spektrafilm Behavior Port Plan

This plan is the active spektrafilm port contract for this workspace.

Reference implementation: `external/spektrafilm`

Historical touch-point notes are evidence for implementation areas and behavior deltas, not an authority for compatibility policy. If historical notes say to keep old profile compatibility, preserve CPU/CUDA production parity, defer legacy cleanup, or retain old resource names, this plan wins: spektrafilm replacement behavior, CUDA-only production rendering, and deletion or hard blocking of replaced product paths are mandatory. Temporary reference scaffolding is allowed only where this plan explicitly names it.

Local historical material is non-authoritative unless a phase explicitly names it as evidence. In particular, `Archive/`, stale generated artifacts, old branch notes, and existing `scripts/spektrafilm_*` helpers may contain retired phase numbering, old Fast-optics assumptions, or earlier compatibility bridges. Agents must not copy implementation from `Archive/` or satisfy a phase by following a script whose contract conflicts with this plan. When a local script conflicts with this plan, update the script in the owning tooling gate or exclude it from acceptance evidence with a recorded reason.

Goal: replace the current agx-emulsion/Film-Juicer processing behavior with the current spektrafilm behavior while preserving Film-Juicer's context-owned CUDA lifetime invariants and custom GPU grain. The existing broad `WorkingState`, `FrameRequest`, `PreparedCudaFrame`, and ResourceManager command surfaces are replacement targets unless this plan explicitly names a temporary bridge. This is a destructive behavior replacement with a spektrafilm-only host-facing effect contract, not a compatibility migration.

Scope boundary: this plan defines behavior, ownership, descriptor identity, lifecycle, admission, failure semantics, and the production architecture/performance shape needed to avoid slow or fragile implementation. It does not define fixed benchmark targets, VRAM budgets, cache-hit targets, or broad optimization evidence gates; those belong in a future performance-optimization plan if needed. Performance, memory-pressure behavior, hot-path data flow, and resource ownership remain first-class constraints in every phase. This plan still requires descriptors to name resource ownership, lifetime, and estimated size where that information is needed for correct admission and diagnosable failure.

Reference authority:
- `external/spektrafilm/src/spektrafilm/runtime/params_schema.py`
- `external/spektrafilm/src/spektrafilm/runtime/stages/filming.py`
- `external/spektrafilm/src/spektrafilm/runtime/stages/printing.py`
- `external/spektrafilm/src/spektrafilm/runtime/stages/scanning.py`
- `external/spektrafilm/src/spektrafilm/runtime/services/*`
- `external/spektrafilm/src/spektrafilm/model/*`

- `external/spektrafilm/src/spektrafilm/profiles/io.py`

Architecture may differ. Colour science, schema meaning, route behavior, units, channel order, and fallback policy must match spektrafilm unless this plan explicitly names a Film-Juicer extension or deferred item.

Every semantic implementation claim must be checked against `external/spektrafilm` before code lands. The phase manifest records the exact file/function or small reference fixture used for each changed schema field, default, formula, route decision, channel order, failure behavior, or stage ordering rule. Do not validate copied spektrafilm resource correctness beyond the consumed fields needed for the active phase; the bundled resources are owner-supplied copies from spektrafilm and are assumed correct.

Intentional exceptions:
- Film-Juicer remains CUDA-only for production rendering. Validation uses `external/spektrafilm` reference fixtures plus narrow stage-contract checks; the old OFX C++ CPU pixel renderer is not retained as validation authority and must never become an OFX production fallback.
- Film-Juicer retains its visual grain implementation; spektrafilm supplies only the grain schema/statistical contract.
- Exact numerical parity with the Python reference is not expected because Film-Juicer is a GPU production renderer with performance-oriented execution paths. The visual target for comparable implemented routes is below a just noticeable difference when compared under matched route/profile/scanner/output settings.
- Film-Juicer UI controls should follow the spektrafilm reference contract. Reuse, rewire, rename, hide, or remove existing Film-Juicer controls so the host-facing value has current spektrafilm meaning; do not preserve old behavior behind old labels.
- The existing input and output color-space option sets remain available. They map into recipe-owned input/output color and CCTF fields and are treated as Film-Juicer host color-management controls over the spektrafilm render contract.
- Film-Juicer exposes the spektrafilm reference/custom dichroic model plus `durst_digital_light`, `thorlabs`, and `edmund_optics` dichroic choices as typed spektrafilm-resource-backed product controls. The reference/custom model is the parity default; the measured Film-Juicer sets are selectable product extensions and must be recorded in recipe/hash/fixture identity when selected.
- Film-Juicer exposes UV/IR controls under an Advanced UI group. Hanatos 2025 adaptation window, adaptation surface, and spectral Gaussian blur controls belong under the same Advanced group.
- Exact optics is the spektrafilm-matching optics backend. It implements the spektrafilm optics path, including PSF/FFT behavior where the reference uses it. Minor deviations such as lower precision and GPU execution optimizations are accepted only when visual inspection and phase evidence show no material difference under matched route/profile/scanner/output settings.
- Fast optics is out of scope for this plan. A later approved plan may add a lower-cost backend after spektrafilm parity is established, but this port must not add Fast backend identity, Fast resources, Fast UI behavior, or Exact-to-Fast fallback.

- Upstream spektrafilm's ICC/export metadata support is out of render scope. Do not replace Film-Juicer output-color recipes with ICC profile parsing unless a later approved file-export feature explicitly adds that scope.
- Film-Juicer's user-facing halation default is `off`. This is an explicit Film-Juicer product default divergence from spektrafilm's `HalationParams.active=true`; when halation is enabled, it uses spektrafilm parameters, profile-derived presets, stage ordering, and Exact optics behavior.

Runtime/UI defaults:
- Halation defaults off as the explicit Film-Juicer exception above. Parity fixtures must align halation state deliberately: compare with spektrafilm halation disabled for Film-Juicer defaults, or enable Film-Juicer halation and compare against spektrafilm's active halation path.
- DIR defaults active.
- Scanner/print glare defaults active where the selected route consumes glare.
- Film-Juicer visual grain defaults off.
- Diffusion filters default off.
- Default enlarger/print illuminant is `TH-KG3`. Film reference illuminant and scanner/viewing illuminant resolve from the selected profile unless this plan explicitly names an override.
- Default dichroic filter set is the spektrafilm reference/custom model used by `color_enlarger` when no explicit filter set is passed.
- Auto-exposure defaults enabled and its method defaults to `center_weighted`; the supported spektrafilm methods are `center_weighted`, `average`, `median`, `partial`, `matrix`, `multi_zone`, and `highlight_weighted`. Production metering follows spektrafilm `small_preview`: before any crop/rescale, downsample by nearest neighbor to max long edge 256 when needed, otherwise meter the input as-is.
- Print exposure compensation defaults enabled and print exposure normalization defaults enabled.
- Scanner black/white correction controls default to `black_correction=false`, `white_correction=false`, `black_level=0.01`, and `white_level=0.98`.
- Film-Juicer intentionally keeps its current host color-management controls and defaults: input color space `DaVinci Wide Gamut`, input CCTF decode `false`, output color space `sRGB`, and output CCTF encode `true`. These are Film-Juicer product defaults over the spektrafilm render contract, not old compatibility behavior and not spektrafilm's Python `IOParams.input_color_space="ProPhoto RGB"` default. Reference comparisons must set matching input/output color and CCTF controls, or label the comparison as Film-Juicer extension-mode evidence.
- `density_min` defaults to C/M/Y density `(0.07, 0.08, 0.12)` and remains active for direct film scanner bounds and enlarger film-density bounds even when visual grain is off.
- Hanatos adaptation defaults match current spektrafilm settings: adaptation window on, adaptation surface off, and spectral Gaussian blur `0.0`.
- Spektrafilm runtime defaults `settings.use_enlarger_lut=false` and `settings.use_scanner_lut=false`; Film-Juicer intentionally treats LUTs as GPU execution strategy, not behavior toggles. Production LUTs must still match spektrafilm spectral math and interpolation semantics. Scanner/enlarger 3D LUT acceleration follows spektrafilm `compute_with_lut`: endpoint-inclusive per-channel grids from `xmin`/`xmax`, normalized lookup

coordinates `(data - xmin) / (xmax - xmin)`, and `apply_lut_3d` default `pchip` interpolation with prepared per-axis Hermite slopes and per-cell min/max clamping. Hanatos filming TC LUT sampling is separate and follows spektrafilm `apply_lut_cubic_2d` reflected-boundary Mitchell cubic interpolation plus the `_rgb_to_tc_b` brightness scale. Trilinear, bilinear, Mitchell-3D, or current Film-Juicer scanner/film LUT sampling may be used only as an explicitly approved output-equivalent implementation strategy with fixture evidence.
- Film-Juicer keeps selectable illuminant options comparable to the current product, but recipe identity stores the typed resolved illuminant key and applies spektrafilm illuminant semantics.

The plan is route-first and recipe-first. The diagram shows ownership flow, not permission to pass broad objects through every layer. Consumers take the narrowest subrecipe, descriptor, prepared view, or lease that fully defines their behavior.

```text
ProfileCatalog / ProfileAssets
      |
      v
RenderRecipe / named subrecipes
      |
      v
FrameRequest frame-local facts + stable recipe snapshot/reference
      |
      v
Resource-family descriptors / scratch-role descriptors / CUDA payload descriptors
      |
      v
PreparedCudaFrame stage views, leases, handles, and frame scratch
      |
      v
CUDA render
```

Everything in the port should make that path clearer, smaller, and easier to validate.

Lean-port amendments from the code-debloating review:
- This is a deletion-first port. When a spektrafilm behavior lands, the old Film-Juicer behavior it replaces is deleted or hard-blocked in the same owning phase unless this plan names a temporary bridge and removal phase.
- Do not create scaffolding ahead of consumers. A recipe group, field, descriptor, hash lane, diagnostic field, script, wrapper type, resource-family contract, or bridge exists only when a current phase consumes it for validation, hashing, CUDA packing, route behavior, diagnostics, or lifecycle safety.
- Validation is minimal consumed-contract validation, not file/resource certification, user-asset hardening, or legacy migration. Production profiles/resources are direct owner-supplied copies

from spektrafilm except for explicitly named Film-Juicer resource exceptions, and may be assumed correct. User-supplied profiles/resources are expected to be compatible; it is not the plugin's job to lint, repair, migrate, or explain them. Runtime checks exist only to select visible profiles, build the current recipe, prepare required resources, and avoid undefined CUDA/render behavior: key presence, route metadata needed for menus, consumed array dimensions, wavelength-axis identity for consumed spectral data, consumed finite/NaN/null policy, selected resource presence, duplicate key checks, and explicit unsupported-state failures. Delete validation whose purpose is old Film-Juicer project compatibility, arbitrary folder/CSV ingestion, silent substitution, broad user guidance, migration diagnostics, or proving copied spektrafilm files are well-formed.
- Diagnostics are compact gated failure paths, not telemetry or user-file troubleshooting. Add only the fields needed to make a current product failure actionable; do not create universal diagnostic records, per-phase class hierarchies, always-collected verbose evidence, or profile/resource lint reports.
- Resource-manager changes must reduce old ceremony over time. Preserve context ownership, epoch correctness, in-flight lifetime, finish/abort, admission failure reporting, and descriptor identity. Do not extend old registry/policy/cache/scratch systems unless the phase removes more legacy surface than it adds and records why the hook is mechanically required.
- The port is not accepted if Phase 10 becomes the first large cleanup sweep. Each phase owns deletion of the old behavior it replaces.

Stable terminology:
- `RenderRecipe` means resolved output-affecting policy.
- `FrameRequest` means immutable per-frame inputs and frame-local tokens.
- `PreparedCudaFrame` means prepared handles, leases, views, scratch, staging, and finish/abort state.
- `ScannerSpectralLutDescriptor` means LUT-generation identity only.
- `scanner post-effect descriptor` means black/white correction, glare, blur, unsharp, output color, and output CCTF identity.
- `FilmJuicerMainCFilterShift` means the user C-filter extension.

Non-Negotiables

1. spektrafilm behavior is the only supported behavior after the port.
   - Do not add a global legacy mode.
   - Do not preserve saved-project compatibility for old integer `FilmStock` / `PrintPaper` params.
   - Do not detect, migrate, diagnose, test, or preserve old Film-Juicer saved-project state.
   - Keep the existing OFX identifier. Retaining that identifier does not create any obligation to understand old saved projects or removed params.
   - Treat spektrafilm as the only effect contract in architecture, parameters, validation, and render behavior. Old-project behavior remains outside the product contract and must not drive shims.
   - Remove old exposure normalization, print filter units, scanner defaults, DIR behavior, halation behavior, print behavior, and profile loading as their callers move.

2. Production rendering is CUDA-only.
   - CUDA is the production backend now.
   - Do not add abstraction for hypothetical future GPU backends. The host `RenderRecipe` should stay clear enough for a future backend to consume, but no future-backend machinery is part of this port.
   - CUDA-only means no CPU production pixel-render path; it does not prohibit host CPU control-plane work where CPU code is simpler and fast enough.
   - Host CPU code may load profiles, build recipes, perform minimal consumed-field checks, hash, diagnose, and prepare resources.
   - Host CPU work may include parsing, minimal consumed-schema checks, recipe construction, hashing, diagnostics, descriptor construction, LUT/profile derivation, and resource-preparation orchestration when that work is not a render fallback.
   - Host CPU code must not remain a production pixel-render fallback.
   - The old OFX CPU pixel renderer is not a spektrafilm reference. Do not retain `processImpl()`, `multiThreadProcessImages()`, old scanner-runtime CPU rendering, old `PipelineRunner` paths, or any broad C++ CPU pixel path as validation authority.
   - Validation authority comes from `external/spektrafilm` plus targeted reference fixtures, reference-code review, and stage-contract checks at the consumed boundaries: loader, recipe/digest, descriptor/hash, CUDA payload packing, upload/staging, and small kernels where direct structural checks are possible.
   - A production OFX render must never select CPU pixels because CUDA is unavailable, because a route is unsupported, because resource admission failed, because the host asked for draft/preview/interactivity, or because the GPU implementation is incomplete.
   - A host non-CUDA render entry for the current spektrafilm product contract must fail before producing output or mutating render state. Phase 1A hard-blocks product access to `processImpl()` / `multiThreadProcessImages()` for spektrafilm output.
   - If the old CPU renderer source remains temporarily for buildability or staged deletion, the blocking wrapper or unreachable bridge is source-marked `SF_TEMP_BRIDGE_CPUProductRendererBlocked`, has no product OFX call site, and records owner/removal phase. Unmarked `processImpl()`, `multiThreadProcessImages()`, old scanner-runtime CPU rendering, or old `PipelineRunner` code is not allowed to survive as an implicit reference, fallback, or validation hook.
   - Production pixel reads are GPU work, including auto-exposure metering. Host CPU code may derive metering bounds, weights, descriptors, temporal tokens, diagnostics, and validation fixtures, but it must not read production pixels as a hidden render stage or fallback.
   - CPU math helpers are allowed only when they are explicitly validation-only or host-side derivations that feed a recipe/resource descriptor.
   - Host-side LUT/profile derivation helpers are allowed only as descriptor-owned resource preparation. A production scanner/enlarger LUT builder must consume a narrow descriptor such as `ScannerSpectralLutDescriptor` / print-filter descriptor plus parsed payload identity, and must not consume broad `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime slots, mutable `JuicerProcessor` members, live OFX images, or old route modes. Current broad CPU builders such as `build_scan_lut_cpu(Scanner::ScannerMediumRuntime...)` are replacement

targets when touched; retaining one requires an `SF_TEMP_BRIDGE_*` marker that states whether it is validation-only or descriptor-owned host derivation, its allowed call sites, and its removal/narrowing phase.
  - Tiny validation-only C++ helpers are allowed only when they check a narrow consumed contract or stage invariant and cannot publish pixels as product output. They must consume explicit fixture data or narrow recipe/subrecipe inputs, not broad `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime slots, mutable `JuicerProcessor` side channels, or old route modes.
  - Validation-only C++ helpers must be source-obvious and mechanically unreachable from OFX product rendering. Prefer focused validation files or `scripts/spektrafilm_*` helpers. A helper added to or retained in `mainProcessing.*`, `ScannerOptics.cpp`, or another current product-render file must be source-marked as validation-only and its phase evidence must prove no `JuicerEffect::render`, `JuicerProcessor::process`, `processImpl()`, `multiThreadProcessImages()`, OFX image fetch/copy path, or CUDA product launch path can call it. Current product-adjacent test-facing wrappers such as `JuicerProcTest` are part of this audit surface when their host file is touched: they are moved, source-marked narrow validation-only, or explicitly allowlisted with owner/removal status.
  - Intermediate release guard: builds that have accepted Phase 1 spektrafilm identity but have not reached final readiness are internal-only unless the owner explicitly approves a blocked-output release whose current-contract failure diagnostics are the intended product behavior. Under no circumstance may an intermediate build ship old Film-Juicer/agx pixels under spektrafilm profile keys.

3. Resource/profile payload replacement is owner-supplied.
  - The product resource root is `Resources/`. The current `Resources/` tree has been updated with the latest spektrafilm resources.
  - Phase 1A profile metadata is read from `Resources/profiles/*.json` in product runs.
  - Do not spend implementation phases copying, converting, or curating production `Resources/profiles`.
  - Avoid malformed-resource fixture sets unless they prove a product failure boundary that cannot be covered by direct structural checks.
  - Because the replacement bundle is owner-supplied and reference-matched, neither production runtime nor phase acceptance should full-validate the copied spektrafilm bundle. Phase 2 may record the spektrafilm reference source/hash and run a cheap inventory/smoke check when useful, but it must not require a bundle-lint framework or broad copied-file validation.
  - Product runtime performs cheap catalog metadata reads for visibility and parses selected resources only as far as the current recipe/resource consumer needs. A malformed non-selected user profile or resource must not affect unrelated product use. A malformed selected user profile/resource may fail with the smallest useful load/preflight diagnostic; Film-Juicer does not owe broader validation, repair, or troubleshooting.

4. Profile selection is key-based.
  - Use Resolve `StrChoiceParam` values for film and print profile keys.

- Use new parameter identities such as `FilmProfileKey` and `PrintProfileKey`.
- Do not keep integer-choice, hidden-string, or cross-host compatibility alternate paths.
- The local Resolve canary confirmed `kOfxParamTypeStrChoice` works even though Resolve reports host API `1.4`.

5. Required routes are visible from the start.
- `NegativeDirectScan`
- `NegativePrintScan`
- `PositiveDirectScan`
- `PositivePrintScan`
- A route may be temporarily blocked only by a named phase-local preflight diagnostic that names the missing dependency and removal acceptance item.
- Positive print scan must follow spektrafilm's normal print route. It must not be hidden, remapped to negative behavior, or deferred to cleanup.
- `ScanRoute` is the only route identity at the UI/state/catalog/recipe-publication boundary after Phase 1 and in any Phase 1-owned hash input. Old `PrintBypass`, `NegativeOnly`, `Print`, or `ScannerMedium` decisions may exist only as local backend bridge details until the phase that owns the touched behavior removes them.
- CUDA payloads, resource descriptors, and backend route consumption change in the first phase that changes the corresponding behavior. Until then, any old route value that survives behind an existing backend call must be named as an `SF_TEMP_BRIDGE_*` with an owner and removal phase.
- If old `PrintBypass` cannot be deleted in Phase 1, it is a named `SF_TEMP_BRIDGE_PrintRouteUi` UI adapter. Its raw value must not cross the recipe boundary, frame-request boundary, descriptor boundary, CUDA payload packing, or hashes; only the resolved `ScanRoute` may. Phase 1B is not accepted if normal recipe/state code still stores raw `PrintBypass`, `NegativeOnly`, `Print`, or `ScannerMedium` as route identity outside a source-adjacent bridge marker. Direct-route dependency on this adapter must be removed before Phase 3 direct-route pixel behavior is accepted, and print-route dependency must be removed before Phase 4 print-route pixel behavior is accepted. Phase 2 must not be forced to perform render-behavior cleanup it does not own.

6. Film-Juicer grain is not replaced.
- spektrafilm defines the grain schema/statistical contract only.
- Film-Juicer remains the production visual grain renderer.
- Final grain matching is deferred to real grain-plate tuning, not spektrafilm rendered grain.
- Dust, scratches, and gate weave remain Film-Juicer visual/temporal systems and are not compared to spektrafilm rendered output.
- Glare is different: glare is part of the spektrafilm scanner/print behavior and should match spektrafilm semantics for the active route.
- `density_min` remains part of film-density scanner/enlarger bounds even when visual grain is disabled. It must not be applied to print-media scanner bounds.

7. Optional optics default to `Off`, except where spektrafilm behavior is explicitly enabled by the route/control.
   - While an optics effect is off, no optics backend is resolved, no output-affecting optics math runs, and no optics resources are requested.
   - Halation is the only default divergence in this group: Film-Juicer defaults it off, but enabled halation must follow spektrafilm's active halation path. Camera diffusion remains inactive by spektrafilm default, and lens blur is identity at its default value.
   - Exact optics is the spektrafilm-matching backend. It follows the reference optics behavior with PSF/FFT resources where required, subject only to reviewed GPU precision/execution deviations that pass visual inspection.
   - Fast optics is out of scope for this plan. Do not expose or implement a lower-cost optics backend in the spektrafilm port.
   - If Exact is requested and cannot be admitted, the render fails clearly. There is no Fast optics backend or silent lower-cost substitute in this plan.
   - spektrafilm `settings.preview_mode` is not supported in Film-Juicer. Film-Juicer renders the full production path.
   - Host preview/draft/interactivity hints may affect scheduling or diagnostics only. They must not change output or select an optics backend.

8. Resource management preserves context-owned CUDA lifetime invariants while replacing broad prepared-frame/resource-manager surfaces.
   - This resource architecture uses `docs/rm/resource-manager-simplification-plan.md` as background, not as an imported checklist. The normative invariants for this port are: exact `DeviceContextKey {deviceId, contextOpaque}` plus context epoch for durable residency; context-owned durable resources; prepared-frame-owned scratch, staging, scan-error, transient event, and auto-exposure frame work; explicit `finish()` / `abort()` release; no instance-owned durable GPU cache; no public render-path `ensure_*` chain; and no new device-wide scheduler, allocator framework, graph system, or cache layer unless the implementing phase proves immediate correctness or measured performance need.
   - `WorkingState` publishes immutable host recipe state only.
   - `FrameRequest` is the immutable per-render input to preparation.
   - `PreparedCudaFrame` owns frame-local leases, views, staging, scratch, and completion/abort behavior.
   - Durable GPU residency is context-owned and keyed by complete descriptor identity plus `DeviceContextKey` and context epoch.
   - The current public `PreparedCudaFrame::prepare_*`, public render-path `ensure_*`, and `ResourceManager::command_ensure_*` expansion pattern is bridge debt, not the destination architecture. New spektrafilm resources enter through descriptor-driven preparation unless an owner records a named temporary bridge with allowed call sites and a removal gate. Accepted Phase 3/4 product routes must not use `Root::prepare_cuda_frame(..., const WorkingState&)`, `PreparedCudaFrame::prepare_*` methods that accept `WorkingState` / `Print::Runtime` / `Print::Params`, or ResourceManager commands that consume those broad objects as their normal preparation contract.

- Render code must not perform file discovery, ad hoc uploads, raw allocations, public `ensure_*` chains, effect-specific late preparation, schema interpretation, hash construction, old-state translation, or instance-owned durable GPU caching.

9. Clang formatting and static checks are mandatory for touched code.
- Treat `Release-Clang|x64` as the main build/tidy configuration unless told otherwise.
- Run clang-format or the repository's clang-format wrapper on touched C++/CUDA files.
- Run targeted clang-tidy on touched C++/CUDA files with the checked-in broad `.clang-tidy`.
- Do not weaken `.clang-tidy` or clang-format policy to make a phase pass.
- Keep current Clang/compiler math settings unless code-level review, reference comparison, or a targeted performance change gives a concrete reason to change them.

10. Keep the implementation direct, explicit, and production-debuggable.
- The port must produce clear, direct, maintainable C++/CUDA, not clever infrastructure.
- The target is code that is easy to read, easy to debug under production pressure, and hard to misuse.
- Use plain data, explicit ownership, and direct control flow by default.
- Keep helpers local to the owning domain unless multiple current call sites prove a shared helper is simpler.
- Use small closed structs for recipes, descriptors, and payloads.
- Do not use extensible option bags, generic policy objects, descriptor registries, callback registries, manager-manager layers, or framework-like abstractions.
- Maintain one obvious runtime path for spektrafilm behavior.
- Do not add mode stacks, compatibility branches, fallback ladders, silent downgrades, or hidden substitutions.
- Delete old paths when their owning behavior moves. Temporary bridges must be named, local, searchable, and assigned a removal phase.
- Use concrete resource descriptors owned by the resource family. Do not introduce generic resource systems.
- Use typed boundaries where swaps would create real bugs: units, channel order, route identity, profile identity, backend identity, ownership lifetime, and host/device payload boundaries.
- Do not add abstraction for hypothetical future backends, resources, profile schemas, render stages, diagnostics, scheduling, caching, or allocation needs.
- Add a helper or abstraction only when it removes current complexity, prevents a real bug, or creates a concrete ownership/lifecycle boundary needed by the current phase.
- Do not apply SOLID, SRP, dependency inversion, or interface-first design as abstract virtues. Split code only when the split creates clearer current ownership, lifecycle control, failure handling, unit/channel safety, resource identity, or repeated-use simplification.
- A coherent domain flow in one module is preferable to scattered tiny classes, services, factories, interfaces, or adapters that make the render path harder to trace.
- Closed route, backend, profile, and resource behavior should use explicit typed data and direct branching where that is clearer than polymorphic extension points.

- Failure behavior must be explicit. Do not replace required data, unsupported modes, resource admission failures, exactness failures, or invalid host state with quiet defaults or best-effort behavior.
- Simplicity must not weaken lifecycle rules, hash/resource identity, type boundaries, route correctness, CUDA context ownership, prepared-frame isolation, or host-facing failure clarity.
- Output-affecting decisions must be resolved into `RenderRecipe`. `FrameRequest`, `PreparedCudaFrame`, CUDA packers, resource managers, and kernels must not become secondary policy owners.
- Consumer-first rule: do not add a recipe field/group, descriptor, hash lane, diagnostic field, bridge entry, wrapper type, cache, scratch descriptor, or helper API before the current phase has a concrete consumer.
- A phase may not add a placeholder recipe field, descriptor field, hash lane, CUDA payload member, wrapper type, diagnostic enum, or helper API merely because a later phase is expected to need it. The first implemented consumer owns the field, names the consumed source data, defines hash/diagnostic behavior, and adds the acceptance note for that behavior.
- Litmus tests before accepting new code in this port:
  - Can a reviewer trace an output-affecting decision from host params/profile data to `RenderRecipe`, descriptor/hash, CUDA payload, and kernel without finding a second policy owner?
  - Does every new field, descriptor, hash lane, diagnostic, wrapper type, helper, or API have a current-phase consumer?
  - If a helper/type were inlined or moved next to its only caller, would the code become easier to follow? If yes, keep it local or delete it.
  - Does a split prevent a real bug around ownership, lifetime, units, channel order, route identity, profile identity, backend identity, or host/device payload boundaries?
  - Can a production failure name the route, profile/resource identity, backend, descriptor identity, CUDA context/epoch, and exact missing or unsupported state?
  - Does missing required data, unsupported state, failed admission, or exactness failure stop before output instead of substituting another value?
  - When a feature is off, does it produce no descriptor, upload, scratch, kernel work, or output-affecting hash contribution except required recipe identity?
  - Are old paths deleted or hard-blocked when their behavior moves, with any temporary bridge named `SF_TEMP_BRIDGE_*`, searchable, local, and assigned a removal phase?
  - Can `rg` find the policy owner, bridge, diagnostic, descriptor, and hash lane by concrete typed names?
  - Is the only justification "future extension", "clean architecture", "SRP", "OCP", or "a later backend might need this"? If yes, reject it for this port.

11. Performance and architecture quality are first-class constraints.
- This is not a benchmark plan, but each phase must keep hot render paths cheap, traceable, and free of repeated packing, redundant uploads, hidden CPU/GPU synchronization, layout churn, and render-time discovery.

- GPU payload, resource, and scratch changes must expose producer, consumer, update frequency, host layout, device layout, ownership/lifetime, descriptor/hash identity, and upload/preparation point before code lands.
- Memory pressure and overlap behavior must be handled through admission, bounded residency, prepared-frame leases, clear failure diagnostics, or output-stable strategies. Repeated steady-state one-shot fallback, unbounded VRAM growth, or broad policy surfaces are design failures.
- A future performance plan may add numeric targets or benchmarks, but it is not needed to reject slow architecture in this port.

12. ICC profiles stay out of the render port.
- Upstream spektrafilm now uses ICC profiles for saved-image tagging/embedding, not for simulation colour science.
- Film-Juicer's OFX render output remains numeric pixels in the host pipeline. Current output-color UI choices should continue to map to recipe-owned primaries, white point, matrices, and CCTF data.
- Do not add ICC asset loading, ICC CMM dependencies, or ICC-derived GPU payloads as part of this port.

Implementation Discipline Guardrails

The port should produce direct, debuggable C++/CUDA, not a generic rendering framework.

Reject these implementation shapes unless an owner explicitly approves a narrow exception with a removal or follow-up plan:
- `RenderRecipe` passed through broad layers "just in case" instead of passing the narrow owning subrecipe.
- abstract factories, visitor-style stage graphs, generic policy objects, descriptor registries, or manager-manager layers.
- open-ended descriptors using `std::any`, `void*`, stringly typed option maps, unrelated optional fields, or long unrelated `needX` boolean lists.
- member-by-member mirrors of old `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime structs, or CUDA payload structs as the real spektrafilm contract.
- hidden globals, implicit singleton caches, or hash builders that perform lookup/discovery while hashing.
- repo-wide result/error frameworks for leaf helpers when a local boolean, enum, or compact boundary result is clearer.
- public `PreparedCudaFrame::prepare_*`, public render-path `ensure_*`, or `command_ensure_*` expansion for new spektrafilm behavior.
- CPU pixel-render fallbacks for spektrafilm output after a touched path moves to the new contract.
- placeholder recipe fields, descriptor fields, hash lanes, CUDA payload members, wrapper types, diagnostic enums, or helper APIs added only because a later phase might need them.

Required implementation shapes:
- small structs with explicit units/order in their names
- local free functions near the owning domain
- one hash helper beside the descriptor or recipe group it describes
- one packer per CUDA payload, consuming the smallest recipe/subrecipe that fully defines the payload
- deletion or hard blocking of old paths as soon as the owning behavior moves
- the first implemented consumer owns each new field, names the consumed source data, defines hash/diagnostic behavior, and adds the acceptance note for that behavior

Reference-Parity Policy

Primary correctness comes from code-level port fidelity before broad numerical testing:
- `external/spektrafilm` is the local reference source for code review, formula checks, stage ordering, defaults, profile/resource semantics, and reference-output generation.
- Local C++ validation helpers are secondary evidence only. They are useful only for checking narrow consumed contracts or debugging C++/CUDA boundaries, and they never define parity when they differ from `external/spektrafilm`.
- Each phase manifest that performs `ReferenceCodeReview` or `ReferenceNumeric` must record the exact `external/spektrafilm` git ref or dirty-tree status used. A dirty reference tree may be used only when the manifest names the changed reference files and why they are intentional reference inputs.
- Match spektrafilm stage order, formulas, channel ordering, clamp/extrapolation behavior, defaults, profile/resource inputs, and route-specific decisions.
- Preserve existing Film-Juicer GPU numerical-precision and acceleration choices only where code-level spektrafilm review, targeted reference comparison, or owner visual evidence shows they serve the same spektrafilm semantic role and stay below visible drift. Old agx-emulsion or old Film-Juicer parity is historical evidence only; it is not acceptance authority for the spektrafilm port. Do not change GPU execution merely to chase exact numerical parity. Change it when spektrafilm review, targeted comparison, visible drift, determinism/lifecycle risk, or a concrete performance/resource reason shows that the current choice is wrong for spektrafilm.
- Do not add phase-wide numerical contracts, broad tolerance gates, or benchmark-style evidence requirements only for their own sake.
- Add explicit numeric checks only at approximation or acceleration boundaries where code-level comparison is not enough to prove equivalent intent.

Approximation/acceleration boundaries that may need targeted checks:
- `ScannerSpectralLutDescriptor` evaluation versus the equivalent spectral scanner path
- Exact optics GPU execution versus the spektrafilm reference PSF/FFT behavior where the reference uses optics
- glare behavior versus spektrafilm scanner/print glare
- final comparable base-pipeline renders versus spektrafilm reference output
- any newly introduced Film-Juicer approximation that changes spektrafilm math rather than only changing execution strategy

Reference comparisons:
- Do not require end-to-end DeltaE checks before enough implementation exists to produce comparable renders.
- The first visual/reference milestone is the base comparable pipeline without optics, visual grain, dust, scratches, or gate weave.
- Final comparable-render checks use DeltaE 2000 against spektrafilm reference output with matched route, profiles, scanner settings, output transform, LUT resolution, and explicitly recorded disabled/enabled effects.
- GPU approximation targets are visually below JND under matched route, profile, scanner settings, output transform, LUT resolution, and explicitly recorded disabled/enabled effects. DeltaE 2000 summaries are evidence for that target, not broad phase-wide tolerance gates unless a phase note explicitly adds one.
- Numeric comparison evidence must record the fixture set, reference source/ref status, output transform, CCTF, LUT resolution where relevant, disabled/enabled effects, mean/p95/p99/max DeltaE 2000, and whether errors are localized or structured. Averages alone cannot justify a visible local hue, density, channel-order, or edge artifact.
- Old Film-Juicer output is not a target except as local evidence for intentionally retained approximations.

Intentional Film-Juicer product exceptions:

Exception	Scope	Consequence
Current GPU precision and acceleration choices	Retained when they preserve spektrafilm semantics and stay visually below JND	Do not rewrite for exact numerical parity without specific evidence or reason.
Film-Juicer visual grain renderer	Production visual grain	spektrafilm grain is schema/statistical contract only; rendered spektrafilm grain is not a visual target.
Reference fixture/stage-contract workflow	Validation authority and debugging aid	Generate tiny fixtures from `external/spektrafilm` and verify them at consumed boundaries: loader, recipe/digest, descriptor/hash, CUDA payload packing, upload/staging, and narrow kernels. The old OFX C++ CPU pixel renderer is not a reference aid. CPU may still load, perform minimal consumed-field checks, hash, diagnose, derive immutable resources, prepare CUDA resources, and run narrow validation-only helpers that consume explicit fixture data or narrow recipe/subrecipe inputs.
Active Film-Juicer color/CCTF controls	Host product color management	The existing input and output color-space option sets remain available. These are spektrafilm-era host controls, not compatibility behavior; match them in reference comparisons or mark the comparison as extension-mode. Input color/CCTF decode belongs to the film raw recipe; output color/CCTF encode belongs to scanner/output identity.
UV/IR camera filters	Advanced Film-Juicer controls	Defaults are inactive; output-affecting values are typed recipe/hash inputs.

Hanatos 2025 adaptation window/surface/spectral blur	Advanced Film-Juicer controls	Defaults match current spektrafilm settings: window on, surface off, spectral Gaussian blur `0.0`; output-affecting values are typed recipe/hash and LUT-identity inputs.
Runtime defaults	Halation off, DIR active, print-route glare active, Film-Juicer visual grain off, diffusion filters off	Disabled/off features emit no descriptors, uploads, scratch, kernels, or hash contribution except required recipe data such as density bounds.
Print illuminant	Default enlarger/print illuminant `TH-KG3`, with supported current Film-Juicer illuminant choices retained	Resolve through spektrafilm `standard_illuminant`; unsupported values fail before recipe publication. Film reference and scanner/viewing illuminants remain profile-resolved unless explicitly overridden.
Unsupported auto-exposure method handling	Recipe/preflight hardening	Film-Juicer hard-fails unsupported current-contract auto-exposure method values before recipe publication instead of matching spektrafilm's current identity-exposure fallback for unknown strings.

Breaking-Replacement And Failure Policy

This port is a breaking replacement. Treat spektrafilm as the only product contract. Old Film-Juicer profile selection, route identity, CPU render behavior, saved-project state, defaults, and compatibility behavior are outside the product contract.

Rules:
- Retain the existing OFX identifier. This is still a destructive contract replacement: old projects, removed params, old saved state, and old defaults are outside the product contract.
- Do not add code whose purpose is to detect, migrate, explain, emulate, preserve, or diagnose old Film-Juicer state. If an old project opens under the retained identifier, its old saved params are irrelevant; only current spektrafilm descriptor params and current spektrafilm minimal consumed-field checks exist.
- Old `FilmStock`, `PrintPaper`, `PrintBypass`, binary render-mode, and integer/index profile params are deleted from the spektrafilm contract. Any temporary code reference to them is implementation bridge debt only and must not create host-visible compatibility behavior.
- Only spektrafilm-era host params and recipe fields are valid product inputs. A current spektrafilm instance missing current params needed by the selected route, such as `FilmProfileKey`, route, or backend params, is a current-contract descriptor/state error and must fail before recipe publication; that diagnostic is not an old-project compatibility path. `PrintProfileKey` remains current UI/project state, but direct routes keep it as opaque route-inert selection state and do not resolve, load, validate, hash, or prepare print payloads unless the selected `ScanRoute` consumes print behavior.
- Old render modes, old profile/index selection, old print-filter units, old exposure normalization, old DIR behavior, old scanner defaults, and old CPU pixel rendering are deleted, blocked, or named as short-lived implementation bridges. The old OFX CPU renderer is not a special retained reference and must not be reachable as product rendering, a fallback, host-visible compatibility behavior, or validation authority.
- Unsupported current spektrafilm state, missing required spektrafilm params, unsupported route combinations, missing selected GPU resources, selected consumed data that cannot be parsed

safely enough for the active route, product CPU render attempts, and final-render exactness failures hard-fail before output is produced.
- Only defaults present in spektrafilm semantics or explicitly named in this plan may be applied. Locally invented fallback defaults are forbidden.
- Optional calibration defaults, when allowed by this plan, must be identified as optional-calibration defaults rather than compatibility fallbacks, must be included in hashes when output-affecting, and must emit diagnostics/status that name the missing optional calibration.
- Temporary bridges are implementation scaffolding only. They must be named, local, searchable, recorded in the phase manifest, and removed by the owning cleanup gate.

Required Codebase Cut Points

These existing structures are replacement targets, not long-term extension points:

- `Profiles::AgxFilmProfile`: do not adapt it into the spektrafilm schema. Replace it with typed spektrafilm profile payloads and allow `AgxFilmProfile` only inside a named temporary bridge with an owning phase and removal gate.
- `Profiles::GrainMetadata`, `Profiles::HalationMetadata`, `Profiles::ProfileGlare`, and old top-level profile `grain` / `halation` / `glare` objects: do not preserve them as spektrafilm profile schema or rename them into new profile payloads. Runtime `density_min` belongs to `GrainContract` and uses the spektrafilm `GrainParams` default/override, not a profile-authored grain field. Retained Film-Juicer visual grain controls belong to the visual grain recipe/control surface. Halation low-level presets are the profile-digest output in this area, derived from `info.use` and `info.antihalation`; scanner/print glare is a runtime `ScannerOutputRecipe` / `PrintRenderingParams` control with spektrafilm `GlareParams` defaults, not profile data. Any temporary parser bridge for current Film-Juicer assets must be local, searchable, and removed by the phase that accepts the corresponding spektrafilm route.
- `ResourceAssetLibrary` index APIs: replace normal film/print selection by integer index with profile keys and typed catalog entries.
- `Print::Params` Durst-step fields and `kEnlargerSteps`: remove from the spektrafilm path. Retained Film-Juicer UI sliders are Kodak CC offsets after the UI boundary.
- `WorkingStateCorePayload`: treat as `SF_TEMP_BRIDGE_WorkingStateCorePayload` from Phase 1 if it survives temporarily. It must not gain spektrafilm fields, own hashes, interpret schema, or become the real recipe contract. Remove it or replace member-by-member mirroring with narrow recipe/shared-derivation references as soon as touched CUDA behavior no longer needs it.
- CPU `processImpl()` render behavior: hard-block or sever product OFX access in Phase 1A. Product spektrafilm OFX renders must not call it as a backend, fallback, or validation reference. Narrow validation-only C++ helpers may exist only outside OFX product render call sites and must consume explicit fixture data or narrow recipe/subrecipe inputs.
- Hash-time calls into `root().assets()` or catalog lookup: replace with hashes built from already-resolved recipe identities and asset/version tokens.

- Render-path public `ensure_*` expansion and effect-specific `PreparedCudaFrame::prepare_*` growth: replace with descriptor-based prepared-frame preparation plus prepared views/leases. Any retained effect-specific preparation method must be a named temporary bridge with a removal phase.
- `PrintBypass`, `NegativeOnly`/`Print` render-mode switches, and `ScannerMedium`-only route selection: replace as route identity with `ScanRoute`. `ScannerMedium` may remain only as a derived backend medium after the route is known.
- CUDA payload and resource packers that read scattered legacy state: replace with `RenderRecipe` or narrow subrecipe inputs before the first spektrafilm pixel behavior lands. Legacy packers may remain only as named bridges with allowed call sites and removal gates.

Reference Fixture And Stage-Contract Policy

The current CPU render path is not a product backend or validation reference for spektrafilm. Correctness is protected by executable reference fixtures from `external/spektrafilm` plus targeted stage-contract checks that prove fragile semantics at each consumed boundary before they are hidden inside full GPU renders.

Rules:
- Product spektrafilm renders must fail or route to CUDA before CPU source-pixel reads, destination writes, CPU auto-exposure, `processImpl()`, `multiThreadProcessImages()`, or frame/resource preparation can produce product output. Phase 1A must establish this product-entry cutoff.
- The existing `processImpl()` / `multiThreadProcessImages()` CPU path, scanner runtime lease path, and old `PipelineRunner` path are not accepted as validation authority. Do not preserve them behind a macro, hidden route, or reference entry point.
- Each phase that ports fragile behavior names one reference fixture command/artifact or one reference-code review target from `external/spektrafilm`, plus the smallest C++/CUDA boundary check that proves the consumed contract has not changed while crossing loader, recipe, descriptor/hash, payload packing, upload/staging, or kernel boundaries.
- For NaN/null/non-finite semantics, checks must be boundary-local rather than only end-to-end: density-like fields preserve `NaN`/JSON `null` through load, recipe/digest, hash/resource identity, and upload until `density_to_light`; `data.log_sensitivity` applies `10 ** value` before `nan_to_num`; `Inf` fails selected consumed data before recipe publication or resource preparation.
- For channel/order semantics, checks use asymmetric RGB/CMY/XYZ sentinel values that catch swaps before visual comparison.
- Tiny C++ validation helpers are allowed only when they consume explicit fixture data or narrow recipe/subrecipe inputs and return structural/numeric evidence. They must not consume broad `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime slots, mutable `JuicerProcessor` side channels, old route modes, live OFX images, or CUDA product fallback state.

- If a local C++ helper disagrees with `external/spektrafilm`, the phase records the mismatch and follows `external/spektrafilm` unless an owner explicitly approves a Film-Juicer GPU execution deviation.
- Phase 10 proves the old OFX CPU renderer is not reachable from product rendering and that retained validation helpers, if any, are narrow stage-contract helpers with no product render call site. No CPU render backend, fallback, host-visible render route, broad old renderer allowlist, or product-code allowlist for CPU rendering may remain after final readiness.

