data = `cl_mem` buffers, `kOfxImageEffectPropOpenCLCommandQueue` set (first GPU device found). |

| Path | What your plug-in gets |
|---|---|
| CPU | Host memory, always available as the fallback. |

The host tries CUDA first, then OpenCL, then the CPU; if a GPU render fails, the frame is rendered on the CPU.

- **CUDA, zero-copy:** when CUDA runs on the graphics card Hope Creative renders on, your images (the source, other moments, extra inputs and the output) live in GPU memory that Hope's Direct3D 12 device and CUDA share. Nothing goes through host memory: on an RTX 5080 moving a 4K frame to the plug-in and back takes about 1.5 ms instead of about 35 ms. The pointers are ordinary CUDA device pointers (packed rows); there is nothing to do on your side.
- **Otherwise** (OpenCL, CUDA on another card, sharing not available) images are copied through host memory on the way in and out. That trip is optimised: about 7.5 ms for a 1080p frame, and a CPU plug-in reads the picture where it lands, without extra copies.
- The host waits for your CUDA work to finish (stream and context synchronised) before it uses the result, so you may work on your own streams as well.

## 9. Parameters

Parameters become ordinary effect properties: users can keyframe them, drive them with expressions and modifiers, and undo every change. Groups become collapsible groups; pages are not used.

| OFX type | Shown as |
|---|---|
| Double / Integer | Number with slider (range from display min/max, limits from min/max) |
| Boolean | Checkbox |
| Choice / StrChoice | Menu (a StrChoice keeps its value as text, so it survives reordered entries) |
| RGB / RGBA | Colour |
| Double2D (XY, normalised) | Point with a viewer handle |
| Double2D / Integer2D (other) | Two numbers |
| Double3D / Integer3D | Three numbers in one row |
| String | Text |
| PushButton | Button (a press reaches you as `InstanceChanged`) |
| Custom | Kept and saved, not shown |
| Parametric, Page | Not shown yet; the plug-in keeps its defaults |

- **Dynamic menus:** filling or changing a menu's options at run time (for example a list of installed profiles) is supported, including `propReset` on the options. Menus filled in `CreateInstance` are already known when the plug-in is described.
- Values you set yourself with `paramSetValue` in `InstanceChanged` (a preset that sets several sliders) or while an overlay is dragged are written into the project, in the same undo step as the user's change.
- Secret parameters you hide or show at run time are followed by the user interface.
- The host does not animate Boolean, Choice and String parameters (see the host properties).

## 10. Extra input clips

Every clip besides Source and Output (a matte, a displacement map, a second picture) appears in the effect as a layer picker labelled with the clip's label. The picked layer, as it sits in the composition, is your clip's image; `kOfxImageClipPropConnected` tells whether a layer is picked. Clips can be optional.

## 11. Overlays (on-screen controls)

- **V1** overlay interacts ( `kOfxImageEffectPluginPropOverlayInteractV1` ) draw with classic OpenGL (immediate mode works) in a context of the host's own; the result is laid over the viewer.
- **V2** overlays (OFX 1.5) draw through the **Draw Suite**: lines, stipple, rectangles, polygons, ellipses and text in the viewer's standard colours. V2 is used when a plug-in offers both.
- Your coordinates are canonical coordinates of the layer's picture; `kOfxInteractPropPixelScale` tells you the size of one screen pixel.
- **Events:** draw, pen down / motion / up (pen motion is also sent while hovering), and **keys** (down, repeat, up with `kOfxPropKeySym` and `kOfxPropKeyString` ) while the selection tool is active. Keys your overlay doesn't take stay shortcuts; Ctrl / Cmd combinations always stay the application's.
- Overlays are shown for 2D layers.

## 12. Messages

- `message` (Error, Warning, Message) appears in the application's status line, each distinct text once; Log messages go to the log only.
- `setPersistentMessage` is shown directly under your effect until you call `clearPersistentMessage` : the right place for "licence missing" or "profile not found".
- Questions are answered with "No", since renders are never interrupted by dialogs.
- Session logs: **Developer ▸ Show Log Files**.

## 13. Threading, instances and stability

- All actions of one plug-in are called one at a time.
- One effect on a layer may have several instances at once: the preview renderer, the export and the viewer's overlays each keep their own. Don't rely on a single instance per effect.
- Every call into the plug-in is guarded: if the plug-in crashes, it is switched off until the next start, its effects pass their picture through, and the user is told. The project keeps the effect and all its values.
- Projects keep effects whose plug-in is missing or switched off, with all their values; on a machine with the plug-in they work again.

## 14. Resolve compatibility extensions

Many plug-ins are developed and tested in DaVinci Resolve first. Hope Creative sets `kOfxImageClipPropThumbnail` = 0 on clips and images (documented by Blackmagic in `ofxImageEffectExt.h` ) because plug-ins built with that header expect it. Other host-specific properties are not provided: please treat host-specific properties as optional and fall back to standard behaviour when they are missing.

## 15. Not supported yet

- macOS (planned, including Metal rendering)
- OpenGL and Metal render paths; zero-copy for OpenCL
- Tiles (the whole image is always rendered)
- Parametric and custom parameter interfaces; parameter pages
- Temporal access on 3D and adjustment layers
- Overlays on 3D layers

## 16. Testing checklist

1. Install the bundle in `Common Files\OFX\Plugins`, start Hope Creative and find your plug-in in the **Effects** panel ▸ **OFX** (**Developer ▸ OFX Plug-ins…** lists every plug-in found, with the reason if one can't be used).
2. Apply it to a video layer, a text layer and a small shape layer (Region of Definition, layer-sized images).
3. Change every parameter, use your presets and buttons, undo / redo.
4. Switch the viewer to Half and Quarter resolution (render scale).
5. Play back, then export (H.264 and ProRes; with motion blur for fractional times).
6. Check messages (licence states) under the effect and in the status line.
7. Save the project, reopen it, also on a machine without the plug-in.

**Diagnostics:** the session log is in `%APPDATA%\Hope Studios\Hope Creative\logs` (**Developer ▸ Show Log Files**). The plug-in cache is `%APPDATA%\Hope Studios\Hope Creative\ofx_plugins.json`; delete it to have every plug-in described again. Setting `HOPE_NO_OFX=1` starts the application without OFX plug-ins.

## 17. Further reading

- OpenFX documentation and programming reference: openfx.readthedocs.io
- OpenFX headers, examples and the C++ support library: github.com/AcademySoftwareFoundation/openfx
- The Open Effects Association: openeffects.org

Questions, test builds or something your plug-in needs from the host? Please get in touch, we're happy to help make your plug-in work at its best in Hope Creative.

Contact: Florian Düe · florian.duee@gmx.de