Current object replacement map:

Current object	spektrafilm destination
`Profiles::AgxFilmProfile`	`LoadedFilmProfile` / `LoadedPrintProfile` or equivalent selected consumed-profile payloads. The old type, `load_agx_*` loaders, and `load_film_stock_into_base()` use of that payload are Phase 2 bridge debt, not a destination schema.
`BaseState`	Process-owned selected profile asset or phase-local bridge; not a long-term schema owner.
`WorkingState`	Immutable publication envelope containing `RenderRecipe`, compact identity, and narrow retained references.
`WorkingStateCorePayload`	Deleted or `SF_TEMP_BRIDGE` validation-only adapter; not the recipe contract.
`Print::Runtime`	Process-owned print derivation or `PrintRecipe`-referenced immutable derivation.
`Print::Params`	UI-boundary input converted once into `PrintRecipe` fields.
`Profiles::GrainMetadata` / `Profiles::HalationMetadata` / `Profiles::ProfileGlare` profile fields	Deleted or temporary profile-loader bridge only. Final grain, defect, and visual-control behavior is owned by typed recipe/control groups; halation profile digest consumes only spektrafilm `info.use` / `info.antihalation`; scanner/print glare and `density_min` are runtime recipe/control values, not old top-level profile metadata structs.
`Scanner::Options` / `Scanner::Settings`	Scanner recipe/post-effect fields or validation-only LUT/debug controls. They must not remain mutable processor side-channel state or scanner LUT identity owners.
`Scanner::ScannerMediumRuntime`	Scanner recipe/resource descriptor output, not route identity.

Long-term `WorkingState` must not retain scanner runtimes, `Print::Runtime`, large spectral/profile arrays, CUDA upload state, or derived resource payloads as flat fields. During migration it may publish `RenderRecipe`, compact build identity, and narrow immutable references only. Any retained broad field is a named temporary bridge or validation-only adapter with a removal phase.

Temporary Prepared-Frame Bridge Removal Table

These are known current bridge surfaces. Agents must not grow them into the new architecture. If one is still needed during a phase, that phase manifest must list the exact allowed call sites, owner, reason, cleanup-gate symbol, and removal phase.

Current bridge/API	Why temporary	Replacement	Removal gate
`JuicerProcessor::setWorkingState`, `setPrintRuntime`, `setPrintParams`	Render glue still passes broad host state and print runtime objects separately.	A single immutable `FrameRequest` built from `RenderRecipe` and compact prepared recipe references.	No direct `Print::Runtime`/`Print::Params` injection after Phase 4; final rejection in Phase 10.
`JuicerProcessor::FrameRequest` fields that store `WorkingState`, `Print::Runtime`, or `Print::Params` as independent identities	The frame boundary can accidentally preserve old schema ownership and hash divergence.	`FrameRequest` holds one stable recipe snapshot/reference plus per-frame route/extent/temporal inputs; long-term behavior consumes recipe/subrecipe references, not mutable processor side-channel members.	Phase 3 removes direct-route side-channel consumption; Phase 4 removes print side-channel consumption; after Phase 4 any remaining broad field is validation-only, pure recipe envelope, or rejected by cleanup.
`JuicerProcess::Root::prepare_cuda_frame(..., const WorkingState&)`	Preparation can keep interpreting monolithic state instead of recipe descriptors.	`prepare_cuda_frame(FrameRequest)` or equivalent descriptor bundle produced from `FrameRequest` and its stable `RenderRecipe` snapshot/reference.	Accepted Phase 3 direct-route product renders must not use the broad `WorkingState` signature as their normal preparation input; Phase 4 removes it from product preparation. Phase 10 rejects any remaining broad signature unless validation-only and unreachable from product render.
`PreparedCudaFrame::prepare_current_medium(const WorkingState&)`	Current medium resources can be keyed from state fields instead of prepared density/resource descriptors.	Medium density/spectral resource descriptor from film develop outputs and `DensityBoundsRecipe`.	Phase 3.
`PreparedCudaFrame::prepare_scan_lut(const WorkingState&)`	Scanner LUT identity can be recomputed from local state instead of the canonical scanner recipe.	`ScannerSpectralLutDescriptor` built from the canonical descriptor row: `DensityBoundsRecipe`, route, medium, polarity, scanned profile channel/base density identity, scan illuminant, observer/CMF, XYZ normalization, LUT resolution, interpolation, stored value domain/format, and schema version. Scanner post-effect identity is separate.	Phase 3D for canonical direct-route execution; Phase 8 verifies reuse and removes remaining route/post-effect static-key bridges.
`PreparedCudaFrame::prepare_print_illuminant_filtered(const WorkingState&, const Print::Runtime&, const Print::Params&)`	Print resource prep can retain old `Print::Runtime`/`Print::Params` semantics and old filter units.	`PrintRecipe` print-filter fields and `PrintIlluminantDescriptor`.	Phase 4.
`PreparedCudaFrame::prepare_optics_scratch`	Effect-specific scratch preparation can bypass descriptor-owned scratch roles.	`FrameScratchDescriptor` plus prepared-frame views/leases.	Phase 6 or touched optics acceptance.

`PreparedCudaFrame::prepare_spatial_dir_scratch`	Effect-specific scratch preparation can keep spatial DIR outside recipe/resource ownership.	Spatial DIR scratch descriptor plus prepared-frame views/leases.	Phase 5.
`PreparedCudaFrame::prepare_*_kernel` gaussian/halation/grain methods	Public effect-specific kernel preparation expands the prepared-frame surface one feature at a time.	Descriptor-driven preparation plus pure prepared views/leases.	Do not add new public methods; retain only as named bridges or pure view/lease access.
`PreparedCudaFrame::validate_density_primitives(const WorkingState&)`	Validation can remain coupled to monolithic state rather than descriptor identity.	Validation-only density descriptor hooks that consume the same recipe values as CUDA.	Phase 3 allowlist or removal.
`PreparedCudaFrame::validate_print_primitives(const WorkingState&, const Print::Runtime&, const Print::Params&)`	Validation can keep print runtime side channels alive.	Validation-only print descriptor hooks that consume `PrintRecipe` prepared data.	Phase 4 allowlist or removal.
Any public render-path `ensure_*` method outside prepared-frame/context-owned internals	Public ensure chains bypass the prepared-frame resource contract.	Private context/resource-manager internals called only from descriptor-based preparation.	Rejected by every phase cleanup gate.
`ResourceManager::command_ensure_*`	Render-time command negotiation can become the implementation shape for new spektrafilm resources.	Descriptor-driven preparation behind `PreparedCudaFrame`.	Do not add new command families for spektrafilm; replacement is descriptor-driven preparation.
`ResourceManager::command_ensure_auto_exposure_buffers` and related auto-exposure scratch preparation	Auto-exposure can become hidden CPU/readback work or generic frame scratch instead of a device-resident metering resource with explicit ordering.	Auto-exposure metering descriptor and prepared-frame-owned device result/scalar consumed by film exposure kernels in the same CUDA submission.	Phase 3 direct-route acceptance proves same-frame auto-exposure stays device-resident, has no required GPU-to-CPU readback/synchronize before film exposure, and is absent when auto exposure is disabled.
`PreparedCudaFrame::launch_base_pipeline_graph` / `ResourceManager::command_launch_base_pipeline_graph` and base graph cache	Graph replay can become a second execution contract whose hash/resource identity diverges from direct launch.	Direct launch remains the correctness path; graph replay may survive only as an optional context-local replay of the same descriptors, payloads, prepared views, and execution keys.	Phase 3/4 pixel acceptance must record whether graph replay is disabled, retained as an optional bridge, or updated to consume the same recipe/descriptors as direct launch. Phase 10 rejects graph replay that owns policy, resource lookup, fallback behavior, or divergent hashes.
`ResourceKind`, `ResourceKindContract`, `kResourceKindContract`	Existing ResourceManager resource classification/onboarding table can become a central registry by accident.	Spektrafilm resource families own plain descriptor structs and local hash builders. Extending this table is allowed only as a named temporary bridge with cleanup ownership and must not add generic policy lanes.	Bridge inventory and Phase 10 resource-ownership review.

Public effect-specific `PreparedCudaFrame::prepare_*` methods that perform resource discovery, upload, allocation, schema interpretation, hash construction, or old-state translation after the frame has nominally been prepared	They turn `PreparedCudaFrame` into a second expanding preparation surface instead of the prepared result.	`Root::prepare_cuda_frame(...)` produces admitted resources; render code consumes prepared views/leases.	Every phase inventory; final rejection unless validation-only and allowlisted.
`WorkingStateCorePayload` member-by-member mirrors used for sharing, upload, or payload packing	It can make `RenderRecipe` an envelope around legacy state instead of the real contract.	`RenderRecipe` or narrow subrecipe/shared derivation references.	Phase 3 for direct behavior, Phase 4 for print behavior, final validation-only only.
Unconditional static-noise/STBN/Wang/static-grain upload during frame preparation	Static noise can become hidden durable residency even when visual grain, dust, scratches, gate weave, and other consumers are disabled.	A descriptor-gated static-noise resource request emitted only when an explicit current consumer is enabled.	Removed before Phase 3 direct-route pixel behavior is accepted; any temporary path is `SF_TEMP_BRIDGE_StaticNoiseUpload` with allowed call sites and resource impact.

No new `WorkingState`, `Print::Runtime`, or `Print::Params` parameter may be added to a CUDA preparation method without also adding a phase-local bridge table row and a removal gate. The preferred direction is narrower descriptor parameters, not more state plumbing.

Temporary Bridge Policy

A temporary bridge is allowed only when it is local, named, searchable, and assigned to a specific removal phase. This applies especially to prepared-frame APIs that still pass legacy objects such as `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime structs, or old resource-manager command shapes.

Bridge naming:
- A temporary bridge is debt, not architecture.
- A bridge may exist only when deleting the old path in the same phase would prevent buildable progress.
- A bridge must be smaller than the behavior it replaces.
- A bridge must not grow new fields after creation.
- A bridge must not become a public contract.
- A bridge must be removed before the first phase that claims ownership of that behavior is accepted.
- Every temporary bridge must include `SF_TEMP_BRIDGE` or `SpektrafilmBridge` in source adjacent to the bridge declaration, wrapper, old-API call site, or cleanup gate symbol. A manifest-only bridge name is not enough for source code a developer can edit.
- For a broad legacy API that cannot be renamed immediately, add a short source-adjacent marker at each still-allowed call-site family or at the wrapper that centralizes those calls. The marker names the bridge and removal phase, for example `SF_TEMP_BRIDGE_PrepareCudaFrameWorkingStateInput remove Phase 4`.

- If source-adjacent marking is impossible because the symbol is generated or external, the phase manifest must record that exception and the cleanup gate must still use a matching searchable bridge name.
- Use code-facing names that describe the actual role, such as `PrintRouteUiAdapter`, `WorkingStateCorePayloadAdapter`, or `PreparedScanLutAdapter`. Do not use migration labels in public type/function names unless the bridge is tiny, local, and deleted by the owning phase.
- Anonymous transitional plumbing is forbidden.
- New direct call sites into old bridge families are forbidden unless the call site is listed in the bridge manifest before use.
- Bridges must not provide user-visible compatibility behavior; they are only implementation scaffolding while the owning phase is being completed.
- A bridge must not be the path that makes product pixel acceptance pass by translating current-contract spektrafilm identity into old Film-Juicer/agx render behavior. Any owner-approved pixel-producing exception is non-production evidence only, must be named as such in the phase manifest, and cannot satisfy a phase that claims rendered spektrafilm output.

Current bridge families that agents must not extend without an explicit manifest entry:
- `WorkingState`-derived render data used as CUDA/resource input.
- `SF_TEMP_BRIDGE_WorkingStateCorePayload`, if retained temporarily, including any member-by-member `WorkingState` mirror used for sharing, upload, or payload packing.
- Prepared-frame methods that accept `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime structs, boolean route flags, or old resource-manager command shapes.
- Public effect-specific `PreparedCudaFrame::prepare_*` methods that perform resource discovery, upload, allocation, schema interpretation, hash construction, or old-state translation after the frame has nominally been prepared.
- Integer/index-based `FilmStock` / `PrintPaper` params and asset lookup.
- `PrintBypass`, `NegativeOnly`, `Print`, or `ScannerMedium`-only render identity.
- Product CPU pixel-render entry points and old CPU stage paths. Narrow validation helpers are allowed only when they consume explicit fixtures or narrow recipe/subrecipe inputs and cannot render OFX product pixels.
- Render-time `ensure_*` / `command_ensure_*` resource preparation outside context-owned internals and the prepared-frame boundary.
- CUDA payload/resource packers that consume scattered local state instead of `RenderRecipe` or named subrecipes.

Every bridge entry must record bridge name, owning phase and owner, source-adjacent marker or documented generated/external exception, allowed call sites, legacy inputs it still accepts, new recipe/result that will replace it, output or resource impact, hash inclusion/exclusion behavior where relevant, cleanup-gate symbol/search term, removal phase, and manifest entry.

Cleanup allowlist entries for bridge symbols must reference the same bridge name recorded in the phase manifest and in source-adjacent bridge markers. A bridge mentioned only in plan prose, a manifest, or a remote allowlist is not accepted for editable source.

Prepared-frame bridge rules:
- New render behavior may be implemented behind existing prepared-frame methods only if those methods are listed as temporary bridges before use.
- The first phase that changes pixel behavior for a route must already pack CUDA payloads and resource descriptors from `RenderRecipe` or a narrow subrecipe. It must not introduce spektrafilm behavior by repopulating legacy `WorkingState`, `Print::Runtime`, `Print::Params`, or scanner runtime structs as the real contract.
- No phase may add a new public effect-specific preparation method or public render-path `ensure_*` / `command_ensure_*` method as the implementation shape for a new spektrafilm resource family. If an owner-approved exception is unavoidable, it is a temporary bridge, must not perform policy selection outside the descriptor boundary, and must have a removal phase.
- Pure view/lease accessors on `PreparedCudaFrame` are not bridge debt when they expose resources that were already declared and admitted by the preparation boundary.
- A bridge must not perform file discovery, schema interpretation, default policy selection, or hash-time asset lookup.
- Phase acceptance must state whether bridge count decreased, stayed flat with reason, or increased with owner approval.
- Phase 10 must not discover untracked bridges; any remaining bridge must be validation-only or explicitly approved with owner, reason, and allowlist.

Final Runtime Shape

ProfileCatalog / ProfileAssets

Process-owned, immutable after load unless a later runtime-reload feature is explicitly added.

Owns:
- profile discovery metadata
- typed film/print keys
- profile path/source identity
- minimal consumed-field parsing/checks
- authored spektrafilm `info.*` and `data.*` payloads
- neutral-filter database as optional calibration data
- dichroic filter resource identities for Film-Juicer's exposed spektrafilm filter sets
- identity/version tokens consumed by recipes and resource keys

Does not own:
- UI state
- render routes
- CUDA handles
- frame scratch
- derived runtime decisions
- recipe construction policy
- hash policy

- render diagnostics policy

First-cut catalog API contract:
- Use these names unless inspection finds an existing collision; if a phase chooses different names, its manifest must record the mapping and cleanup/search terms.
- Preferred home is focused `ProfileCatalog.*` and `ProfileAssets.*` files. If those names collide with existing project files, use `SpektrafilmProfileCatalog.*` / `SpektrafilmProfileAssets.*` and record the mapping. `ResourceAssetLibrary.*` may contain only process-root access glue and temporary bridges while the old index APIs are being cut away.
- Once selected spektrafilm payloads, profile digests, catalog role logic, or consumed-field diagnostics exist, focused profile/catalog files are mandatory. Do not place those owners in `ResourceAssetLibrary.*`, `JuicerState.*`, `Print.*`, or CUDA/resource-manager files except as source-marked bridge glue that forwards to the focused owner. A phase that adds `LoadedFilmProfile`, `LoadedPrintProfile`, `ProfileDigest`, role visibility, selected-resource diagnostics, or profile hash policy to `ResourceAssetLibrary.*` as the real owner is not accepted.
- `ProfileCatalogEntry`: immutable metadata needed for visibility and menu publication, including profile key, label, source path/token, effective support, effective stage, effective polarity/type, defaulted-field status, and cheap source identity. Effective role metadata is the selected profile's `ProfileInfo` after applying spektrafilm dataclass defaults. It does not contain full spectral arrays.
- `ProfileCatalog`: process-owned catalog view over `Resources/profiles/*.json`, with typed accessors for visible film and print entries.
- Required accessor semantics: film entries are exposed by key from effective `support=film && stage=filming`; print entries are exposed by key from effective `stage=printing` regardless of effective `support`; duplicate keys fail inside the same typed role. Missing `support`, `stage`, and `type` do not hide a profile; they resolve through spektrafilm `ProfileInfo` defaults (`support=film`, `stage=filming`, `type=negative`). Unsupported explicit bounded values remain invalid for the affected role.
- Catalog role truth table:

Effective metadata after spektrafilm defaults	Film menu visibility	Print menu visibility	Route/use policy
`support=film`, `stage=filming`, `type=negative`	Yes	No	Capture-film profile for `NegativeDirectScan` and `NegativePrintScan`.
`support=film`, `stage=filming`, `type=positive`	Yes	No	Capture-film profile for `PositiveDirectScan` and `PositivePrintScan`.
`stage=printing`, any supported `support`, any supported `type`	No, unless it also has a separate explicit filming role in a future schema	Yes	Print-medium profile. `type`/polarity does not decide print-menu visibility unless a later phase names a print-specific consumer.
Unsupported `support`, `stage`, or selected-route `type`	No for the affected role	No for the affected role	File is unavailable for that role; do not infer from folders, old `type=paper`, neutral-filter coverage, or profile key/name. Missing bounded role fields are not unsupported; they have already been defaulted above.

A single `info.stock` may appear once per typed catalog role. Duplicate keys inside the film role or inside the print role are catalog errors. If a future schema needs one file to expose multiple roles, that phase must add an explicit multi-role representation rather than reinterpreting the single-role table above. Defaulted role metadata is schema defaulting, not folder/name inference; diagnostics record when `support`, `stage`, or `type` came from spektrafilm defaults.
- Selected payload loading uses key-based functions named `load_film_profile_by_key()` and `load_print_profile_by_key()`. Those functions return typed spektrafilm payloads or compact selected-resource diagnostics; they do not fall back to old index `0`, first discovered entries, folder guesses, neutral-filter coverage, or `AgxFilmProfile`.
- Current legacy APIs `film_stock_for_index`, `print_paper_for_index`, `load_agx_*`, print-folder payload lookup, invalid-index clamp-to-zero, and selected-neutral-DB-to-default-DB fallback are replacement targets. If one survives a phase, it must be a named non-rendering `SF_TEMP_BRIDGE_*` with allowed call sites and removal gate. New normal product code must not call them for profile identity, recipe construction, hashes, resource descriptors, or render behavior.

Rules:
- Profile files are classified before catalog visibility. A profile candidate has profile-shaped JSON data; a missing top-level `info` object is treated like spektrafilm's empty `ProfileInfo` for defaulting, but catalog publication still requires a stable profile key.
- Non-profile resource JSONs found under the selected profile metadata root, including old neutral-filter databases under old `Resources/profiles` layouts, are ignored for profile visibility, ordering, defaults, hashes, and fallback behavior. They may be reported as non-profile resource-placement evidence only.
- `Resources/filters/neutral_print_filters.json` and any legacy `Resources/profiles/enlarger_neutral*.json` files are non-profile resources. They never participate in profile catalog visibility, ordering, defaults, duplicate-key checks, hashes, or fallback behavior.
- Once a JSON file is classified as a profile candidate, apply spektrafilm `ProfileInfo` defaults before catalog role classification. Missing `support`, `stage`, or `type` is therefore valid defaulted metadata; unsupported explicit values make the file invisible or unavailable for the affected role without taking down unrelated valid profiles. Do not infer catalog metadata from old Film-Juicer profile shape, profile key/name, folders, neutral-filter coverage, or old index tables.
- Build film menus from effective `support=film && stage=filming`.
- Build print menus from effective `stage=printing`, regardless of whether schema `support` is `paper` or `film`.
- Positive capture-film profiles are visible as film entries from `ProfileCatalog`.
- Neutral-filter coverage must not affect profile menu visibility.
- Duplicate keys inside the same typed role are catalog errors.
- Catalog/menu enumeration should parse only the metadata needed for stable keys, labels, support/stage/polarity, source identity, and cheap visibility diagnostics.

- Phase 1 metadata bootstrap is intentionally shallow; Phase 2 expands only the consumed selected-profile/resource parsing needed by recipe construction and GPU resource preparation. It does not add copied-bundle validation or user-file linting.
- Lazy selected-profile/resource parsing must still fail before recipe publication or resource preparation when the active route cannot be represented safely. Diagnostics should stay compact: profile/resource key, source path when cheap, consumed field/resource, route, and phase.
- Do not grow `ResourceAssetLibrary` into a manager for recipe building, hashing, diagnostics, render routing, or CUDA preparation. If profile/schema logic needs helpers beyond loading and validation, place small helpers near the owning profile or recipe domain.

RenderRecipe

`RenderRecipe` is the one host-owned render contract. `WorkingState` should hold this recipe, compact hashes/build identity, and immutable asset references needed to define the current render. It should not become a flat parking lot for schema data or CUDA state.

Hard ownership rule:
- If an output-affecting value can be resolved once during recipe build, it belongs in `RenderRecipe` or a named subrecipe, not in `FrameRequest`, `PreparedCudaFrame`, CUDA payload packers, kernels, resource-manager policy code, or live instance state.
- `WorkingState` is only the publication envelope for `RenderRecipe` plus narrow immutable references.
- No new spektrafilm production behavior may be implemented by adding flat legacy-style fields to `WorkingState`. A new output-affecting field must land in the owning recipe/subrecipe first, and any `WorkingState` exposure is only a publication reference or named bridge.
- Any retained reference added to `WorkingState` must be classified in the phase manifest as a recipe input reference, process-owned immutable asset reference, context-owned resource reference, compact identity/cache, or temporary bridge reference with removal phase. Unclassified retained references fail phase acceptance.
- `FrameRequest` is only the stable recipe snapshot/reference plus host frame extent/layout, temporal tokens, clip/source/session identity, derived `pixelSizeUm`, and genuinely per-frame inputs such as auto-exposure metering request tokens. CUDA context key, stream, submission snapshot, context epoch, and durable residency acquisition belong to the frame-preparation call/input, not to the recipe snapshot object.
- `PreparedCudaFrame` is only prepared handles, leases, views, staging/scratch ownership, and finish/abort state.

When phase guidance conflicts with these ownership rules, the ownership rules win: `RenderRecipe` owns resolved render behavior, `FrameRequest` owns frame-local facts, `PreparedCudaFrame` owns prepared CUDA handles/leases/views, and diagnostics/preflight own validation status.

Use a small number of nested or neighboring aggregates. These names describe ownership boundaries, not a mandate to keep all C++ recipe definitions inside one large type:

```text
RenderRecipe
  ProfileRoute
  FilmRawRecipe
  FilmDevelopRecipe / DirCouplersRecipe
  DensityBoundsRecipe
  PrintRecipe
  ScannerOutputRecipe
  SpatialOptics
  GrainContract
```

`ExposureDensity` remains useful plan shorthand for the film-raw plus film-development area while consumers are few, but implementation must split it before it becomes a mixed bucket for input color, auto exposure, film raw conversion, density development, DIR, profile digests, and render-scale policy. `PrintScanner` remains useful plan shorthand for the print/scanner handoff, but implementation should prefer `PrintRecipe` and `ScannerOutputRecipe` once both areas have active consumers. Do not create a durable C++ `PrintScanner` bucket that becomes the new `WorkingState`.

Output-affecting policy owner index:

Policy area	Owner	Non-owner rule
Profile identity, typed metadata, selected route, and asset/version tokens	`ProfileRoute`	Catalog, UI, CUDA packers, descriptors, and kernels consume resolved profile/route identity; they do not infer route or profile policy from old params, strings, folders, or index tables.
Input color/CCTF decode, RGB-to-raw method, auto/manual exposure ordering, film sensitivity, density development inputs, and DIR controls	`ExposureDensity` plan area, implemented as `FilmRawRecipe`, `FilmDevelopRecipe`, and `DirCouplersRecipe` once those consumers land	`FrameRequest`, CUDA packers, resource managers, and kernels do not choose exposure ordering, metering method, film raw defaults, DIR defaults, or profile digest policy.
Direct-film and print-route scanner/enlarger density bounds	`DensityBoundsRecipe`	Scanner LUT builders, print resources, grain controls, and CUDA payloads do not recompute range semantics from profile payloads or visual-grain state.
Print filters, neutral calibration, print exposure normalization, preflash, and print handoff	`PrintRecipe`	Print, descriptors, and CUDA code consume the owning print subrecipe. They do not read `Print::Params`, `Print::Runtime`, old Durst units, or neutral-filter fallbacks as behavior contracts.
Scanner LUT inputs, scanner route correction, scanner post effects, output color, and output CCTF	`ScannerOutputRecipe`	Scanner, descriptors, CUDA code, and LUT builders consume

the owning scanner/output subrecipe. Post-LUT effect identity must not enter scanner LUT identity. |
Optional optics domains, exact/off backend identity, enabled component policy, and exactness failure policy	`SpatialOptics`	Resource admission may report availability and pressure outcomes, but does not decide whether optics are active, exact, downgraded, or substituted.
spektrafilm grain statistical contract, density-layer presence, layer semantics, and `density_min` source	`GrainContract`	Film-Juicer visual grain controls do not own scanner/enlarger density bounds or spektrafilm layer schema semantics.
Frame extent/layout, concrete `pixelSizeUm`, temporal/source/session tokens, and metering request tokens	`FrameRequest`	These are host frame-local facts. CUDA context key, stream, submission snapshot, context epoch, and durable residency acquisition are preparation-call inputs/results. Neither side becomes an alternate route, profile, scanner, print, optics, grain, density-bound, or resource-policy owner.
Durable GPU handles, frame scratch/staging, prepared views/leases, admission status, finish/abort state	`PreparedCudaFrame` / context-owned resource internals	Prepared-frame code consumes descriptors and reports preparation outcomes. It does not perform profile lookup, schema interpretation, output default selection, or compatibility fallback.

This index is an onboarding map, not a second architecture. It is also an acceptance artifact: Phase 1C must add a compact source-adjacent owner map near the `RenderRecipe` C++ type or focused recipe header, and every phase that adds an output-affecting value must keep that map accurate. If a phase adds an output-affecting value, its manifest must name the row it belongs to or explain why a new row/domain is needed.

When the `RenderRecipe` C++ type first lands, keep a compact source-adjacent owner map near the type definition or in the focused recipe header. It should mirror the rows above at a high level so a new developer can find the owner of route identity, film raw policy, film development/DIR policy, density bounds, print policy, scanner/output policy, optics, grain, frame-local facts, and CUDA preparation without reading this full plan. Keep it as orientation comments or a small table, not a generated registry, runtime map, or review checklist.

Use nested structs by default. A nested group becomes a public top-level type only when it has an independent hash, failure policy, validation boundary, resource descriptor, or CUDA/backend consumer.

`ExposureDensity` and `PrintScanner` are umbrella names in this plan, not permission to build broad long-term C++ buckets. Phase 3 should split film raw/exposure inputs from film development/DIR inputs once both are active and Phase 3D should establish canonical direct scanner LUT identity/execution separately from post-LUT effects. Phase 4 should keep print-filter, neutral calibration, print exposure, preflash, and print-medium handoff together as print-owned data. Phase 8 should keep remaining scanner color correction, glare, blur/unsharp, output encoding, and cross-route LUT reuse verification together as scanner/output-owned data. If unrelated consumers are active in code, split before phase acceptance rather than letting an umbrella absorb policy from multiple domains.

Hashes are derived identities over consumed recipe fields and asset/version tokens. They may be cached in the recipe publication envelope or beside the resource descriptor that consumes them, but they are not source-of-truth recipe data and must not contain fields absent from the recipe/assets they identify.

Skeleton groups are allowed only when they block old behavior from leaking forward or define a hard boundary needed by the current phase. A field may be added only when the current phase has a concrete consumer: validation, hash/resource identity, CUDA packing, diagnostics, route behavior, or lifecycle safety. `DensityBoundsRecipe` may be named as final recipe structure, but Phase 1 must not add density-bound fields before direct scanner/enlarger consumers exist.

Recipe contract:
- Recipes are the only host-side render contract consumed by resource keys, CUDA payload packing, hashes, and diagnostics.
- Recipes contain selected authored profile identity, resolved typed metadata, and derived runtime fields. Authored profile fields and derived runtime fields remain separate by type/name and diagnostics.
- CUDA payload packers and resource descriptors consume `RenderRecipe` or named subrecipes. They do not re-derive route, profile defaults, density bounds, scanner ranges, print normalization, optional calibration policy, or resource identity from raw profile/catalog/UI state.
- Consumers must take the narrowest named subrecipe that fully defines their behavior. Passing the full `RenderRecipe` is acceptable only for recipe assembly, publication, top-level validation, or code that genuinely combines multiple recipe groups. A CUDA payload packer, resource descriptor builder, hash helper, or diagnostic emitter that only needs one group must accept that group directly.
- A recipe group must not become a flat replacement for legacy `WorkingState`. If a group starts accumulating fields that are consumed by unrelated phases, split it or move the fields to the owning group before the phase is accepted.
- Hash builders consume the same recipe fields and asset/version tokens that resource descriptors and kernels consume.
- `WorkingState` may temporarily envelope the recipe during the port, but new spektrafilm behavior must not be implemented by refilling flat legacy `WorkingState` fields as the real contract.
- `WorkingState` is not a profile payload container. New spektrafilm spectral/profile/sample arrays live in process-owned immutable assets or shared host derivations referenced by recipe identity.

ProfileRoute

Owns:
- selected `FilmProfileKey`
- selected `PrintProfileKey`
- immutable profile references

- typed profile support/stage/polarity and other bounded profile metadata
- scan route enum
- asset identity/version tokens

Final bounded-domain inventory:
- `ProfileSupport { Film, Paper }`
- `ProfileStage { Filming, Printing }`
- `ProfilePolarity { Negative, Positive }`
- `ProfileUse { Still, Cine }`
- `ProfileAntihalation { Strong, Weak, No }`
- `ProfileChannelModel { Color, Bw }`
- `ScanRoute { NegativeDirectScan, NegativePrintScan, PositiveDirectScan, PositivePrintScan }`
- `DensitometerLabel` as an opaque string wrapper for profile metadata, not an enum rejection surface, unless a future consumer defines bounded behavior
- `IlluminantKey` for supported reference/viewing/enlarger illuminants
- `RgbToRawMethod { Hanatos2025, Mallett2019 }`
- `PrintNormalizationMode` for the print normalization truth table
- `ScannerCorrectionMode` or equivalent typed flags for black/white correction behavior

This list is not permission to create a global types header or to implement every enum in Phase 1/2. `ProfileRoute` owns only profile identity, profile metadata needed for route/menu decisions, asset tokens, and `ScanRoute`. `RgbToRawMethod` belongs with film raw, `PrintNormalizationMode` and `DichroicFilterSet` belong with print, and scanner correction flags belong with scanner/output. Define each bounded domain in the smallest owning header when the first current consumer lands.

`PrintNormalizationMode` may be represented as one four-state enum or as two typed booleans, but the recipe mapping must be explicit:
- `None`: `normalize_print_exposure=false`, `print_exposure_compensation=false`
- `CompensationOnly`: `normalize_print_exposure=false`, `print_exposure_compensation=true`
- `NormalizeOnly`: `normalize_print_exposure=true`, `print_exposure_compensation=false`
- `NormalizeAndCompensate`: `normalize_print_exposure=true`, `print_exposure_compensation=true`

The default is `NormalizeAndCompensate`, matching spektrafilm defaults where both booleans are `true`.

Strings may exist at JSON/UI boundaries and final diagnostic formatting. They should not be normal cross-module identity.

Profile role visibility is stage-first:

- `FilmProfileKey` selects capture profiles from `info.stage=filming`; current spektrafilm capture profiles are `support=film`, and explicit unsupported capture-support values block selection instead of being inferred from key/folder names.
- `PrintProfileKey` selects output/print-medium profiles from `info.stage=printing`, regardless of whether `info.support` is `paper` or `film`. Spektrafilm includes printing-stage film profiles such as Kodak 2383/2393; do not hide them by filtering print choices to `support=paper`.
- The `Negative`/`Positive` prefix in `ScanRoute` comes only from selected capture profile `info.type`. Selected print profile `info.type` does not choose the route; it controls print-medium semantics such as black/white print exposure correction support.

Required behavior:
- `ScanRoute` is resolved once at the recipe boundary from selected capture profile polarity (`film.info.type`) plus the user's direct-vs-print selection after role validation. Stage/support determine catalog role and support diagnostics; selected print profile polarity never changes route identity.
- New-instance/profile-selection route defaults follow spektrafilm GUI behavior after profile digest: positive capture-film profiles default to direct film scan, and negative capture-film profiles default to print scan. Spektrafilm's lower-level `IOParams.scan_film=false` construction default is not the post-selection product default. Users may still choose direct or print explicitly for supported selected profiles, including `PositivePrintScan`.
- Direct scan routes consume no print profile data in pixel math, scanner bounds, output-affecting hashes, CUDA resources, CUDA payloads, or render preflight. The resolved params may retain `PrintProfileKey` as opaque UI/project state so the user can switch to a print route later, but direct-route render/preflight does not resolve or load that print profile and does not fail because the route-inert print selection is unavailable or malformed. Python comparison fixtures should use valid print state because the reference runtime constructs it, but Film-Juicer records this as an intentional output-inert setup divergence, not a product dependency. Print resource preparation and print profile payload hash participation are required only for `NegativePrintScan` and `PositivePrintScan`.
- Neutral-filter database lookup is print-route-only. Phase 2 may record source/hash or inventory evidence for `Resources/filters/neutral_print_filters.json`, but selected-entry parsing, malformed-entry failure, final neutral C/M/Y values, and calibration-status hashes belong to `NegativePrintScan` / `PositivePrintScan` when database neutral filters are enabled. Direct routes do not query neutral calibration and cannot fail because a selected neutral entry is missing or malformed.
- Downstream render code, scanner selection, hashes, resource descriptors, and CUDA payload packing consume `ScanRoute` or route-derived typed fields, never raw `PrintBypass`/integer render-mode state.
- Backend-only medium tags such as negative/print are derived from `ScanRoute` after route validation. They are not sufficient route identity.
- A temporarily retained direct-vs-print UI parameter must be listed as a local shim and removed by its phase gate; its raw value is excluded from hashes because the resolved `ScanRoute` is the hashed output-affecting field.

- Raw `PrintBypass`, binary `NegativeOnly`/`Print`, and `ScannerMedium`-only route values are rejected past the UI adapter after Phase 1B. If an untouched backend still needs one of those values, the adapter must be source-marked as `SF_TEMP_BRIDGE_PrintRouteUi` and the accepted route behavior must consume only the `ScanRoute` result.
- Route support checks, missing-dependency failures, and preflight status are recipe-build/preflight diagnostics, not recipe fields.

ExposureDensity

`ExposureDensity` is the film exposure/development plan area. It is not a request for one durable C++ struct that owns every film-side policy. As soon as active consumers make the split clearer, implement this area as focused recipe groups such as `FilmRawRecipe`, `FilmDevelopRecipe`, and `DirCouplersRecipe`, with packers and descriptors consuming the narrow group they need.

Owns:
- render-scale policy: film-format long edge and physical-unit conversion policy
- Film-Juicer active input color space and input CCTF decode settings where output-affecting
- RGB-to-raw method: `RgbToRawMethod { Hanatos2025, Mallett2019 }`
- advanced film-raw controls: UV/IR camera filter triplets, Hanatos 2025 adaptation window, Hanatos 2025 adaptation surface, and Hanatos 2025 spectral Gaussian blur
- spektrafilm auto-exposure enabled state
- spektrafilm auto-exposure ordering and method selection
- manual camera exposure compensation
- `HighlightBoostRecipe` with `boost_ev=0.0`, `boost_range=0.3`, and `protect_ev=4.0`, owned by film exposure rather than optics
- final film sensitivity set and sensitivity hash
- density model and density-to-light identity
- DIR controls and DIR runtime identity
- digested profile-derived runtime defaults

Required behavior:
- Profile-derived runtime defaults are resolved once while building the recipe, after loader defaults and before hashing/resource descriptors/payload packing.
- The digest step is equivalent in intent to spektrafilm `digest_params()` / `_apply_film_specifics()`: it consumes profile polarity, stock key, `info.use`, and `info.antihalation` and produces derived runtime defaults without mutating authored profile data.
- C++ `ProfileDigest` should be a pure local derivation function plus a plain result struct, not a service, registry, cache, policy object, or second recipe owner. It matches spektrafilm parity by producing the same post-digest runtime values that Python's `digest_params()` mutates into its params object; it does not need to copy Python's in-place mutation shape.
- Required digest outputs include spektrafilm negative/positive DIR gamma defaults, stock-specific DIR overrides for `fujifilm_velvia_100` and `fujifilm_provia_100f`, and low-level halation presets keyed by `(use, antihalation)`.

- DIR post-digest constants are output-affecting recipe data and override the pre-digest `DirCouplersParams` schema defaults:

Capture film condition	`gamma_samelayer_rgb`	`gamma_interlayer_r_to_gb`	`gamma_interlayer_g_to_rb`	`gamma_interlayer_b_to_rg`
`film.info.type == positive`	`(0.12, 0.08, 0.06)`	`(0.12, 0.06)`	`(0.08, 0.06)`	`(0.06, 0.06)`
`film.info.type == negative`	`(0.336, 0.319, 0.273)`	`(0.353, 0.302)`	`(0.154, 0.353)`	`(0.168, 0.226)`
`film.info.stock == fujifilm_velvia_100`	`(0.108, 0.072, 0.054)`	`(0.108, 0.054)`	`(0.072, 0.054)`	`(0.054, 0.054)`
`film.info.stock == fujifilm_provia_100f`	`(0.156, 0.104, 0.078)`	`(0.156, 0.078)`	`(0.104, 0.078)`	`(0.078, 0.078)`

Stock-specific overrides run after polarity defaults. They are spektrafilm runtime defaults, not authored profile fields, and they must not be confused with the initial dataclass values in `params_schema.py`.

- Halation post-digest preset constants apply only when the selected capture profile is film-like under spektrafilm's `params.film.is_film` check. They seed low-level halation fields even while Film-Juicer's user-facing halation toggle is off, so the enabled path is deterministic:

`info.use`	`info.antihalation`	`halation_first_sigma_um`	`halation_strength`
`still`	`strong`	`(65.0, 65.0, 65.0)`	`(0.015, 0.005, 0.0)`
`still`	`weak`	`(65.0, 65.0, 65.0)`	`(0.08, 0.02, 0.0)`
`still`	`no`	`(65.0, 65.0, 65.0)`	`(0.30, 0.10, 0.015)`
`cine`	`strong`	`(50.0, 50.0, 50.0)`	`(0.015, 0.005, 0.0)`
`cine`	`weak`	`(50.0, 50.0, 50.0)`	`(0.08, 0.02, 0.0)`
`cine`	`no`	`(50.0, 50.0, 50.0)`	`(0.30, 0.10, 0.015)`

If a `(use, antihalation)` pair has no spektrafilm preset, keep the schema/default halation low-level values and record that no digest preset applied. Do not synthesize a fallback from folder names, film stock names, or old Film-Juicer metadata.

- Derived digest fields are hashed as recipe data when they affect output or descriptor/resource identity. Diagnostics distinguish authored profile fields from defaulted/derived runtime fields. Disabled halation keeps its derived preset seeds available for the enabled path but emits no optics descriptor/hash contribution while off.
- Active Film-Juicer input color space and input CCTF decode controls are intentional spektrafilm-era host-product controls, not old compatibility behavior. Reference comparisons must either match those settings or mark the comparison as extension-mode. Output-affecting input color/CCTF choices participate in film raw recipe/hash identity. Output color space and output CCTF encode belong to `ScannerOutputRecipe`, not to film exposure or film raw

conversion. Internal spektrafilm print-balance midgray simulation remains fixed to the reference defaults.
- Auto EV meters input RGB using the active Film-Juicer input color/CCTF settings.
- Auto-exposure follows spektrafilm preprocessing order and preview semantics: meter before `io.crop` and `io.upscale_factor`; because Film-Juicer does not implement those transforms, parity fixtures must use `io.crop=false` and `io.upscale_factor=1.0`. The metered image is spektrafilm `small_preview`: if the long edge is greater than 256, downsample by nearest-neighbor/order-0 resampling so the long edge is 256; otherwise use the input image as-is.
- Auto-exposure enabled state is a typed recipe field and defaults to `true`, matching spektrafilm `camera.auto_exposure`. When disabled, the measured auto scale is identity and manual exposure still follows the ordering below.
- Auto-exposure method is a typed recipe field with the current spektrafilm methods: `center_weighted`, `average`, `median`, `partial`, `matrix`, `multi_zone`, and `highlight_weighted`. Default is `center_weighted`. Each method follows spektrafilm's luminance, region/mask, and EV formula on the spektrafilm preview image, including the `Inf` auto-EV clamp back to `0 EV`. Unsupported current-contract method values fail recipe build/preflight rather than falling back to identity exposure; this is an explicit Film-Juicer hardening deviation from spektrafilm's unknown-string fallback.
- Auto scale multiplies source RGB samples before any input CCTF decode, RGB-to-XYZ conversion, Hanatos/Mallett conversion, or film raw sensitivity integration.
- Manual EV applies after RGB-to-film-raw conversion.
- Do not keep one ambiguous `2^(autoEV + manualEV)` render scale.
- Production auto-exposure measurement and same-frame scale finalization stay device-resident. The GPU implementation must meter the spektrafilm preview image, not a full-resolution or render-window substitute unless a later approved divergence explicitly changes output and fixture expectations. A metering pass may write a prepared-frame device scalar/result consumed by the film exposure/raw kernels in the same CUDA submission, but it must not require a GPU-to-CPU readback, `cudaStreamSynchronize`, host EV calculation, or host scalar upload before film exposure can launch. Host-visible EV diagnostics or temporal writeback, when present, are optional post-frame results guarded by `FrameRequest` tokens and never part of the same-frame correctness path.
- Auto-exposure scratch, staging, and launch descriptors are sized from the spektrafilm preview image. Phase 3 evidence must prove that the metering descriptor/request has long edge `<=256` after nearest-neighbor preview downsampling when the source long edge is larger than 256, and that no full-frame weights, partials, histograms, staging buffers, or same-frame host scalars are allocated for auto-exposure unless a later approved divergence explicitly changes output and fixture expectations.
- Auto-exposure payloads keep auto scale, manual EV, and validity state explicit enough to preserve spektrafilm ordering. CUDA packers must not collapse auto and manual exposure into one CPU scalar or make the resource manager decide metering method, bounds, or exposure ordering.
- Highlight boost is its own film-linear exposure component. It runs after manual exposure compensation and before camera diffusion, camera lens blur, and the halation block.

`halation.active` gates spektrafilm's in-emulsion scatter plus back-reflection halation block; it does not gate highlight boost.
- `HighlightBoostRecipe` is not a `SpatialOptics` / Exact optics component, scratch request, PSF resource, FFT resource, or optics backend selector. Phase 6 may verify that Exact optics preserves the stage boundary, but the owner, hash identity, CUDA payload packing, and default-disabled identity behavior land with film exposure before any route accepts pixels that expose the control.
- With the default `boost_ev=0.0`, highlight boost is output identity and contributes no spatial resource requirement.
- Final sensitivity order is `10^profile.log_sensitivity` -> `nan_to_num` -> optional camera UV/IR band-pass with per-channel reference-illuminant normalization. The UV/IR filter triplets are advanced user controls with spektrafilm defaults `(0, 410, 8)` and `(0, 675, 15)`, so the default path is inactive.
- Hanatos 2025 adaptation is not a profile sensitivity band-pass field. For `RgbToRawMethod::Hanatos2025`, the filming TC LUT descriptor and hash are built from final sensitivity, profile-authored adaptation window params, profile-authored adaptation surface params, profile reference illuminant, `apply_hanatos2025_adaptation_window`, `apply_hanatos2025_adaptation_surface`, and `spectral_gaussian_blur`. Window adaptation preserves white balance by per-channel reference-illuminant normalization; surface adaptation multiplies the raw LUT by `2^surface`.
- Hanatos 2025 filming TC LUT lookup follows spektrafilm `rgb_to_raw_hanatos2025`: convert RGB to TC plus brightness through `_rgb_to_tc_b`, sample the TC LUT with `apply_lut_cubic_2d` reflected-boundary Mitchell cubic interpolation, then multiply raw layer exposures by the brightness scale `b`. The production GPU sampler must not use bilinear TC interpolation or skip the brightness rescale unless an approved fixture proves exact output equivalence.
- spektrafilm currently applies `spectral_gaussian_blur` by passing the control value directly as the wavelength-axis sigma argument to `scipy.ndimage.gaussian_filter` on the Hanatos spectra LUT before sensitivity contraction. Film-Juicer matches that implementation for parity; if a later owner changes the unit interpretation to physical nanometers, it must be recorded as an intentional output-changing divergence with fixture expectations.
- When `RgbToRawMethod::Hanatos2025` and `apply_hanatos2025_adaptation_window=true`, the selected profile must provide a finite four-value `data.hanatos2025_adaptation_window_params` vector. Missing, empty, malformed, or non-finite required window params fail before recipe publication as `MissingRequiredResource` or `MalformedRequiredProfileData`.
- When `apply_hanatos2025_adaptation_surface=true`, the selected profile must provide a finite `[3][15]` `data.hanatos2025_adaptation_surface_params` matrix. Missing, empty, malformed, or non-finite required surface params fail before recipe publication as `MissingRequiredResource` or `MalformedRequiredProfileData`.
- Hanatos 2025 adaptation window, adaptation surface, and spectral Gaussian blur are advanced Film-Juicer controls beside UV/IR. They are typed recipe fields, included in hashes and LUT resource identity when they affect film raw, and default to current spektrafilm settings.

Any non-reference behavior from these controls is an explicit Film-Juicer extension, not an old compatibility branch.
- Only the selected `RgbToRawMethod` creates durable-resource descriptors, upload/admission requests, hash lanes, or payload bindings for method-specific resources. `Hanatos2025` may request the filming TC LUT and Hanatos support tables; `Mallett2019` may request the Mallett basis and method-local normalization data. Global availability of the unselected method's source tables is not enough to upload or retain those resources, and a method switch invalidates only the resource family actually consumed by the new method.
- Hanatos/default film raw does not use old shared green-midgray normalization.
- Mallett remains selectable and uses spektrafilm's method-local green-midgray normalization: convert active input RGB to linear sRGB with the selected input color space and CCTF decode flag, clip the linear sRGB triplet to nonnegative values, contract the Mallett 2019 basis multiplied by the profile reference illuminant with final sensitivity, apply `nan_to_num` to raw responses, and divide all raw channels by the green-channel midgray response from `standard_illuminant(reference_illuminant) * 0.184`. Mallett does not use the Hanatos TC brightness path, old DWG/S-inversion reconstruction, or a shared Film-Juicer midgray scale.
- Full film develop uses normalized film density curves computed as `authoredDensityCurves - nanmin(authoredDensityCurves, axis=0)` before DIR/grain.
- Print develop and print-balance midgray develop use authored density curves unchanged.
- Density interpolation is per-channel linear interpolation over `log_exposure / gamma[channel]`; values outside the authored exposure range clamp to endpoint densities.
- Raw/density triplet indices are preserved through spektrafilm film development: index 0 is R-sensitive exposure that develops C density, index 1 is G-sensitive exposure that develops M density, and index 2 is B-sensitive exposure that develops Y density. Code may label the exposure side RGB and the dye side CMY, but loaders, recipes, CUDA payloads, LUTs, and validation sentinels must not reorder this triplet except at an explicitly named boundary.
- Film-Juicer does not implement spektrafilm `io.crop` or `io.upscale_factor`. These fields are absent from `RenderRecipe` and from the Film-Juicer UI contract.
- `pixelSizeUm = filmFormatLongEdgeMm * 1000 / max(fullFrameWidth, fullFrameHeight)` is derived once for a frame from film recipe render-scale policy plus `FrameRequest` concrete host-provided full-frame extent.
- Host/Resolve scaling and cropping are outside the spektrafilm simulation contract. Imported reference fixtures used for Film-Juicer parity must have `io.crop=false` and `io.upscale_factor=1.0`; nonconforming fixtures fail with `UnsupportedSpektrafilmIoTransform`.
- Recipe hashes include the film-format policy and shape-independent output decisions. Extent-dependent descriptor hashes include the concrete frame extent and derived `pixelSizeUm` only where the consuming resource, scratch family, or launch-visible execution descriptor depends on them.

DensityBoundsRecipe

Owns final route-specific density bounds consumed by scanner/enlarger LUTs, CUDA payloads, hashes, and diagnostics.

Inputs:
- authored profile density-curve extrema
- selected route and medium identity
- profile polarity
- `GrainContract::density_min`
- positive/negative layered-density semantics when active

Owns:
- `dataMinCmy`
- `dataMaxCmy`
- `invSpanCmy`
- bounds source identity
- density-bounds hash
- final validated density bounds consumed by print/scanner/range code

Rules:
- `GrainContract` owns the spektrafilm `density_min` contract but never owns final scanner/LUT ranges.
- `FilmRawRecipe` and `FilmDevelopRecipe` own film exposure/development data but do not publish independent scanner bounds.
- `PrintRecipe`, `ScannerOutputRecipe`, and scanner LUT resources consume `DensityBoundsRecipe`; they do not recompute schema semantics or choose local default bounds.
- Route support checks, missing-bound failures, and validation status belong to recipe-build/preflight diagnostics, not to `DensityBoundsRecipe`.
- Route/source truth table:
  - Direct film scanner bounds for `NegativeDirectScan` and `PositiveDirectScan`: `dataMinCmy = -GrainContract::density_min`, `dataMaxCmy = nanmax(captureFilm.data.density_curves, axis=0)`.
  - Enlarger film-density bounds for `NegativePrintScan` and `PositivePrintScan`: `dataMinCmy = -GrainContract::density_min`, `dataMaxCmy = nanmax(captureFilm.data.density_curves, axis=0)`.
  - Print-media scanner bounds for `NegativePrintScan` and `PositivePrintScan`: `dataMinCmy = nanmin(printProfile.data.density_curves, axis=0)`, `dataMaxCmy = nanmax(printProfile.data.density_curves, axis=0)`. These bounds do not consume `GrainContract::density_min`.
- For every bounds consumer, `dataMinCmy` and `dataMaxCmy` are lower and upper endpoints. `invSpanCmy = 1 / (dataMaxCmy - dataMinCmy)` after validating a positive finite span, and normalized density is `(D - dataMinCmy) * invSpanCmy`. Do not store a legacy positive `density_min` offset, use `dataMaxCmy` as the span, or recompute an alternate range in scanner/enlarger code.
- Missing or malformed data required to compute selected-route bounds fails before resource preparation with profile key, route, field path, and bounds identity.

- Disabled visual grain does not disable the density-min contribution to direct film scanner bounds or enlarger film-density bounds.
- Any default bounds must be explicitly named by this plan, state whether output changes, and must not be used to paper over unusable selected consumed profile data.

PrintScanner

`PrintScanner` names the combined plan area where print-route inputs meet scanner/output inputs. It is prose-only guidance, not permission to add a durable C++ type, public recipe group, cache key, descriptor, or resource family with that name. If a tiny local bridge is needed while only one active consumer exists, source-mark it `SF_TEMP_BRIDGE_PrintScannerLocal`, keep it private to that consumer, and give it an owner/removal phase. Default implementation ownership is split: `PrintRecipe` for print-route policy and `ScannerOutputRecipe` for scanner/output policy. When Phase 4 touches print-route fields, prefer small nested print structs such as `PrintFilterRecipe` and `PrintExposureRecipe` rather than one flat print/scanner bucket. When Phase 8 touches scanner post effects, scanner/output fields live in their own subrecipe.

Phase acceptance must leave no durable resource family, cache, descriptor, public C++ type, public header surface, or long-lived recipe group named `PrintScanner`. A manifest may retain only a private `SF_TEMP_BRIDGE_PrintScannerLocal` bridge with one active consumer, no resource ownership, no hash ownership, and a removal gate before the next phase that activates the other side of the print/scanner boundary.

Owns:
- print illuminant
- Film-Juicer enlarger dichroic selection: spektrafilm reference/custom default, or measured extension sets `durst_digital_light`, `thorlabs`, and `edmund_optics`
- Kodak CC C/M/Y neutral/filter values
- Film-Juicer Y/M/C user slider offsets in Kodak CC units
- neutral-filter optional calibration status
- print exposure normalization
- preflash
- print route identity
- color reference and black/white correction
- scanner LUT recipe inputs consumed by the `ScannerSpectralLutDescriptor` builder
- scanner glare/blur/unsharp identity
- output color space and output CCTF encode behavior

Required behavior:
- `neutral_print_filters.json` is optional calibration data, not profile schema and not a catalog.
- The reference source for neutral calibration is `external/spektrafilm/src/spektrafilm/data/filters/neutral_print_filters.json`; the production resource location is `Resources/filters/neutral_print_filters.json`.

- The bundled neutral-filter data must be traceable to the current `external/spektrafilm` reference used by the phase. Phase evidence records the reference ref and JSON source/hash; production runtime does not need a permanent provenance framework beyond asset identity and optional-calibration status used by the recipe/hash.
- Inspection note: the local reference and `Resources/` both provide the single spektrafilm-shaped neutral database; `Resources/` also contains old per-set neutral files under `Resources/profiles`. If the production neutral database values differ from the inspected reference, the owning phase must record the chosen source/hash before accepting print calibration.
- spektrafilm reference neutral calibration lookup is keyed by print profile, print illuminant, and capture-film profile. It has no dichroic-filter-set dimension.
- Film-Juicer does not implement per-dichroic-set neutral calibration in this port. Current `Resources/profiles/enlarger_neutral_ymc_filters*.json` files are old Film-Juicer resource shape and must not be a destination schema, fallback source, catalog input, or menu filter. A later owner-approved measured calibration feature may add a spektrafilm-keyed per-set resource, but it is outside this plan.
- Missing neutral-calibration data follows spektrafilm's runtime behavior: a missing file or missing selected entry preserves the current neutral C/M/Y values. For a fresh/default Film-Juicer recipe those current values are the spektrafilm schema defaults `C=0`, `M=65`, `Y=55`; if future UI or project state authors different neutral values, a DB miss must not reset them. For print routes, missing-calibration status is recorded and the resulting neutral values are hashed because output changes relative to a calibrated entry.
- Neutral database parsing/lookup runs only while building a print route that consumes database neutral filters. Malformed found entries for the selected `[print stock][enlarger illuminant][film stock]` identity are print-route failures; direct film scans do not parse or look up selected neutral entries.
- Malformed required print profile data fails.
- Print filter units are Kodak CC units, not old Durst/170-step normalized units.
- Film-Juicer may reuse existing user-facing Y/M/C sliders and enlarger dichroic choices only after rewiring/renaming them to the spektrafilm print-filter model. These are Film-Juicer controls over spektrafilm resources/models, not compatibility behavior.
- Reused sliders and choices must use spektrafilm CC math and the selected spektrafilm dichroic model: analytic reference/custom filters for the default, or `Resources/filters/dichroics/*` CSV resources for measured extension sets. They must not route to old Durst/170-step wheel math, normalized wheel values, old per-set neutral DB semantics, or a compatibility branch.
- UI Y/M/C order is converted once to internal C/M/Y CC order at the recipe boundary. Downstream recipe, resource, hash, and CUDA payload code uses C/M/Y CC values.
- `userC` is `FilmJuicerMainCFilterShift`, an output-affecting Film-Juicer extension to spektrafilm. Its default is zero, so default Film-Juicer main-filter C behavior matches spektrafilm's neutral-only C path.
- `userC`, `userM`, and `userY` are all included in `PrintIlluminantDescriptor` and print-route hashes because each changes output when nonzero.
- Neutral calibration remains separate from manual shifts. Neutral DB values populate `neutralC`, `neutralM`, and `neutralY`; manual sliders never rewrite calibration payloads.

- The spektrafilm reference/default dichroic model is `custom_dichroic_filters`, the model used by `color_enlarger` when no filter set is passed. Film-Juicer exposes it as a typed reference/default choice and makes it the new-instance default. `durst_digital_light`, `thorlabs`, and `edmund_optics` are selectable Film-Juicer product extensions; selecting them changes output and must participate in recipe/hash/fixture identity. Missing or malformed resources/model inputs for the selected set fail; do not silently fall back to the reference/custom set.
- Positive print scan enters the same print route as negative print scan, with typed polarity and route-specific correction.
- `ScannerOutputRecipe` owns scanner/output recipe values, but the scanner LUT resource family owns the `ScannerSpectralLutDescriptor` struct, canonicalization, hash builder, and prepared view shape. Correction, glare, blur, unsharp, output color, output CCTF, and other post-LUT identities are route-specific scanner post-effect descriptor fields and are excluded from `ScannerSpectralLutDescriptor`.
- Output color space and output CCTF encode are scanner/output subrecipe fields. They are not film raw or exposure-density fields, and they must not rebuild `ScannerSpectralLutDescriptor` because this port's scanner LUT stores only density-to-log-XYZ generation data.
- Scanner output transform must be equivalent to spektrafilm's `colour.XYZ_to_RGB` call: compute scanner illuminant XYZ from the scanner/viewing illuminant with scanner Y-normalization, derive scanner illuminant xy, convert XYZ to the selected output colourspace with scanner illuminant xy as the source illuminant and output CCTF disabled, then apply scanner RGB blur/unsharp, optional output CCTF encoding, and final clip. Output color/CCTF choices remain post-LUT scanner/output identity and do not enter `ScannerSpectralLutDescriptor`.
- Scanner/output fixture evidence must prove both sides of that boundary: changing output color space or output CCTF changes scanner/output results without rebuilding the density-to-log-XYZ LUT descriptor, and the linear RGB transform matches the spektrafilm `colour.XYZ_to_RGB(..., apply_cctf_encoding=False, illuminant=scannerIlluminantXY)` call for at least one non-D65 scanner/viewing illuminant.
- Illuminants use spektrafilm `standard_illuminant` semantics: align to the 380-780 nm, 5 nm axis; normalize by mean spectral value; build `TH-KG3` as blackbody 3400 K through KG3; build `TH-KG3-L` as `TH-KG3` through lens transmission.
- `Resources/illuminants` remains the standard-illuminant resource family for supported D/T/K75P/reference/viewing/enlarger keys, but it is not profile/catalog data and must not drive profile visibility, defaults, or fallback behavior. `Resources/illuminants/D65.py` is non-runtime unless a phase explicitly owns turning it into a production input.
- `Resources/cie1931_2deg.csv` remains relevant for CIE 1931 2-degree CMFs unless a later phase explicitly replaces it with a compiled/generated table from the same observer data. Scanner XYZ normalization depends on the Y channel and must stay aligned to the canonical 380-780 nm, 5 nm axis.
- Scanner XYZ integration applies its own Y-normalization by `sum(scanIlluminant * CIE1931Y)`; do not reuse mean spectral normalization for scanner XYZ scaling.

Color reference and black/white correction:

- The correction controls default to `black_correction=false`, `white_correction=false`, `black_level=0.01`, and `white_level=0.98`.
- `black_level` and `white_level` are sRGB-encoded scalar controls. Recipe construction stores the authored encoded values and the derived linear values. Correction math consumes the derived linear values after applying spektrafilm's sRGB CCTF decode rule.
- The correction policy is owned by `ScannerOutputRecipe` / color reference recipe data, but stage-specific correction factors are consumed at the spektrafilm stages where the reference applies them.
- Positive direct film scans apply filming exposure correction after film optics and before `log10` film-density conversion, then scanner XYZ correction before glare. Negative direct film scans do not apply black/white color-reference correction.
- Print routes apply print exposure correction after print raw exposure scaling and before enlarger diffusion/`log10`, then scanner XYZ correction before print glare.
- Positive direct film-scan black/white references come from scanner Y of `cmy_black = nanmax(film.data.density_curves)` and `cmy_white = zeros`. Print-route black/white references for negative print profiles come from the printing stage's reference raw exposures: film black is `zeros - density_min`, film white is `nanmax(film.data.density_curves)`, both developed through the print density curves before scanner Y measurement.
- If only black correction or only white correction is enabled, the disabled side uses the measured reference side (`_y_black` or `_y_white`) as its level. Do not force both sides to encoded user defaults when one correction toggle is off.
- Positive direct filming exposure correction uses spektrafilm's midgray density correction: target density is `-log10(0.184)`, the authored film density-curve average (not the normalized film-develop copy) is corrected by film `base_density` average, interpolation is performed in negated density space, and the applied scale is the inverse exposure correction. Negative film routes return identity.
- Print exposure correction applies only when black/white corrections are enabled and the selected print profile type is negative. It uses the authored print density-curve average and print `base_density` average, interpolates in print-density space against `-log10(0.184)`, and returns the exposure correction scale. If a selected `stage=printing` profile has `info.type` other than `negative` and black/white print exposure correction would be consumed, fail before recipe publication until explicit reference-compatible support is defined; do not silently return identity.
- Scanner XYZ correction runs before glare, scales XYZ by corrected Y over measured Y with the spektrafilm epsilon behavior, and clips the correction target into `[0, 1]`. Direct negative film scans remain identity for this correction.
- Print glare follows spektrafilm `add_glare`: when print glare is active and `glare.percent > 0`, generate the lognormal glare field from mean `percent` and standard deviation `roughness * percent`, blur it with the configured Gaussian sigma, divide by 100, and add `glare_amount * illuminant_xyz` in XYZ before output RGB conversion. Direct film scan glare remains disabled by passing no glare settings.
- Film-Juicer production glare must be reproducible for the same frame, recipe, source identity, and render window. A deterministic GPU seed is an execution policy, not a color-science toggle, and it must not change the spektrafilm distribution, blur placement, illuminant scaling, or direct-vs-print route policy. Pixel-exact fixtures with glare active must seed or bridge the

spektrafilm reference side explicitly; otherwise base color-science fixtures disable glare and glare-enabled fixtures use statistical or owner-approved visual tolerances.
- Scanner lens blur is RGB-space Gaussian sigma in pixels. Scanner unsharp mask is `(sigma_px, amount)` and runs after scanner blur. Output CCTF encoding and final clip run after scanner blur and unsharp.
- A phase that exposes or accepts these controls must validate the route-specific skip/apply behavior instead of implementing a single scanner-only correction.

Normative print math:
- Main enlarger filters use internal `C/M/Y` Kodak CC values:
  - `C = neutralC + userC`
  - `M = neutralM + userM`
  - `Y = neutralY + userY`
- `userC`, `userM`, and `userY` default to zero. With `userC=0`, the main C filter matches spektrafilm reference behavior.
- User-facing sliders may remain named/displayed as `Y/M/C`, but values are converted once at the recipe boundary. No normalized 170-step value crosses that boundary.
- Preflash filters use internal Kodak CC values:
  - `C = neutralC`
  - `M = neutralM + preflashM`
  - `Y = neutralY + preflashY`
- If Film-Juicer has no preflash-shift UI in a phase, `preflashM=0` and `preflashY=0` are explicit recipe defaults. There is no preflash C slider in the spektrafilm contract.
- spektrafilm parity fixtures must set `userC=0`. Nonzero `userC` evidence is labeled as Film-Juicer extension evidence.
- Print exposure normalization truth table:
  - `normalize=false`, `compensation=false` -> `normalizer = 1`
  - `normalize=false`, `compensation=true` -> `normalizer = factorMidgrayComp / factorMidgray`
  - `normalize=true`, `compensation=false` -> `normalizer = factorMidgray`
  - `normalize=true`, `compensation=true` -> `normalizer = factorMidgrayComp`
- The default state is `normalize_print_exposure=true` and `print_exposure_compensation=true`; therefore the default normalizer is `factorMidgrayComp`.
- `factorMidgray` and `factorMidgrayComp` are computed from spektrafilm's print-balance midgray references:
  - Re-simulate neutral RGB `0.184` through film raw, authored film density curves, film `channel_density`, and film `base_density`.
  - This midgray simulation uses `_rgb_to_film_raw` call defaults for the synthetic midgray input: `color_space=sRGB` and `apply_cctf_decoding=false`, not the active Film-Juicer input color/CCTF settings.
  - It still consumes the active spektrafilm film-raw path: selected `settings.rgb_to_raw_method`, final sensitivity after UV/IR filter normalization, film reference illuminant, Hanatos 2025 adaptation window/surface/spectral-blur/profile adaptation state when Hanatos is selected, Mallett basis identity when Mallett is selected, and film density-curve gamma.

  - `factorMidgrayComp` exists only when print exposure compensation is enabled and uses `0.184 * 2^manualCameraExposureEV`.
  - Both factors are reciprocal geometric means of print raw midgray under the filtered print illuminant and print sensitivity.
- These factors are immutable print-owned recipe derivations or named host derivation resources prepared before CUDA payload assembly. Kernels consume prepared scalar payload values; render code must not recompute the midgray simulation, perform asset lookup, or repack profile tables for this step.
- Phase 4 evidence must record the update frequency and descriptor/cache identity for `factorMidgray`, `factorMidgrayComp`, filtered main illuminant, filtered preflash illuminant, and preflash raw. These derivations may rebuild on selected film/print profile payload identity, print illuminant, selected dichroic set/resource identity, C/M/Y CC values, preflash values, print-normalization mode, manual camera exposure EV when compensation is enabled, print sensitivity changes, RGB-to-raw method, UV/IR filter settings, film reference illuminant, Hanatos adaptation/profile/settings identity, Mallett basis identity, final sensitivity identity, or film density-curve gamma. They must not rebuild per frame, per tile, or during render launch when those identities are unchanged.
- Print raw order is:
  - `rawBase = filmDensityToPrintRaw(filteredMainIlluminant)`
  - `rawPreflash = preflashExposure > 0 ? captureFilmBaseDensityToPrintRaw(filteredPreflashIlluminant) * preflashExposure : 0`
  - `rawPrint = (rawBase * normalizer + rawPreflash) * printExposure * blackWhitePrintCorrection`
  - enlarger diffusion, if enabled, runs after this scaling and before `log10`.
- `captureFilmBaseDensityToPrintRaw` means spektrafilm's preflash raw calculation from capture-film `data.base_density`, filtered preflash illuminant, and print sensitivity. It is not print `base_density`, zero density, or `density_min`.
- Therefore preflash is scaled by `printExposure` unless a later approved Film-Juicer divergence explicitly changes output and validation expectations.

SpatialOptics

Owns:
- enabled components by domain
- domain tags
- requested backend
- resolved backend
- backend source
- recipe-level component descriptors
- exact-required failure policy

Domains:
- `FilmLinearExposure`
- `PrintLinearExposure`

- `ScannerXyz`
- `ScannerLinearRgb`

Backends:
- `Off`
- `Exact`

Rules:
- Disabled components do not affect output, hashes, resource keys, diagnostics, or scratch.
- Exact is the spektrafilm-matching backend for optics behavior. `SpatialOptics` recipe data owns enabled components, authored controls, requested backend, and exactness policy. Frame-derived exact execution descriptors request PSFs, FFT dimensions, frequency resources, cuFFT work, and scratch only during preparation from `SpatialOptics` plus `FrameRequest` extent/`pixelSizeUm`.
- Fast optics is not a backend in this plan. Do not add Fast recipe values, descriptors, hashes, resources, kernels, or UI behavior.
- Convert micrometer-valued camera, enlarger diffusion, DIR, halation, and exact-optics quantities to pixels from the `FrameRequest::pixelSizeUm` value derived from film recipe render-scale policy plus concrete frame extent.
- Scanner lens blur is pixel sigma in scanner RGB space and must not use micrometer conversion.
- Canonicalize descriptors before hashing/resource lookup.
- Skip inactive, zero-strength, zero-radius, and identity components before descriptor creation.
- Keep recipe identity separate from execution/resource shape. Recipe hashes include shape-independent output decisions; exact execution/resource/scratch descriptor hashes include frame extent, padded dimensions, pixel size, precision, channel grouping, and other frame-derived shape fields only when the consuming resource or launch-visible descriptor depends on them.
- Use one component list per domain over per-effect launch paths when ordering permits.
- Admission results, memory-pressure choices, prepared scratch decisions, exact resource availability, and output-stable pressure strategies are preparation results. They belong to `PreparedCudaFrame` preparation diagnostics/views, not to `RenderRecipe`.

Implementation shape:
- use plain structs for descriptors
- use fixed arrays or small vectors for enabled components
- use direct `switch`/loop dispatch
- keep CUDA payload packing explicit
- do not introduce component inheritance, visitors, callback registries, abstract factories, `std::function` launch lists, generic stage graphs, or an optics framework

GrainContract

Owns:

- spektrafilm grain statistical/schema contract
- derived `density_min`
- sublayer activation
- `density_curves_layers` presence/dimensions
- positive/negative layer interpolation mode
- Film-Juicer grain-control handoff

Rules:
- Film-Juicer visual grain controls do not own scanner/LUT density bounds.
- `density_min` defaults to C/M/Y density `(0.07, 0.08, 0.12)`. If Film-Juicer exposes an override, the override is a typed `GrainContract` recipe field, participates in hashes for consumers whose bounds change, and is recorded in diagnostics. It is not owned by visual grain controls.
- When Film-Juicer exposes or consumes spektrafilm grain fields, their schema/statistical meaning and defaults are owned here: `agx_particle_area_um2`, `agx_particle_scale`, `agx_particle_scale_layers`, `uniformity`, `blur`, `blur_dye_clouds_um`, `micro_structure`, and `n_sub_layers`. These fields are not a mandate to visually match spektrafilm rendered grain; final visual tuning remains deferred to real grain plates.
- `data.density_curves_layers` is authored and consumed as `[N][3 sublayers][3 channels]`. Spektrafilm `n_sub_layers` is not authority to vary this profile shape and must not be used as a resource dimension for authored layer curves; in the inspected reference it only affects the non-sublayer statistical grain path. If Film-Juicer maps `n_sub_layers` into its custom visual grain controls later, that mapping is an explicit Film-Juicer grain handoff, not a profile-schema interpretation.
- `density_min` feeds direct film scanner and enlarger film-density bounds even when visual grain is off; print-media scanner bounds come from print density curves.
- Missing or malformed density-layer data fails only when the selected route/option requires it.
- spektrafilm CPU `use_fast_stats` has no production runtime effect.

FrameRequest

Per-frame immutable input built from one stable published `RenderRecipe` snapshot/reference plus frame-local execution inputs. During migration that recipe may be carried by a narrow `WorkingState` publication envelope, but `WorkingState` is not a long-term peer contract or source of output-affecting policy.

The long-term `FrameRequest` type is a render-boundary contract, not a `JuicerProcessor` implementation detail. Phase 1C should create or move it to a focused frame/request header or equivalent boundary file unless the manifest records a source-marked temporary bridge with a Phase 3 removal/narrowing gate. Once Phase 3 accepts pixel behavior, render code must consume this boundary directly instead of copying it into mutable processor side-channel members.

Rules:
- Host-only frame-local normalization work only.

- No file discovery.
- No live mutable instance reads after snapshot.
- No CUDA context acquisition.
- No durable GPU residency acquisition.
- No ad hoc policy negotiation.
- No route, profile, print, scanner, optics, grain, density-bound, default, or resource-identity policy that could have been resolved into `RenderRecipe`.
- Expected long-term contents are recipe snapshot/reference, concrete full-frame extent, render window/layout/components, derived `pixelSizeUm`, temporal frame/session tokens, clip/source token, frame-bounds version, and per-frame auto-exposure metering request inputs.
- CUDA context key, stream identity, submission snapshot, context epoch, and durable residency acquisition are not `FrameRequest` fields. They are supplied to, derived inside, or returned from the preparation boundary because they describe the concrete CUDA execution context rather than the immutable host frame snapshot.
- `FrameRequest` derives and carries `pixelSizeUm` as an immutable frame-local value from recipe-owned film-format policy plus concrete frame extent. After `FrameRequest` construction, render and preparation code must consume that value or descriptor fields derived from it; they must not recompute pixel size ad hoc from unrelated state.
- Production auto-exposure pixel metering is GPU work. `RenderRecipe` owns `auto_exposure`, `auto_exposure_method`, input color/CCTF settings, manual exposure compensation, and output-affecting auto-exposure policy. `FrameRequest` owns frame-local metering inputs: concrete full-frame extent, render window/source extent mapping, spektrafilm preview descriptor (max long edge 256, nearest-neighbor/order-0 semantics), metering bounds, weights/LUT identity, temporal/cache token, source identity token, and stable recipe snapshot. The measured EV/scale is produced by the GPU metering pass during prepared-frame execution and consumed by film raw exposure before RGB-to-film-raw. A cached measured value is valid only when its frame/clip/source/bounds/preview/method token matches the current `FrameRequest`.
- A fresh measured EV/scale is not part of immutable `FrameRequest`. It is a prepared-frame/run-local result with token-checked optional writeback. This preserves spektrafilm stage order (`auto_exposure(image)` before `_rgb_to_film_raw`, then manual `exposure_compensation_ev` after film raw) without making the frame input object mutable or turning CUDA packers into auto-exposure policy owners.
- The same-frame production path must not read the measured EV/scale back to the CPU and synchronize before continuing. GPU reductions, histograms, partials, validity flags, and final scale are frame-local CUDA resources or prepared-frame views; host readback is diagnostics/writeback only and happens after the render no longer depends on it.
- Host render-window/ROI invocations are not durable resource identity by default. A full-frame-only product policy is acceptable only when it fails before resource preparation and before any tile-local substitute can act as product rendering. If a route accepts partial render windows, durable resources, stable host derivations, LUTs, filters, profile tables, auto-exposure cache tokens, and route payload identity must be keyed by full-frame/source/recipe descriptors where those are the semantic inputs; render window may scope output writes, launch geometry, and frame scratch dimensions only when the family contract says so. If the host cannot expose

full source memory for the spektrafilm preview meter from a partial render-window invocation, auto-exposure blocks with a diagnostic instead of substituting tile-local or CPU metering.
- Host CPU code must not read source pixels to compute production spektrafilm auto-exposure. CUDA packers and resource managers must not decide the auto-exposure method or bounds.
- Before any spektrafilm pixel behavior lands, the render path must build one immutable `FrameRequest` from the published `RenderRecipe` plus frame-local values. Resource preparation and CUDA payload packing consume that `FrameRequest` or named subrecipes derived from it.
- After `FrameRequest` construction begins, render code must not read live processor/instance fields for route, profile identity, print params, scanner settings, overrides, hashes, resource identity, or output-affecting temporal values.
- New spektrafilm render code must not copy `FrameRequest` fields into mutable `JuicerProcessor` side-channel members and then consume those members as the real contract. Existing setters such as `setPrintParams`, `setPrintRuntime`, `setWorkingState`, scanner setters, and override setters are bridge debt once the corresponding behavior moves; they may remain only behind named temporary bridges with removal phases.
- Direct-route product behavior accepted in Phase 3 must consume route, profile, scanner, density-bound, exposure, and direct-resource identity through `FrameRequest` / `RenderRecipe` / named subrecipes, not through mutable processor side channels copied from `FrameRequest`.
- Print-route product behavior accepted in Phase 4 must consume print params, print runtime identity, scanner handoff, and print-resource identity through `FrameRequest` / `RenderRecipe` / named subrecipes or immutable host derivations, not through mutable processor side channels copied from `FrameRequest`.
- Output-affecting temporal/session values must be snapshotted before preparation begins. Any completion writeback to instance/session/clip state must compare the expected instance token, session token, clip token, frame-bounds version, recipe hash, and source/metering token where relevant. Stale completion writeback is dropped; it must never update current state after an overlapping render, abort, clip change, session reset, or recipe replacement.
- Any retained old render input is a named `SF_TEMP_BRIDGE` adapter with allowed call sites and a removal phase.

Phase 1C must create a current-field disposition table for `JuicerProcessor::FrameRequest` and the mutable `JuicerProcessor` side-channel members/setters that mirror its values before Phase 1C acceptance and before Phase 3 pixel behavior can start. The full table may live in the phase manifest or in a short plan appendix referenced by that manifest, but while any `TemporaryBridge`, old setter, or mirrored mutable processor member remains, the focused `FrameRequest` / adapter source must carry a compact source-adjacent pointer or bridge ledger that names the disposition artifact, bridge names, allowed call-site family, and removal phase. A manifest-only disposition is not enough for editable source a new developer must work in. The table must classify every field in the current `FrameRequest` shape and every mirrored current render-input member into one of these buckets:
- `FrameLocal`: render window, extent/layout/components, frame time, frame rate, session/temporal tokens, clip/source token, frame-bounds version, `pixelSizeUm`,

auto-exposure metering request descriptors/tokens, and a validated prior auto-exposure cache token only when the token proves the cached value belongs to the current frame request. Context key/epoch, stream identity, and submission snapshot are preparation-boundary inputs/results, not `FrameRequest` policy.
- `RecipeOwned`: output-affecting values resolved before render, including route, profile identity, scanner options/settings, print params, output encoding, optics settings, grain contract, DIR controls/defaults, halation/glare settings, density bounds, hashes, resource identity, auto-exposure method, and render-scale policy that is independent of concrete frame extent.
- `HostDerivation`: host-side derivations that are immutable for the recipe and referenced by recipe identity or asset/version token.
- `TemporaryBridge`: legacy values retained only behind a named bridge with allowed call sites and removal phase.
- `Deleted`: values used only by removed old behavior.

Minimum disposition for the current codebase:

Each row must name the current request field or field family, any mirrored processor setter/member that currently carries the same value (`setFrameRequest`, `setPrintParams`, `_printParams`, `setWorkingState`, `_ws`, scanner setters, override setters, `_dirRT`, and similar fields), the final owner, allowed consumers during the phase, and the removal phase for any bridge. The table must explicitly call out setter side effects such as `setWorkingState()` updating DIR runtime values from `WorkingState`. A disposition table that classifies `FrameRequest` but leaves the mutable processor copy path unnamed is not accepted. A bridge ledger in source may be short, but it must let `rg SF_TEMP_BRIDGE` or the bridge name lead a developer from the old field/setter to the owning disposition row without reading the full port plan.

Current `FrameRequest` field or family	Required disposition
`workingState`	Keep only as the publication envelope carrying `RenderRecipe` during migration. Long-term consumers use recipe/subrecipe references, not flat legacy fields.
`printRuntime`	`TemporaryBridge` or `HostDerivation`; cannot remain the print behavior contract after Phase 4.
readiness booleans	`FrameLocal` preflight status only; not output identity.
`components`, `renderWindow`	`FrameLocal`.
`scannerOptions`, `scannerSettings`	`RecipeOwned` scanner recipe/post-effect fields.
`printParams`	`RecipeOwned` `PrintRecipe` print-filter fields; raw `PrintBypass` excluded from recipe hashes.
halation, grain, and glare overrides	`RecipeOwned` or deleted/renamed current-contract controls; bridge only until owning optics/grain/scanner phase.
`dirRuntime`	`RecipeOwned` DIR recipe or host derivation; bridge only until Phase 5.
`exposureScale`, camera auto/manual fields	Split: per-frame metering request descriptors/tokens are `FrameLocal`; typed auto-exposure method, enabled state, input color/CCTF settings, and resolved exposure ordering/manual scale semantics are

`RecipeOwned`; a fresh GPU-measured auto scale is not a `FrameRequest` field and belongs to a named prepared-frame/run-local auto-exposure result unless an already measured value is carried by a validated cache token. |
`autoExposureMeterBounds*`	`FrameLocal`.
`outputEncoding`	`RecipeOwned` scanner/output identity in `ScannerOutputRecipe`; it is copied into `FrameRequest` only as part of the stable recipe snapshot/reference.
session, instance, clip, frame time/rate, frame bounds version	`FrameLocal` / temporal identity.
`gateWeaveAmount`	Film-Juicer temporal/defect system; not part of spektrafilm parity and must be explicitly classified as retained Film-Juicer visual behavior or deleted.
`pixelSizeUm`	Split ownership: film-format/render-scale policy is `RecipeOwned`; the concrete value is `FrameLocal`, derived once in `FrameRequest` from that policy plus full-frame extent. No ad hoc recomputation after `FrameRequest` construction.
preview/draft/sequential render hints	`FrameLocal` scheduling/diagnostic hints only; must not change output or select an optics backend.

PreparedCudaFrame

The only render preparation boundary.

Long-term API target:
- `Root::prepare_cuda_frame(...)` is the public preparation boundary.
- Preparation/admission/build/upload work happens before or inside this boundary from `FrameRequest`, `RenderRecipe`, resource descriptors, and context-owned services.
- After preparation, render code consumes prepared views, frame-owned leases, durable bundle handles, status, `finish()`, and `abort()`.
- Public effect-specific `PreparedCudaFrame::prepare_*` methods are temporary bridges unless they only expose or bind already-declared frame scratch/views without resource discovery, upload, allocation, schema interpretation, hash construction, or old-state translation.
- A public `PreparedCudaFrame::prepare_*` bridge may survive only if it exposes or binds already-declared prepared-frame views/leases. It must not perform file/resource discovery, hash construction, schema interpretation, old-state translation, upload, allocation, or route/resource policy decisions after the frame has nominally been prepared.
- Any bridge that still performs those actions must be removed before the phase that implements that resource family's spektrafilm pixel behavior.
- Do not add new public effect-specific `PreparedCudaFrame::prepare_*` or public render-path `ensure_*` / `command_ensure_*` methods for spektrafilm resource families. New resources enter through descriptor-driven frame preparation. Any exception is an owner-approved temporary bridge with allowed call sites, cleanup gate, and removal phase.

Owns or retains:
- context epoch association
- durable GPU bundle handles
- frame scratch/workspace leases

- staging and temporary buffers
- frame-local transient CUDA/pinned/event resources such as scan-error flags, readbacks, auto-exposure temporaries, and preparation staging
- resource views
- completion and abort state

Rules:
- Render after preparation binds prepared handles and launches kernels.
- Mutable frame work is never shared across overlapping renders.
- Durable resources are context-owned, not instance-owned.
- Context reset/retire invalidates only the affected context/epoch.
- Render code does not introduce standalone CUDA allocation ownership after preparation has begun.
- Scan-error staging/readback is diagnostic/lifetime-control work, not product pixel correctness. Before any CUDA route that can signal scan errors is accepted, the family contract must state whether scan-error staging is always present or gated; prove it has bounded device/pinned/event footprint, does not participate in output/resource hashes, does not force a same-frame GPU-to-CPU synchronization dependency, and does not allocate/free raw CUDA or pinned resources every steady-state frame unless that allocation is explicitly admitted as a bounded always-present diagnostic exception.

Host / OFX Lifecycle Matrix

Each phase that touches OFX describe/create, instance bootstrap, render entry, frame preparation, CUDA launch, abort, context lifetime, or session state must keep this lifecycle contract current in its manifest.

Host action / lifecycle path	Required behavior
Describe / create	Publish only the current spektrafilm descriptor under the retained OFX identifier. Old params are not defined, fetched, diagnosed, mapped, or migrated. Required current params have explicit spektrafilm defaults or fail as current-contract descriptor errors.
Attach / bootstrap	Build metadata catalogs, current param handles, and immutable host-side services only. No CUDA context is required. No old saved state is interpreted.
Param change	Rebuild or invalidate `RenderRecipe` from current spektrafilm params only. Hashes use resolved recipe fields and asset/version tokens, not live catalog lookup or old indexes.
Clip / session change	Increment or replace the relevant session, clip, frame-bounds, and source tokens before any new `FrameRequest` is built. Pending stale completion writeback from older tokens is dropped.
Normal CUDA render	Validate current params/profile assets/resources, build `RenderRecipe`, build one immutable `FrameRequest`, prepare through `PreparedCudaFrame`, launch from prepared views/leases, then call `finish()` exactly once after successful completion.

CUDA unavailable	Fail before output. Do not route to CPU pixels, do not fetch source/destination images for a CPU fallback, and do not prepare frame resources as if a fallback could render.
Host non-CUDA product render entry	Fail or use the minimal host-required `NonCudaRenderPathInvoked` stub before output, source reads, destination writes, CPU auto-exposure, `processImpl()`, `multiThreadProcessImages()`, or frame/resource preparation can act as product rendering. The reference-only CPU harness is outside the OFX product route.
Abort before preparation	No `PreparedCudaFrame` exists. Drop the render-local request and perform no completion writeback.
Abort after preparation, before launch	Call `abort()` on the prepared frame, release frame leases/scratch/staging, retain context-owned durable resources only when their context/epoch remains valid, and perform no output writeback.
Abort after launch	Record in-flight use where needed, call `abort()` once, release frame-local resources after stream/event safety requirements are met, and drop completion writeback unless the render completed and tokens still match.
Overlapping renders	Each render owns its own `FrameRequest`, prepared frame, scratch, staging, metering result, and temporal tokens. Durable resources may be shared only through context-owned descriptors and leases.
Context reset / retire	Invalidate only resources for the affected `DeviceContextKey` and context epoch. In-flight prepared frames finish or abort through their existing ownership path; new renders prepare against the new context/epoch.
Shutdown	Stop new preparation, wait for or abort active preparations/renders through prepared-frame ownership, retire context-owned resources in order, then release host services.

Data-Flow And Performance Shape

This plan does not set benchmark targets, but it must prevent slow architecture. Data order and flow are correctness-adjacent for a GPU-only renderer because bad boundaries create repeated packing, redundant uploads, CPU/GPU synchronization, layout churn, and hidden CPU render work.

Review each phase that introduces or changes a CUDA payload, resource descriptor, scratch family, LUT, large table, PSF, FFT resource, or transient image with these questions:
- Is there one clear path: profile assets -> `RenderRecipe` -> `FrameRequest` -> resource descriptors -> `PreparedCudaFrame` -> CUDA payloads/kernels?
- Does each output-affecting datum have one producer, one owner, and a clear consuming phase/kernel/resource?
- Are authored profile fields, derived recipe fields, immutable host derivations, durable GPU resources, frame-local scratch, and CUDA payloads distinct?
- Are hashes built from already-resolved recipe fields and asset/version tokens, not lookup, file discovery, or repacking side effects?
- Are immutable resources uploaded only when descriptor identity changes?

- Are frame-local buffers admitted through `PreparedCudaFrame` views/leases, not allocated, resized, or discovered during render?
- Are CUDA payloads designed around kernel access patterns rather than mirroring host structs or legacy `WorkingState` layout?
- Are channel/data orders explicit at every boundary: RGB, C/M/Y, Y/M/C UI, density channels, XYZ, linear RGB, and spectral wavelength order?
- Are conversions between orders performed once at the owning boundary rather than repeatedly in hot paths?
- Are spectral tables, density curves, `ScannerSpectralLutDescriptor` resources, grain parameters, PSFs, FFT buffers, and temporary images laid out for their consuming kernels?
- Do disabled/off features produce no descriptor, no upload, no scratch admission, no hash contribution, and no kernel work?

This contract is review evidence, not runtime data. Keep C++ descriptors as the smallest plain identity keys that decide reuse/admission for their resource; do not add producer/consumer prose, diagnostics, lifecycle policy, or hash commentary as descriptor fields. For each new CUDA payload/resource/scratch family, the phase manifest or a source-adjacent note referenced by that manifest must name:
- producer
- consumer
- update frequency
- host layout
- device layout
- ownership/lifetime
- descriptor/hash identity
- descriptor schema/version source when applicable, including what layout, generated-table, numeric-domain, channel-order, or resource-semantics changes require incrementing it
- included hash inputs
- intentionally excluded hash inputs
- upload/preparation point
- disabled/off behavior, or `not applicable` with a concrete reason for always-present diagnostic/lifetime resources

Do not add block-size, shared-memory, cache-hit, or benchmark requirements before the implementation has a concrete performance question. This is not permission to accept slow hot-path design: require enough data-flow shape that accidental slow architecture is reviewable before code lands.

Schema And Type Rules

Boundary Names

Keep spektrafilm field names visible at the loader boundary:
- `metadata.version`

- `metadata.copyright`
- `metadata.created`
- `metadata.license`
- `metadata.citation`
- `metadata.datasource`
- `info.stock`
- `info.name`
- `info.support`
- `info.stage`
- `info.type`
- `info.use`
- `info.antihalation`
- `info.target_print`
- `info.channel_model`
- `info.densitometer`
- `info.reference_illuminant`
- `info.viewing_illuminant`
- `info.log_sensitivity_density_over_min`
- `data.wavelengths`
- `data.log_sensitivity`
- `data.hanatos2025_adaptation_window_params`
- `data.hanatos2025_adaptation_surface_params`
- `data.channel_density`
- `data.base_density`
- `data.midscale_neutral_density`
- `data.log_exposure`
- `data.density_curves_model`
- `data.density_curves`
- `data.density_curves_layers`

Do not invent profile-level `grain.*` fields unless the bundled schema deliberately adds them later.

Bounded Domains

Use explicit enums or small validated value types for bounded upstream/profile domains. Do not let normal behavior depend on string comparisons, display names, substrings, or file-name heuristics after the loader/UI boundary.

Required bounded or boundary-normalized domains include:
- `ProfileSupport` from `info.support`
- `ProfileStage` from `info.stage`
- `ProfilePolarity` from `info.type`
- `ProfileUse` from `info.use`

- `AntihalationMode` from `info.antihalation`
- `ChannelModel` from `info.channel_model`
- `DensitometerLabel` from `info.densitometer`, treated as an opaque string wrapper for profile metadata unless a current consumer explicitly defines bounded behavior
- `IlluminantKey` for reference, viewing, print, and enlarger illuminants
- `RgbToRawMethod` for Hanatos/Mallett/raw conversion selection
- `AutoExposureMethod { CenterWeighted, Average, Median, Partial, Matrix, MultiZone, HighlightWeighted }`
- `PrintNormalizationMode`
- `DichroicFilterSet`
- `ScanRoute`
- optics domain/backend enums

Optional bounded fields are parsed and checked only when catalog visibility, selected-route recipe construction, digest/default policy, resource preparation, or compact diagnostics consume them. Unknown consumed values are either a hard selected-route/resource failure or a typed `Unsupported` value that blocks the selected route through a named diagnostic. Unknown unconsumed optional values do not create a product validation failure, and consumed values must not fall back to default strings silently.

Host Bounded Params

Raw OFX integers, strings, and choice values are UI-boundary inputs only. Recipe building converts them once into typed values before publication, hashing, resource descriptors, CUDA payload packing, or diagnostics.

This applies at least to:
- input color space
- input CCTF decode mode
- output color space
- output CCTF encode mode
- spectral/RGB-to-raw method
- auto-exposure method
- UV/IR camera filter controls
- Hanatos 2025 adaptation window, adaptation surface, and spectral Gaussian blur controls
- scanner mode/settings
- route/direct-vs-print selection
- optics backend
- camera diffusion filter active/family/strength/spatial-scale/halo/core/bloom controls
- enlarger diffusion filter active/family/strength/spatial-scale/halo/core/bloom controls
- print normalization mode
- dichroic filter set
- print/enlarger illuminant choice

Invalid current-contract host values must not silently clamp to a default through helpers such as `colorSpaceFromIndex()`. They either:
- map to an explicit spektrafilm-era default named by this plan, or
- fail during recipe build/preflight as `MissingRequiredHostParameter` or `UnsupportedMode`.

Clamp-to-default helpers may remain only at UI display/default-construction boundaries or validation-only code. They must not hide invalid host state during recipe construction.

Runtime Defaults

Runtime defaults are resolved at the UI/recipe boundary and recorded as recipe fields only when they can affect output. These defaults describe Film-Juicer's spektrafilm-era product controls, not old saved-project compatibility.

Domain	Default	Policy
DIR	active, spatial defaults `diffusion_size_um=20.0`, `diffusion_tail_um=200.0`, `diffusion_tail_weight=0.06`; gamma constants come from the post-digest `ProfileDigest` table above	Matches the intended spektrafilm product behavior. Routes that require active/default DIR must either implement the full active spatial DIR contract or fail with `DirNotImplementedForPhase3`; non-spatial DIR is validation-only unless the fixture explicitly sets spatial diffusion to identity and records that state. Agents must not silently disable active DIR, use the pre-digest schema gamma defaults as final runtime values, or accept non-spatial DIR as default product behavior.
Halation	off	Intentional Film-Juicer user-facing default. Enabled halation uses spektrafilm math and profile-derived `(use, antihalation)` presets. Disabled halation emits no optics descriptor, resource request, scratch role, or optics hash contribution.
Highlight boost	`boost_ev=0.0`, `boost_range=0.3`, `protect_ev=4.0`	spektrafilm film-linear exposure behavior. It is independent of `halation.active`, runs after manual exposure and before camera diffusion/lens blur/halation, and is output identity at default `boost_ev=0.0`.
Film-Juicer visual grain	off	Intentional Film-Juicer user-facing default. This does not disable the spektrafilm grain contract fields that feed scanner/enlarger density bounds.
Print-route glare	active, `percent=0.03`, `roughness=0.7`, `blur=0.5`	Matches spektrafilm print-scan behavior and `GlareParams` defaults. Direct film scan routes still disable glare regardless of this default. Glare is runtime scanner/output or print-rendering recipe data, not profile-authored metadata. Production GPU glare is deterministic for a stable frame/recipe/source/window while preserving spektrafilm's lognormal distribution, blur, and XYZ illuminant add. Pixel-exact reference fixtures must seed/bridge the Python reference or disable glare; glare-on parity evidence may use statistical or owner-approved visual tolerances. Before Phase 8, print-scan routes requiring active glare fail with `GlareNotImplementedForPhase4` unless the validation fixture explicitly disables glare. Agents must not silently disable active glare to make an early route render.

Camera diffusion filter	off, family `black_pro_mist`	Matches spektrafilm default inactive diffusion control. The family choice remains a UI-boundary choice until Phase 6 converts it into `SpatialOptics` recipe data.
Enlarger diffusion filter	off, family `black_pro_mist`	Matches spektrafilm default inactive diffusion control. The family choice remains a UI-boundary choice until Phase 6 converts it into `SpatialOptics` recipe data.
Camera lens blur	`0` micrometers	Identity value; no descriptor or scratch when zero. Micrometer-to-pixel conversion consumes `FrameRequest::pixelSizeUm`.
Enlarger lens blur	unsupported/output-inert	spektrafilm exposes `enlarger.lens_blur=0`, but the inspected spektrafilm runtime does not consume it. The port does not implement enlarger lens blur until a later approved Film-Juicer extension gives it behavior.
Scanner lens blur	`0` pixel sigma	Identity value; no descriptor or scratch when zero. This is RGB-space scanner blur and does not use micrometer conversion.
Scanner unsharp mask	`(0.7, 0.7)`	spektrafilm scanner default `(sigma_px, amount)`; runs after scanner blur and before output CCTF/final clip.
Production LUT execution	GPU LUT path enabled as execution strategy, resolution default `17`	Intentional Film-Juicer execution policy even though spektrafilm settings default `use_enlarger_lut=false` and `use_scanner_lut=false`. This is not a user-visible color-science toggle. Scanner/enlarger 3D LUT generation and sampling must be spektrafilm-equivalent to `compute_with_lut`: endpoint-inclusive grids, normalized lookup coordinates, and `apply_lut_3d` default `pchip` interpolation with per-cell min/max clamping. Hanatos filming TC LUT sampling must follow `apply_lut_cubic_2d` reflected-boundary Mitchell cubic interpolation plus brightness rescale.
Input/output color management	input color space `DaVinci Wide Gamut`, input CCTF decode `false`; output color space `sRGB`, output CCTF encode `true`	Intentional Film-Juicer host color-management defaults retained for the spektrafilm-era product. They override spektrafilm Python's new-parameter default input color space of `ProPhoto RGB`, but do not change spektrafilm stage ordering or internal midgray print-balance defaults. Reference comparisons must set these values explicitly or mark the result as extension-mode. Input color/CCTF belongs to `FilmRawRecipe`; output color/CCTF belongs to `ScannerOutputRecipe`.
Print illuminant	`TH-KG3`	spektrafilm enlarger default. Current Film-Juicer illuminant choices remain available when resolved by `standard_illuminant`.
Dichroic filter set	reference/custom	Matches spektrafilm `color_enlarger` default. Film-Juicer measured sets remain selectable extension-mode choices and must be explicit recipe/hash/fixture identity when selected.
Neutral print filters	C/M/Y CC `(0, 65, 55)` for fresh/default params	spektrafilm schema defaults. Calibrated DB entries override the current neutral values. Missing neutral DB/file entries preserve the current neutral values; therefore a fresh/default recipe remains `(0, 65, 55)`, while future authored/manual neutral values are not reset by a DB miss.
UV/IR camera filters	UV `(0, 410, 8)`, IR `(0, 675, 15)`	Advanced controls; default amplitudes are inactive.
Auto-exposure enabled	`true`	Matches spektrafilm `camera.auto_exposure`. Disabled auto exposure produces identity measured auto scale; manual exposure still follows spektrafilm ordering.

Auto-exposure method	`center_weighted`	Typed method domain is `center_weighted`, `average`, `median`, `partial`, `matrix`, `multi_zone`, and `highlight_weighted`. Method math follows spektrafilm's luminance, mask/region, `small_preview` long-edge 256 nearest-neighbor metering image, and `Inf -> 0 EV` behavior. Unsupported values fail before recipe publication as an explicit Film-Juicer hardening deviation from spektrafilm's unknown-string identity fallback.
Print exposure normalization	`normalize_print_exposure=true`, `print_exposure_compensation=true`	Matches spektrafilm defaults. If represented as `PrintNormalizationMode`, default is `NormalizeAndCompensate` and the normalizer is `factorMidgrayComp`.
Scanner black/white correction	`black_correction=false`, `white_correction=false`, sRGB-encoded `black_level=0.01`, sRGB-encoded `white_level=0.98`	Matches spektrafilm scanner defaults. Recipe construction derives linear levels through spektrafilm's sRGB CCTF decode rule. Route-specific correction application is defined under `ScannerOutputRecipe`; negative direct film scans skip color-reference correction.
`density_min`	C/M/Y density `(0.07, 0.08, 0.12)`	Owned by `GrainContract`; consumed by direct film scanner and enlarger film-density bounds, including when visual grain is off.
Hanatos 2025 adaptation controls	window on, surface off, spectral Gaussian blur `0.0`	Advanced controls beside UV/IR. Profile-authored window/surface params are preserved; setting defaults match current spektrafilm behavior and participate in hash/LUT identity when output-affecting.

Any default not listed here must come from `external/spektrafilm` reference behavior or be added to this table with an explicit output-impact policy before implementation.

Profile Digest Boundary

The profile loader owns authored JSON fields. Recipe building owns derived runtime fields.

Required digest behavior:
- Run a named `ProfileDigest` or equivalent after loader validation/defaulting and before recipe hash/resource/payload construction.
- Inputs are authored profile metadata, selected route, stock key, and user/runtime controls. Optional neutral calibration data may be inspected only for Phase 2 source/availability inventory evidence or by the Phase 4 print-owned recipe builder/helper. The digest must not write derived values back into the authored profile payload.
- Output-affecting derived fields include DIR gamma defaults, stock-specific DIR overrides, halation low-level presets, and any route/color-reference correction factors that can be computed before frame preparation. Neutral-print-filter choices are print-owned derived fields: Phase 2 may record source identity and broad availability as inventory, but selected-entry parsing, final neutral C/M/Y values, print-route fallback status, output-affecting hash identity, resource descriptors, and resource preparation land first in Phase 4 `PrintRecipe`. Direct-route acceptance requires neutral calibration exclusion, not setup-only preflight.

- `density_min` and glare parameters are not profile-digest outputs. They are runtime recipe/control defaults or user overrides, with halation low-level presets being the profile-derived exception in this behavior area.
- Profile-authored Hanatos adaptation window/surface params remain profile payload data; Hanatos setting defaults (`apply_window`, `apply_surface`, `spectral_gaussian_blur`) remain recipe fields. The digest may assemble their status for diagnostics, but must not invent the removed `data.bandpass_hanatos2025` contract or write derived adaptation values back into authored profile data.
- spektrafilm `settings.preview_mode` and debug deactivation switches are not production Film-Juicer features. Do not model upstream `settings.preview_mode` as a recipe output change. Film-Juicer has no separate Preview optics backend in this port.
- Only output-affecting defaults, optional-calibration availability/fallback status recorded by the owning phase, and unsupported bounded modes record source/default status. Do not add parallel provenance fields for values already unambiguous from the recipe.

Payload Value Policy

The loader must preserve authored payload semantics until the owning spektrafilm consumer applies its field-specific policy.

Density-like authored fields preserve `NaN`, JSON `null`, and finite negative samples through ingestion, digesting, recipe construction, and GPU upload until density is converted to transmitted light:

```text
transmitted = 10 ** (-density) * light
transmitted[NaN] = 0
```

Replacing density `NaN` with `0` earlier is wrong. `NaN` density becomes zero transmitted light. Zero density is transparent.

`log_sensitivity` has a different policy:

```text
sensitivity = 10 ** log_sensitivity
sensitivity = nan_to_num(sensitivity)
```

Do not generalize the sensitivity policy to density arrays.

Consumed Data Contract

The loader contract is a consumed-field contract, not a profile/resource validation product. The bundled profile/resource tree is an owner-supplied spektrafilm copy and may be assumed correct. User-supplied files are expected to follow the same contract; Film-Juicer does not lint, repair, migrate, or explain arbitrary malformed user files.

Runtime checks are limited to the fields needed by the current action:
- catalog visibility reads only enough metadata for stable keys, labels, support/stage/polarity, and duplicate-key checks
- selected capture-film routes parse only the profile data consumed by film raw, density development, scanner/enlarger bounds, digest defaults, and active route resources
- selected print routes parse only the print profile/resource data consumed by print exposure, filters, print density development, scanner bounds, and active route resources
- direct scan routes parse selected capture-film and scanner/output fields only. They may retain `PrintProfileKey` as opaque UI/project state, but selected print-profile payload loading, print-profile schema checks, print resource preparation, neutral database lookup, output-affecting hashes, and route failures are not allowed until a print route is selected. Direct-route parity fixtures that compare against Python should still use valid print state because Python constructs print services before dispatch, but that setup quirk is not Film-Juicer product behavior.
- optional fields are parsed only when a current recipe, descriptor, diagnostic, or route decision consumes them
- copied bundle evidence, when recorded, is a source/hash or cheap inventory/smoke check, not a full profile/resource linter

Selected consumed data must still be safe enough to avoid undefined render behavior. If a selected field/resource needed by the active route is missing, has an unusable shape, has the wrong wavelength axis, contains an unsupported bounded value, or violates a consumed finite/NaN/null policy, fail at recipe build or resource preparation with the smallest useful diagnostic. Non-selected user files and unused fields are ignored.

`data.midscale_neutral_density` is not a consumed product field today. Treat it as optional copied-bundle inventory or reference evidence only; do not require its presence or shape for selected direct or print routes, and do not include it in render recipes, output-affecting hashes, descriptors, resources, GPU uploads, neutral calibration, print normalization, or final sensitivity unless a future phase names a current consumer. If a diagnostic or inventory command chooses to inspect it, preserve density-like `null`/`NaN` semantics and keep any validation in that explicit non-render evidence path.

Captured route data consumed when selected includes:
- `data.log_sensitivity`
- `data.channel_density`
- `data.base_density`
- `data.log_exposure` and `data.density_curves`

- fields needed by the selected film raw method, density route, scanner/enlarger bounds, and active DIR/grain/optics consumers

Print route data consumed when selected includes:
- print profile `data.log_sensitivity`
- print profile `data.channel_density`
- print profile `data.base_density`
- print profile `data.log_exposure` and `data.density_curves`
- typed print polarity from `info.type`
- selected print filter, neutral-calibration, illuminant, and scanner-handoff resources that the active route consumes

Optional upstream fields remain optional until a current consumer uses them:
- `info.name`
- `info.log_sensitivity_density_over_min`
- `data.hanatos2025_adaptation_window_params`
- `data.hanatos2025_adaptation_surface_params`
- `data.density_curves_model`
- `data.density_curves_layers`
- `info.reference_illuminant`
- `info.viewing_illuminant`
- `info.target_print`
- `info.densitometer`
- `info.use`
- `info.antihalation`
- `info.channel_model`
- `neutral_print_filters.json` entries

Optional does not mean stringly typed once consumed. Optional bounded fields parse into their domain enum/value type only at the boundary where a current consumer needs them. Invalid consumed values block that selected route or resource; invalid unconsumed values do not create a product validation obligation.

Concrete `ProfileInfo` defaults from spektrafilm are part of the schema lock:
- `type=negative`
- `support=film`
- `stage=filming`
- `use=still`
- `antihalation=weak`
- `channel_model=color`
- `densitometer=status_M`
- `log_sensitivity_density_over_min=0.2`
- `reference_illuminant=D55`
- `viewing_illuminant=D50`

For Film-Juicer's catalog, `info.support`, `info.stage`, and `info.type` follow spektrafilm `ProfileInfo` defaulting before role visibility is decided. Catalog-visible role identity must not depend on folders, old Film-Juicer profile shape, neutral-filter coverage, or profile key/name inference beyond the selected profile key itself. User-created profiles follow the same spektrafilm defaulting policy; unsupported explicit bounded values make the profile unavailable for the affected role. Record defaulted status only when it is output-affecting or diagnostics are enabled; do not collect always-on provenance for every default.

Unusable selected consumed data fails with profile/resource key, source path when cheap, consumed field/resource, route, and phase. Include expected/actual shape only when the code already has it locally.

JSON `null` inside numeric arrays is accepted only where the current spektrafilm profile writer can emit NaN as JSON null and the consuming field's schema lock records a NaN policy. Loader code must not silently replace NaN/null values except at spektrafilm-defined math points such as final sensitivity `nan_to_num`. If a selected consumed field contains `Inf` or another value that cannot be represented safely by the active route, fail that selected route; do not add broad bundle linting to prove copied resources never contain it.

Selected numeric profile token policy:
- JSON numbers are accepted when the field row allows the resulting value.
- JSON `null`, bare/numeric `NaN`, and string `"NaN"` parse to NaN only for fields whose schema-lock row explicitly allows NaN. JSON `null` is not a generic missing-value marker.
- Missing samples, malformed rows, wrong rank, wrong channel count, wrong curve length, and wrong wavelength axis fail selected-profile loading. Do not insert placeholder rows or synthesize NaN rows to keep rendering.
- `Inf`, `Infinity`, `-Inf`, and string equivalents fail for every selected consumed numeric profile/resource field before recipe publication or resource preparation. This is an intentional Film-Juicer hardening deviation from Python/NumPy's ability to represent infinities; it prevents cross-platform overflow drift and accidental transparent/opaque reinterpretation.
- Exact canonical wavelength-axis validation is also intentional Film-Juicer hardening beyond spektrafilm Python's shape-only profile validation. Current spektrafilm resources already satisfy the 380-780 nm, 5 nm axis, and downstream color science assumes that axis. GPU recipes must prove the selected consumed spectral samples are on that axis rather than accepting any 81-sample vector that would produce plausible but wrong output.
- Density-like fields never use `nan_to_num`. `data.log_sensitivity` is the only current profile array that uses `10 ** value` followed by spektrafilm-equivalent `nan_to_num`.
- Any retained parser helper must be field-gated by the schema row. A helper named or shaped like "allow_nan" is not sufficient authority to accept `null`, `NaN`, `Inf`, booleans, malformed rows, or string tokens for a selected field.

Loader Schema Lock

Before Phase 2 implementation, the agent must add or reference a compact consumed-field schema lock that is specific enough for a serial coding agent to implement the loader without inventing shape, unit, channel, default, or finite-value policy. The lock may live in the phase manifest, a short appendix, or a source-adjacent note. It is an implementation checklist for consumed fields, not a checked-in validation suite and not a general profile linter.

The schema lock is not a general validation registry. It covers fields consumed by catalog visibility, recipe construction, resource descriptors, digest/default policy, compact failure diagnostics, or selected-profile schema parity when this plan explicitly names the exception. Optional upstream/source metadata that is not consumed by the render path is ignored; it does not require a permanent row, wrapper type, hash lane, or failure policy.

Each consumed field row records only the fields needed by the loader/recipe/resource consumer:
- field path
- owning payload or recipe group
- required/optional route role
- shape and axis rule
- upstream channel order
- internal channel order if different
- units/value domain
- default or absent-field policy
- finite/NaN/null policy
- failure class and diagnostic identity fields when a selected consumed field can fail the active route/resource
- reference file/function checked in `external/spektrafilm` when the policy is not obvious from the profile dataclass or checked-in JSON shape

Minimum locked fields for a phase:
- catalog-visible fields consumed by that phase: normally `info.stock`, `info.name`, `info.support`, `info.stage`, and `info.type`
- selected-route fields consumed by that phase for recipe construction, resource preparation, digest/default policy, or compact diagnostics
- selected spectral sample axes only where selected spectral samples are consumed
- optional upstream fields only when a current recipe, descriptor, digest rule, or diagnostic consumes them
- legacy `info.fitted_cmy_midscale_neutral_density` and `info.log_exposure_midscale_neutral` only as ignored-key cleanup evidence when encountered in touched loader code, not as consumed behavior

The table below is a compact reference for known consumed or potentially consumed fields. It is not a mandate to parse, validate, fixture, or store every row in Phase 2. A phase includes only the rows needed by current catalog visibility, selected-profile parsing, recipe construction,

resource descriptors, digest/default policy, compact failure diagnostics, or an explicit inventory/evidence command.

Selected-profile schema-lock row reference for Phase 2 loader work:

Field	Shape / order at loader boundary	Unit / domain	Required policy	Default / finite policy
Consumed source metadata	scalar strings	source diagnostic metadata	Optional unless a current consumer uses it	If absent, ignore it unless diagnostics explicitly consume it. Metadata is not profile identity and must not affect pixel output except through raw asset/version tokens. Do not fail bundled or user assets solely because unused `metadata.*` fields are absent.
`info.stock`	scalar string key	profile identity	Required for selected profiles and catalog-visible profiles	No fallback; duplicate role/key is catalog error.
`info.name`	scalar string label	UI/diagnostic label	Optional for menu label	May fall back to `info.stock` for label only; not profile identity.
`info.support`	bounded string -> `ProfileSupport`	`film` / `paper`; spektrafilm default is `film`	Optional/defaulted for selected profiles and catalog-visible profiles	Apply spektrafilm default before catalog role classification. Unsupported explicit value makes the profile unavailable for that role or fails selected-route loading. Do not infer catalog role from folders, key/name, neutral coverage, or old index tables.
`info.stage`	bounded string -> `ProfileStage`	`filming` / `printing`; spektrafilm default is `filming`	Optional/defaulted for selected profiles and catalog-visible profiles	Apply spektrafilm default before catalog role classification. Unsupported explicit value makes the profile unavailable for that role or fails selected-route loading.
`info.type`	bounded string -> `ProfilePolarity`	negative / positive; spektrafilm default is `negative`	Optional/defaulted for selected/catalog-visible route construction	Apply spektrafilm default before route/catalog classification. No polarity inference from key/name beyond the schema default; unsupported explicit values fail before route publication.
`info.use`	bounded string -> `ProfileUse`	still / cine; spektrafilm default is `still`	Optional/defaulted where spektrafilm defaults exist	Default status recorded when output-affecting digest fields change.
`info.antihalation`	bounded string -> `ProfileAntihalation`	strong / weak / no; spektrafilm default is `weak`	Optional/defaulted where spektrafilm defaults exist	Default status recorded when halation presets change.
`info.channel_model`	bounded string -> `ProfileChannelModel`	color / bw; spektrafilm default is `color`	Optional/defaulted where spektrafilm defaults exist	Unsupported model fails if selected route cannot represent it.
`info.densitometer`	scalar string label	densitometer label; spektrafilm default is `status_M`	Optional unless selected behavior consumes it	Opaque/profile-visible metadata in current spektrafilm. Do not reject unknown labels or define a bounded domain unless a current Film-Juicer consumer explicitly assigns output semantics and records the intentional hardening exception.

`info.reference_illuminant`	bounded string -> `IlluminantKey`	reference illuminant; spektrafilm default is `D55`	Optional unless route/resource consumes it	Loader records defaulted status; route-consumed unsupported values fail before recipe publication.
`info.viewing_illuminant`	bounded string -> `IlluminantKey`	scanner/viewing illuminant; spektrafilm default is `D50`	Optional unless scanner route consumes it	Loader records defaulted status; route-consumed unsupported values fail before recipe publication.
`info.target_print`	scalar profile key	intended print target	Optional metadata	Not used as print menu filter unless a later approved feature says so.
`info.log_sensitivity_density_over_min`	scalar float	profile-visible metadata for sensitivity interpretation	Optional unless a future consumer requires it	If absent, default is spektrafilm `0.2`. Accepted/profile-visible but unconsumed by this render port unless a phase names a consumer. No output effect and no direct hash participation except through raw asset/version identity.
Legacy `info.fitted_cmy_midscale_neutral_density`, `info.log_exposure_midscale_neutral`	legacy upstream keys only	unsupported legacy metadata	Not consumed by current spektrafilm plan	Current spektrafilm strips these keys before constructing `ProfileInfo`; Film-Juicer must not depend on them for print normalization or neutral calibration.
`data.wavelengths`	`[81]`	nm	Required when selected profile spectral samples are consumed	Must match 380-780 nm at 5 nm; no resampling of profile samples. Values are finite numeric only; `null`, `NaN`, `Inf`, missing samples, duplicate samples, or off-axis samples fail selected-profile loading only for selected consumed spectral data. This exact-axis check is Film-Juicer GPU hardening over spektrafilm Python's shape-only validation, not an invitation to add broad bundle linting or to resample authored profile samples.
`data.log_sensitivity`	`[81][3]` RGB-sensitive exposure order: R-sensitive/C-density, G-sensitive/M-density, B-sensitive/Y-density	log sensitivity	Required for capture-film and print routes	JSON `null`, bare/numeric `NaN`, and string `"NaN"` are stored as NaN in the authored log domain, then the consumer computes `10 ** log_sensitivity` and applies spektrafilm-equivalent `nan_to_num`. `Inf` fails selected-profile loading as an intentional hardening deviation. Do not apply density-to-light NaN semantics here.
`data.hanatos2025_adaptation_window_params`	`[4]` finite vector	Hanatos spectral band-pass window parameters	Required for `RgbToRawMethod::Hanatos2025` when `apply_hanatos2025_adaptation_window=true`	Missing, empty, malformed, or non-finite required window params fail before recipe publication as `MissingRequiredResource` or `MalformedRequiredProfileData`. Do not synthesize a no-op window, switch methods, disable adaptation, invent `data.bandpass_hanatos2025`, or apply a legacy spectral multiplier.
`data.hanatos2025_adaptation_surface_params`	`[3][15]` finite channel-major matrix when consumed; empty `[]` allowed only while surface adaptation is disabled	Hanatos log-exposure correction surface parameters	Required only when `apply_hanatos2025_adaptation_surface=true`	Missing, empty, malformed, or non-finite required surface params fail before recipe publication as `MissingRequiredResource` or `MalformedRequiredProfileData`.
`data.channel_density`	`[81][3]` CMY	spectral density	Required for selected capture-film and print profiles	Convert once if internal storage differs. Preserve finite values, finite negatives, JSON `null`, bare/numeric `NaN`, and string `"NaN"` through ingestion, digesting,

recipe construction, hashing, and GPU upload. No `nan_to_num`, no finite-count completeness gate, no CSV/folder substitution. NaN becomes zero transmitted light only at `density_to_light`. `Inf`, malformed rows, missing samples, and wrong axis fail selected-profile loading. |
`data.base_density`	`[81]`	spectral density	Required for selected capture-film and print profiles	Same density-like policy as `data.channel_density`. Preserve NaN/null until `density_to_light`; no fake baseline fallback and no zero-density substitution. `Inf`, malformed shape, missing samples, and wrong axis fail selected-profile loading.
`data.midscale_neutral_density`	`[81]` when inspected	spectral density	Optional copied-bundle inventory/reference field; not consumed by current product routes	Do not require this field for selected direct or print routes. Preserve density-like `null`/`NaN` semantics only if a non-render inventory/diagnostic path stores it, and exclude it from render recipes, output-affecting hashes, resource descriptors, GPU uploads, neutral-filter calibration, print normalization, and final sensitivity unless a future phase names a current consumer. Missing or malformed values do not fail selected-profile loading; they fail only an explicit inventory/diagnostic command that chose to inspect this unconsumed field.
`data.log_exposure`	`[N]`	log exposure axis	Required for selected profiles	Finite numeric and non-decreasing enough for spektrafilm interpolation. Repeated samples are allowed and follow spektrafilm `fast_interp`/`np.interp`-style right-biased exact-match semantics. Missing, `null`, `NaN`, `Inf`, decreasing samples, or length mismatch with density arrays fail selected-profile loading. Interpolation clamps out-of-range exposure queries to endpoint densities; loader must not sort, deduplicate, clamp, synthesize, or repair the authored axis. If future owners want strict-increasing rejection, record it as intentional hardening divergence with source/bundle evidence.
`data.density_curves_model`	current spektrafilm `DensityCurvesModel` payload when present	profile-visible density-curve model data	Optional unless a future phase names a consumer	If absent, default to the reference dataclass's empty model. Accepted/profile-visible but unconsumed by this render port unless a phase names a consumer. No output effect and no direct hash participation except through raw asset/version identity.
`data.density_curves`	`[N][3]` CMY density channels aligned by index to R/G/B-sensitive raw exposures unless schema lock proves otherwise	density over log exposure	Required for selected capture-film and print profiles	Density-like policy. Preserve finite values, finite negatives, JSON `null`, bare/numeric `NaN`, and string `"NaN"` in authored curves. Full film develop uses a normalized copy `authoredDensityCurves - nanmin(authoredDensityCurves, axis=0)` before DIR/grain; print develop and print-balance midgray develop use authored curves unchanged. Interpolation preserves channel index from raw exposure to density, does not mask NaNs, and clamps out-of-range queries to endpoint density values. Bounds use `nanmin`/`nanmax`. `Inf`, malformed rows, missing samples, wrong channel count, or length mismatch with `data.log_exposure` fail selected-profile loading.
`data.density_curves_layers`	`[N][3 sublayers][3 channels]`, same `N` as `data.log_exposure`	density over log exposure	Optional unless grain/layer route requires it	When unconsumed, do not lint it. When consumed, preserve finite values, finite negatives, JSON `null`, bare/numeric `NaN`, and string `"NaN"` through layer interpolation and CUDA upload. Positive-film layer interpolation uses the spektrafilm negated-density/negated-curve path; negative-film layer interpolation uses authored density values. `n_sub_layers` does not vary this authored profile

shape or upload shape; it is a spektrafilm grain-parameter field, not a schema dimension for `density_curves_layers`. `Inf`, malformed required layer data, wrong shape, or length mismatch fail only when selected behavior requires layers. |
| `neutral_print_filters.json` selected entry | object lookup `[print stock][enlarger illuminant][film stock] -> [C, M, Y]` | Kodak CC filter units, CMY order | Optional calibration looked up only for selected print routes when database neutral filters are enabled | Missing file or missing selected entry preserves the current neutral C/M/Y values and records missing-calibration status; missing entries do not hide profiles. In a fresh/default print recipe the preserved values are spektrafilm defaults `C=0`, `M=65`, `Y=55`, but authored/manual current values must survive a DB miss. Found selected entries must be three finite numeric CC values and are not normalized/clamped to `[0,1]`. `null`, `NaN`, `Inf`, malformed arrays, wrong order, or old normalized Y/M/C units fail selected print recipe digest/preparation with the neutral-calibration identity. Direct routes do not perform selected-entry lookup or neutral-calibration preflight, and direct-route hashes/resources/status exclude neutral values entirely. |
| Selected measured dichroic filter CSVs | one finite wavelength/transmission CSV per C/M/Y filter in selected measured dichroic set | percent transmittance loaded as `value / 100`, Akima-resampled to profile axis | Required only when selected print/filter route consumes a measured extension set | Missing or malformed selected CSVs fail resource preparation. Do not fall back to another dichroic set, the reference/custom analytic model, 170-step wheel semantics, or old profile-folder filter data unless a phase records an output-identical descriptor-local strategy. |

Loaded spektrafilm profile payloads store selected required fixed-size sample data as fixed-axis host containers or equivalently bounded arrays: consumed spectral fields use the canonical 81-sample axis and consumed density-curve fields use their shape-checked curve axis/extent. Authoring-friendly vector-of-pairs, JSON row objects, or resampling scratch may exist only inside the loader bridge and must not become the loaded profile payload, `RenderRecipe`, resource descriptor, hash input shape, or CUDA packing source.

Channel And Unit Boundaries

Boundary field	Upstream order/unit	Internal owner	Rule
`data.log_sensitivity`	RGB-sensitive exposure order	`FilmRawRecipe` / sensitivity derivation	Convert once at profile ingest if internal storage differs.
raw exposure / density triplet index	index 0/1/2 = R-sensitive->C, G-sensitive->M, B-sensitive->Y	`FilmRawRecipe` -> `FilmDevelopRecipe` -> spectral conversion	Preserve the index across raw exposure, density interpolation, layer interpolation, `channel_density`, scanner/enlarger bounds, LUT generation, and CUDA payload packing. Use asymmetric sentinel values in tests; do not infer B/G/R from old Film-Juicer field names.
`data.hanatos2025_adaptation_window_params`	`[4]` finite window parameter vector	`FilmRawRecipe` / Hanatos LUT descriptor	Required for Hanatos film raw when window adaptation is enabled; it is not a sensitivity array. Missing, empty, malformed, or non-finite

required window params fail before recipe publication. Do not invent `data.bandpass_hanatos2025` or a legacy spectral multiplier. |
`data.hanatos2025_adaptation_surface_params`	`[3][15]` finite channel-major surface parameter matrix when surface adaptation is enabled; empty `[]` only while disabled	`FilmRawRecipe` / Hanatos LUT descriptor	Required only when surface adaptation is enabled; surface correction multiplies the Hanatos raw LUT by `2^surface`. Missing, empty, malformed, or non-finite required surface params fail before recipe publication.
`data.channel_density`	CMY spectral density	profile payload -> print-exposure and scanner spectral conversion resources	Convert once; not consumed by `FilmDevelopRecipe`. Scanner/enlarger table code consumes typed CMY to compute spectral density.
`data.base_density`	spectral 81 samples	profile payload -> print-exposure, scanner spectral conversion, and black/white correction math	Required for selected spektrafilm profiles; not a `DensityBoundsRecipe` input.
`data.midscale_neutral_density`	spectral 81 samples	selected-profile schema payload	Required for selected-profile schema parity, render-output-inert, and not neutral-filter calibration.
`data.density_curves_model`	current spektrafilm model payload	profile payload	Accepted/profile-visible but unconsumed unless a phase names a consumer.
`data.density_curves`	CMY over log exposure, same triplet index as raw exposure	`FilmDevelopRecipe`	Preserve authored extrema; store normalized curves separately.
`data.density_curves_layers`	layered density curves	`GrainContract`	Positive/negative interpolation is typed.
`density_min` contract	CMY density lower bound	`GrainContract` -> `DensityBoundsRecipe`	Not owned by visual grain controls.
Dichroic filter set	reference `custom`, plus Film-Juicer `durst_digital_light`, `thorlabs`, `edmund_optics`	`PrintRecipe`	`custom` is the spektrafilm reference baseline; measured Film-Juicer sets are typed extension-mode resources, not a legacy wheel model.
Dichroic filter resources/models	reference/custom analytic erf model, or measured `Resources/filters/dichroics/<set>/filter_{c,m,y}.csv`; C/M/Y channel order; CSV values are percent transmittance	Process-owned filter assets/model constants -> `PrintRecipe` / `PrintIlluminantDescriptor`	The reference/custom default uses spektrafilm's fixed `edges=[516,500,610,607]` and `transitions=[12,8,8,8]` analytic model on the canonical axis. Measured sets divide CSV values by 100 before use, unique duplicate wavelength rows by first occurrence, and resample with Akima interpolation onto the canonical 380-780 nm, 5 nm axis. Missing selected measured filters, malformed values, non-monotonic usable wavelengths after unique handling, or out-of-domain resampling failures abort recipe/resource preparation with a named diagnostic. Descriptor hashes include filter-set key, model-vs-CSV source identity, channel order, interpolation/model method, value unit when applicable, canonical axis identity, and descriptor schema version.
Kodak filters	C/M/Y CC units	`PrintRecipe`	UI may expose Y/M/C; convert once to C/M/Y. Do not store as old normalized 170-step values.
scanner ranges	CMY density bounds	`DensityBoundsRecipe`	Use prepared `dataMinCmy`, `dataMaxCmy`, `invSpanCmy`; no alternate range owners.

scanner density input	normalized C/M/Y density axes	`ScannerSpectralLutDescriptor` / scanner CUDA payload	Normalize once from typed CMY density bounds before LUT lookup or spectral scanner evaluation. LUT input axis 0 is C, axis 1 is M, axis 2 is Y unless a descriptor schema version explicitly changes it.
scanner LUT reference semantics	C/M/Y semantic axes with X/Y/Z `log10` output triplets	`ScannerSpectralLutDescriptor` / scanner LUT resource	Spektrafilm builds LUT meshes with semantic axes C, M, then Y and scanner source values in `log10(XYZ)`. The 3D LUT path uses endpoint-inclusive grids, normalized lookup coordinates, and default `apply_lut_3d(method='pchip')` interpolation with prepared per-axis Hermite slopes and per-cell min/max clamping. This is the behavior contract even if GPU storage is rearranged for coalescing.
scanner LUT GPU storage	implementation-defined, descriptor-versioned storage layout	`ScannerSpectralLutDescriptor` / scanner LUT resource	Current Film-Juicer may keep Y-density-major / M-density / C-density-minor storage and may store `log2(XYZ)` only as a versioned storage detail. The descriptor must record semantic axis order, storage axis order, interpolation convention, stored output triplet order, log base/value domain, numeric format, and schema version, and builder/sampler fixtures must prove equivalence to the C/M/Y `log10(XYZ)` plus PCHIP/clamped interpolation contract. Existing Mitchell-cubic/trilinear/bilinear sampler behavior is not the reference contract.
scanner spectral output	reference `log10(XYZ)` in X/Y/Z order	scanner conversion	Convert from the stored log domain to XYZ exactly once before glare/color adaptation/output matrix work. Do not reinterpret scanner LUT output as RGB or channel-density order.
scanner/output RGB	linear RGB in R/G/B order before output CCTF	`ScannerOutputRecipe` and CUDA payload	Scanner color runtime owns CAT/output matrix, output color space, output CCTF, post-LUT correction, glare, blur, unsharp, and final clipping/encoding order. These do not enter `ScannerSpectralLutDescriptor` identity unless the LUT stored value domain changes.

Use named domain types where they prevent swaps:
- `RgbTriplet`
- `CmyTriplet`
- `DensityCmy`
- `CmyCcTriplet`
- `LogXyzTriplet`
- `XyzTriplet`
- `LinearRgbTriplet`
- `EncodedRgbTriplet`
- `Micrometers`
- `Pixels`
- `AutoExposureMethod { CenterWeighted, Average, Median, Partial, Matrix, MultiZone, HighlightWeighted }`

Do not introduce unused wrappers as Phase 1 scaffolding, but the scanner/output boundary is a demonstrated swap risk. The first phase that implements scanner LUT packing, scanner XYZ correction, glare, scanner blur/unsharp, output matrix/CAT, CCTF encoding, or final output

packing must use the scanner color-domain wrappers above at its host recipe, descriptor, packer, validation, and diagnostics boundaries. CUDA kernels and hot loops may convert once to plain POD after the boundary preserves the value domain and triplet order.

Diagnostics

Use one compact result shape only at coarse host boundaries where a value plus failure detail must cross a boundary:
- profile/catalog loading
- selected-profile/resource preflight for consumed fields
- recipe build/preflight
- resource preparation/admission

```text
StatusResult<T>
  value
  status
  diagnostic
```

Minimal diagnostic identity contract:
- Preferred first home is a small `SpektrafilmDiagnostics.*` or nearest-boundary helper file introduced only when two coarse host boundaries need the same shape. A single-boundary phase may keep the type local, but it must still use the code strings below so cleanup gates and manifests can find the same failures.
- Every boundary diagnostic has a failure class from the taxonomy below plus one searchable diagnostic code string. The code string is the cleanup/search identity; the failure class is the policy category.
- Known diagnostic code strings for planned gates include `SpektrafilmPixelPipelineNotImplementedForPhase1A`, `MissingRequiredHostParameter`, `MalformedRequiredProfileData`, `MissingOptionalCalibration`, `MissingRequiredResource`, `UnsupportedMode`, `UnsupportedSpektrafilmIoTransform`, `DirNotImplementedForPhase3`, `ScannerPostEffectsNotImplementedForPhase3`, `ScannerPostEffectsNotImplementedForPhase4`, `GlareNotImplementedForPhase4`, `CudaUnavailable`, `NonCudaRenderPathInvoked`, `ResourceDescriptorMismatch`, `ScratchAdmissionFailure`, `DurableResourceAdmissionFailure`, `ExactOpticsNotImplementedForPhase6`, `ExactAdmissionFailure`, `ValidationOnlyMissingHook`, and `FinalRenderExactnessFailure`.
- Introduce each diagnostic code only when a phase first emits that distinct product failure, cleanup gate, or owner-run evidence artifact. Do not create placeholder constants, phase-local class hierarchies, or a repo-wide diagnostic framework just to host planned names.
- Temporary phase/blocking diagnostics are cleanup targets, not permanent product API. When the blocked behavior is implemented, hidden, deleted, or replaced by a more specific product failure, remove the temporary diagnostic code, preflight branch, fixture expectation, cleanup

allowlist entry, and manifest requirement unless the phase manifest records a still-useful current product failure with owner and retention reason.
- Detailed diagnostic formatting is gated. Normal Release behavior may carry compact status, failure class, code, hashes, and cheap identity fields; expensive strings, profile dumps, resource inventories, and fixture evidence are emitted only through explicit diagnostics/evidence paths.

Do not wrap leaf math helpers, hash helpers, payload packers, hot-path render code, or local validation helpers in generic result plumbing. Use plain return values, `bool` with `outError`, or small local enums.

Keep result helpers local until repeated boundary code proves a shared helper is simpler. Do not create a repo-wide error/result framework as Phase 1 scaffolding. User-profile debug logs, if added, must be behind an explicit diagnostics gate and should report only profile key/path, field, expected shape/domain, and short detail.

Diagnostic fields are optional per failure. Include only fields that are relevant to the boundary and cheap enough for the active diagnostics level; do not force every diagnostic into a universal record.

Possible diagnostic fields. This is a menu, not a required universal record; each failure path chooses the smallest useful subset:
- OFX param name for current-contract host-param failures
- profile key
- profile role/support/stage/polarity
- route
- phase/subphase
- host action when relevant
- stage/domain
- backend when relevant
- field path for schema failures
- recipe hash
- resource hash
- resource key/path/descriptor identity when relevant
- CUDA device id, context identity, and context epoch when relevant
- request/frame/session/clip/source token identity when relevant
- optional-data status
- execution strategy
- failure class
- short detail string

Do not create a diagnostic class hierarchy per phase, route, or recipe. The failure taxonomy below is a shared label menu for coarse boundary diagnostics; it is not a request for generated types, per-phase subclasses, visitor plumbing, or a repository-wide error framework.

Delete or hide old telemetry-style diagnostics when the owning behavior moves: neutral-filter live reload status, selected-to-default fallback notes, private resource fallback trace records as product behavior, old saved-project interpretation details, and verbose profile metadata reports whose only purpose is legacy/migration support.

Release builds should pay for status fields and hashes, not expensive string formatting or large diagnostic collection. Format detailed traces only behind diagnostics gates.

Failure Taxonomy

Use the same failure classes throughout the port:

Failure class	Example	Policy
`MissingRequiredHostParameter`	a current spektrafilm instance lacks `FilmProfileKey`, `PrintProfileKey`, route, or backend params required by the descriptor	Hard fail before recipe publication; include param name, effect contract, route if known, phase, and host action. This class applies only to current spektrafilm params; removed old params are ignored, not diagnosed.
`MalformedRequiredProfileData`	selected profile has wrong `data.channel_density` shape	Hard fail before recipe publication; include profile key, path, field, expected shape/unit, actual shape/type when cheap, route, and phase.
`MissingOptionalCalibration`	no neutral calibration entry for selected film/paper/illuminant	Continue only when this plan names an explicit optional-calibration default/manual C/M/Y recipe value; emit a compact gated diagnostic and mark `optionalCalibrationDefaultApplied=true`. No per-dichroic-set calibration dimension exists in this port.
`MissingRequiredResource`	selected measured dichroic CSV or required profile payload missing	Preflight/render hard fail; include resource path/key, route, backend, and phase. The reference/custom analytic dichroic model has no CSV fallback requirement.
`UnsupportedMode`	selected route/profile polarity combination has no implemented backend in an intermediate phase	Hard fail through a named phase-local diagnostic; include removal/implementation acceptance item.
`CudaUnavailable`	production render requested without a usable CUDA backend/context	Hard fail before output; include route, backend, host action, and phase.
`NonCudaRenderPathInvoked`	host invokes a CPU pixel-render path for spektrafilm product output	Hard fail before product spektrafilm pixels are produced; include route when known and phase. Reference-only CPU harness use is not this failure class.
`ResourceDescriptorMismatch`	prepared resource identity does not match the current recipe/frame descriptor	Hard fail before launch; include descriptor family, expected hash, actual hash when known, route, backend, and phase.
`ScratchAdmissionFailure`	required frame scratch cannot be admitted for the current descriptor	Hard fail or use only an explicitly output-stable strategy; include route, domain, backend, descriptor hash, required buffer role, requested bytes when known, and resource identity.

`DurableResourceAdmissionFailure`	context-owned durable resource cannot be admitted or retained	Hard fail unless the plan names an output-stable strategy for that family; include descriptor family, resource identity, context identity, requested bytes when known, and backend status.
`ExactAdmissionFailure`	exact optics requested but workspace/plan/cache cannot be admitted	Hard fail when Exact is selected; no lower-cost backend downgrade. Include domain, route, descriptor hash, requested bytes when known, backend status, and resource identity.
`ValidationOnlyMissingHook`	fixture comparator or host evidence unavailable on a developer machine	May skip only with explicit manifest reason; production behavior must not silently downgrade.

Output-stable admission choices such as sequential-channel exact execution, explicit temporary-plane aliasing, or narrower workspace reuse are status/diagnostic fields, not failure classes. They must never change output and should be recorded only when useful for debugging admission behavior.

Every non-fatal default entry in a phase manifest must state whether output changes and why it is spektrafilm-compatible or explicitly approved by this plan. Final renders fail clearly instead of silently downgrading whenever exact behavior is required.

Resource Ownership And Descriptor Rules

This section is an architecture contract, not a benchmark plan, and it is the spektrafilm-specific resource slice of `docs/rm/resource-manager-simplification-plan.md`. It forbids ambiguous ownership, hidden resource lookup, and unplanned allocation paths. It does not require numeric performance or VRAM evidence gates.

1. Build immutable profile assets once per process when selected or needed.
2. Build `RenderRecipe` host-side without requiring a CUDA context.
3. Build `FrameRequest` from one stable recipe snapshot.
4. Prepare all required GPU resources and scratch through `PreparedCudaFrame`.
5. Render only from prepared handles.
6. Finish or abort explicitly.

Resource keys:
- recipe hashes include output-affecting validated inputs
- recipe hashes are built only from resolved recipe fields and asset/version tokens
- an asset/version token is valid descriptor identity only when it uniquely represents the selected parsed payload fields consumed by that recipe/resource family in the current process; profile key, menu index, folder name, first-entry position, or a constant process asset version is not sufficient identity for consumed spectral/profile data
- when a resource consumes authored profile arrays or copied bundle resources, the owning asset loader must provide a stable payload identity such as a content digest, owner-supplied source digest, or monotonic load-generation token tied to that exact parsed payload. The

descriptor/hash then consumes that payload identity instead of re-reading files or hashing packed CUDA bytes
- hash builders are pure functions over canonicalized recipe/subrecipe fields, asset/version tokens, and descriptor fields only
- hash builders must not perform catalog lookup, file discovery, or `root().assets()` calls
- hash builders must not invoke payload packers, hash packed struct bytes or padding, observe upload order, depend on GPU pointers/handles, depend on cache-hit or admission results, depend on stream identity, or include unrelated output/post-effect settings. Durable resource keys may include `DeviceContextKey` and context epoch only through the descriptor/key path that explicitly owns durable residency.
- GPU execution keys include final execution descriptor fields only
- durable resources include `DeviceContextKey` and context epoch
- frame scratch descriptors include requested route/domain/backend, extent, buffer roles, lifetime/alias group, required/optional role, estimated bytes, and shape-affecting execution constraints
- selected pressure/fallback strategies are prepared-frame admission results and diagnostics, not descriptor identity, unless they change requested resource shape, precision, aliasing, or launch-visible descriptor fields
- disabled/off features produce no durable descriptor, no scratch role, no hash contribution, and no resource lookup
- immutable GPU resources are built or uploaded only on descriptor cache miss, descriptor/schema/version identity change, or context/epoch invalidation. Steady-state preparation binds descriptor-matched resources and must not repack, reupload, or rebuild them to compute identity
- Prepared-frame ownership is necessary but not sufficient to approve allocation churn. Accepted steady-state CUDA pixel routes must not allocate/free raw CUDA device memory, pinned host memory, CUDA events, or transient staging every frame for a stable descriptor. Each frame-local scratch/staging family must either bind an admitted/reused prepared-frame lease or carry a family contract that proves the resource is always-present, bounded, not output-affecting, and not part of a same-frame GPU-to-CPU synchronization dependency.
- descriptors are plain, closed structs owned by one resource or scratch family; they are not open option bags, feature-policy containers, or generic dispatch payloads

Descriptor rules:
- Canonicalize descriptors before lookup.
- Cache only immutable host derivations that have a stable descriptor identity and are worth sharing as part of the normal architecture.
- Durable resources are keyed by full descriptor identity plus `DeviceContextKey` and context epoch.
- Reusable kernels, LUTs, PSFs, frequency resources, and exact resources must have named descriptor families rather than being hidden inside unrelated hash lanes.
- Frame scratch is described by role and lifetime before it is admitted or leased.
- Workspace requests are route/domain/backend/buffer/byte descriptors, not an expanding set of booleans.

- Do not add descriptors that contain `std::any`, `void*`, string option maps, generic property maps, unrelated optional fields, or long unrelated `needX` boolean lists.
- Current bool-list shapes such as `PreparedCudaFrame::WorkspaceRequest` and `ScratchRequestDescriptor` are cleanup targets. New spektrafilm phases must not extend `needOptics`, `needSpatialDir`, `needBlurred`, `needAux`, `needGrainTriplet`, `needGrainShared`, `needGateMask`, or similar flags as the destination design; replace touched behavior with named scratch-role descriptors.
- Zero-cost disabled-feature evidence must prove descriptor absence at the owning boundary: no durable descriptor, no scratch role, no static-noise/resource request, no prepare call, no kernel work, and no output-affecting hash contribution for the disabled feature except any required recipe identity.
- Prepared-frame preparation is descriptor-driven. Do not grow the public surface with one method per effect unless the method is an owner-approved temporary bridge.
- Do not add device-wide schedulers, allocators, graph systems, or cache layers in this plan. Defer that work to a future optimization plan if the small descriptor-driven design proves insufficient.

Resource Descriptors

Do not introduce a permanent global resource registry as Phase 1 scaffolding. Each implemented resource family owns its own small descriptor contract when the family first lands.

The existing `ResourceKind` / `ResourceKindContract` / `kResourceKindContract` table in `Cuda/ResourceManager/JuicerCudaResourceCore.h` is current resource-manager scaffolding, not the spektrafilm destination architecture. Existing ResourceManager internals remain the low-level context-owned residency, admission, in-flight accounting, retirement, and reset mechanism. Existing private fallback paths may survive only as named legacy internals until the owning descriptor family is replaced; accepted spektrafilm scanner, print, DIR, optics, grain, and auto-exposure resource families must not use private LUT/resource fallback as normal product behavior. New spektrafilm descriptor families must hard-fail on admission failure unless this plan explicitly names an output-identical, descriptor-local strategy with a cleanup checkpoint. Spektrafilm resource families own descriptor structs, hash builders, validation policy, source asset identity, and route-specific defaults. ResourceManager tables receive only the minimal mechanical hooks needed to admit and retire a concrete resource family. ResourceManager tables must not gain schema interpretation, default selection, route policy, channel-order conversion, profile fallback, scanner correction policy, print-filter semantics, pressure policy layers, or compatibility fallback decisions.

Each resource family owns:
- one descriptor struct or equivalent plain-data key
- canonicalization rules where needed
- one hash helper

- a descriptor schema/version token when layout, stored value domain, numeric format, interpolation convention, channel order, or generated-table semantics could otherwise make two incompatible resources share one key
- the consumed `RenderRecipe` group/subrecipe fields and asset/version tokens
- context identity fields when the resource is durable GPU residency
- prepared-frame view/lease shape when render code consumes the prepared result

Each resource descriptor remains minimal: it contains only canonical identity/admission fields, layout/version tokens needed to prevent incompatible reuse, and context identity when the resource is durable GPU residency. Do not make descriptors carry producer/consumer explanations, diagnostics, disabled-path prose, or broad policy state. Those belong in the family contract and phase evidence.

Each new or materially changed CUDA payload, durable resource, and scratch/staging family must also have a compact source-adjacent family contract in the implementing change. Prefer a short comment beside the descriptor/payload definition or a small manifest note that links to that source location; do not create a standing appendix, registry, or policy table unless the code already has multiple active consumers and the source-adjacent note is no longer simpler. "Materially changed" includes new fields, changed units/channel order, changed descriptor identity, changed packing, changed lifetime/ownership, changed disabled behavior, changed preparation/upload point, or a new kernel/resource consumer for an existing family.

The minimum family contract names:
- producer
- consumer
- update frequency
- host layout
- device layout
- ownership/lifetime
- descriptor/hash identity, including the descriptor schema/version source when applicable
- included hash inputs
- intentionally excluded hash inputs
- upload/preparation point
- disabled/off behavior

Add kernel access pattern, payload packing rationale, scratch byte estimates, admission-pressure behavior, or micro-level launch/storage details only when that detail affects correctness or prevents an accidental slow path for the family being implemented. These details are review evidence for the current resource, not a runtime registry or generic framework.

For optional or gated CUDA families, the family contract must include disabled-path evidence showing no descriptor creation, hash participation, lookup, allocation, upload, scratch/staging admission, or kernel launch when the feature is disabled or blocked by phase admission.

If disabled/off behavior is not applicable to an always-present diagnostic, completion, or lifetime-control family, the family contract must say so explicitly and name why the resource is required, what keeps it frame/prepared-frame owned, why its byte/event footprint is bounded, why it does not affect output identity, and why any host-visible result is asynchronous or otherwise outside same-frame product correctness.

Family contracts are added only as families land. Do not pre-fill contracts for future optics, scanner, grain, or scratch work as Phase 1/2 scaffolding, and do not mirror family contracts into runtime descriptor registries or policy tables.

CUDA payload fields are accepted only when they have a named kernel consumer, unit/channel order, update frequency, constness, and packing/alignment reason. Device payload additions or material layout changes must be covered by the family contract for that phase. Device payloads are compact kernel contracts, not mirrors of host recipe, profile, runtime, or `WorkingState` structs; no device payload field is accepted merely because the source field exists on the host side.

The current broad `PipelineRunParams` shape is a temporary launch bridge, not the final payload contract for spektrafilm behavior. Before a phase accepts direct or print CUDA pixel behavior that touches a stage, that phase must either replace the touched portion with route/stage-specific payload packers or record a source-adjacent `SF_TEMP_BRIDGE_PipelineRunParams` ledger proving inactive-stage fields are not populated, copied into device payloads, used for descriptor/hash/resource requests, or allowed to allocate/upload/launch disabled features. Phase cleanup rejects an accepted route whose real contract is still "fill the whole monolithic run struct and let later code ignore fields."

Expected first-class family contracts include, as they land:
- film sensitivity/profile tables and Hanatos filming TC LUTs
- density curves and density layer curves
- `ScannerSpectralLutDescriptor` resources and separate scanner post-effect descriptors
- print filtered illuminant and print curve/filter resources
- DIR payloads, kernels, and spatial DIR scratch
- exact optics PSFs, frequency-domain resources, cuFFT plan/work resources, and mutable exact work buffers
- grain static assets, visual grain resource bindings, and density-layer grain bindings
- auto-exposure metering buffers
- scan-error staging and transient event resources

LUT execution policy:
- Film-Juicer production GPU rendering uses spectral LUT acceleration for scanner conversion when the LUT represents the same function as the spektrafilm spectral scanner path.
- Film-Juicer production print exposure uses enlarger LUT acceleration when the implemented LUT represents the same function as the spektrafilm spectral enlarger path.
- LUT acceleration is an execution strategy, not a spektrafilm behavior toggle.

- Spektrafilm's Python runtime defaults `settings.use_scanner_lut=false` and `settings.use_enlarger_lut=false`; this port's GPU production policy intentionally differs at the execution layer only. Pixel behavior still follows the same scanner/enlarger spectral functions.
- Production scanner/enlarger LUT generation and sampling must match spektrafilm's LUT semantics: endpoint-inclusive grids from `xmin`/`xmax`, normalized lookup coordinates, and default `apply_lut_3d` PCHIP interpolation with per-cell min/max clamping, unless an owner-approved approximation boundary records the replacement sampler, fixture evidence, and descriptor/schema-version impact.
- `lut_resolution` remains user-controlled through Film-Juicer's advanced scanner/math controls. Default resolution is `17`; valid production range is `17-128` unless the existing ResourceManager clamp changes by approved product decision.
- `ScannerUseLUT` is not a normal behavior control in the spektrafilm-era UI contract. The no-LUT path exists only as validation/debug support for isolating spectral math and must emit clear diagnostics when used.
- Reference evidence for production GPU output records LUT resolution. Production comparisons use matched-resolution LUT paths. Spectral/no-LUT comparisons are validation-only math checks.
- Descriptor identity includes LUT resolution and every LUT-generation input. Scanner post effects are excluded.
- CPU construction of scanner/enlarger LUT tables is allowed only as host-side resource derivation for the GPU path. Such builders must consume the canonical descriptor plus loaded profile/resource payload identity, must emit table data plus diagnostics only, and must not become a callable scanner/enlarger renderer for OFX pixels, broad validation, or route fallback. A retained broad-runtime builder is a temporary bridge, not the semantic owner.
- Spektrafilm's Python services lazily build LUTs on first use, but Film-Juicer's product renderer must treat first use as descriptor admission/preparation work. A descriptor miss may build/upload the table before launch with diagnostics; a descriptor hit must not run CPU spectral calculations, broad runtime reconstruction, sentinel probes, or hidden validation checks in the render hot path.

Canonical `ScannerSpectralLutDescriptor` identity:
- Includes `route`.
- Includes scanned medium kind: film or print.
- Includes polarity.
- Includes `DensityBoundsRecipe` identity.
- Includes scanned profile channel-density identity.
- Includes scanned profile base-density identity.
- Includes scan illuminant identity.
- Includes observer/CMF identity.
- Includes XYZ normalization convention.
- Includes LUT resolution.
- Includes interpolation convention.
- Includes reference semantic input-axis order.
- Includes storage input-axis order.

- Includes stored value domain, log base, and numeric format.
- Includes stored output triplet order.
- Includes descriptor schema version.
- Excludes output color space.
- Excludes output CCTF encoding.
- Excludes black/white correction.
- Excludes glare.
- Excludes scanner lens blur.
- Excludes scanner unsharp mask.
- Excludes frame bounds.
- Excludes auto-exposure.
- Excludes post-LUT route correction identity.

No spektrafilm resource may be hidden inside an old generic hash lane such as scanner/upload/DIR identity. If a legacy lane is temporarily reused, the bridge manifest must name the lane, the consumed recipe fields, and the removal phase.

A documentation table of resource families is useful, but it is not a runtime registry, broad policy layer, or Phase 1 acceptance blocker. Add family contracts as the implementing phase creates real descriptors.

FrameScratchDescriptor

Frame scratch requests use a named descriptor shape, not feature booleans.

Do not require the complete scratch descriptor shape before a phase has a real scratch consumer. Introduce or extend the descriptor in the first phase that needs new mutable scratch, staging, pinned host memory, scan-error state, or transient event ownership.

`FrameScratchDescriptor` is not a universal optional-field bag. Use small descriptor structs per scratch family when fields differ materially.

A scratch descriptor must describe one scratch family or one tightly related alias group. If two consumers do not share allocation shape, lifetime, aliasing, and preparation failure policy, they should not share a descriptor just to reduce type count.

Required fields where applicable:
- route
- domain
- backend
- frame extent
- precision/storage format
- buffer roles
- lifetime/alias group

- required or optional role
- estimated bytes
- shape-affecting execution constraints, if any

Admission decisions may record the selected output-stable strategy, but scratch descriptors must not become policy objects.

All mutable scratch, staging, scan-error, auto-exposure, pinned host, and transient event resources are owned by the prepared frame or by a lease retained by the prepared frame. Resource-manager internals may pool or retain them, but render code consumes only prepared-frame views and completion/abort ownership.

Any retained upload staging owned by resource-manager internals must be bounded, context/epoch keyed, and visible in the owning family contract. It must not become a hidden output-affecting intermediate or a second frame-local workspace.

Exact-Mode Optics

Exact optics is the spektrafilm-matching backend of `SpatialOptics`, not a parallel render pipeline. It is intended to match spektrafilm optics behavior, including PSF/FFT behavior where the reference uses it. Lower precision, CUDA execution order, separable implementation details, and other GPU optimizations are accepted only when the phase evidence shows they pass visual inspection under matched route/profile/scanner/output settings. Exact must not use approximation kernels as substitutes for spektrafilm optics behavior.

Interface

`SpatialOptics` emits recipe-level exact component data for enabled components only. During frame preparation, the exact resource family combines that recipe data with immutable `FrameRequest` shape facts to build canonical execution/resource/scratch descriptors. Each execution descriptor identifies:
- domain
- route
- effect type
- final sampled PSF identity
- `FrameRequest::pixelSizeUm` where identity depends on physical scale
- frame extent where identity depends on extent
- padded FFT dimensions
- precision
- channel grouping or sequential-channel strategy
- normalization policy
- descriptor hash

CUDA/resource code consumes these descriptors. It does not decide whether exact is allowed, whether another backend can replace Exact, or whether the effect is active.

`RenderRecipe` must not absorb frame-local extent, padded FFT dimensions, cuFFT work area sizing, or final per-frame execution hashes. Those are preparation outputs derived from recipe plus `FrameRequest`; durable descriptor-stable artifacts may be retained context-locally when their canonical descriptor identity matches.

First Implementation Shape

Exact optics is first-class in this plan, but the first implementation should be correct before it grows cache and pooling machinery. Use the smallest complete exact implementation:
- one host `SpatialOptics` state
- exact recipe data owned by the recipe/subrecipe, with frame-derived execution/resource descriptors built during prepared-frame preparation
- admission before launch through the prepared-frame/context-owned boundary
- prepared-frame-owned mutable exact buffers and leases
- mutable exact work is prepared-frame-owned
- descriptor-stable immutable exact artifacts may be context-owned when reuse prevents avoidable churn
- one-shot immutable exact artifacts require a family-contract justification that explains why per-frame construction/upload/planning is not avoidable or not worth retaining
- one prepared-frame plan/workspace lease path
- sequential-channel execution as the first memory-pressure strategy
- no graph replay requirement
- no device-wide scheduler
- no general allocator framework

The plan expects Exact optics to be resource heavy because spektrafilm-style PSFs, FFTs, padded workspaces, and frequency-domain resources can be large. Resource pressure is handled through admission, descriptor-stable retention, prepared-frame leases, sequential-channel strategies, and clear failure diagnostics. It is not handled by silently selecting Fast.

Do not add PSF caches, frequency-resource caches, cuFFT plan pools, eviction policy, or max-cacheable-entry policy just for architectural completeness. Add retained resources inside Phase 6 only when the descriptor-stable artifact would otherwise cause avoidable per-frame churn, correctness/lifetime requires retention, or owner-approved measured evidence justifies it. Otherwise they are optimization follow-up, not acceptance blockers.

Resource Ownership

Context-owned durable resources:
- sampled exact PSFs when reused

- frequency-domain PSFs
- exact Gaussian/mixture helper kernels where applicable
- cuFFT plan entries when retained across frames

Retention rules:
- exact durable resources are reclaimable by descriptor family, context key, and epoch
- large exact artifacts are not one-shot by default merely because they are large. The family contract decides: mutable work is prepared-frame-owned; descriptor-stable immutable PSFs, frequency resources, and cuFFT plans may be context-owned when reuse prevents churn; any one-shot immutable artifact needs a documented justification.
- any eviction/readmission policy is deterministic and does not change output
- diagnostics name whether admission failed on scratch, durable resource, cuFFT plan/work area, or descriptor mismatch

Frame-owned mutable resources:
- real input/output planes
- complex frequency buffers
- optional padded planes
- temporary channel buffers
- cuFFT work area when not internally owned by cuFFT

Prepared-frame leases must keep mutable exact resources isolated across overlapping renders. A mutable plan/work area must never be shared as a hidden singleton.

Admission

Admission happens during prepared-frame/context-owned preparation, before exact kernels launch.

Admission estimates:
- frame scratch
- cache-miss durable allocations
- cuFFT work area
- padded dimensions
- temporary planes
- channel scheduling
- aliasing opportunities

Allowed output-stable strategies:
- sequential channels
- explicit temporary-plane aliasing
- narrower workspace reuse with the same math

Not allowed:

- exact-to-fast downgrade
- host preview/draft downgrade
- render-time ad hoc allocation
- late discovery that a kernel needs unplanned scratch

If exact is requested and cannot be admitted, fail with a diagnostic naming route, domain, descriptor hash, requested bytes where known, resident/reclaimable class when known, unavailable resource class, and backend status.

Acceptance

- enabled exact descriptors use only the shared `SpatialOptics` path
- retained durable exact resources, if any, are context-owned and context-epoch keyed
- retained exact resources, if any, have explicit descriptor identity and deterministic retention/readmission behavior
- one-shot immutable exact resources, if any, have family-contract justification; mutable exact work is prepared-frame owned
- Exact mutable buffers are prepared-frame owned.
- overlap renders receive independent leases or fail clearly
- insufficient memory fails clearly or uses an output-stable strategy
- final exact renders do not silently downgrade to another backend
- disabled or identity Exact components for an accepted route emit no descriptor, hash participation, admission request, scratch/resource upload, kernel launch, or diagnostic noise; each exposed component has a disabled/zero-sigma evidence case unless the route explicitly requires an identity/default component for correctness and the owning family contract names that exception
- representative impulse/PSF fixtures validate camera diffusion, enlarger diffusion, and any halation/scatter descriptor that exposes Exact
- visual inspection evidence confirms accepted Exact GPU precision/execution deviations remain visually aligned with spektrafilm optics behavior under matched route/profile/scanner/output settings

Delivery Phases

Cleanup is part of every phase. A phase is not accepted until old paths touched by that phase are deleted, hard-blocked, or named as temporary bridges with owner, allowed call sites, removal phase, cleanup-gate symbol, and manifest entry. Phase 10 is a final verification sweep, not the first cleanup pass.

Every phase manifest must name the old code path, state field, resource path, or bridge that became unreachable, deleted, hard-blocked, or intentionally retained. If a phase changes behavior but deletes nothing, the manifest must state why no old path was made obsolete by that phase.

Every phase must leave the project buildable. If a route or option is not implemented yet, it must be blocked by a named phase-local preflight diagnostic rather than silently remapped to old behavior.

Agent Phase Contract

This plan is the stable architecture and phase contract. It should not become a line-by-line coding recipe. Agents may choose local helper names, private function decomposition, and file layout details that fit the codebase, as long as the phase contract, ownership rules, reference behavior, and validation gates are preserved.

Before coding a phase or sub-phase, create or record a short agent task brief. The brief constrains the next slice of work without prescribing private implementation mechanics.

Each implementation phase must explicitly preserve:

- one clear purpose
- primary allowed write areas by module/domain
- do-not-touch areas owned by later phases
- new or changed public contracts
- named temporary bridges, if any
- hard-fail cases introduced or enforced by the phase
- validation evidence required for acceptance
- cleanup-gate symbols owned by the phase

Agent Task Briefs

An agent task brief is a small execution wrapper for one phase gate or sub-phase. It exists to prevent scope drift, not to over-specify implementation.

Each brief should record:
- phase/sub-phase and stopping point
- inspect-first targets: files, functions, structs, CUDA payloads, resource descriptors, scripts, or reference files the agent must read before editing
- likely write areas and explicitly forbidden write areas
- authoritative reference behavior for the slice, including `external/spektrafilm` files/functions when behavior is being ported
- allowed behavior changes and behavior that must stay blocked or untouched
- public contracts that may change, if any
- old paths expected to be deleted, hard-blocked, or bridged
- required failure behavior and diagnostic identity fields
- one small validation target: structural, diagnostic, reference-code, targeted numeric, cleanup, or host check

- hot-path constraints when the slice touches render, CUDA launch, payload packing, resource lookup, or per-pixel/per-frame code

Briefs must not prescribe:
- exact private helper names unless the name is a public contract or cleanup-gate symbol
- line-by-line edits
- broad file renames before the relevant code has been inspected
- new abstractions merely for future extension
- mandatory heavy fixture or host evidence for a narrow structural change

If the codebase contradicts the brief, the agent stops and records the mismatch instead of expanding scope or inventing policy. If a brief omits an output-affecting policy, schema default, reference formula, resource lifetime rule, or failure mode needed to proceed, the agent treats that item as blocked until the owner decides.

Canonical Agent Start Prompts

Use these short prompts when starting the whole plan or an individual phase in a new agent thread. They are intentionally brief wrappers around this plan, not replacement specs.

Plan kickoff:

```text
Read AGENTS.md and .tmp/spektrafilm-diff/spektrafilm-port-plan.md.

We are starting production implementation of the spektrafilm port. Treat the plan as the source of truth. Do not broaden scope, preserve context-owned CUDA lifetime and finish/abort invariants while shrinking broad prepared-frame/resource-manager surfaces, keep production rendering CUDA-only, and follow spektrafilm colour science unless the plan names an intentional exception.

First, summarize the current entry gates, phase order, required checks, and any blockers before editing code. Then start only the first unlocked phase.
```

Reusable phase starter:

```text
Start Phase <N> of .tmp/spektrafilm-diff/spektrafilm-port-plan.md.

Read the phase entry gate, expected write areas, implement/acceptance bullets, cleanup ledger, and validation requirements. Do not implement later-phase behavior. If a named script/check is missing, add it only when a small repo-side helper is clearer than a manual/owner-run artifact; otherwise record the exact manual/owner-run evidence. Then
```

implement the phase, run phase-local validation and cleanup checks, and produce the phase manifest/evidence notes.
```

Phase-specific one-liners:
- Preflight: tooling contract sync before production code implementation; update or disallow stale `scripts/spektrafilm_*` helpers that contradict this plan.
- Phase 1A: metadata catalog, profile key UI identity, CPU product-render cutoff, and reference-fixture workflow. Do not change pixel math.
- Phase 1B: `ScanRoute` and route/UI contract. Resolve route identity once at the UI/state boundary.
- Phase 1C: recipe envelope and frame-preparation boundary. Prevent new public prepare/ensure surfaces.
- Phase 2: spektrafilm consumed schema, profile payloads, selected-profile/resource preflight, `ProfileDigest`, asset/version tokens, `GrainContract::density_min`, and copied-bundle source/inventory evidence.
- Phase 3: direct-route film raw, exposure-density, direct scanner payload/resource binding, direct CUDA behavior, Phase 3D canonical scanner LUT execution plus first non-pass-through negative-direct parity render, and the DIR foundation needed before print-route pixel acceptance.
- Phase 4: print route, print filter math, print resource descriptors, and print-route CUDA behavior after the DIR foundation is accepted.
- Phase 5: remaining DIR spatial/detail work, route/polarity cleanup, spatial DIR descriptors, scratch binding, and disabled-DIR pruning not completed by Phase 3D.
- Phase 6: spektrafilm-matching Exact `SpatialOptics`, exact descriptors, PSF/FFT resource admission, prepared-frame mutable work, and context-owned durable exact resources where justified.
- Phase 7: intentionally unused/reserved; stale Phase 7 references are tooling debt and not hidden implementation scope.
- Phase 8: remaining scanner route/post-effect behavior, print-route scanner integration, glare, blur, unsharp, and all-four-route completion. Canonical direct-route `ScannerSpectralLutDescriptor` execution is already Phase 3D-owned.
- Phase 9: grain contract, `density_min` ownership, `density_curves_layers`, Film-Juicer visual-grain handoff, and static-noise gating.
- Phase 10: final verification; run final route matrix, final copied-bundle source/inventory evidence, repo-wide cleanup, bridge/resource ownership review, old-CPU-renderer cutoff proof, and manifest. Do not introduce other first-time major cleanup.

Primary write areas are module-level boundaries, not exhaustive file lists:

| Phase | Primary allowed write areas | Do not change before owning phase |
| --- | --- | --- |
```

1	profile metadata identity, UI/state identity, route enum, recipe envelope, diagnostics, CPU product-cutoff/reference-fixture preflight, bridge inventory	full schema ingestion, pixel math, print/scanner/optics/grain behavior
2	consumed profile schema loading, profile/catalog assets, asset/version tokens, copied-bundle source/inventory evidence, profile digest	pixel math except validation-only comparators
3	film raw, density development, canonical direct scanner LUT execution, parity-capable negative-direct CUDA rendering, direct-route CUDA behavior, and Phase 3D DIR foundation	print route, optics, scanner post effects beyond the explicit identity parity fixture, visual grain
4	print route, filter math, print resource descriptors, print-route CUDA behavior	optics, scanner post effects, visual grain
5	remaining DIR gamma/rules, spatial DIR descriptors and scratch binding not completed by Phase 3D	optics, scanner post effects, visual grain
6	exact optics descriptors, lifetime-classified exact resources, prepared-frame-owned mutable exact work, optional retained context-owned immutable exact resources, prepared-frame exact leases	fast optics, scanner post effects, visual grain
8	scanner correction/glare/blur/unsharp ordering and route-specific scanner diagnostics	visual grain rendering
9	grain schema/statistical contract, layer resources, Film-Juicer grain handoff	broad behavior rewrites outside grain/bounds ownership
10	verification evidence, final cleanup allowlists, final route/bundle/resource sweep	major first-time feature implementation

Default File / Module Ownership

Use the existing file structure by default. Rename current files when that clarifies ownership. Add new files only when the new contract would otherwise bloat an unrelated legacy type, create circular dependencies, or make the source-of-truth boundary less clear. Any deviation from this table must be noted in the phase manifest.

Contract / type	Preferred home
`RenderRecipe`, nested recipe groups, recipe hashes	Create a focused `RenderRecipe.*` or domain recipe home when Phase 1C introduces the recipe envelope. `JuicerState.*` may hold only the publication envelope, forward declarations, and tiny glue needed to publish the snapshot. Do not add meaningful recipe behavior, hashes, diagnostics, CUDA packing inputs, resource identity, schema arrays, scanner runtimes, print runtimes, or default policy to `WorkingState` or `JuicerState.*` just because those files currently own state.
`FrameRequest` shape	Create or move to focused `FrameRequest.*`, `RenderFrameRequest.*`, or an equivalent render-boundary header by Phase 1C unless the phase manifest records a source-marked temporary bridge and a Phase 3 removal/narrowing gate. `mainProcessing.*` may host only adapters while `JuicerProcessor` call sites are cut over; accepted Phase 3+ pixel behavior must not consume `FrameRequest` by copying broad fields into mutable processor members.

`ProfileCatalog`, `ProfileAssets`, typed profile payloads	Prefer focused spektrafilm profile/catalog files when adding the new schema would keep `Profiles::AgxFilmProfile`, old parser fallbacks, or catalog defaults central to the product path. `ProfileJSONLoader.*` and `ResourceAssetLibrary.*` may host bridge glue and process-owned asset access, but once selected spektrafilm payloads, catalog visibility rules, or selected-profile consumed-field contracts exist, their real owner must be focused profile/catalog files rather than legacy loaders or asset-library classes.
Domain triplets and small bounded types	Nearest domain header; avoid a generic types dump unless multiple domains consume them.
Diagnostics/result shape	Nearest coarse host boundary first; shared header only if at least two boundaries use it.
Print filter recipe / neutral calibration identity	`Print.*`, focused print recipe files, or the `PrintRecipe` group. Avoid adding print policy to scanner/output recipe code.
Scanner LUT/range identities	`Scanner.*`, focused scanner descriptor files, or `ScannerOutputRecipe`. Keep post-LUT scanner effects out of LUT descriptor identity.
Prepared-frame views/leases and descriptor preparation boundary	`ProcessRoot.*` plus `Cuda/ResourceManager/*` internals.
CUDA payload structs	`Cuda/JuicerCudaPayloads.h`.
CUDA resource descriptors/hashes	Owning CUDA resource family files under `Cuda/ResourceManager/*` or `Cuda/JuicerCudaResources*`.
Cleanup/evidence tooling	Existing `scripts/spektrafilm_*` helpers or the smallest new helper justified by the phase. Scripts are helpers, not a required framework.

Large File And Function Containment

The current implementation has dense modules and long functions in the exact areas the port will touch, especially `JuicerState.cpp`, `mainProcessing.cpp`, `ProcessRoot.cpp`, `Cuda/JuicerCudaResources*`, and `Cuda/ResourceManager/*`. The port must reduce cognitive load while preserving phase-local deletion and buildability.

Rules:
- A phase that touches a long render/rebuild/preparation function must record a containment note in the phase manifest: which policy moved into `RenderRecipe` or a named subrecipe, which bridge/local helper was deleted or narrowed, and whether the touched function became smaller, stayed equivalent, or grew with reason.
- The containment note must name dense touched functions explicitly when applicable, including `JuicerProcessor::processImagesCUDA`, `rebuild_working_state`, `Root::prepare_cuda_frame`, `PreparedCudaFrame::prepare_*`, and CUDA/resource serving functions. If a phase adds route, payload, resource, scanner, print, DIR, optics, or grain logic to one of those functions, it must also state why the logic could not live in an owning-domain helper or narrow packer introduced by that phase.
- Do not add new broad functions that combine profile parsing, recipe policy, descriptor construction, resource admission, CUDA payload packing, and launch behavior. Split only along real ownership boundaries named by this plan.

- If a phase must keep a dense function temporarily, the phase should prefer local extraction of a coherent block into an owning-domain helper over adding generic services, registries, managers, visitors, or callback layers.
- New local lambdas inside dense render/preparation functions are acceptable only for small mechanical glue. They must not become the owner of output policy, schema interpretation, density bounds, resource identity, or CUDA payload semantics. When replacing an existing large lambda block, prefer a named domain-local helper or packer so future readers can find the behavior without reading the whole render path.
- A phase that touches `JuicerProcessor::processImagesCUDA` for payload policy, feature setup, scanner selection, print setup, DIR, halation, optics, grain, or scratch requests must move at least the touched policy into an owning-domain helper/packer or delete/narrow an existing policy block. Adding new policy lambdas to `processImagesCUDA` without reducing an existing block is accepted only when the phase manifest explains the temporary bridge and removal phase.
- Any phase write area that lists `mainProcessing.*` permits only deletion, hard-blocking, adapter glue, source-marked temporary bridge narrowing, or calls into focused recipe/descriptor/payload helpers for the touched behavior. It is not permission to add new output-policy lambdas, schema interpretation, resource identity construction, scanner/print/DIR/optics/grain defaults, or broad payload packing inside `JuicerProcessor::processImagesCUDA` or `processImpl()` unless an equal-or-larger legacy policy block is deleted or moved in the same phase and the manifest names the focused owner.
- A phase that touches `JuicerState.*` for recipe behavior must state why the behavior could not live in focused recipe/domain files. Adding new schema arrays, profile payloads, scanner runtimes, print runtimes, or CUDA/resource identity directly to `WorkingState` is rejected unless it is a named temporary bridge with a removal phase.
- Phase 1C recipe work must record a compact current-block-to-owner extraction map for `rebuild_working_state` before accepted recipe behavior lands. The map should classify the existing rebuild blocks into future owners such as profile digest/loader validation, `FilmRawRecipe`, `FilmDevelopRecipe`, `DirCouplersRecipe`, `DensityBoundsRecipe`, `PrintRecipe`, `ScannerOutputRecipe`, retained Film-Juicer visual controls, resource descriptors, hashes, and publication glue. This map is onboarding evidence and a deletion/narrowing guide, not a permanent process artifact.
- Phase 3 and Phase 4 are high-risk because they touch direct pixel behavior, print behavior, scanner handoff, CUDA payloads, and resource preparation. They must use their serial sub-gates as implementation stop points; each sub-gate leaves the code buildable and narrows at least one legacy-shaped path.
- A large-file split is not required merely because a file is long. It is required only when adding the new contract to the existing file would make ownership less clear, hide the policy owner, or make a new developer read unrelated render stages to understand the changed behavior.

Phase Manifest Rule

The phase manifest is the authoritative touched-file and evidence list. Do not infer phase scope from whatever is dirty in the working tree.

Rules:
- The current working branch is `testing`; agents may use git diff only as a helper to build the phase manifest.
- When an agent task brief exists, the phase manifest must say whether the implementation stayed inside the brief, which touched files were expected, which were added after inspection, and why any allowed scope changed.
- A phase manifest must list the touched paths, validation commands/results, cleanup-gate reports, remaining bridges, skipped checks with reasons, and any optional-calibration defaults or blocked modes introduced by the phase. The manifest may be a concise manual note when a script would add more machinery than value.
- If a diff base is needed, use the phase-start commit on `testing` or an explicitly recorded base ref in the manifest. The manifest file list, not the raw git diff, is the source of truth for phase validation.
- Dirty files outside the phase manifest are ignored for the phase and must not be reformatted, reverted, or used as evidence.

Shared Phase Evidence

Each phase records objective evidence without turning the plan into implementation code. Keep the evidence proportional to the phase:

Baseline evidence:
- explicit touched-file list used by acceptance commands.
- phase manifest written by `scripts/spektrafilm_phase_manifest.py` when that helper exists and is useful, otherwise a concise manual manifest with the same facts.
- clang-format check for touched C++/CUDA code, or a concrete not-applicable reason for doc/script-only phases.
- targeted clang-tidy for touched C++/CUDA code using `Release-Clang|x64`, or owner-run/tool-unavailable reason with the exact file list.
- `Release-Clang|x64` build result or owner-run build requirement with reason.
- one relevant automated route, fixture, structural, diagnostic, cleanup, or reference-code check owned by that phase.
- cleanup status for old paths touched by that phase, including any named bridge or narrow validation-helper allowance.
- skipped checks with concrete reasons.

Additional hard gates are used when the touched files or owner policy justify them. They are not a mandatory process framework for narrow doc/script/structural slices.

Conditional evidence:
- cleanup-symbol-gate report when a phase touches cleanup targets, bridge allowlists, CPU product-cutoff/reference-fixture access, route identity, old profile identity, old resource preparation, or Phase 10 final verification.

- bridge inventory delta when a phase adds, removes, touches, or relies on a bridge. A phase with no bridges and no bridge-touching work may record "not applicable" in the phase manifest instead of producing an empty bridge file.
- host checks only when the phase changes render behavior, resource lifetime, host-visible failure behavior, or manually validated integration behavior.
- owner manual visual validation once parity-capable renders exist. It supplements automated checks and records route, profiles, settings, reference source, rendered artifact path, expected visual behavior, observed result, and pass/fail.

Acceptance is based on the phase manifest and evidence reports. Manual confidence is not acceptance evidence.

Standard Agent Evidence Commands

Keep evidence proportional to the phase while making acceptance deterministic. A broad phase or owner-directed review may add fixture matrices, host evidence packages, or extra cleanup reports. Narrow phases should not become process-heavy when their validation surface is narrow, but every implementation phase still needs an explicit touched-file list, manifest, one phase-local validation or structural check, and cleanup status for old paths it touched.

Default evidence is:
- explicit touched-file list, not dirty-tree fallback
- phase manifest written by `scripts/spektrafilm_phase_manifest.py` when useful, otherwise a concise manual manifest
- clang-format check for touched C++/CUDA files after each code-edit batch and before phase acceptance
- targeted clang-tidy for touched C++/CUDA files after each code-edit batch and before phase acceptance
- `Release-Clang|x64` build result, or owner-run build requirement with reason
- one relevant automated validation check for the behavior touched
- cleanup gate status for old paths touched by the phase
- remaining temporary bridges touched by the phase, if any, with removal phase
- skipped checks with concrete reasons

Use the Visual Studio 18 direct tool paths from `AGENTS.md` when PATH lookup is unreliable. The canonical build command is:

```powershell
& "C:\Program Files\Microsoft Visual Studio\18\Community\MSBuild\Current\Bin\MSBuild.exe" juicer.sln /p:Configuration=Release-Clang /p:Platform=x64
```

Optional acceptance command shape for implementation phases. These commands are examples of a deterministic evidence path, not mandatory ceremony for every narrow slice. A

concise manual manifest is acceptable when it records the same touched-file list, format/tidy result, validation result, cleanup status, skipped-check reason, and bridge/default facts more clearly than adding or broadening scripts.

```bash
scripts/format_touched_files.sh --check $(cat .tmp/spektrafilm-phase-N-touched.txt)
scripts/hard_gate_touched_files.sh $(cat .tmp/spektrafilm-phase-N-touched.txt)
pwsh scripts/tidy_touched_files.ps1 -FileList .tmp/spektrafilm-phase-N-touched.txt
-Configuration Release-Clang -Platform x64 -Report .tmp/spektrafilm-phase-N-tidy.txt
cleanup_targets=()
while IFS= read -r touched_file; do cleanup_targets+=(--target "$touched_file"); done <
.tmp/spektrafilm-phase-N-touched.txt
scripts/spektrafilm_cleanup_symbol_gates.sh --report .tmp/spektrafilm-phase-N-cleanup.txt
"${cleanup_targets[@]}" --allowlist .tmp/spektrafilm-phase-N-cleanup.allowlist
python3 scripts/spektrafilm_phase_manifest.py --phase phase-N --touched-files
.tmp/spektrafilm-phase-N-touched.txt --output .tmp/spektrafilm-phase-N-manifest.json
--build-config "Release-Clang|x64" --command "<commands actually run>" --check
cleanup=pass
```

Adjust `phase-N` to the actual phase or sub-phase such as `phase-1A`. If a command cannot run in the agent environment, record an owner-run build/check requirement with reason in the manifest. Do not substitute dirty-tree fallback output for acceptance evidence.

For Phases 1A-9, cleanup gates are scoped to files touched by the phase plus any explicitly named phase-owned symbols. A phase must not fail because unrelated future-phase legacy symbols still exist in untouched files. Repo-wide cleanup gates that scan all `src` are Phase 10 evidence, or owner-approved broad sweep evidence only. Phase-local allowlists still apply to every touched-file match and must use the bridge/default names recorded in that phase manifest.

If a phase-owned cleanup scope is narrower than the cleanup script's fixed check set, that phase must add or use phase/check selection before acceptance. Do not satisfy a narrow phase by broad-allowlisting unrelated future-phase legacy symbols in files touched for that phase. The cleanup report must name the phase-owned check set that ran; for Phase 1A that set is profile identity/catalog fallback/default checks plus CPU product-render cutoff/reference-fixture access checks.

Rules:
- `-UseDirtyTree` for `scripts/tidy_touched_files.ps1` is exploratory only and must not be used for acceptance evidence.
- If `msbuild`, `pwsh`, `clang-tidy`, host tools, or fixture tooling are unavailable, record a skipped check with the concrete missing tool or reason in the manifest.

- A phase may use narrower checks only when the manifest says why the narrower evidence still covers the touched behavior.

Pre-Implementation Tooling Contract Sync

Before production code implementation starts, run a tooling contract sync against this plan. This is a scripts-only preflight unless a script's report format is documented in this plan. It exists to keep agents from validating against stale local assumptions.

Required checks and edits:
- Inspect every `scripts/spektrafilm_*` helper that a phase brief intends to use. If it encodes Fast optics in Phase 6, Exact optics in Phase 7, dirty-tree-only touched-file discovery, broad copied-bundle validation, hardcoded recipe homes that conflict with the Default File / Module Ownership table, or old compatibility/fallback expectations, update it or mark it disallowed for acceptance in the phase manifest.
- `scripts/spektrafilm_validate_fast_optics.py` is not valid acceptance evidence for this port. Fast optics is out of scope.
- `scripts/spektrafilm_validate_exact_optics.py` is valid only after it is updated to Phase 6 Exact optics terminology, sub-gates, schema names, and cleanup expectations. Phase 7 is retired/reserved in this plan; agents must not infer a hidden Phase 7 from stale scripts or old notes.
- `scripts/spektrafilm_cleanup_symbol_gates.sh` must expose phase/check selectors that match this plan's phase ownership. Phase 6 Fast and Phase 7 Exact cleanup names are stale until renamed or removed.
- Validators used for acceptance must take an explicit touched-file list or manifest path. They may record a git diff against the phase-start ref as helper context, but they must not derive acceptance scope from accidental dirty working tree state.
- Frame-boundary validators must accept focused `RenderRecipe.*` and `FrameRequest.*`/`RenderFrameRequest.*` homes. A validator that requires meaningful recipe behavior in `JuicerState.*` or the final request contract in `mainProcessing.*` is stale until updated.
- Bundle/resource validators, including `spektrafilm_validate_bundle.py` and final-sweep bundle sections, are optional inventory/smoke evidence only. They must not become full copied-resource correctness gates.

The tooling sync acceptance artifact is a concise note or manifest that lists the helpers inspected, helpers updated, helpers disallowed, and the exact phase-local validation commands that remain authoritative. Do not create a validation framework; prefer small script edits, targeted `rg` checks, and explicit manifests.

Phase dependency matrix:

| Phase | Identity/catalog | Schema/assets | Render behavior | Resource/prepared-frame | Tooling evidence |

| --- | --- | --- | --- | --- | --- |
1	minimal metadata catalog, profile keys, route enum	bounded metadata types only	CPU product cutoff; reference-fixture workflow	recipe shell, descriptor rules, bridge inventory	identity/cutoff/reference-fixture cleanup
2	catalog no longer DB/index based	selected-profile/resource consumed-field preflight and copied-bundle source/inventory check	no new pixel behavior required	asset/version tokens	profile/resource smoke or structural evidence
3	direct-route profile identity consumed	film raw, density, bounds	direct scans	CUDA payloads from recipe	direct-route fixtures
4	print-route identity consumed	print/filter assets	print scans	print resource descriptors	print-route fixtures
5	DIR identity consumed	DIR recipe	DIR behavior	spatial DIR scratch descriptors	DIR fixtures
6	exact identity consumed	exact descriptors	spektrafilm-matching exact optics	lifetime-classified exact PSF/FFT resources; prepared-frame-owned mutable exact work; optional retained context-owned immutable exact resources	exact admission and visual-inspection fixtures
8	scanner route identity refined	correction/glare placement	scanner effects and print-route scanner completion	post-effect identity cleanup and verification/reuse of the Phase 3D canonical scanner LUT contract	scanner fixtures
9	grain contract identity consumed	density layers	Film-Juicer grain handoff	layer upload/binding	grain contract fixtures
10	final verification	final copied-bundle source/inventory evidence	final route matrix plus GPU-output evidence	final bridge/resource evidence plus old-CPU-renderer cutoff proof	build/manifest/cleanup gates

Phase 1: Contracts, Metadata Catalog, UI Identity, CPU Product Cutoff

Purpose: make the irreversible spektrafilm-only cutover visible before changing processing behavior.

Phase 1 is intentionally split into small buildable gates. No Phase 1 gate changes spektrafilm pixel math. Phase 1 has hard sequential sub-gates for this serial implementation. Complete Phase 1A and record its manifest before accepting 1B; complete 1B before accepting 1C. The gates may be implemented in one reviewable changeset when the owner approves and the evidence remains separately attributable to 1A, 1B, and 1C; do not turn the split into three manifest rituals when one focused change proves all three gates cleanly.

Phase 1A: catalog identity and CPU product-render cutoff
- Entry gate:
  - The Phase 1A metadata source is the product resource root `Resources/profiles/*.json`.
  - The current `Resources/` tree has been updated with the latest spektrafilm resources before implementation starts.

  - At minimum, `Resources/profiles/*.json` contains catalog-valid entries for one negative filming profile, one positive filming profile, and one printing profile.
  - Catalog-visible profiles provide a stable `info.stock` key; `info.name` may fall back to `info.stock` for labels only. `info.type`, `info.support`, and `info.stage` may be explicit or may come from spektrafilm `ProfileInfo` defaults before role classification.
  - Missing `info.support`, `info.stage`, or `info.type` is not inferred from old Film-Juicer profile shape, profile key/name, folders, or neutral-filter coverage; it is defaulted only through spektrafilm `ProfileInfo`. Unsupported explicit catalog metadata makes the file unavailable for the affected Phase 1A catalog role.
  - The phase note records `Resources/profiles/*.json` as the metadata source and records the profile-file classification/defaulting rule. `Resources/filters/neutral_print_filters.json` and legacy `Resources/profiles/enlarger_neutral*.json` files are classified as non-profile resources and ignored by the profile catalog; profile-shaped JSON with unsupported role metadata is unavailable for the affected catalog role without taking down unrelated visible profiles.
- Phase 1A does not copy, convert, or curate production `Resources/profiles`; fixtures may be used only for intentionally malformed diagnostic evidence or owner-approved structural checks that cannot be represented by the product resources.
- Purpose: introduce current-contract profile identity, prevent CPU product spektrafilm output before any pixel behavior changes, and establish the external-reference fixture/stage-contract workflow used for fragile semantics such as NaN/null handling.
- Expected write areas: `ProfileJSONLoader.*`, `ResourceAssetLibrary.*`, `JuicerState.*`, `JuicerEffect.*`, `JuicerEffect.h`, `ParamNames.h`, `main.cpp`, `mainProcessing.*`, and `scripts/spektrafilm_*` only where needed for metadata identity, `StrChoiceParam` descriptor publication/fetching, parameter snapshot identity, Phase 1A cleanup selection, CPU product-render cutoff, and reference-fixture evidence.
- Do not change: full spektrafilm schema ingestion, film/print/scanner pixel math, CUDA payload layout, CUDA resource behavior, print/filter math, optics, DIR, or grain behavior.
- Implement: minimal metadata catalog from `Resources/profiles/*.json`; typed profile keys and bounded profile metadata; `FilmProfileKey` / `PrintProfileKey` OFX `StrChoiceParam` descriptors whose enum values are `info.stock` keys and whose labels use `info.name` with `info.stock` fallback for labels only; old integer profile identity removed from the normal selection path; product spektrafilm non-CUDA render access hard-blocked; narrow validation helpers, if needed, consume explicit fixtures or narrow subrecipe inputs and cannot render OFX product pixels.
- Descriptor/default rules:
  - `FilmProfileKey` and `PrintProfileKey` are the only normal host-facing profile identity params after Phase 1A. Old `FilmStock` and `PrintPaper` integer params are deleted from the normal descriptor/fetch/snapshot path or retained only as named hidden bridges that are not read as compatibility fallbacks.
  - `StrChoiceParam` enum values are stable profile keys from `info.stock`; display labels use `info.name` or `info.stock` for labels only. Labels are not identity.
  - New-instance default profile keys are `kodak_portra_400` and `kodak_portra_endura`. Missing either key is a Phase 1A catalog/descriptor failure. Do not substitute old Film-Juicer

defaults, old index `0`, first discovered entries, neutral-filter coverage, folder guesses, or owner-local alternatives.
- Phase 1A render-disposition rules:
  - After `FilmProfileKey` / `PrintProfileKey` become the normal current-contract profile identity, old Film-Juicer/agx CUDA pixel output must not be produced under those keys.
  - Until Phase 3/4 implements spektrafilm CUDA pixels for the active route, product render attempts fail through the named Phase 1A diagnostic `SpektrafilmPixelPipelineNotImplementedForPhase1A` before `Root::prepare_cuda_frame`, durable/static resource upload, product CPU pixel work, product CPU auto-exposure metering, product CPU source-pixel reads, product CPU destination writes, or legacy CUDA payload packing.
  - The Phase 1A not-implemented diagnostic is placed before any render-time path can translate spektrafilm profile keys back into the old pixel contract. This includes pending `WorkingState` rebuild, `prepareWorkingState()`, old profile-index `hash_params*` / upload-core hash construction, `load_film_stock_into_base`, old print-runtime preparation, old asset index lookup for render behavior, and any old `WorkingStateCorePayload` publication or upload path. If an owner approves `SF_TEMP_BRIDGE_ProfileKeyToLegacyRenderAsset`, the bridge must remain non-rendering and the bridge manifest must explicitly name each retained structural-check or diagnostic call-site family.
  - In the current code shape, the required product-entry cutoff is in `JuicerEffect::render()` immediately after enough current-contract host values are available to name the diagnostic and before `begin_frame_preparation()`, OFX image fetch, non-float CPU copy, pending state rebuild, CPU auto-exposure, `prepareWorkingState()`, `FrameRequest` construction, `proc.process()`, or any old CUDA preparation call. If later refactoring moves the product entry, the same before-side-effect requirement follows the new entry. Lower-level CPU/CUDA render functions may still defensively fail with the same code, but lower-level failure is not sufficient Phase 1A evidence if product-entry side effects already happened.
  - Phase 1A acceptance must identify the exact product-entry cutoff call site and the minimal host values it reads. Those values are limited to current-contract parameters and context needed to name the diagnostic, route/profile key when available, backend, and phase. The cutoff must not fetch OFX images, call `begin_frame_preparation()`, rebuild or publish `WorkingState`, construct old `FrameRequest` policy, compute old hashes, trigger resource preparation, or call old CPU/CUDA payload paths before deciding that the current route is blocked.
  - Do not add a key-to-index bridge by default. The only allowed exception is an owner-approved `SF_TEMP_BRIDGE_ProfileKeyToLegacyRenderAsset` for non-rendering catalog, UI, diagnostic, manifest, or structural-check paths. Render attempts still fail through `SpektrafilmPixelPipelineNotImplementedForPhase1A` before old pixel-contract rebuild, hash construction, loading, payload publication/upload, frame preparation, or legacy CUDA pixel execution. The bridge entry must name allowed call sites, hash inclusion/exclusion behavior, cleanup search term, owner, and a Phase 3 direct-route and Phase 4 print-route removal gate.
  - Prefer one product render cutoff at the product render entry, or a single small local helper called by CPU and CUDA product entries. Lower-level functions may assert or fail defensively, but they must not each reimplement route compatibility, fallback, or not-implemented policy.

- Acceptance:
  - `FilmProfileKey` and `PrintProfileKey` are the normal current-contract identity.
  - Discovered `StrChoiceParam` enum/default values come from stable profile keys plus effective metadata after spektrafilm `ProfileInfo` defaults.
  - Catalog evidence proves `FilmProfileKey` includes catalog-visible `support=film && stage=filming` profiles for both negative and positive `info.type`.
  - Catalog evidence proves `PrintProfileKey` includes `stage=printing` profiles regardless of whether schema `support` is `paper` or `film`, and regardless of `info.type`.
  - `info.stock` is identity; `info.name` is label only.
  - Duplicate `info.stock` values inside the same typed role fail catalog admission.
  - Missing `info.support`, `info.stage`, or `info.type` is filled only from spektrafilm `ProfileInfo` defaults before catalog visibility; unsupported explicit role metadata makes that file unavailable for the affected role and is not repaired from folder names, neutral-filter coverage, old profile shape, or profile key/name guesses.
  - Non-profile JSONs found in the selected metadata root are reported as ignored non-profile resources or covered by an explicit profile-source manifest; they do not fail the profile catalog and do not influence profile visibility, ordering, defaults, hashes, or fallback behavior.
  - Neutral-filter data, print folders, first discovered entries, and old default arrays do not affect catalog visibility or default substitution.
  - Old integer profile params are not read as compatibility fallbacks.
  - The old OFX CPU render path is not retained as validation authority. Any validation-only C++ helper is narrow, consumes explicit fixture data or subrecipe inputs, has no OFX product render call site, and is subordinate to `external/spektrafilm`.
  - Phase 1A blocks the production CPU pixel path for the current spektrafilm product contract, including `processImpl()`, `multiThreadProcessImages()` CPU rendering, non-float CPU copy fallback for spektrafilm output, and CPU auto-exposure metering. This is a product-entry cutoff, not a requirement to delete unrelated host derivation code. Any remaining CPU code is host recipe/resource derivation, diagnostics, or a narrow validation helper with no product render call site.
  - CUDA render attempts cannot produce old Film-Juicer/agx pixels under spektrafilm profile keys; they reach `SpektrafilmPixelPipelineNotImplementedForPhase1A` before pending `WorkingState` rebuild, old profile-index hash construction, `prepareWorkingState()`, old profile/index asset loading, old payload publication, or `Root::prepare_cuda_frame` until the owning pixel phase implements spektrafilm behavior. Any owner-approved `SF_TEMP_BRIDGE_ProfileKeyToLegacyRenderAsset` retained in Phase 1A is non-rendering and does not allow pixel output or old pixel-contract preparation.
  - Phase 1A manifest records touched files, build/check status, CPU product-cutoff/reference-fixture evidence, catalog/default evidence, render-disposition evidence, Phase 1A-owned cleanup check set/status, and any retained bridge.
- Implementation order:
  - Add or adjust cleanup tooling so Phase 1A evidence can run only Phase 1A-owned cleanup checks.
  - Build the minimal metadata catalog and mark files without required current-contract metadata unavailable for catalog visibility.

  - Publish/fetch `FilmProfileKey` and `PrintProfileKey` as `StrChoiceParam` descriptors from that catalog, with explicit defaults.
  - Move `ParamSnapshot` and Phase 1A hash inputs to profile keys, leaving any old profile params only as named non-fallback bridges.
  - Hard-block product spektrafilm non-CUDA render access before it can produce product output or use CPU auto-exposure/source/destination pixel work; record the external-reference fixture/stage-contract workflow for validation evidence. Do not label or retain the old CPU renderer as a spektrafilm reference path.
  - Enforce Phase 1A render disposition so spektrafilm profile keys cannot drive old CUDA pixels unless the owner-approved bridge exception above exists.

Phase 1B: route and UI/state contract
- Purpose: make route identity typed and spektrafilm-only before any route pixel behavior moves.
- Entry gate: Phase 1A manifest exists and records buildable status or owner-run build requirement.
- Expected write areas: `JuicerEffect.*`, `JuicerState.*`, route/UI parameter definitions, and small nearby helpers needed to resolve current host values into `ScanRoute`.
- Do not change: full schema ingestion, CUDA resource preparation, scanner/print/film pixel math, optics, DIR, grain, or prepared-frame bridge APIs except to name untouched old route values as bridges.
- Implement: `ScanRoute` enum and route matrix; spektrafilm-only UI/state assumptions; old `PrintBypass`, binary print/negative mode, and `ScannerMedium`-only route identity deleted or converted once through `SF_TEMP_BRIDGE_PrintRouteUi`; saved-project compatibility behavior explicitly absent.
- Acceptance: all four required routes exist in typed metadata; `ScanRoute` is the only recipe route identity for Phase 1-owned hashes; raw `PrintBypass`, `NegativeOnly`, `Print`, and `ScannerMedium` values do not cross the recipe, frame-request, descriptor, CUDA payload, or hash boundary except through a source-adjacent `SF_TEMP_BRIDGE_PrintRouteUi` adapter whose output is `ScanRoute`; Phase 1B manifest records route matrix evidence and bridge status.

Phase 1C: recipe/frame/resource boundary skeleton
- Purpose: create the minimum recipe/frame/prepared-resource boundary that prevents later spektrafilm pixels from landing through scattered legacy state, and put those boundary types in focused files before they become another `JuicerState.*` / `mainProcessing.*` expansion.
- Entry gate: Phase 1B manifest exists and records buildable status or owner-run build requirement.
- Expected write areas: focused `RenderRecipe.*` or domain recipe files for recipe types; focused `FrameRequest.*`, `RenderFrameRequest.*`, or an equivalent render-boundary header for the request shape; `JuicerState.*` only for publication-envelope glue; `mainProcessing.*` only for cutover adapters and old side-channel classification; `ProcessRoot.*` only if the preparation entry/bridge must be named; and narrow CUDA boundary headers only when needed to preserve buildability without changing pixel behavior.

- Do not change: full schema ingestion, film/print/scanner pixel math, new CUDA resource families, new public effect-specific `PreparedCudaFrame::prepare_*` methods, or render-path `ensure_*` / `command_ensure_*` methods.
- Implement: `RenderRecipe` skeleton and nested groups only where needed by Phase 1; immutable `FrameRequest` cutover shape and field disposition table; recipe-to-CUDA packer boundary rule; descriptor and scratch ownership rules; prepared-frame bridge inventory for existing public preparation methods and mixed prepared-frame view/workspace surfaces.
- Acceptance: `WorkingState` is only the publication envelope for the Phase 1 recipe shell; `FrameRequest` construction captures the stable recipe snapshot/reference plus frame-local facts; any retained `Root::prepare_cuda_frame(WorkingState...)` path is recorded as `SF_TEMP_BRIDGE_PrepareCudaFrameWorkingStateInput`; recording the bridge is not enough for pixel acceptance, and Phase 3+ product pixel phases are blocked if that bridge is still the normal preparation input for their accepted route; Phase 1C manifest records the field disposition, prepared-frame bridge inventory, focused-file status, source-adjacent recipe owner map, source-adjacent disposition pointer or bridge ledger for retained `FrameRequest` / setter bridges, and build/check evidence.

Phase 1C new work:
- `RenderRecipe` skeleton with only the groups that block old state from leaking into later phases. Expected final groups are `ProfileRoute`, focused film raw/develop/DIR groups, `DensityBoundsRecipe`, focused print and scanner/output groups, `SpatialOptics`, and `GrainContract`; do not add future fields to those groups until a current phase has a consumer.
- The C++ home for meaningful recipe behavior is a focused recipe/domain file, not a flat expansion of `JuicerState.*`. A tiny Phase 1C publication envelope may be declared through `WorkingState`, but any active recipe group that owns hashes, diagnostics, CUDA packing, resource descriptors, schema payload references, scanner/print runtime identity, or default policy lives in `RenderRecipe.*` or a focused domain recipe file in the phase that gives it behavior.
- Empty future groups are not acceptance evidence. If a group has no Phase 1C consumer, leave it as plan structure rather than C++ surface. Any `WorkingState` or `FrameRequest` addition must either replace/delete an old field path, create the Phase 1C boundary needed by Phase 3, or be recorded as a named temporary bridge with removal phase.
- a compact source-adjacent owner map in the focused recipe header, naming where a new developer should look for profile/route identity, film raw policy, film development/DIR policy, density bounds, print policy, scanner/output policy, optics, grain, frame-local facts, and CUDA preparation. This is orientation text, not a runtime registry.
- named domain value types at host boundaries where they prevent real bugs in this phase: start only with values needed by recipe/frame/preparation boundaries, such as `RgbTriplet`, `CmyTriplet`, `DensityCmy`, `CmyCcTriplet`, `Micrometers`, and `Pixels`; defer scanner color-domain wrappers until the first scanner/output recipe, descriptor, packer, validation, or diagnostics consumer, then require `LogXyzTriplet`, `XyzTriplet`, `LinearRgbTriplet`, and `EncodedRgbTriplet`.
- Type wrappers are boundary tools, not noise generators. Use them in profile loaders, recipes, descriptors, validation, diagnostics, and host/device packing boundaries when they prevent

unit/channel/identity swaps. Convert once to plain CUDA payload POD where kernels or hot loops benefit from simple arrays/scalars.
- compact diagnostic/status result shape only for the coarse host boundaries touched by Phase 1C.
- host-bounded-param validation for touched UI-boundary helpers. Once output or input color settings enter `RenderRecipe`, `FrameRequest`, scanner/output recipe data, hashes, descriptors, or CUDA payload packing, invalid color indices must fail recipe build/preflight as a current-contract host-parameter diagnostic. `OutputColor::colorSpaceFromIndex()` must not remain the silent sRGB fallback at that boundary; if Phase 1C does not carry color settings, Phase 8 blocks scanner/output acceptance until this cleanup is done.
- `FrameRequest` frame-local identity where needed: render window, concrete full-frame extent/layout, frame time, temporal/source/session tokens, stable recipe snapshot pointer, and concrete `pixelSizeUm` derived from recipe-owned film-format policy plus extent. Context key/epoch, stream identity, and submission snapshot stay with the CUDA preparation input/result.
- resource descriptor rules and naming conventions; each real resource family adds its descriptor/hash when implemented by its owning phase.
- `FrameScratchDescriptor` rule for route/domain/backend/buffer/lifetime/estimated-byte ownership; the concrete shape is introduced or extended by the first phase that needs new mutable scratch.
- descriptor-driven prepared-frame preparation shape; new spektrafilm resource families must not add public effect-specific `prepare_*` or public render-path `ensure_*` methods.
- immutable `FrameRequest` cutover shape; before Phase 3 pixel behavior lands, route, profile identity, print params, scanner settings, overrides, hashes, resource identity, and output-affecting temporal values must enter render through the stable `RenderRecipe` snapshot/reference carried by `FrameRequest`, through named subrecipe references, or through a named temporary adapter.
- by Phase 3, `FrameRequest` is a focused render-boundary type or a clearly marked temporary bridge. New pixel behavior must not be accepted if the real contract is still `JuicerProcessor::setFrameRequest()` copying broad fields into mutable members and consuming those members as policy.
- current `FrameRequest` field disposition must be recorded in the Phase 1C manifest or a short plan appendix referenced by it, and any retained old `FrameRequest` / mutable setter bridge must have a compact source-adjacent pointer or bridge ledger that names the disposition artifact and removal phase. Phase 1C is not accepted, and Phase 3 is blocked, until this table and source-discoverable bridge pointer exist.
- The disposition table must explicitly account for current CPU auto-exposure call sites and caches, including `computeAutoExposure(...)` from `JuicerEffect::render()`, `InstanceState` auto-exposure cache/mask fields, `JuicerProcessor::FrameRequest` auto-exposure fields, and any CPU source-pixel read used for metering. Each must be classified as `Deleted`, `FrameLocalDescriptor`, `HostDerivationNoPixelRead`, or `SF_TEMP_BRIDGE_CPUAutoExposureBlocked` with no product OFX pixel read. Phase 3 auto-exposure product acceptance is blocked until metering is GPU-resident and same-frame as the spektrafilm render path.

- recipe-to-CUDA packer boundaries for `RenderRecipe` or narrow subrecipes, including the first-pixel-behavior entry gate used by Phase 3+.
- CUDA payloads, resource descriptors, and backend route consumption change only in the first phase that changes that behavior. Until then, any old route value that survives behind an existing backend call must be named as an `SF_TEMP_BRIDGE_*` with an owner/removal phase.
- `SF_TEMP_BRIDGE_WorkingStateCorePayload` inventory if the current sharing payload survives temporarily.
- prepared-frame bridge inventory for current public `PreparedCudaFrame::prepare_*` methods, `Root::prepare_cuda_frame(...WorkingState...)`, `WorkspaceRequest` boolean families, `OpticsKernelView`, scanner/grain/static-noise view families, and methods that do not accept legacy inputs but still perform effect-specific preparation work after the frame-preparation boundary. The inventory must classify whether each surface is a pure view/lease accessor, a mechanical scratch lease, or a policy/upload/allocation bridge with an owning removal phase.

Phase 1C verifies but does not reimplement Phase 1A/1B work:
- typed profile keys `FilmProfileKey` and `PrintProfileKey`
- minimal metadata catalog identity, stable keys, labels, support/stage/polarity, source identity, duplicate-key checks, and cheap diagnostics
- `ScanRoute` enum and route matrix
- explicit new-instance profile defaults from spektrafilm-era product keys; default substitution by old integer index or first catalog entry is forbidden
- old `PrintBypass`/binary route identity deletion or `SF_TEMP_BRIDGE_PrintRouteUi` status
- Phase 1A CPU product-render cutoff/reference-fixture status and `SpektrafilmPixelPipelineNotImplementedForPhase1A` render disposition

Phase 1C must not introduce `DichroicFilterSet`, print filter fields, full profile support/stage/use/antihalation/channel-model parsing, neutral calibration fields, `ScannerSpectralLutDescriptor` fields, DIR payload fields, optics descriptors, or grain layer fields unless Phase 1C has a concrete consumer and records why the owning later phase cannot introduce the field itself.

UI/state assumptions:
- State/UI contract is spektrafilm-only. Old Film-Juicer saved projects are not an input, migration target, diagnostic target, or acceptance target.
- The existing OFX identifier may be retained. This does not make old Film-Juicer project state, removed params, or defaults valid product inputs, and it must not drive shims.
- The descriptor publishes required `StrChoiceParam` identities such as `FilmProfileKey` and `PrintProfileKey`. A current spektrafilm instance missing those required params fails before recipe publication with `MissingRequiredHostParameter`; old integer params are not read as compatibility fallbacks or remapped into profile keys.
- New-instance profile defaults are current-contract product defaults, not compatibility fallbacks. The default source is recorded in Phase 1A evidence and participates in descriptor/catalog admission.

- If a bridge is needed while descriptors are being replaced, it must be named, hidden from normal identity, excluded from recipe hashes, and assigned to the phase that owns removing the touched behavior. `SF_TEMP_BRIDGE_PrintRouteUi` may survive Phase 2 only as a UI adapter; direct-route dependency is removed by Phase 3 acceptance and print-route dependency by Phase 4 acceptance.
- `ParamSnapshot` and hash inputs use profile keys and resolved route identity, not profile indices or raw `PrintBypass`, by the end of this phase.
- If asset/version tokens are not fully resolved until Phase 2, any Phase 1 hash-time asset lookup is a named bridge with allowed call sites, excluded from new recipe-resource keys, and removed by the Phase 2 cleanup gate.

Acceptance and cleanup:
- all four required routes exist in typed metadata
- positive print scan is represented from the start
- `ScanRoute` is published as the sole recipe route identity and is the only route value used by Phase 1-owned hashes. Raw `PrintBypass`, `NegativeOnly`, `Print`, or `ScannerMedium` values do not cross the recipe, frame-request, descriptor, CUDA payload, or hash boundary except behind a named `SF_TEMP_BRIDGE_*` owned by the phase that removes the touched backend behavior.
- old `PrintBypass` is deleted or listed as `SF_TEMP_BRIDGE_PrintRouteUi`; its only output-affecting product is resolved `ScanRoute`, direct-route dependency has a Phase 3 removal gate, and print-route dependency has a Phase 4 removal gate
- `StrChoiceParam` choices come from the metadata catalog bootstrap, not from ad hoc folders, first-entry defaults, neutral-filter coverage, or old index lists
- minimal catalog visibility comes from profile metadata, not neutral-filter database coverage, folder guesses, or default catalog entries
- old integer film/print index state is no longer the normal identity path
- normal integer profile identity is removed only after catalog-backed profile keys are live
- spektrafilm profile keys do not drive old Film-Juicer/agx CUDA pixels after Phase 1A; the render fails through `SpektrafilmPixelPipelineNotImplementedForPhase1A` before pending old `WorkingState` rebuild, old profile-index hash construction, old profile/index asset loading, old payload publication, or preparation until the owning pixel phase implements spektrafilm behavior. Any owner-approved `SF_TEMP_BRIDGE_ProfileKeyToLegacyRenderAsset` is non-rendering only, has recorded call-site/hash impact and Phase 3/4 removal gates, and does not satisfy rendered spektrafilm output acceptance.
- first spektrafilm pixel behavior cannot read live processor/instance fields for route, profile identity, print params, scanner settings, overrides, hashes, resource identity, or output-affecting temporal values after `FrameRequest` construction begins
- current `FrameRequest` field disposition is recorded by Phase 1C; Phase 1C is not accepted, and Phase 3 is blocked, until the table exists
- descriptor ownership rules exist before Phase 3; each durable or frame-local resource family adds its own descriptor/hash when implemented

- `FrameScratchDescriptor` is introduced or extended before any spektrafilm phase requests new mutable scratch, staging, scan-error, auto-exposure, pinned host, or transient event resources
- no new public effect-specific `PreparedCudaFrame::prepare_*` or public render-path `ensure_*` / `command_ensure_*` method is added for spektrafilm resource families
- before first spektrafilm CUDA pixel behavior, static-noise resources are descriptor-gated: visual grain, dust, scratches, gate weave, or another explicit static-noise consumer must be enabled before `GrainStaticAssets`, STBN, Wang tiles, or related static-noise uploads are requested
- any retained unconditional static-noise upload is named `SF_TEMP_BRIDGE_StaticNoiseUpload`, has allowed call sites and output/resource impact recorded in the phase manifest, and is removed before Phase 3 direct-route pixel behavior is accepted. Current source targets include `Root::resolve_cuda_grain_static_resources`, `PreparedCudaFrame::GrainStaticAssets`, and `JuicerCuda::ensure_grain_static_assets_uploaded`; the phase manifest must either show they are descriptor-gated by an enabled static-noise consumer or list the exact bridge/removal evidence.
- no output-affecting field is added outside the recipe envelope without an owner/hash rule
- CPU product rendering fails unconditionally for spektrafilm output before frame-preparation resource work and before any spektrafilm behavior can render through CPU. This must not depend only on optional compile-time switches such as `JUICER_CUDA_ONLY`; the normal Release-Clang product path must not perform CPU pixel copy, CPU auto-exposure metering, CPU source-pixel reads, CPU destination writes, or fall through from `JuicerEffect::render` / `JuicerProcessor::process()` to production `processImpl()` for spektrafilm output. Any remaining CPU math is host derivation or a narrow validation helper outside product render, not a named product-route exception.
- if `processImpl()`, `multiThreadProcessImages()`, old scanner-runtime CPU rendering, or old `PipelineRunner` code remains in touched source after Phase 1A, the source contains `SF_TEMP_BRIDGE_CPUProductRendererBlocked` next to the unreachable bridge/stub, its allowed call sites exclude product OFX rendering, and its removal gate is recorded. A broad unmarked CPU renderer in `mainProcessing.*` is not accepted as "blocked" merely because an earlier product-entry check exists.
- any retained CPU math is named narrow validation-helper or host-derivation code, not a production render fallback
- old structs may be used only through named temporary bridges with owning phase, allowed call sites, and removal criteria
- `WorkingState` may remain as a small immutable recipe envelope; `WorkingStateCorePayload` is not allowed to become that envelope and must not gain spektrafilm fields
- no CUDA allocation is required to build/publish `WorkingState`
- prepared-frame bridge inventory exists and records any public method still passing `WorkingState`, `Print::Runtime`, `Print::Params`, scanner runtime structs, or performing effect-specific preparation work after the frame-preparation boundary
- any public prepared-frame bridge that still discovers resources, builds hashes, interprets schema, translates old state, uploads, allocates, or makes route/resource policy decisions is

assigned to the owning resource family and removed before that family's spektrafilm pixel behavior is accepted
- phase cleanup gate records old integer selection, catalog fallback/default status, CPU product-render cutoff/reference-fixture status, and the Phase 1A-owned cleanup check set used

Phase 2: Consumed Schema And Profile Assets

Purpose: make spektrafilm profiles first-class process-owned assets and parse only the consumed fields needed by recipes/resources. The owner-supplied spektrafilm bundle is assumed correct; this phase does not validate the whole bundle.

Entry gate:
- Phase 2A creates the consumed-field schema lock before loader behavior changes. Phase 2B entry requires that lock to exist and cover every `info.*` and `data.*` field consumed by Phase 2B, plus `metadata.version` only if current diagnostics consume it. It includes Hanatos adaptation fields only when enabled/consumed, `info.log_sensitivity_density_over_min` and `data.density_curves_model` only when a current consumer needs them, JSON-null/NaN policy only for consumed numeric fields, and ignored legacy keys only when those keys are still present in touched loader code.
- The phase manifest records the exact `external/spektrafilm` reference ref or dirty-tree status used for schema/default/formula review.
- Any schema question not resolved by `external/spektrafilm` before implementation is recorded as a blocked field with owner decision required; agents must not invent shape, finite, or default policy locally.
- The agent task brief must name the `external/spektrafilm` files/functions used to verify each consumed schema field, default, digest rule, Hanatos adaptation param, auto-exposure method, profile/resource lookup rule, JSON-null/NaN policy, and legacy stripped profile key. If the brief cannot name the reference source for a consumed behavior, that behavior is blocked.

Serial sub-gates:
- Phase 2A, schema and reference lock: complete the consumed-field schema lock, bounded-domain map, route-role table, and reference-source list before touching loader behavior. It must include only fields consumed by catalog visibility, selected-profile parsing, recipe construction, resource descriptors, digest/default policy, or compact failure diagnostics.
- Phase 2B, loader, assets, and digest: implement profile payload loading, catalog expansion, typed asset/version tokens, selected-profile preflight, and `ProfileDigest`. Runtime parsing is selected-profile/selected-route parsing, not all-files validation or user-file hardening.
- Phase 2C, bundle/source inventory: record the owner-supplied bundle source/ref/hash or a cheap inventory/smoke check when useful, then record optional calibration status and resource identities only for resources this phase actually consumes. Do not add or require a bundle validator across copied spektrafilm resources.

Expected write areas:

- focused `ProfileCatalog.*` / `ProfileAssets.*` or the mapped equivalent names are required for selected spektrafilm payload ownership, catalog role logic, consumed-field diagnostics, selected-resource identity, and profile digest policy
- `ProfileJSONLoader.*` only for loader-local parsing helpers, schema-lock enforcement, narrow selected-payload construction, or temporary bridge deletion
- `ResourceAssetLibrary.*` only for process-root asset access glue, legacy API deletion/hard-blocking, or source-marked temporary bridges with exact non-rendering call sites; it must not own new spektrafilm payloads, digest policy, catalog role policy, selected-resource diagnostics, fallback/default selection, or recipe/hash decisions
- focused recipe/profile headers where needed for typed payload references, payload identities, and digest consumption
- `JuicerState.*` only for publication-envelope glue that points at the focused recipe/profile payloads; it must not own selected profile arrays, digest policy, resource descriptor identity, or output-affecting defaults unless the call site is listed as a named `SF_TEMP_BRIDGE_*` with a Phase 2/3 removal gate
- `Print.*` only for tiny source identity structs if Phase 2 needs them for selected metadata or digest inputs; product print/filter behavior waits for Phase 4
- `scripts/spektrafilm_*` only for the smallest schema-lock, inventory, cleanup, or manifest evidence command this phase actually uses
- small fixtures under `.tmp` or owner-approved fixture locations only when they prove a failure boundary that cannot be checked structurally

Do not change:
- film raw, density, direct scanner, print-route pixel math, DIR, optics, scanner post effects, visual grain, CUDA kernels, or production render behavior except narrow selected-resource failure/preflight code needed by this phase.
- `AgxFilmProfile` as a destination schema. It may appear only behind a named temporary bridge with no new spektrafilm fields.

Old paths owned by Phase 2:
- old profile-folder discovery, CSV profile substitution, first-entry defaults, neutral-filter-coverage menu filtering, old profile index asset lookup, and hash-time asset lookup must be deleted, hard-blocked, or listed as a temporary bridge with allowed call sites and a removal phase.
- legacy index/catalog APIs such as `film_stock_for_index`, `print_paper_for_index`, `neutral_filter_database_for_dichroic_set`, print-paper folder payload lookup, selected-neutral-DB-to-default-DB fallback, invalid dichroic/index clamp-to-zero, and missing selected dichroic identity filters must be deleted or hard-blocked in the spektrafilm path before Phase 2 acceptance unless a named non-rendering bridge records exact call sites and removal phase.
- The Phase 2 cleanup gate must explicitly account for phase-owned or touched normal spektrafilm loader-path legacy symbols and call-site families, such as `Profiles::AgxFilmProfile`, `load_agx_film_profile_json`, `load_agx_film_profile`, `load_agx_print_profile`, `load_film_stock_into_base()` as a schema-ingestion path, `PrintPaperFolderProfilePayload`, `print_paper_folder_profile_payload`, `load_print_paper_folder_profile_payload`,

`load_print_paper_pairs`, `paper_dir_for_folder`, `claim_print_folder`, `append_default_film_stocks`, `append_default_print_papers`, and neutral-filter catalog ordering from old `Resources/profiles/enlarger_neutral*.json`. Do not require broad repo scans or symbol dispositions for untouched future-phase code. A cleanup report that only says "folder/CSV fallback removed" without dispositions for the touched or phase-owned loader-path symbols is not accepted.
- the current broad non-finite literal sanitizer, JSON-null-to-NaN substitution, and NaN insertion for malformed profile rows must be narrowed to the schema-lock policy for each consumed selected field. Selected consumed arrays must not be accepted through global `Inf` token replacement, silent malformed-row NaNs, or old "keep rendering with placeholders" behavior. If an old parser helper is retained for non-selected inventory/debug parsing, it must be named as a non-product bridge with exact allowed fields and removal/review phase.
- any Phase 1 asset/version-token bridge must be removed or converted to a typed asset/version token before Phase 2 acceptance.
- old resource path constants may remain only if they identify current spektrafilm resources or are recorded as non-consumed legacy files outside the spektrafilm path.

Required evidence:
- Record the spektrafilm reference source/ref/hash used for the copied resource bundle when available, plus a cheap catalog/resource inventory if it helps prove the loader sees expected roles. Do not run or add `spektrafilm_validate_bundle.py` as a required acceptance gate.
- Phase 2 evidence records profile role counts, optional neutral-calibration source/availability status when consumed as evidence, current neutral-filter JSON source/hash when consumed as evidence, selected dichroic model/resource identities only if consumed by this phase, Hanatos adaptation shape policy from the schema lock, nullable-numeric policy from the schema lock, and unavailable selected-resource failure behavior. It must not publish final neutral C/M/Y recipe values or grow into a general profile-lint framework for copied resources, unused metadata, user-profile hardening, or legacy migration.

Implement:
- expansion of the Phase 1 metadata catalog into first-class profile assets from `Resources/profiles/*.json`
- fresh spektrafilm profile structs, not extensions of `AgxFilmProfile`
- immutable `LoadedFilmProfile` and `LoadedPrintProfile` payloads, or equivalently named selected consumed-profile payloads
- selected-payload identity tokens for each loaded film/print profile and consumed copied-bundle resource; the token must represent the exact parsed/authored payload fields consumed by recipes, descriptors, and CUDA resource builders, and must not be a menu index, profile label, folder name, or constant process asset version
- loader-boundary `SpektrafilmProfileInfo`, `SpektrafilmFilmData`, and `SpektrafilmPrintData` or equivalent structs that preserve upstream field names
- loader-boundary profile metadata and Hanatos adaptation payload fields using current spektrafilm schema names

- typed bounded domains for `info.use`, `info.antihalation`, `info.channel_model`, illuminant keys, and every other bounded upstream choice consumed by Phase 2 recipe/digest/catalog code; `info.densitometer` is an opaque typed label wrapper, not a bounded enum, unless upstream spektrafilm adds a bounded domain before implementation. Film raw, print, scanner, and optics enums land with their owning consumers unless Phase 2 directly consumes them
- route-aware consumed-field parsing
- field-gated finite/NaN/null handling that rejects unsafe values in selected consumed fields before recipe publication; NaN/null may survive only for fields whose schema-lock row records spektrafilm NaN semantics and whose consumer preserves those semantics until the named math point
- one or a few targeted selected-profile loader fixtures or structural checks proving density-like `null`/`NaN` survives into the loaded payload and resource upload for consumed density fields, while `data.log_sensitivity` applies `10 ** value` before `nan_to_num`; targeted checks must also prove selected consumed-field safety failures such as `Inf`, malformed rows, missing samples, wrong axes, or old placeholder-row behavior fail the selected route. Do not expand this into per-field malformed-profile matrices or copied-bundle validation.
- copied-bundle source/inventory evidence only; no full bundle validation gate
- authored-vs-derived separation
- `ProfileDigest` contract for spektrafilm-derived runtime defaults, without mutating authored profile payloads
- reference/default notes for auto-exposure enabled default, print normalization booleans/default mode, scanner black/white correction defaults, and `density_min` default only as owner handoff evidence. Phase 2 must not implement output-affecting recipe fields, hashes, descriptors, or resource behavior for later-phase owners until the owning phase consumes them.
- fixed-axis containers or equivalent bounded arrays for selected required 81-sample spectral data and shape-checked density-curve sample data; vector-of-pairs or row-object storage is allowed only as loader-local bridge scratch before selected consumed-field checks
- neutral-filter database source/availability identity under `Resources/filters/neutral_print_filters.json` as Phase 2 inventory/evidence only. Product lookup/math, selected-entry parsing, malformed-entry failure, final neutral C/M/Y values, print-route fallback status, output-affecting hashes, descriptors, and resource preparation wait for Phase 4 `PrintRecipe`; direct routes must not perform selected-entry preflight or fail on malformed neutral calibration.
- measured dichroic filter resource identity under `Resources/filters/dichroics/{durst_digital_light,thorlabs,edmund_optics}` plus the reference/custom analytic model identity only as inventory/source identity unless Phase 4 consumes the filters
- authoritative `GrainContract` fields needed by density bounds, including `density_min`
- removal of old profile-folder/CSV substitution from the spektrafilm path, including print-paper folder payload APIs and default catalog insertion paths listed in the Phase 2 old-path inventory

Acceptance and cleanup:
- menus are built from profile metadata, not neutral-filter coverage
- positive capture-film profiles appear in film selection

- `stage=printing` profiles appear in print selection even when schema support is `film`
- unusable selected consumed fields/resources fail with compact diagnostics
- digest evidence covers spektrafilm negative/positive DIR gamma defaults, `fujifilm_velvia_100` / `fujifilm_provia_100f` overrides, halation `(use, antihalation)` presets, Hanatos adaptation payload/default status, and unsupported/defaulted source status
- non-selected bundled and user profiles are not parsed for correctness merely because they exist
- bundle/source evidence records profile count, route-role count, optional calibration status when consumed, current neutral-filter JSON source/hash when consumed, selected dichroic model/resource identities only if consumed, Hanatos adaptation shape policy, nullable numeric policy, and every skipped check with reason
- missing neutral calibration does not hide profiles
- missing selected measured dichroic resources produce an asset/preflight diagnostic with the selected filter-set key and resource path only when the current phase consumes that measured set; otherwise Phase 2 records inventory/source identity and leaves selected-resource preflight to Phase 4
- `density_min` is owned by `GrainContract`, not by Film-Juicer visual grain controls
- `density_min` default is C/M/Y density `(0.07, 0.08, 0.12)` unless a typed current-contract override is provided; any override participates in affected bounds/resource hashes and diagnostics
- old `AgxFilmProfile`, `load_agx_*` APIs, print folder/CSV payload APIs, default catalog insertion helpers, and index APIs are gone from normal call sites or named as local temporary bridges with removal gates
- `AgxFilmProfile` does not gain spektrafilm fields except inside a named temporary bridge
- old global non-finite/NaN substitution helpers are either removed from the spektrafilm loader path or reduced to field-gated helpers for selected consumed fields; selected consumed profile data rejects malformed rows and `Inf` with compact diagnostics rather than inserting placeholder samples
- profile/resource content copying is not part of this phase
- selected profile/resource payload identities are consumed by recipe/resource hashes where those payloads affect output or descriptor identity
- cleanup gate rejects default catalog entries, folder/profile guessing, neutral DB coverage menu filtering, and fake required profile defaults

Phase 3: Film Raw, Density, Direct Scanner

Purpose: make direct negative and direct positive film-scan base conversion work through the new recipe path without old exposure or scanner semantics, and produce the first parity-capable host render no later than Phase 3D. Phase 3D owns canonical direct-route scanner LUT execution and a real negative-direct CUDA render/reference comparison; structural launch evidence, a loading plug-in, pass-through output, or a blocked-output diagnostic is not rendering evidence. Default production route acceptance also requires every default non-identity scanner/output effect consumed by the route to be implemented from `ScannerOutputRecipe`;

otherwise the route remains blocked or the artifact is validation-only with identity controls recorded until the owning scanner phase lands.

Entry gate:
- Phase 2 manifest exists and records consumed-schema status, copied-bundle source/inventory status, asset/version token status, selected direct-route capture-profile availability, direct-route print-profile/neutral-calibration exclusion evidence, and any skipped checks.
- Current `FrameRequest` field disposition table exists from Phase 1C and classifies every current field as `FrameLocal`, `RecipeOwned`, `HostDerivation`, `TemporaryBridge`, or `Deleted`.
- `FrameRequest` is either a focused render-boundary type or a source-marked temporary bridge with a Phase 3 removal/narrowing note. Direct-route pixel behavior is blocked if the accepted path still relies on copying broad `FrameRequest` fields into mutable `JuicerProcessor` members and reading those members as route/profile/scanner/resource policy.
- CUDA payload packing and resource descriptors for direct routes consume `RenderRecipe` or narrow `FilmRawRecipe`, `FilmDevelopRecipe`, `DensityBoundsRecipe`, `ScannerOutputRecipe`, and `ProfileRoute` subrecipes. They must not consume spektrafilm behavior through member-by-member legacy `WorkingState` mirrors as the real contract.
- Current CUDA `PipelineRunParams` is treated as a high-risk temporary bridge until narrowed. Phase 3 accepted direct-route behavior must name any inactive-stage fields touched by the broad struct and prove they are not populated, copied, hashed, or used for resource admission unless the active direct-route recipe consumes them.
- Any retained prepared-frame method that accepts `WorkingState`, scanner runtime structs, or old resource-manager command shapes is listed in the bridge inventory before direct-route code uses it, with allowed call sites, output impact, cleanup-gate symbol, and removal phase.
- Hash builders for the direct route are free of catalog lookup, file discovery, and `root().assets()` calls.
- Static-noise/STBN/Wang/static-grain uploads are descriptor-gated before direct-route CUDA pixel behavior starts. If the current unconditional upload path is temporarily retained, it is listed as `SF_TEMP_BRIDGE_StaticNoiseUpload` and removed before Phase 3 acceptance.
- Direct-route setup intentionally skips spektrafilm's output-inert print-service initialization. Direct routes must not load selected print-profile payloads, parse selected neutral calibration entries, emit setup-status tokens for print/neutral data, or fail because route-inert print/neutral state is unavailable or malformed. Any fixture that compares against Python keeps valid print setup state on the Python side and records that Film-Juicer excludes it from product preflight.
- The agent task brief must name the reference functions/files for camera exposure ordering, auto-exposure method semantics, film raw conversion, Hanatos adaptation LUT construction, density interpolation, density bounds, and direct scanner conversion. Any direct-route behavior not covered by the brief remains blocked or explicitly no-op.
- Phase 3D starts from an owner-observed host-render disposition. If the plug-in loads but the image passes through, stays unchanged, produces no output, or fails before the direct CUDA

launch produces destination pixels, that is a Phase 3D entry defect to diagnose and fix, not an acceptable skipped host check.

Serial sub-gates:
- Phase 3A, direct recipe and descriptor ownership: land or complete the consumed `FilmRawRecipe`, `FilmDevelopRecipe`, `DensityBoundsRecipe`, `ProfileRoute`, direct scanner/LUT descriptor inputs, direct-route print/neutral exclusion checks, and scanner post-effect disposition needed for direct routes. This sub-gate is structural; it does not claim direct-route pixel parity.
- Phase 3B, film raw and density CUDA payload cutover: replace old exposure ordering, old shared Hanatos/default midgray normalization, direct density development inputs, direct CUDA payload packing, and direct auto-exposure dispatch with recipe/subrecipe-owned inputs. Same-frame auto-exposure scale remains device-resident, metering uses the spektrafilm preview dimensions rather than full-frame bounds, and auto-exposure scratch/staging has a named descriptor or family contract before direct pixels are accepted.
- Phase 3C, direct route launch and cleanup: wire and structurally validate negative and positive direct CUDA launch only after direct scanner resources consume `DensityBoundsRecipe`, direct payloads no longer rely on broad `WorkingStateCorePayload` mirroring, static-noise upload is descriptor-gated, and pre-Phase-8 scanner post effects are either implemented from the scanner/output recipe or explicitly blocked through `ScannerPostEffectsNotImplementedForPhase3` when the reference default is non-identity. Phase 3C cannot claim rendered pixels or route acceptance from structural launch/resource evidence alone. A scanner post effect may be pruned as identity only when the selected fixture/control state explicitly makes it identity and records that state. Because spektrafilm scanner unsharp defaults non-identity, a Phase 3 direct-route render that has not implemented scanner blur/unsharp/output order from `ScannerOutputRecipe` is validation-only unless the fixture explicitly sets the relevant post effects to identity.
- Phase 3D is split into three strictly ordered serial sub-gates. Completing an earlier sub-gate does not imply acceptance of a later one:
  - Phase 3D-1, negative-direct CUDA destination-write proof: first eliminate pass-through/no-output behavior and prove that a real Resolve negative-direct CUDA invocation writes and publishes non-pass-through destination pixels with DIR disabled and scanner post effects explicitly identity. This sub-gate proves the working CUDA destination path only; its baseline/output numeric comparison is not a spektrafilm reference-parity comparison, and it does not accept current film-raw or scanner interpolation semantics as reference-correct.
  - Phase 3D-2, matched negative-direct parity and canonical scanner LUT: implement the canonical direct scanner LUT contract with C/M/Y semantic axes, endpoint-inclusive grids, normalized coordinates, `log10(XYZ)` reference values, PCHIP interpolation with prepared per-axis Hermite slopes, and per-cell min/max clamping; an alternative storage representation is accepted only with numeric evidence proving equivalent decoded output and interpolation behavior. Correct output-affecting direct film-raw mismatches exposed by the matched comparison, explicitly including any missing or non-consumed Hanatos adaptation window/surface/spectral-blur state, `_rgb_to_tc_b` conversion, reflected-boundary cubic TC LUT

sampling, or brightness rescale. Compare the host render against a matched `external/spektrafilm` reference artifact and record the result before accepting Phase 3D-3.
  - Phase 3D-3, DIR foundation before print: land the recipe-owned DIR foundation needed for later print-route work. Full default-product acceptance requires the active spatial DIR contract: `DirCouplersRecipe`, digest-driven negative/positive gamma defaults and stock overrides, disabled-DIR pruning, active/amount/inhibition controls, `diffusion_size_um`, `diffusion_tail_um`, `diffusion_tail_weight`, Gaussian/exponential spatial correction, route/polarity failure behavior, and cleanup of old live OFX DIR reads as the production contract. A non-spatial CUDA DIR path is validation-only unless the selected fixture explicitly sets spatial diffusion to identity and records that state; Phase 4 must remain structural/blocked if Phase 3D-3 has not accepted full active spatial DIR.
- The Phase 3D manifest records `3D-1 destination writes`, `3D-2 matched parity`, and `3D-3 DIR foundation` status separately. A later sub-gate remains pending or blocked until its own acceptance evidence exists even when an earlier sub-gate passes.

Expected write areas:
- focused `RenderRecipe.*` / film-development recipe files, `mainProcessing.*`, `Scanner.*`, `FilmProcessing.*` / `Couplers.*` for Phase 3D direct-render diagnosis, canonical direct scanner LUT execution, and the DIR foundation, `Cuda/JuicerCudaPayloads.h`, `Cuda/JuicerCudaResources*`, `Cuda/ResourceManager/*`, and `scripts/spektrafilm_*` only where needed for direct-route film raw/density/scanner recipe fields, direct CUDA payload/resource descriptors, canonical direct `ScannerSpectralLutDescriptor` building/sampling, Phase 3D DIR recipe/payload ownership, Phase 3 bridge cleanup, and validation evidence. `JuicerState.*` is limited to publication-envelope glue.
- `mainProcessing.*` is limited to direct-route adapter deletion/narrowing, calls into focused direct-route payload/resource packers, and removal of old side-channel or payload blocks; no new output-policy lambdas, scanner defaults, or broad payload assembly may become the direct-route contract there.

Do not change:
- print-route behavior, remaining Phase 5 spatial DIR/tail work, optics behavior, scanner post effects beyond the explicitly identity negative-direct parity fixture and direct scanner conversion required by this phase, or visual grain behavior.

Old paths owned by Phase 3:
- direct CPU pixel render fallback must be confirmed absent from the production path. Targeted comparison uses `external/spektrafilm` fixtures, reference-code review, and narrow stage-contract checks rather than retaining a broad CPU renderer.
- direct-route dependency on `SF_TEMP_BRIDGE_PrintRouteUi`, old shared green-midgray normalization, old direct scanner range recomputation, direct scanner LUT packing from scattered `WorkingState` fields, direct CUDA payload packing from `WorkingStateCorePayload`, and unconditional static-noise upload must be deleted, hard-blocked, or listed as a temporary bridge that does not affect accepted direct-route behavior.

- current direct scanner LUT log2/Mitchell behavior is a Phase 3D-2 replacement or equivalence-proof target. It must not remain the production direct-route sampler or be deferred to Phase 8 while Phase 3D-2 claims a parity-capable negative-direct render.

Implement:
- spektrafilm camera exposure ordering
- `HighlightBoostRecipe` as film exposure-owned CUDA payload/launch behavior: after manual exposure compensation and before camera diffusion, lens blur, and halation; default `boost_ev=0.0` is output identity and requests no spatial resources
- typed spektrafilm auto-exposure enabled state and methods: default enabled, default method `center_weighted`, supported methods `center_weighted`, `average`, `median`, `partial`, `matrix`, `multi_zone`, and `highlight_weighted`
- final sensitivity builder
- advanced UV/IR controls and final sensitivity order: `10^log_sensitivity`, `nan_to_num`, optional UV/IR camera band-pass with per-channel reference-illuminant normalization
- Hanatos 2025 adaptation LUT identity from profile-authored window/surface params plus `apply_window`, `apply_surface`, `spectral_gaussian_blur`, and reference illuminant under the same advanced tab as UV/IR
- Mallett method-local green-midgray normalization
- removal of old shared Hanatos/default midgray normalization
- density model from `channel_density`, `base_density`, and authored/normalized curves
- `DensityBoundsRecipe` production values for direct routes, including `dataMinCmy`, `dataMaxCmy`, `invSpanCmy`, authored extrema, route medium, and authoritative `GrainContract.density_min`
- negative direct scan
- positive direct scan
- CUDA payload packing from `RenderRecipe`
- Phase 3D-1 host-render diagnosis and destination-write proof that rejects plug-in load success, pass-through, unchanged output, blocked-output diagnostics, or launch-only structural evidence as substitutes for rendered pixels; this evidence proves non-pass-through CUDA destination publication only and is not reference-parity evidence
- Phase 3D-2 canonical direct scanner LUT builder/sampler behavior matching spektrafilm `compute_with_lut` / `apply_lut_3d`: endpoint-inclusive C/M/Y grids, normalized coordinates, `log10(XYZ)` reference semantics, PCHIP interpolation, prepared per-axis Hermite slopes, and per-cell min/max clamping
- Phase 3D-2 direct film-raw parity corrections exposed by the matched comparison, including Hanatos adaptation state consumption and exact TC conversion/cubic lookup/brightness semantics where mismatched
- Phase 3D-2 matched negative-direct host/reference parity artifact with DIR disabled and scanner post effects explicitly identity, including rendered output paths and a numeric comparison appropriate to the matched render
- Phase 3D-3 DIR foundation: `DirCouplersRecipe`, digest-driven gamma/default consumption, full active spatial CUDA DIR payload/scratch binding or validation-only non-spatial isolation with

explicit spatial-identity fixture state, disabled-DIR pruning, route/polarity failure behavior, and removal or source-marking of old live OFX DIR runtime reads as production behavior
- CPU product-render cutoff and reference-fixture/stage-contract verification

Phase boundary:
- Active/default spatial DIR behavior is implemented in Phase 3D before print-route pixel acceptance. Before Phase 3D lands, routes or fixtures that require active/default DIR fail through `DirNotImplementedForPhase3`; validation fixtures that render before Phase 3D must explicitly disable DIR or explicitly set spatial diffusion to identity and record that state in evidence. Old DIR must not be kept alive as a hidden substitute.
- Phase 3C structural launch/resource evidence does not satisfy Phase 3D-1. Phase 3D-1 must demonstrate that a host invocation writes and publishes non-pass-through negative-direct destination pixels through the new CUDA path. Host rendering may not be skipped in the Phase 3D manifest.
- Phase 3D-1 destination-write evidence does not satisfy Phase 3D-2 reference parity. Phase 3D-2 must correct direct-route film-raw and scanner mismatches exposed by the matched comparison and record the matched reference-comparison artifact before Phase 3D-3 begins.
- Canonical direct scanner LUT work is Phase 3D-2-owned. Phase 8 may reuse, verify, and complete route/post-effect integration around that LUT contract, but it must not be the first phase to implement or validate canonical direct scanner sampling.
- Spatial DIR diffusion-tail scratch/admission may remain Phase 5-owned only for routes or fixtures that are explicitly blocked or validation-only before Phase 5. The manifest must state whether Phase 3D implemented the full active spatial DIR contract; if it did not, no default-active product route may claim spektrafilm DIR behavior.
- Phase 3D is the product gate for default-active DIR, not a label for partial DIR scaffolding. If the phase accepts only non-spatial, DIR-disabled, or spatial-identity evidence, the manifest must mark affected rendered artifacts as validation-only and Phase 4 product print pixels remain blocked until Phase 5 returns to and closes the Phase 3D/4 gates.
- Scanner post effects remain Phase 8-owned unless the task explicitly includes a narrow direct scanner slice. Phase 3 direct-route acceptance must not inherit old scanner blur/unsharp/glare/color-correction defaults through `Scanner::Options`, scanner runtime structs, or old static-key hashes. Any post-LUT effect that is not implemented from recipe-owned scanner/output fields is blocked by `ScannerPostEffectsNotImplementedForPhase3` when the reference default is non-identity, and the phase manifest records which choice was used. "Direct routes render" therefore means base-route or validation-only evidence until required default post effects are implemented; it is not permission to silently bypass spektrafilm default unsharp.

Acceptance and cleanup:
- direct negative and direct positive routes render through CUDA only for the behavior actually accepted by the manifest. Before full Phase 3D DIR acceptance, any DIR-disabled, non-spatial, or spatial-identity rendered artifact is validation-only and cannot satisfy default-active product acceptance. After Phase 3D, default-active DIR uses the full spatial `DirCouplersRecipe` path rather than old runtime behavior.

- Phase 3D-1 cannot pass unless a real owner-host negative-direct render produces and publishes non-pass-through pixels through the new CUDA path with DIR disabled and scanner post effects explicitly identity. A structural validator, successful build, plug-in load, unchanged/pass-through image, launch-only evidence, visual inspection alone, or skipped host run cannot satisfy this sub-gate.
- Phase 3D-2 cannot pass unless the Phase 3D-1 render is compared against a matched spektrafilm reference, output-affecting direct-route mismatches found by that comparison are corrected, and the matched comparison artifact is recorded. The Phase 3D-2 negative-direct parity artifact uses the canonical scanner LUT contract. Current log2 storage is allowed only when decoded-output equivalence is proven; Mitchell-cubic, trilinear, bilinear, or other non-PCHIP direct scanner sampling is not accepted.
- direct-route product acceptance proves selected print-profile payloads and neutral-filter selected entries are not loaded, validated, hashed, prepared, or converted into setup-status tokens for direct-route rendering. Any Python reference fixture uses valid print setup state but the Film-Juicer product path treats that state as output-inert and excludes it completely.
- positive direct scan uses typed positive-film semantics, not a remap through negative behavior
- full film develop uses normalized film density curves; direct `ScannerSpectralLutDescriptor` bounds use `-density_min` lower bounds and film curve maxima
- UV/IR filters default inactive but are exposed as advanced typed controls; auto-exposure defaults enabled with method `center_weighted`; Hanatos adaptation window/surface/spectral-blur defaults match current reference behavior and are hashed when output-affecting
- profile RGB/CMY/channel conversions happen once at profile/recipe boundary
- `ScannerSpectralLutDescriptor` consumes `DensityBoundsRecipe`; kernels and payload packers do not recompute bounds from profile schema
- `ScannerSpectralLutDescriptor` resource identity is owned by the scanner resource family and follows the canonical descriptor row in the Resource Descriptors section
- direct scanner ranges use authored extrema and `GrainContract` lower bounds through `DensityBoundsRecipe`
- direct-route CUDA payloads/resource descriptors do not depend on member-by-member `WorkingStateCorePayload` mirroring; `SF_TEMP_BRIDGE_WorkingStateCorePayload` is removed for direct-route behavior or retained only behind one named adapter with no schema interpretation, no hash ownership, no new fields, and a Phase 4 removal gate
- direct-route product acceptance rejects broad `Root::prepare_cuda_frame(...WorkingState...)`, `command_ensure_uploaded(WorkingState)`, or equivalent monolithic-state preparation as the normal route path; accepted direct-route preparation consumes `FrameRequest`, `RenderRecipe` / named subrecipes, and owned descriptors
- no production CPU film raw, density, or direct scanner render path remains for touched behavior; any retained CPU code is host derivation or a narrow validation helper with no product render call site and no parity authority over `external/spektrafilm`
- resource preparation stays behind `PreparedCudaFrame`
- grain/static-noise disabled state produces no static-noise descriptor, no upload request, no scratch admission, no hash contribution, and no kernel work unless an explicit Film-Juicer static-noise consumer is enabled

- final sensitivity, selected RGB-to-raw method resources, and scanner table hashes are built from consumed recipe data; unselected Hanatos/Mallett resource families produce no upload/admission request and no hash contribution for accepted direct-route behavior
- cleanup gate rejects old shared green-midgray normalization, schema interpretation in CUDA payload assembly, direct-route payload packing from scattered legacy state, and hash-time asset lookup

Phase 4: Print Route And Filter Math

Purpose: make negative and positive print-scan print/exposure/filter handoff work through typed print recipes and spektrafilm CC filter math. Default production print-route acceptance also requires active default DIR and any default non-identity scanner/output effects, including print glare, to be implemented from their owning recipes; otherwise the route remains blocked or artifacts are validation-only.

Entry gate:
- Phase 3 manifest exists and records accepted direct negative/direct positive route behavior or an owner-approved direct-route gap that does not affect print-route implementation.
- Phase 3D-1 records a real non-pass-through negative-direct owner-host render, and Phase 3D-2 records the matched spektrafilm reference comparison using the canonical direct scanner LUT contract. Phase 4 must not start from structural-only direct-route evidence or a plug-in that still passes input pixels through unchanged.
- Phase 3D-3 full active spatial DIR foundation is accepted before any default-active print-route pixel behavior is accepted. If Phase 3D-3 is incomplete or only non-spatial validation DIR is available, Phase 4 may perform only structural print recipe/resource work and must not claim rendered print-route product behavior except for fixtures that explicitly disable DIR or set spatial diffusion to identity.
- If the Phase 3D manifest does not say `default-active spatial DIR product-ready`, Phase 5 becomes the next required implementation gate before any Phase 4 product pixel acceptance. Phase 4 work in that state is limited to non-pixel structural recipe/resource ownership or validation-only fixtures with explicit DIR-disabled/spatial-identity settings; it must not land a product print-render route that later relies on Phase 5 to become correct.
- Phase 2 copied-bundle source/inventory evidence covers the availability of print profiles, capture-film profiles, neutral calibration data when consumed, and exposed dichroic filter sets when consumed. It is not a full resource validator.
- The task brief names the `external/spektrafilm` files/functions reviewed for printing, print normalization, preflash, filter construction, positive print route, print density development, and scanner handoff.
- `SF_TEMP_BRIDGE_PrintRouteUi`, if it still exists, is only a UI adapter whose raw value does not cross the recipe, frame-request, descriptor, CUDA payload, or hash boundary. Print-route dependency on it is removed before Phase 4 acceptance.
- Any retained `prepare_print_illuminant_filtered`, `Print::Runtime`, `Print::Params`, or `WorkingStateCorePayload` bridge is listed with allowed call sites, output impact, cleanup-gate symbol, and removal phase.

- Current CUDA `PipelineRunParams`, if still broad, is treated as a high-risk temporary bridge. Phase 4 accepted print-route behavior must name any inactive-stage fields touched by the broad struct and prove they are not populated, copied, hashed, or used for resource admission unless the active print-route recipe consumes them.

Serial sub-gates:
- Phase 4A, print recipe and resource identity: land the consumed `PrintRecipe` or print-owned subrecipe fields for print filters, neutral calibration, print exposure, preflash, dichroic resources, and print-medium handoff. Remove old Durst/170-step units from the recipe boundary before any print-route pixel claim.
- Phase 4B, print resource preparation and CUDA payload cutover: replace `Print::Params` / `Print::Runtime` side-channel behavior for accepted print-route behavior with recipe/subrecipe-owned descriptors, immutable host derivations, and CUDA payload packers. Any retained print bridge is source-marked and cannot be the accepted product path.
- Phase 4C, print route launch and scanner handoff cleanup: accept negative and positive print-route CUDA renders only after print/enlarger bounds consume `DensityBoundsRecipe`, print resources are descriptor-owned, and scanner post-effect/glare behavior is implemented from scanner/output recipe fields, blocked through `ScannerPostEffectsNotImplementedForPhase4` / `GlareNotImplementedForPhase4` when the reference default is non-identity, or pruned only when the selected fixture/control state explicitly makes it identity.

Expected write areas:
- focused `RenderRecipe.*`, `PrintRecipe.*`, or nearby print-domain files for print-owned recipe fields; `JuicerState.*` only for publication-envelope glue
- `Print.*`
- `mainProcessing.*` only for print-route adapter deletion/narrowing, calls into focused `PrintRecipe` / print payload/resource packers, and removal of old `Print::Params` / `Print::Runtime` side-channel blocks; no new print policy lambdas or broad payload assembly may become the print-route contract there
- `Scanner.*` only where print route hands prepared print medium into scanner resources
- `Cuda/JuicerCudaPayloads.h`, `Cuda/JuicerCudaResources*`, `Cuda/ResourceManager/*` for print payload/resource descriptors
- `scripts/spektrafilm_*` for print-route validation and cleanup evidence

Do not change:
- remaining Phase 5 spatial DIR/tail work, optics behavior, scanner post-effect/glare behavior beyond a named `GlareNotImplementedForPhase4` boundary, visual grain behavior, Exact optics, or broad scanner LUT policy unrelated to print-medium handoff.

Old paths owned by Phase 4:
- print-route CPU render fallback must be confirmed absent from the production path. Targeted print-route comparison uses `external/spektrafilm` fixtures, reference-code review, and narrow stage-contract checks rather than retaining a broad CPU renderer.

- old Durst/170-step filter units, old normalized Y/M/C values past the UI boundary, old neutral DB substitution/default semantics, `Print::Params` as behavior contract, `Print::Runtime` as side-channel state, print-route `WorkingStateCorePayload` packing, print-route dependency on `SF_TEMP_BRIDGE_PrintRouteUi`, and `prepare_print_illuminant_filtered` old signatures must be deleted, hard-blocked, or bridged with an explicit cleanup gate.

Implement:
- print-filter fields nested in `PrintRecipe` or an equivalent print-owned subrecipe
- typed `DichroicFilterSet` for the spektrafilm reference/custom default plus Film-Juicer's exposed `durst_digital_light`, `thorlabs`, and `edmund_optics` choices
- Kodak CC C/M/Y filter units
- spektrafilm reference/custom analytic dichroic model and measured CSV resources for every exposed Film-Juicer measured dichroic set
- Film-Juicer Y/M/C slider offsets as Kodak CC values, converted once to internal C/M/Y order
- selected dichroic identity: reference/custom model constants for the default set, or measured resource identity from `Resources/filters/dichroics/<set>/filter_{c,m,y}.csv`, with C/M/Y channel order, percent-transmittance division by 100, duplicate-wavelength uniquing by first occurrence, Akima resampling onto the canonical 380-780 nm axis, and named diagnostics for missing or malformed selected measured filter resources
- optional neutral calibration lookup owned by `PrintRecipe`, not `ProfileDigest`
- explicit manual/current C/M/Y CC recipe values, with missing neutral-calibration data preserving those current values. For a fresh/default recipe the preserved current values are the spektrafilm schema defaults `C=0`, `M=65`, `Y=55`.
- Phase 4 is the first phase that may publish output-affecting neutral C/M/Y values into `PrintRecipe`; this is the Film-Juicer adaptation of spektrafilm's `apply_database_neutral_print_filters()` mutation. Phase 2 calibration evidence/status must not be consumed as final print behavior.
- print exposure normalization truth table and explicit recipe mapping for `normalize_print_exposure` / `print_exposure_compensation`; default is both booleans true (`NormalizeAndCompensate`)
- preflash exposure and preflash M/Y shifts with C held at neutral
- print density development
- `DensityBoundsRecipe` production values for print routes, with bounds prepared once for `ScannerSpectralLutDescriptor`/resource consumers
- black/white correction slots needed by scanner/color reference
- negative print scan
- positive print scan through the normal print route

`PrintRecipe` / print-owned fields own:
- selected `DichroicFilterSet`
- dichroic resource identity/hash
- neutral C/M/Y CC values
- user C/M/Y CC offsets converted once from UI Y/M/C, including `FilmJuicerMainCFilterShift` as the output-affecting C-filter extension with default zero

- preflash M/Y CC offsets and explicit C-neutral rule
- print illuminant
- print exposure normalization mode
- print/preflash scaling order

Phase boundary:
- Print-route glare is not implemented here unless the task explicitly includes the Phase 8 scanner glare slice. Because print-route glare defaults active, a route or fixture that requires active glare before Phase 8 fails through `GlareNotImplementedForPhase4` rather than silently disabling glare or keeping old glare-compensation behavior alive. Phase 4 "render" evidence is product evidence only when active print glare and any non-identity scanner effects consumed by the selected settings are implemented from `ScannerOutputRecipe`; otherwise it is structural/validation-only evidence or blocked behavior.
- Active/default spatial DIR is Phase 3D-owned and is a precondition for default product print-route pixel acceptance. Phase 4 print-math render fixtures must either consume the Phase 3D full spatial `DirCouplersRecipe` path or explicitly record a validation-only DIR-disabled/spatial-identity fixture that does not satisfy default product print-route acceptance. When current product defaults would require active DIR and Phase 3D is incomplete, print-route renders fail through `DirNotImplementedForPhase3` rather than silently disabling DIR, dropping spatial diffusion, or using old DIR behavior.
- Other scanner post effects remain Phase 8-owned unless the task explicitly includes a narrow scanner/output slice. Phase 4 print-route acceptance must not inherit old scanner blur/unsharp/color-correction defaults, old glare-compensation removal, or scanner effects mixed into LUT identity. Any post-LUT effect other than glare that is not implemented from recipe-owned scanner/output fields is blocked by `ScannerPostEffectsNotImplementedForPhase4` when the reference default is non-identity, and the phase evidence records which choice was used. Glare remains blocked through `GlareNotImplementedForPhase4` until the owning scanner/glare slice lands.
- Phase 4 validation must include one print-math fixture with glare explicitly disabled for the fixture or not required and DIR consumed through the Phase 3D full spatial recipe-owned path, one optional validation-only DIR-disabled/spatial-identity print fixture when useful for isolating print math, and one default-active print-route glare path that fails through `GlareNotImplementedForPhase4` when active glare would otherwise be required before Phase 8.
- Early-phase acceptance evidence must record whether DIR and glare were active, blocked, or explicitly disabled for each fixture. DIR-disabled print fixtures are diagnostic isolation evidence only; they are not default product print-route acceptance evidence.

Acceptance and cleanup:
- positive print scan is supported, not merely visible
- enlarger film-density bounds use `-density_min` and capture-film curve maxima; print-media scanner bounds use print profile curve min/max and do not consume `density_min`

- Film-Juicer's existing Y/M/C sliders and `durst_digital_light` / `thorlabs` / `edmund_optics` choices may remain exposed, but their behavior is replaced by spektrafilm CC math and spektrafilm filter resources; the spektrafilm reference/custom set remains the default filter model.
- selected dichroic filter set, dichroic model/resource identity, neutral calibration identity, manual C/M/Y CC values including `FilmJuicerMainCFilterShift`, preflash values, print exposure normalization mode, print/preflash scaling order, and print illuminant are included in `PrintRecipe` identity and hashes
- spektrafilm parity fixtures set `FilmJuicerMainCFilterShift=0`; nonzero C-filter-shift fixtures are Film-Juicer extension evidence
- default print exposure normalization maps to `normalize_print_exposure=true` and `print_exposure_compensation=true`, producing default normalizer `factorMidgrayComp`
- print-balance factors, filtered main/preflash illuminants, and preflash raw are prepared from print-owned recipe/resource identities and remain stable across repeated frames with unchanged identities; Phase 4 acceptance evidence records that render launch does not recompute midgray simulation, filter application, asset lookup, or profile-table packing for these values
- print-route scanner/enlarger ranges consume `DensityBoundsRecipe`, not local recomputation from print profile schema
- missing neutral calibration may continue only when explicit user/manual C/M/Y CC recipe values are present, or when the named spektrafilm fallback C/M/Y CC `(0, 65, 55)` is applied with optional-calibration status; otherwise, routes requiring neutral balance fail through a named preflight diagnostic
- old `Print::kDefaultNeutral*` values and old per-set neutral DB substitutions must not be used as spektrafilm defaults unless they match the named spektrafilm fallback and are renamed as spektrafilm-era defaults
- old Durst/170-step filter-unit behavior and old per-set neutral DB semantics are gone from the spektrafilm path
- print raw scaling and preflash are packed from `PrintRecipe` / print-owned subrecipe fields
- print route hashes include output-affecting correction/filter/preflash inputs
- print-route CUDA payloads/resource descriptors do not depend on member-by-member `WorkingStateCorePayload` mirroring; any retained `SF_TEMP_BRIDGE_WorkingStateCorePayload` adapter has no schema interpretation, no hash ownership, no new fields, and a removal gate before Phase 5
- print-route pixel acceptance consumes the Phase 3D recipe-owned full spatial DIR foundation for default-active DIR; any DIR-disabled or spatial-identity print fixture is clearly labeled validation-only
- print-route product acceptance rejects broad `Root::prepare_cuda_frame(...WorkingState...)`, `command_ensure_uploaded(WorkingState)`, or equivalent monolithic-state preparation as the normal route path; accepted print-route preparation consumes `FrameRequest`, `RenderRecipe` / named subrecipes, immutable host derivations, and owned descriptors
- no production CPU print render path remains for touched behavior; any retained CPU code is host derivation or a narrow validation helper with no product render call site and no parity authority over `external/spektrafilm`

- cleanup gate rejects `kEnlargerSteps`, old normalized Y/M/C values past the UI boundary, old per-set neutral DB default/substitution semantics, and print-filter compatibility branches

Phase 5: DIR Completion, Spatial Tail Work, And Cleanup

Purpose: complete any DIR work not accepted in Phase 3D, especially spatial diffusion/tail scratch binding, and prove old DIR runtime paths no longer act as the production behavior contract. If Phase 3D did not accept full active spatial DIR, this phase is pulled forward as a gate return before default direct-route or print-route product pixel acceptance; any earlier rendered route evidence is validation-only or blocked until this phase closes the gap.

Entry gate:
- Either Phase 4 structural work has already run and its manifest records direct and print route status, print side-channel bridge status, and any `GlareNotImplementedForPhase4` boundary still active, or Phase 5 is being run as the explicit Phase 3D/4 gate return required before default product pixel acceptance because full active spatial DIR was not accepted earlier.
- Phase 3D manifest exists and records whether the full active spatial DIR contract was accepted before print-route pixel behavior, or whether only validation-only non-spatial/spatial-identity evidence exists. If Phase 3D did not accept full default-active spatial DIR, Phase 4 default-product print-route pixel acceptance is invalid and Phase 5 must return to the Phase 3D/4 gates before proceeding.
- Phase 2 `ProfileDigest` evidence covers DIR gamma defaults and stock overrides needed by `DirCouplersRecipe`.
- The task brief names the `external/spektrafilm` files/functions reviewed for DIR defaults, same-layer/interlayer gamma, positive-film silver-density behavior, spatial Gaussian diffusion, spatial exponential diffusion-tail behavior, and disabled-DIR behavior.
- `prepare_spatial_dir_scratch`, if retained, is listed as a temporary bridge with allowed call sites and removal phase.
- Current `Couplers::define_params`, `Couplers::fetch_runtime`, `Couplers::Runtime`, and `JUICER_ENABLE_COUPLERS` stub behavior are classified before Phase 5 implementation starts. If any survive while Phase 5 lands, they must be listed as `SF_TEMP_BRIDGE_CouplersLiveOfxRuntime` or a narrower named validation-only bridge with exact allowed call sites, live-OFX-read status, output/hash impact, cleanup-gate symbol, and removal phase.

Serial sub-gates:
- Phase 5A, DIR recipe and defaults completion: complete or verify `DirCouplersRecipe`, default-active policy, disabled-DIR pruning, digest consumption, `diffusion_size_um`, `diffusion_tail_um`, `diffusion_tail_weight`, and failure behavior left after Phase 3D.
- Phase 5B, non-spatial CUDA DIR cleanup: complete any remaining non-spatial DIR gaps and verify old non-spatial DIR math/payload packing is no longer the production contract.
- Phase 5C, spatial DIR scratch and binding: add spatial DIR descriptors, prepared-frame scratch admission/binding, exponential-tail handling, and spatial CUDA launch wiring when

diffusion is enabled. This sub-gate is required before default-active DIR can be claimed when Phase 3D did not already implement full active spatial DIR.

Expected write areas:
- focused recipe files for `DirCouplersRecipe`
- `JuicerState.*` only for publication-envelope glue; it must not own DIR defaults, live OFX runtime state, descriptor identity, scratch policy, or output-affecting DIR fields unless the call site is listed as a named `SF_TEMP_BRIDGE_*` with a Phase 5 removal gate
- `FilmProcessing.*` / `Couplers.*` where the current DIR math lives
- `mainProcessing.*` only for DIR adapter deletion/narrowing, calls into focused DIR payload/scratch helpers, and removal of old runtime DIR side-channel blocks; no new DIR defaults, scratch policy, or broad payload assembly may become the production DIR contract there
- `Cuda/JuicerCudaPayloads.h`, CUDA kernels touched by DIR, `Cuda/JuicerCudaResources*`, and `Cuda/ResourceManager/*`
- `scripts/spektrafilm_*` for DIR fixture/cleanup evidence

Do not change:
- optics behavior, scanner post effects, visual grain behavior, print-filter behavior except where DIR output feeds already accepted routes, or Exact optics resources.

Old paths owned by Phase 5:
- CPU DIR render fallback must be confirmed absent from the production path. Targeted DIR comparison uses `external/spektrafilm` fixtures, reference-code review, and narrow stage-contract checks rather than retaining a broad CPU renderer.
- old `ratio_rgb`, `diffusion_interlayer`, `high_exposure_shift`, old DIR profile fields, old pre-correction/runtime correction substitutes, ad hoc spatial scratch allocation, and `prepare_spatial_dir_scratch` effect-specific public surface must be deleted, hard-blocked, or reduced to validation-only bridges.
- old live OFX DIR parameter definition/fetch/runtime paths, including `Couplers::define_params`, `Couplers::fetch_runtime`, `Couplers::Runtime` as the behavior contract, project-size/ROD reads inside DIR runtime fetching, and compile-time DIR stubs that preserve the old runtime ABI, must be deleted, hard-blocked, or reduced to named validation-only bridges. Production DIR behavior is built from `DirCouplersRecipe` captured before render preparation, not from live instance reads.

Implement:
- any remaining typed `DirCouplersRecipe` fields beside the film-development recipe fields, not inside a broad live OFX runtime or mixed film/raw bucket
- any remaining explicit same-layer/interlayer gamma controls
- any remaining active/amount/inhibition controls plus `diffusion_size_um`, `diffusion_tail_um`, and `diffusion_tail_weight`
- any remaining positive-film silver-density rules
- any remaining non-spatial CUDA DIR gaps from Phase 3D

- spatial DIR descriptors, admission, scratch accounting, exponential-tail handling, and CUDA binding when diffusion is enabled
- route-specific diagnostics for malformed or missing DIR inputs

Acceptance and cleanup:
- active DIR uses spektrafilm gamma controls, not old `ratio_rgb`, `diffusion_interlayer`, or `high_exposure_shift`
- negative/positive defaults and stock-specific overrides from `ProfileDigest` are consumed by `DirCouplersRecipe`
- positive film uses the positive-film silver-density rule where spektrafilm requires it
- spatial DIR uses spektrafilm `diffusion_size_um`, `diffusion_tail_um`, and `diffusion_tail_weight`; when spatial DIR is enabled, correction follows spektrafilm's Gaussian/exponential mixture: `(1 - diffusion_tail_weight) * Gaussian(diffusion_size_um) + diffusion_tail_weight * Exponential(diffusion_tail_um)`. Tail parameters do not allocate scratch, upload resources, or affect resource identity when spatial DIR is inactive.
- spatial DIR scratch is prepared-frame owned and never allocated ad hoc during render
- DIR hashes are built from the consumed `DirCouplersRecipe`
- no production CPU DIR render path remains for touched behavior; any retained CPU code is host derivation or a narrow validation helper with no product render call site and no parity authority over `external/spektrafilm`
- cleanup gate rejects old DIR profile fields, old runtime correction paths, live OFX DIR fetch/define surfaces used as product behavior, compile-time DIR ABI stubs used as product behavior, and ad hoc DIR pixel-scale derivation except validation-only helpers with allowlists

Phase 6: Spektrafilm-Matching Exact Optics And Resource Admission

Purpose: implement spektrafilm-matching optics as the Exact backend of `SpatialOptics`, with mutable exact work owned by the prepared frame and descriptor-stable immutable exact artifacts allowed to be context-owned when reuse prevents avoidable churn. Exact is the reference-matching PSF/FFT optics path. Fast optics is out of scope and must not be used to satisfy Exact behavior.

Entry gate:
- Phase 5 manifest exists and records active/default DIR behavior or explicitly owner-approved DIR gaps that do not affect the optics route being implemented.
- Phase 2 `ProfileDigest` evidence covers halation low-level `(use, antihalation)` preset defaults needed by Exact optics.
- `FrameRequest` carries stable concrete `pixelSizeUm` derived from recipe-owned film-format policy plus full-frame extent before Exact optics consume micrometer values.
- The task brief names the `external/spektrafilm` files/functions or owner-approved exact reference behavior for every Exact component exposed by this phase, including PSF/FFT behavior where the reference uses it.
- Any retained public optics preparation bridge is listed with allowed call sites, resource impact, cleanup-gate symbol, and removal phase.

- The phase note names whether retained PSF, frequency, or cuFFT-plan resources are required for correctness/lifecycle or only justified by avoiding repeated descriptor-stable churn.
- Before Exact behavior is accepted, the current mixed optics/scanner/grain preparation surface has a disposition. Scanner blur/glare/unsharp, visual grain blur/dye-cloud kernels, gate-mask/static-noise scratch, and film/print Exact optics may share mechanical workspace leases only after each has its own recipe/descriptor owner. Phase 6 must not treat the current broad `OpticsKernelView`, `WorkspaceRequest` booleans, or public `prepare_*_kernel` methods as the semantic owner of Exact optics.

Exact optics component table:

Component	Domain	Required spektrafilm reference	Required behavior	Required evidence
Camera diffusion filter	`FilmLinearExposure`	Phase note names the reference file/function, filter semantics, and PSF/FFT behavior where used.	Exact execution descriptors own sampled PSF identity, `FrameRequest::pixelSizeUm`, padded extent, normalization, and scratch/admission. Recipe data owns only the enabled component and authored controls.	Impulse/PSF fixture plus reference-code review.
Camera lens blur	`FilmLinearExposure`	Phase note names the reference file/function and micrometer-to-pixel conversion semantics.	Consume `FrameRequest::pixelSizeUm` through descriptors; no render-time ad hoc scale derivation.	Impulse fixture or targeted comparison.
In-emulsion scatter / halation	`FilmLinearExposure`	Phase note names the reference file/function, stage placement, and profile preset fields.	Consume `ProfileDigest` presets plus user high-level scalars. Old Film-Juicer `strength/size_um/scattering_strength/scattering_size_um` fields are not semantic inputs.	Targeted comparison or reference-code review.
Enlarger diffusion	`PrintLinearExposure`	Phase note names the reference file/function and print-stage placement.	Run only after print raw scaling order from `PrintRecipe`; disabled state emits no descriptor, hash, scratch, resource, or kernel work.	Print-linear impulse/PSF fixture plus reference-code review.

Phase 6 must still name and verify the spektrafilm stage order around highlight boost when it touches adjacent camera diffusion, lens blur, or halation code. That verification is a boundary check only: highlight boost remains owned by `HighlightBoostRecipe` in film exposure and must not be moved into Exact optics descriptors, scratch admission, PSF/FFT resources, or optics backend selection.

If Exact remains hidden or blocked during Phase 6 instead of being exposed, the product preflight uses `ExactOpticsNotImplementedForPhase6` and the phase manifest records the blocked controls/routes. `ExactAdmissionFailure` is reserved for the later case where Exact is selected/exposed but required scratch, durable resources, cuFFT plan/work area, or prepared-frame leases cannot be admitted.

Serial sub-gates:
- Phase 6A, Exact recipe and execution descriptors: define exact descriptors, domain tags, canonicalization, hash identity, precision/channel grouping, padded extent, and requested-vs-resolved backend status.
- Phase 6B, Exact admission and resource lifetime: add scratch/admission estimates, context-owned immutable resource policy when justified, prepared-frame mutable leases, and failure diagnostics.
- Phase 6C, Exact kernels and fixtures: wire exact kernels/plans for exposed components and prove no downgrade or late allocation path remains.

Expected write areas:
- focused `SpatialOptics` descriptor files
- `JuicerState.*` only for publication-envelope glue; it must not own Exact optics defaults, descriptor identity, scratch/admission policy, backend selection, or output-affecting optics fields unless the call site is listed as a named `SF_TEMP_BRIDGE_*` with a Phase 6 removal gate
- `JuicerEffect.*`, `JuicerEffect.h`, and `ParamNames.h` only for Phase 6-owned optics UI-boundary params where an existing Film-Juicer control does not already publish the current-contract value
- `mainProcessing.*` only for Exact optics adapter deletion/narrowing, calls into focused `SpatialOptics` descriptor/payload/scratch helpers, and removal of old mixed optics setup blocks; no new Exact optics policy, backend selection, or scratch/admission policy may become the contract there
- exact optics CUDA kernels and launch glue
- `Cuda/JuicerCudaPayloads.h`, `Cuda/JuicerCudaResources*`, `Cuda/ResourceManager/*`
- `ProcessRoot.*` only where exact resource admission/preparation must expose prepared-frame views/leases
- `scripts/spektrafilm_*` only for the smallest exact/admission fixture or cleanup evidence helper this phase actually uses

UI placement locked by recovery context:
- Camera diffusion filter choice is a film/camera-stage optics control. If Film-Juicer exposes it, publish it near the film/camera optics controls, with UI-boundary identity equivalent to spektrafilm `camera_diffusion_filter_family`.
- Enlarger diffusion filter choice is a print/enlarger-stage optics control. If Film-Juicer exposes it, publish it near print/enlarger optics controls, with UI-boundary identity equivalent to spektrafilm `diffusion_filter_family`.
- Supported family values are the spektrafilm diffusion families (`glimmerglass`, `black_pro_mist`, `pro_mist`, `cinebloom`) unless the phase note blocks or hides a family. The raw family string or UI choice must be converted once into typed `SpatialOptics` recipe data before hashing, descriptor construction, or CUDA payload packing.

Do not change:

- scanner post effects, visual grain behavior, print-filter math, profile loading, or future performance-backend behavior except where shared descriptor/admission code is required for Exact.

Old paths owned by Phase 6:
- Exact-to-Fast downgrade, render-time scratch allocation outside prepared-frame admission, hidden context-wide mutable buffers, thread/instance/device-id-only exact caches, public effect-specific exact preparation APIs, unkeyed cuFFT/PSF lifetime shortcuts, old Film-Juicer halation strength/size/scattering runtime paths, old ad hoc pixel-size derivation, and disabled optics still affecting hashes/resources must be deleted, hard-blocked, or recorded as rejected unsupported modes.
- The current combined optics/scratch path that mixes scanner post effects, visual grain workspaces, gate-mask/static-noise work, halation kernels, and future Exact PSF/FFT work must be split, narrowed, or explicitly bridged before accepted Exact pixel behavior. Any retained shared workspace is a mechanical lease/aliasing detail, not the owner of scanner, grain, or Exact optics policy.
- CPU optics fallback must be confirmed absent from the production path. Targeted optics comparison uses `external/spektrafilm` fixtures, reference-code review, impulse/PSF checks, and narrow stage-contract checks rather than retaining a broad CPU renderer.

Implement:
- enforced optics domain tags across recipes, resource descriptors, CUDA payloads, and kernels
- stage-boundary verification that spektrafilm highlight boost was already implemented by the film exposure owner and still runs after manual exposure compensation and before camera diffusion, lens blur, and halation. Phase 6 does not implement or re-own highlight boost.
- halation low-level defaults from `ProfileDigest` `(use, antihalation)` presets
- exact PSF descriptors for every exposed exact component
- PSF/FFT execution that matches spektrafilm optics behavior, subject only to approved GPU precision/execution deviations that pass visual inspection
- context-owned PSF/frequency resources where descriptor-stable reuse prevents repeated per-frame construction/upload/transform churn
- context-owned cuFFT plan entries where descriptor-stable reuse prevents repeated plan churn or CUDA API lifetime requires retention
- one-shot immutable exact artifacts only when the family contract justifies one-shot ownership; mutable exact work remains prepared-frame-owned
- explicit descriptor identity and deterministic readmission policy for any retained exact resource
- prepared-frame plan/workspace leases
- admission estimates for frame scratch, cache-miss durable allocations, cuFFT work area, padded dimensions, temporary planes, and channel scheduling
- output-stable strategies: sequential channels, explicit temporary-plane aliasing, and narrower workspace reuse with the same math
- disabled/identity component pruning before hashing/resource lookup

Acceptance and cleanup:
- optional optics default to `Off`
- disabled optics request no resources, scratch, kernels, or hash contribution and do not affect output/diagnostics
- exact descriptors use only the shared `SpatialOptics` path
- exact recipe data stays shape-independent; frame extent, padded FFT dimensions, cuFFT work sizing, and final execution descriptor hashes are derived during prepared-frame preparation from `SpatialOptics` plus `FrameRequest`
- Exact descriptors and prepared views are distinct from scanner post-effect descriptors and visual-grain descriptors. If an allocation is shared for aliasing or pressure reasons, the descriptor owners remain separate and the phase evidence explains the alias group and why output policy is not shared.
- retained durable exact resources, if any, are context-owned and context-epoch keyed
- one-shot immutable exact artifacts, if any, have family-contract justification and do not create avoidable PSF/frequency/cuFFT churn
- exact mutable buffers are prepared-frame owned
- overlap renders receive independent leases or fail clearly
- insufficient memory fails clearly or uses an output-stable strategy
- final exact renders do not silently downgrade to another backend
- exact diagnostics name route, domain, descriptor hash, requested bytes where known, unavailable resource class, backend status, and resource identity
- visual inspection evidence confirms accepted lower-precision or GPU-optimized Exact behavior remains visually aligned with spektrafilm optics behavior under matched route/profile/scanner/output settings
- no production CPU optics render path remains for touched behavior; any retained CPU code is host derivation or a narrow validation helper with no product render call site and no parity authority over `external/spektrafilm`
- cleanup gate rejects exact-to-fast downgrade, render-time scratch allocation outside prepared-frame admission, effect-specific public preparation APIs that are not named temporary bridges, old halation `strength/size_um/scattering_strength/scattering_size_um` runtime paths, and ad hoc `pixelSizeUm` derivation

Phase 7: Intentionally Unused / Reserved

Phase 7 is intentionally unused/reserved in this plan; no implementation scope, cleanup ownership, or validation obligation is hidden there.

Phase 8: Scanner Corrections And Route Effects

Purpose: align remaining scanner route/post-effect behavior and keep `ScannerSpectralLutDescriptor` identity limited to inputs that actually change the LUT. Canonical direct scanner LUT implementation and the first parity-capable negative-direct render are Phase 3D-2-owned after Phase 3D-1 proves destination writes; Phase 8 verifies reuse of that accepted contract for the remaining routes and must not defer or re-open either gate.

Entry gate:
- Phase 6 Exact manifest exists, or Exact remains hidden/blocked with a manifest-recorded diagnostic that does not affect scanner route work. Phase 8 must not depend on Fast optics because Fast is out of scope for this plan.
- Phase 4 `GlareNotImplementedForPhase4` boundary is removed for accepted print routes by this phase, unless the route is explicitly blocked with a named diagnostic before output.
- Phase 3D manifest records separate Phase 3D-1 destination-write evidence and Phase 3D-2 canonical direct scanner LUT plus matched negative-direct host/reference parity evidence. If either artifact is missing, return to the owning Phase 3D sub-gate rather than implementing canonical direct scanner sampling for the first time in Phase 8.
- Phase 3/4 `ScannerSpectralLutDescriptor` identity and density bounds already exclude scanner post-effect identity for accepted direct and print routes. This phase may complete scanner post-effect descriptors and remove legacy static-key remnants, but it must not rely on Phase 8 to rescue post effects that were mixed into accepted LUT identity.
- The task brief names the `external/spektrafilm` files/functions reviewed for scanner correction, glare, black/white correction, blur/unsharp, output color/CCTF ordering, and route-specific illuminant selection.
- Any retained `prepare_scan_lut`, scanner static-key, scan-error staging, or scanner runtime bridge is listed with allowed call sites, output/hash impact, cleanup-gate symbol, and removal phase.

Serial sub-gates:
- Phase 8A, canonical-LUT reuse verification, color, and post-effect identity cleanup: verify the Phase 3D canonical scanner LUT builder/sampler contract is reused without semantic divergence for accepted routes, verify LUT-generating fields are already separate from scanner color/post-effect fields, finish any route slices still blocked before acceptance, and remove legacy static-key/hash remnants that mix post-LUT effects into LUT hashes.
- Phase 8B, route correction and glare: implement direct-vs-print glare policy, black/white correction order, scanner correction rules, and print glare defaults.
- Phase 8C, scanner blur, unsharp, output ordering, and route fixtures: wire route-specific blur/unsharp/output CCTF order and validation fixtures/artifacts for all four routes.

Expected write areas:
- `Scanner.*`
- focused recipe/descriptor files for scanner subrecipes
- `JuicerState.*` only for publication-envelope glue; it must not own scanner defaults, LUT identity, post-effect policy, glare/correction behavior, or output-affecting scanner fields unless the call site is listed as a named `SF_TEMP_BRIDGE_*` with a Phase 8 removal gate
- `mainProcessing.*` only for scanner adapter deletion/narrowing, calls into focused scanner/output payload/resource packers, and removal of old scanner side-channel/static-key blocks; no new scanner defaults, LUT identity, post-effect policy, or output-color policy may become the contract there

- `Cuda/JuicerCudaPayloads.h`, scanner CUDA kernels/resources, `Cuda/JuicerCudaResources*`, `Cuda/ResourceManager/*`
- `scripts/spektrafilm_*` for scanner route fixtures and cleanup evidence

Do not change:
- profile schema loading, print-filter math, DIR behavior, optics behavior, or visual grain behavior except where scanner density bounds are consumed as already accepted recipe inputs.

Old paths owned by Phase 8:
- CPU scanner fallback must be confirmed absent from the production path. Targeted scanner comparison uses `external/spektrafilm` fixtures, reference-code review, LUT/spectral boundary checks, and narrow stage-contract checks rather than retaining a broad CPU renderer.
- old scanner defaults, glare-compensation removal, direct film scan inheriting print glare, scanner effects mixed into `ScannerSpectralLutDescriptor` identity, `ScannerStaticKey::glareHash` in the LUT key, and output color/CCTF rebuilding `ScannerSpectralLutDescriptor` resources must be deleted, hard-blocked, or moved behind named validation-only bridges.
- Phase 8 does not own first implementation of canonical direct scanner LUT sampling; any surviving direct-route log2/Mitchell acceptance gap is a failed Phase 3D-2 gate and returns to Phase 3D-2.

Implement:
- route-specific scanner correction rules, including positive direct apply and negative direct skip behavior from the spektrafilm color-reference service
- direct film scan glare disabled unless a future approved Film-Juicer divergence says otherwise
- print scan glare from `ScannerOutputRecipe` after the print route has selected print-scan behavior
- scanner black/white correction behavior
- scanner blur/unsharp route placement
- scanner conversion order after any route-stage exposure correction: density -> spectral density -> transmitted light -> XYZ -> route-allowed black/white XYZ correction -> optional print glare -> linear output RGB -> scanner blur/unsharp -> optional output CCTF encode -> final clip
- scan illuminant selection from the scanned medium's `info.viewing_illuminant`
- scanner XYZ Y-normalization and illuminant chromaticity for `XYZ_to_RGB`
- `ScannerSpectralLutDescriptor` cleanup so all LUT bounds come from `DensityBoundsRecipe`
- scanner diagnostics from recipe/result identities

`ScannerSpectralLutDescriptor` policy:
- The canonical builder/sampler behavior for direct routes is implemented and host/reference-tested in Phase 3D-2. Phase 8 extends and verifies the same contract across remaining routes; it does not substitute a new direct-route interpolation contract.
- The existing Film-Juicer `ScannerSpectralLutDescriptor` acceleration remains the production GPU execution path when it represents the same scanner function as the spectral path.

- LUT acceleration is an execution strategy, not a spektrafilm behavior toggle. `ScannerUseLUT` is not a normal behavior control in the spektrafilm-era UI contract.
- Production scanner LUT sampling must reproduce spektrafilm's `compute_with_lut` / `apply_lut_3d` default path: endpoint-inclusive LUT grids, normalized lookup coordinates, PCHIP interpolation, and per-cell min/max clamping. Current Film-Juicer sampler/storage choices are acceptable only as versioned implementation details proven equivalent to that contract.
- Default LUT resolution is `17`; user-controlled production resolution remains in the `17-128` range unless a later approved product decision changes the ResourceManager clamp.
- Reference comparisons for production GPU output record LUT resolution and use matched-resolution LUT paths. Spectral/no-LUT comparisons are validation-only math checks.
- `ScannerSpectralLutDescriptor` identity follows the canonical descriptor row in the Resource Descriptors section.
- Scanner color/post-effect identity is separate from `ScannerSpectralLutDescriptor` identity. Output color runtime, output CCTF, black/white correction, glare, blur, unsharp, frame-bounds/runtime effects, and route post-correction identity are excluded from LUT identity.
- Replace or split the current scanner static-key shape so `ScannerSpectralLutDescriptor` identity excludes post-LUT scanner effects.
- `ScannerStaticKey::glareHash` is a current cleanup target: glare identity must move out of the LUT key.
- Glare, blur, unsharp, frame-bounds/runtime effects, and route post-correction identity live outside the LUT descriptor, preferably in one plain `ScannerPostEffectsDescriptor` plus a smaller color descriptor only if the implementation needs a distinct resource lifetime or upload boundary.

Scanner post-effect descriptor policy:
- Start with the smallest plain structs that match actual preparation boundaries: usually one post-LUT color/correction descriptor and one post-effect descriptor. Split them only when there is a real independent resource lifetime, upload point, scratch shape, or cache identity.
- Do not create one descriptor family per scalar or per UI control. Scanner color/runtime, black/white correction, glare, blur, unsharp, output color, output CCTF, frame-bounds/runtime effects, and route post-correction values should be grouped by the CUDA payload/resource boundary that consumes them.
- Scanner post-effect descriptors must not share `ScannerSpectralLutDescriptor` schema/version identity, LUT hashes, or LUT resource lifetime. They may consume LUT output through prepared-frame views, but they do not rebuild or invalidate the LUT unless the LUT stored value domain or LUT-generation inputs change.
- Disabled scanner post effects emit no descriptor, hash participation, scratch admission, upload, or kernel work unless a route explicitly requires an identity/default post-effect for correctness; any such exception must be named next to the owning descriptor.

Acceptance and cleanup:
- direct negative, direct positive, negative print, and positive print routes use route-specific scanner rules

- direct film scans use film viewing illuminant, film `channel_density`/`base_density`, `-density_min`/film-max scanner bounds, and no glare
- print scans use print viewing illuminant, print `channel_density`/`base_density`, print curve min/max scanner bounds, and print glare
- positive direct film scans apply filming exposure correction after film optics and before `log10`, and apply scanner XYZ correction before glare; negative direct film scans skip black/white color-reference correction even when controls are enabled
- print routes apply print exposure correction after print exposure scaling and before enlarger diffusion/`log10`, then apply scanner XYZ correction before print glare
- black/white XYZ correction runs before glare where the route permits it; scanner blur/unsharp runs after `XYZ_to_RGB` and before output CCTF encoding/final clip
- scanner correction, glare, blur, unsharp, output color, and output CCTF do not enter `ScannerSpectralLutDescriptor` identity
- `ScannerSpectralLutDescriptor` hashes follow the canonical descriptor row in the Resource Descriptors section.
- scanner post-effect identity is separate from `ScannerSpectralLutDescriptor` identity
- repeated-frame descriptor/cache evidence covers direct negative, direct positive, negative print, and positive print route identities, proving unchanged recipe/context/extent only binds descriptor-matched scanner LUT/post-effect resources and does not rebuild/reupload LUTs, repack large host structs, or let render-window/output-effect identity perturb stable resource keys
- glare-compensation removal is gone from the spektrafilm path
- no production CPU scanner render path remains for touched behavior; any retained CPU code is host derivation or a narrow validation helper with no product render call site and no parity authority over `external/spektrafilm`
- cleanup gate rejects old scanner defaults, glare-compensation removal, and scanner effects mixed into unrelated LUT identity

Phase 9: Grain Contract Alignment

Purpose: align the spektrafilm grain schema/statistical contract while preserving Film-Juicer visual grain.

Entry gate:
- Phase 8 manifest exists and records accepted route-specific scanner behavior, density-bound ownership, and static-noise gating status.
- Phase 2 schema lock covers `data.density_curves_layers`, `density_min`, positive/negative interpolation semantics, and malformed-layer diagnostics.
- The task brief names the `external/spektrafilm` files/functions reviewed for grain schema/statistics, default statistical fields, density-layer meaning, positive/negative layer semantics, and density-bound usage.
- The phase note states which visual behavior remains Film-Juicer-owned: visual grain, dust, scratches, and gate weave.

Serial sub-gates:
- Phase 9A, grain contract and density-bound ownership: finalize `GrainContract` ownership of `density_min` and spektrafilm grain statistical fields, and prove film-density bounds consume `density_min` while print-media scanner bounds do not.
- Phase 9B, density-layer resources and positive/negative semantics: add `density_curves_layers` validation, upload, binding, and route/polarity-specific interpolation semantics.
- Phase 9C, Film-Juicer visual grain handoff and static-noise gating: ensure visual grain controls do not re-own density bounds and grain-off produces no static-noise descriptor/upload/scratch/kernel work.

Expected write areas:
- focused `GrainContract` recipe files
- `JuicerState.*` only for publication-envelope glue; it must not own grain statistical defaults, density-bound identity, density-layer resource identity, static-noise policy, or output-affecting grain fields unless the call site is listed as a named `SF_TEMP_BRIDGE_*` with a Phase 9 removal gate
- `ProfileJSONLoader.*` only for density-layer validation gaps left from Phase 2
- grain/layer resource descriptors and CUDA upload/binding code under `Cuda/*`
- `mainProcessing.*` only for visual-grain handoff and static-noise gating
- `scripts/spektrafilm_*` for grain fixture/cleanup evidence

Do not change:
- scanner route correction/glare, print-filter math, DIR behavior, optics behavior, profile catalog visibility, or broad visual effects unrelated to the grain handoff.

Old paths owned by Phase 9:
- visual grain controls owning scanner density bounds, unconditional static-noise/STBN/Wang uploads when visual consumers are off, large visual-grain payload policy living inside `JuicerProcessor::processImagesCUDA`, `WorkingStateCorePayload` member-by-member layer upload mirrors, spektrafilm rendered grain as a visual target, and malformed layer data being silently ignored when required must be deleted, hard-blocked, or reduced to validation-only allowlisted helpers.

Implement:
- spektrafilm grain statistical contract in `GrainContract`, including field/default semantics for `agx_particle_area_um2`, `agx_particle_scale`, `agx_particle_scale_layers`, `uniformity`, `blur`, `blur_dye_clouds_um`, `micro_structure`, and `n_sub_layers` when those fields are exposed or consumed
- `density_curves_layers` resource upload and binding
- positive/negative layered grain semantics
- Film-Juicer visual grain control handoff through a focused visual-grain recipe/packer or equivalent helper, so `processImagesCUDA` binds prepared views and launches kernels

instead of owning visual-grain constants, density-bound policy, static-noise decisions, and layer math
- verification that `DensityBoundsRecipe` still consumes authoritative `density_min` for direct film scanner and enlarger film-density bounds, and does not consume it for print-media scanner bounds

Acceptance and cleanup:
- Film-Juicer remains the visual grain renderer
- dust, scratches, and gate weave remain Film-Juicer visual/temporal behavior and are not matched to spektrafilm rendered output
- glare is not covered by the grain exception and remains part of scanner/print reference matching
- grain-off still preserves density-bound contract
- grain-off and Film-Juicer visual-grain routing still preserve spektrafilm statistical field meaning for any exposed/consumed grain controls
- Film-Juicer visual grain controls do not re-own or override film-density `density_min`
- sublayer grain works for valid positive and negative profiles
- unusable selected consumed layer data fails clearly when required
- spektrafilm rendered grain is not used as a visual target
- no `WorkingStateCorePayload` member-by-member mirror remains in the grain/layer upload path; any surviving `SF_TEMP_BRIDGE_WorkingStateCorePayload` is validation-only, named, allowlisted, and scheduled for final removal
- cleanup gate rejects spektrafilm rendered grain as a visual target and visual-grain ownership of scanner density bounds

Phase 10: Final Verification And CPU Product Cutoff Proof

Purpose: prove that cleanup has happened continuously, validate GPU output, and prove the old OFX CPU renderer is unreachable from product rendering. No compatibility path, silent downgrade, locally invented default, broad CPU validation renderer, or CPU render backend may survive.

Entry gate:
- Phase 9 manifest exists, or the owner has explicitly approved a validation-only Phase 9 gap with a named reason and no hidden production behavior dependency.
- Phase 1 through Phase 9 manifests, touched-file lists, cleanup reports, bridge inventories, skipped-check reasons, and owner-run artifacts are available or explicitly marked missing with owner approval.
- Final verification starts from the phase evidence; it must not rediscover architecture, introduce major first-time behavior, or perform the first large deletion sweep for paths that earlier phases owned. The old OFX CPU pixel renderer is not a validation authority; any retained validation helper must be narrow, fixture/subrecipe-driven, and unreachable from product OFX rendering.

- Any owner-approved intermediate blocked-output release before final readiness is recorded with its exact intended diagnostics and proof that spektrafilm profile keys cannot render old Film-Juicer/agx pixels.

Implement:
- final route matrix validation for all four routes
- final validated GPU-output evidence plus reference-fixture/stage-contract evidence sufficient to prove product rendering does not depend on a retained CPU renderer
- product cutoff proof for the old OFX CPU renderer, including no CPU pixel render entry point from OFX product render, no CPU auto-exposure metering used for output, no CPU source/destination pixel render loops reachable from product render, and no product-code allowlists for CPU render behavior
- proof that any retained validation helper consumes explicit fixtures or narrow subrecipe inputs, has no product render call site, and has no parity authority over `external/spektrafilm`
- final cleanup symbol/search gates with reviewed allowlists
- final targeted clang-tidy through `scripts/tidy_touched_files.ps1` using the checked-in broad `.clang-tidy`
- final evidence manifests assembled from phase manifests and recorded phase-start refs where useful
- final sweep validation through the smallest available final-route/cleanup command or owner-approved evidence bundle; a named `spektrafilm_validate_final_sweep.py` script is allowed only if it already exists or is clearly smaller than combining existing phase evidence
- final review of bridge inventory and hard-failure/default policies
- final owner-supplied bundle source/inventory evidence for catalog-visible copied profiles and exposed required resource families, through the smallest available command, owner-run artifact, or manifest note
- final resource-management evidence for context-owned durable resources and prepared-frame scratch
- final reference-parity review of comparable implemented behavior, using DeltaE 2000 only when enough of the base pipeline exists to render comparable Film-Juicer and spektrafilm outputs
- final owner output-validation handoff once rendering works: automated evidence should produce the route/profile/settings/artifact package needed for the owner to run the last visual/output validation check without guessing hidden settings

Acceptance:
- no major bridge, legacy-path, resource-family, or payload deletion is first introduced in Phase 10
- missed major cleanup discovered in Phase 10 fails readiness and returns to the owning earlier phase, or to an owner-approved cleanup phase with its own touched-file manifest, bridge manifest, validation evidence, and cleanup gate results. Phase 10 may verify, report, and block on missed cleanup, but it must not convert itself into the first cleanup phase.

- no CPU render path remains in product code. The Phase 10 manifest records GPU validation evidence, old-CPU-renderer cutoff evidence, and exact proof that any retained validation-helper symbols/files are narrow, fixture/subrecipe-driven, and unreachable from product rendering.
- no intermediate-release exception remains in force unless the owner explicitly carries it as a blocked-output release policy; a ready production release must render spektrafilm CUDA output for accepted routes and must never render old Film-Juicer/agx pixels under spektrafilm keys
- old profile/index/folder/print/exposure/DIR/scanner/halation/glare-compensation compatibility paths are removed or explicitly validation-only with reviewed allowlists
- final copied-bundle source/inventory evidence is recorded, with any skipped smoke/inventory check reason
- comparable-render checks, when available, record route, profiles, disabled/enabled effects, scanner settings, LUT resolution, output transform, and DeltaE 2000 summary
- the final owner output-validation artifact is either passed by the owner or recorded as pending owner-run validation; automated agents do not replace this final visual/output judgment with manual confidence
- no untracked prepared-frame bridge remains; any retained bridge is validation-only or explicitly approved with owner, reason, cleanup-gate allowlist, and removal policy
- `scripts/spektrafilm_phase_manifest.py`, when present and useful, records phase evidence from explicit base/file manifests; otherwise the Phase 10 manifest records the same facts manually.
- `scripts/spektrafilm_cleanup_symbol_gates.sh --phase10-final` or `--all-checks`, when present and useful, has reviewed allowlists so gates catch old compatibility paths without blocking valid recipe/resource code. Otherwise Phase 10 records equivalent targeted `rg`/structural cleanup evidence.
- all phase manifests identify any skipped validation with concrete reasons

Phase Review Gates

Phase 1 Gate: Contracts, Metadata Catalog, UI Identity, CPU Product Cutoff

- Profile and print selection are key-based `StrChoiceParam` identities, not integer index compatibility paths.
- `StrChoiceParam` choices are backed by the metadata-only profile catalog bootstrap, so keys are authoritative before old integer identity is removed.
- All four routes exist in typed metadata from the start.
- The existing OFX identifier is retained, but old saved-project params are not detected, migrated, diagnosed, or mapped. Phase 1 evidence proves old profile params are absent from descriptor/state/hash/render contract except named implementation bridges.
- Product spektrafilm output cannot use the CPU render path. Any retained CPU validation helper is narrow, consumes explicit fixtures or subrecipe inputs, absent from product route matrices, isolated from OFX rendering, and subordinate to `external/spektrafilm`.
- `WorkingState` can publish the recipe shell without a CUDA context or allocation.
- Phase 1C records a current-field disposition table for `JuicerProcessor::FrameRequest`, every mirrored mutable `JuicerProcessor` setter/member, and setter side effects such as

`setWorkingState()` mutating DIR runtime values. The disposition table must be source-discoverable from retained bridge code and must classify each field as `FrameLocal`, `RecipeOwned`, `HostDerivation`, `TemporaryBridge`, or `Deleted`.
- Phase 1C records the prepared-frame bridge inventory for `Root::prepare_cuda_frame(...WorkingState...)`, public `PreparedCudaFrame::prepare_*` methods, `WorkspaceRequest` boolean families, `OpticsKernelView`, static-noise/grain/scanner view families, and any resource-preparation method that still performs old-state translation, upload, allocation, schema interpretation, hash construction, or policy decisions after the preparation boundary.
- First spektrafilm pixel behavior is blocked until the real render contract is `RenderRecipe` / named subrecipes through an immutable `FrameRequest`, not `JuicerProcessor::setFrameRequest()` copying broad fields into mutable processor members and consuming those members as policy.
- No new public effect-specific `PreparedCudaFrame::prepare_*`, public render-path `ensure_*`, or `ResourceManager::command_ensure_*` surface is added for spektrafilm resource families during Phase 1.
- Cleanup evidence records all old identity/cutoff/reference-path access paths touched by the phase.

Phase 2 Gate: Consumed Schema, Profile Assets, Bundle Inventory

- Catalog visibility comes from profile schema metadata, not neutral-filter coverage.
- Profile support, stage, polarity, use, antihalation, channel model, illuminants, RGB-to-raw method, print normalization mode, and dichroic filter set are explicit typed values where present/consumed. Densitometer is an explicit opaque typed label where consumed, not a locally bounded enum unless the upstream reference adds a bounded domain.
- Authored spektrafilm field names remain visible at the loader boundary for consumed data. Current Hanatos adaptation fields, `info.log_sensitivity_density_over_min`, `data.density_curves_model`, and stripped legacy-key policy are named only when selected consumers or touched cleanup code need them; unconsumed optional upstream fields do not require rows, wrappers, hashes, fixtures, or diagnostics.
- Required consumed fields fail when unusable. Optional fields fail only when selected behavior consumes them, with compact diagnostics: profile key, path when cheap, field, expected shape/unit when locally available, route, and phase.
- A copied-bundle source/inventory manifest covers catalog-visible bundled profiles and selected-route required resource availability as evidence. Normal product runtime parses selected payloads lazily only as far as current consumers need.
- Neutral calibration and dichroic filters are optional resources/calibrations, not profile menu filters.
- `density_min` is established as a `GrainContract` input to the route/media `DensityBoundsRecipe` truth table before any route consumes bounds, with default C/M/Y density `(0.07, 0.08, 0.12)` unless a typed current-contract override is provided.
- Cleanup evidence records old profile container, index API, folder/CSV substitution, and fake-default status.

Phase 3 Gate: Film Raw, Density, Direct Scanner

- Film raw exposure uses spektrafilm profile sensitivity, typed auto-exposure enabled/method semantics, selected RGB-to-raw method semantics, and Hanatos adaptation LUT identity.
- Old shared Hanatos/default green-midgray normalization is gone.
- Direct negative and direct positive scan use typed polarity and route-specific `DensityBoundsRecipe`.
- CUDA payloads bind recipe-prepared data; kernels do not re-interpret profile schema.
- Direct-route CUDA behavior consumes `FrameRequest` or named subrecipes derived from it; it does not read live processor/instance fields for route, profile identity, scanner settings, resource identity, or output-affecting temporal values after frame-request construction.
- Phase 3D-1 accepts a real negative-direct CUDA destination-write/publication artifact before parity work is accepted. Structural launch evidence, successful plug-in loading, pass-through/unchanged output, visual inspection alone, and skipped host rendering do not satisfy this sub-gate.
- Phase 3D-2 accepts canonical direct scanner LUT execution and a matched negative-direct host/reference parity artifact before Phase 3D-3 DIR work is accepted. The matched comparison must correct or disposition output-affecting direct film-raw mismatches, including Hanatos adaptation and TC lookup semantics; Phase 3D-1 non-pass-through evidence alone does not satisfy this sub-gate.
- Phase 3D-3 accepts the recipe-owned full active spatial DIR foundation needed before print-route pixel acceptance, or records that print-route pixel work remains blocked. Default-active DIR must not be reduced to non-spatial behavior or deferred to Phase 5 while Phase 4 claims default product print-route pixels.
- `ScannerSpectralLutDescriptor` identity is owned by the scanner resource family for direct scanner use and follows the canonical descriptor row in the Resource Descriptors section.
- Direct-route scanner LUT building/sampling follows canonical endpoint-inclusive C/M/Y grids, normalized lookup coordinates, `log10(XYZ)` reference semantics, PCHIP interpolation, prepared per-axis Hermite slopes, and per-cell min/max clamping. Alternative storage is accepted only with equivalence evidence; non-PCHIP direct scanner sampling is not deferred to Phase 8.
- Hanatos filming TC LUT identity is separate from `ScannerSpectralLutDescriptor` identity and includes only final sensitivity plus Hanatos adaptation window/surface/spectral-blur inputs actually consumed by film raw conversion.
- Cleanup evidence records old exposure normalization, CPU direct render, and hash-time asset lookup status.

Phase 4 Gate: Print Route

- Phase 3D-3 full active spatial DIR foundation has been accepted before print-route pixel behavior is accepted.
- Film-Juicer may reuse the existing user-facing Y/M/C print sliders after rewiring/renaming them to spektrafilm Kodak CC semantics.

- Film-Juicer may reuse the existing user-facing enlarger choices `durst_digital_light`, `thorlabs`, and `edmund_optics` as measured extension sets, and must also expose or otherwise preserve the spektrafilm reference/custom default.
- Reused controls are Film-Juicer controls over spektrafilm's model and resources, not compatibility behavior.
- Slider values are Kodak CC offsets, not old normalized 170-step wheel values.
- UI Y/M/C order converts once at the recipe boundary to internal C/M/Y CC order.
- The selected dichroic set resolves spektrafilm filter curves from the reference/custom analytic model for the default, or from `Resources/filters/dichroics/<set>/filter_{c,m,y}.csv` for measured extension sets.
- Print normalization/preflash math follows the normative formula in this plan.
- `PrintRecipe` hashes include selected dichroic set, resource identity, neutral calibration identity, manual C/M/Y CC values, preflash values, print exposure normalization mode, print/preflash scaling order, and print illuminant.
- Print `ScannerSpectralLutDescriptor`/resource hashes consume `DensityBoundsRecipe`; `PrintRecipe` and `ScannerOutputRecipe` do not own a parallel range calculation.
- Cleanup evidence records old Durst/170-step behavior, old neutral DB interpretation, and print-filter compatibility branch status.

Phase 5 Gate: DIR Completion And Spatial Tail Work

- DIR uses explicit spektrafilm gamma controls, with the full active spatial default foundation already accepted before print-route pixel acceptance.
- Positive-film silver-density rules are represented in `FilmDevelopRecipe`.
- Spatial DIR, when enabled, flows through descriptors and prepared-frame resources, including `diffusion_size_um`, `diffusion_tail_um`, `diffusion_tail_weight`, and the Gaussian/exponential mixture contract.
- Old DIR fields are removed or validation-only with allowlists.
- Cleanup evidence records old DIR profile/runtime path status.

Phase 6 Gate: Spektrafilm-Matching Exact Optics

- Exact mode matches spektrafilm optics behavior, including PSF/FFT behavior where the reference uses it. Lower precision and GPU optimizations are accepted only when visual inspection evidence shows no material difference under matched route/profile/scanner/output settings.
- Exact mode admits required resources before launch and never silently downgrades to a lower-cost backend.
- Exact mode keeps shape-independent optics policy in `SpatialOptics` recipe data and derives execution/resource/scratch descriptors during prepared-frame preparation from that recipe plus `FrameRequest` shape facts.
- Exact optics resources follow the ownership rule: mutable exact work is prepared-frame owned and dies with the frame; descriptor-stable immutable exact artifacts may be context-owned,

epoch-keyed, re-admitted on context changes, and covered by lifecycle evidence when reuse prevents avoidable churn.
- Retained Exact PSF, frequency, and cuFFT-plan resources, if any, have explicit descriptor identity and deterministic retention/readmission policy; one-shot immutable exact artifacts, if any, have documented family-contract justification.
- Exact mutable resources are prepared-frame owned.
- Disabled optics produce no output changes, resource keys, scratch, or diagnostics noise.
- Cleanup evidence records exact-to-fast downgrade, render-time allocation, old halation/scatter, CPU optics, public preparation bridge status, and ad hoc pixel-size derivation status.

Phase 8 Gate: Scanner Route Effects

- Scanner correction, glare, blur, and unsharp are route-specific.
- Canonical direct scanner LUT implementation and first negative-direct parity rendering were accepted in Phase 3D-2 after Phase 3D-1 destination-write proof; Phase 8 verifies and reuses that contract for remaining routes rather than introducing it for the first time.
- These effects do not enter `ScannerSpectralLutDescriptor` identity.
- `ScannerSpectralLutDescriptor` identity is owned by the scanner resource family and follows the canonical descriptor row in the Resource Descriptors section.
- Scanner reference comparisons record LUT resolution and use matched-resolution LUT paths for production GPU output. Spectral/no-LUT paths are validation-only math checks.
- Current scanner static-key shape is replaced or split so `ScannerSpectralLutDescriptor` identity is separate from scanner post-effect identity; glare, blur, unsharp, frame-bounds/runtime effects, and route post-correction identity do not enter LUT keys.
- Direct film scans do not inherit print glare.
- Cleanup evidence records old scanner defaults and glare-compensation status.

Phase 9 Gate: Grain Contract

- Film-Juicer remains the visual grain renderer.
- spektrafilm grain is used for the schema/statistical contract, default statistical controls, `density_min`, and layer semantics only.
- Grain visual matching to real plates is explicitly deferred.
- Dust, scratches, and gate weave remain Film-Juicer visual/temporal behavior and are excluded from spektrafilm rendered-output parity.
- Glare remains a scanner/print reference-matching behavior and is not covered by the grain/defect exclusion.
- No `WorkingStateCorePayload` member-by-member mirror remains in production CUDA payload/resource packing. Any surviving `SF_TEMP_BRIDGE_WorkingStateCorePayload` is validation-only, named, allowlisted, and scheduled for final removal.
- Cleanup evidence records visual-grain/density-bound ownership status.

Phase 10 Gate: Final Verification And CPU Product Cutoff

- All previous cleanup gates have passed or have reviewed allowlists with owner and reason.
- All bridges have been removed or explicitly carried as validation-only with owner approval and manifest entries.
- Final copied-bundle source/inventory evidence covers catalog-visible bundled profiles and exposed required resources.
- Comparable-render reference review is complete where enough implementation exists, and records DeltaE 2000, route, profiles, disabled/enabled effects, scanner settings, LUT resolution, and output transform.
- any retained CPU validation helper is narrow, fixture/subrecipe-driven, unreachable from product rendering, and subordinate to `external/spektrafilm`; no CPU render backend, fallback, product-code harness, broad CPU renderer, or product allowlist remains.
- Final route, build, tidy, manifest, and resource-management evidence is complete.

Per-Phase Definition Of Done

Every phase records the applicable subset of this checklist. The minimum is touched files, manifest/status, clang-format and clang-tidy for touched C++/CUDA code or concrete not-applicable/tool-unavailable reasons, build or owner-run build reason, one phase-local automated validation/structural check, cleanup status for touched old paths, and skipped checks with reasons. The remaining rows apply only when the phase touches that subject.

Lean evidence cap: a phase should normally have one primary evidence artifact plus targeted cleanup/search checks. More artifacts are justified only when the phase changes multiple independent route/resource families or the owner explicitly asks for broader evidence. Do not block implementation on fixture matrices, scripts, or host artifacts for behavior the phase did not touch.

Checklist menu:
- agent task brief reference or "not needed" reason for very small owner-directed changes
- explicit touched-file list used by acceptance commands
- phase manifest path and status
- phase-gate audit answers: intended architectural end state, old architecture deleted/hard-blocked or bypassed by a named temporary bridge, completion evidence, legacy-shaped/slow implementation escape risks, and explicit stop conditions before the next phase
- old path deleted, blocked, or named as a temporary bridge
- temporary prepared-frame bridge table status: APIs removed, APIs still allowed, new bridge rows added, and removal gates
- prepared-frame and other temporary bridges touched by the phase: added, removed, unchanged, or not applicable, with reason and manifest bridge entry when retained
- cleanup gate status for the symbols owned by that phase
- any cleanup allowlist entries, with owner, reason, and removal phase when applicable
- owning `RenderRecipe` group

- new types added and why each could not be nested/private
- hash fields included and excluded
- CUDA payloads that consume the recipe
- resource descriptors, prepared-frame views/leases, scratch/staging descriptors, and CUDA payload/resource/scratch family contracts added by that phase, if any
- diagnostics and failure modes
- hard-failure/default policies, including whether any non-fatal default changes output
- validation evidence used: reference code review, targeted reference numeric, structural route/resource check, diagnostic check, cleanup-only check, or not applicable with reason
- whether `WorkingState` became smaller, neutral, or grew, with reason
- `WorkingState` / `JuicerState.*` delta classification when either changes: recipe envelope, compact identity/cache, immutable asset reference, or temporary bridge with removal phase. Schema arrays, CUDA handles, resource policy, hash ownership, and output-affecting fields outside `RenderRecipe` / named subrecipes are not accepted.
- large-file/function containment gate when touching dense render/rebuild/preparation code, especially `rebuild_working_state()`, `JuicerProcessor::processImagesCUDA()`, `Print::load_profile_from_asset()`, `PreparedCudaFrame`, `JuicerCudaResources`, and `JuicerCudaResourceManager*`: record functions changed, ownership boundary improved, helper/split/deletion performed, and confirmation that no output-affecting policy was added to a dense legacy function as the real owner. Deferral is accepted only for mechanical glue that forwards to a focused recipe, descriptor, prepared view, or packer, and the phase note must name the owning follow-up/removal phase. A phase that adds route/profile/print/scanner/DIR/optics/grain/resource policy to these dense functions without moving or deleting an equal-or-larger legacy policy surface is not accepted.
- whether steady-state `begin_frame` work increased, decreased, or stayed equivalent
- full-frame scratch added, with lifetime/aliasing notes
- touched-file source: explicit phase manifest/filelist; git diff against the phase-start commit on `testing` may be recorded as helper evidence, but accidental dirty state is not a source of truth
- clang-format result for touched C++/CUDA code, or not-applicable reason
- targeted clang-tidy result for touched C++/CUDA code, or owner-run/tool-unavailable reason
- hard-gate result when required by touched files or owner policy
- `Release-Clang|x64` build result
- host/fixture validation performed or skipped with reason
- owner manual visual validation artifact once comparable parity renders are possible, or reason it is not yet possible. Phase 3D is explicitly the latest allowed point for the first comparable negative-direct parity render and may not record this check as skipped.
- evidence manifest using the phase identifier; use `scripts/spektrafilm_phase_manifest.py --phase` when useful, otherwise record the same facts manually with the reason the helper is not applicable

Validation Fixtures And Evidence

Keep fixtures deterministic and small. Each phase must say which automated fixtures or structural checks it runs and which it intentionally leaves to a later phase. Recommended

automated checks are the smallest scripts or commands that prove the touched behavior: consumed-field schema/catalog checks, route matrix checks, descriptor/hash structural checks, CUDA payload/resource binding checks, cleanup-symbol gates, and targeted reference-code or numeric comparisons where code review alone cannot prove parity.

Validation is capped by touched behavior. A phase selects the smallest useful row from the menu below, then narrows it to the specific route, descriptor, recipe group, or diagnostic it changed. The fixture route list is a reusable menu, not a backlog, and no phase creates future route/scanner/optics/grain scripts merely because those rows exist. Any phase that accepts a CUDA pixel route with new or changed durable resources, scratch, staging, or host derivation caches includes a two-frame steady-state resource check for at least one representative accepted route: frame N may create/cache resources, while frame N+1 with the same recipe/context/extent must only bind descriptor-matched resources and must not repack, reupload, rebuild stable LUTs/tables/factors, or allocate/free raw frame resources except for a family-contract-approved always-present diagnostic. For a route that can be invoked over partial render windows, acceptance also includes a same-frame multi-window or tile-style check: repeated windows for the same recipe/context/full-frame source may allocate per-window output/scratch only as declared, but must not rebuild or reupload durable resources, rederive stable LUT/filter/profile data, treat render-window coordinates as durable resource identity, or substitute tile-local metering for spektrafilm full-image preview metering.

Phase-local validation commands are part of acceptance when they exist and match the touched behavior. If a named command does not exist when the owning phase starts, that phase either adds the smallest needed script/check, records an equivalent targeted shell/structural check, or records an owner-run host artifact with a concrete skipped-automation reason in the phase manifest. Do not add a broad validation framework when a small structural check covers the slice.

Owner-run host evidence is accepted only when it contains a concrete artifact, not manual confidence. Each artifact records build configuration, route, profile keys, relevant settings, host action or command, generated log/trace/output path, expected result, observed result, and pass/fail criterion. Manual visual validation by the owner is required once comparable parity renders are possible, and should record matched route/profile/scanner/output settings plus artifact paths. When a repo-side command or structural check is feasible in the agent environment, owner-run evidence may supplement but not replace it. If automation is skipped, the phase note records the concrete reason and the follow-up automation debt.

Phase-specific validation menu:

Use the relevant row for the behavior touched by the task brief. A phase may use a narrower check when the manifest states why it covers the changed behavior. Prefer one primary automated or owner-run artifact plus targeted cleanup/search evidence; add a second primary artifact only when one check cannot cover the independent behavior touched by that phase.

For the Phase 3 validation row, evidence is accumulated in the Phase 3D serial order rather than collapsed into one implied check: Phase 3D-1 proves non-pass-through CUDA destination writes/publication, Phase 3D-2 proves matched negative-direct spektrafilm parity after canonical scanner and direct film-raw corrections, and Phase 3D-3 proves the DIR foundation. Each sub-gate records its own status and artifacts.

Phase	Validation evidence to choose from
1A catalog	Metadata-only catalog evidence: discovered `FilmProfileKey` / `PrintProfileKey` `StrChoiceParam` enum values from `Resources/profiles/*.json` stable profile keys plus effective spektrafilm metadata after `ProfileInfo` defaults; explicit metadata-source and profile-file classification/defaulting evidence, including ignored `Resources/filters/neutral_print_filters.json`, ignored legacy `Resources/profiles/enlarger_neutral*.json`, or an explicit profile-source manifest; proof `FilmProfileKey` includes negative and positive `support=film && stage=filming` profiles; proof `PrintProfileKey` includes `stage=printing` profiles regardless of `support` or `type`; duplicate-key and unsupported-role-metadata diagnostic evidence.
1A defaults/fallbacks	Explicit default key status for `kodak_portra_400` and `kodak_portra_endura`; proof neutral-filter data, print folders, first discovered entries, old default arrays, and old integer profile params are not visibility/default/compatibility fallbacks; any owner-approved `SF_TEMP_BRIDGE_ProfileKeyToLegacyRenderAsset` manifest entry names each retained non-rendering structural/diagnostic call-site family.
1A CPU cutoff	Product spektrafilm renders cannot use CPU pixel copy, CPU auto-exposure metering, CPU source-pixel reads, CPU destination writes, `processImpl()`, or `multiThreadProcessImages()` to produce output; any retained CPU validation helper is narrow, consumes explicit fixtures or subrecipe inputs, isolated from product rendering, subordinate to `external/spektrafilm`, and recorded with owner/purpose/removal gate.
1A render disposition	`SpektrafilmPixelPipelineNotImplementedForPhase1A` occurs before pending old `WorkingState` rebuild, old profile-index hash construction, old profile/index asset loading, old payload publication, frame preparation, or old CUDA pixels; Phase 1A-owned cleanup check status is recorded separately from future-phase cleanup symbols.
1B	Structural route matrix evidence for `NegativeDirectScan`, `NegativePrintScan`, `PositiveDirectScan`, and `PositivePrintScan`; no raw old route value crosses the recipe, frame-request, descriptor, CUDA payload, or hash boundary except a source-adjacent recorded bridge.
1C	Current-field `FrameRequest` disposition table exists before Phase 1C acceptance and before Phase 3 pixel behavior starts. Always includes preparation-boundary bridge inventory and cleanup gate report proving no new public prepared-frame/ensure surface was added.
2	Consumed-field/schema lock and reference evidence for Phase 2A, including consumed metadata only, Hanatos adaptation params only when consumed, JSON-null/NaN/Inf token policy for consumed numeric fields, optional inventory-only status for unconsumed fields such as `midscale_neutral_density`, and ignored legacy-key handling when current assets/code expose those keys. Loader/digest fixtures for Phase 2B are one or a few targeted selected-route checks, not a malformed-profile matrix: prove density-like `channel_density`, `base_density`,

and `density_curves` preserve `null`/`NaN` through loaded payload/hash/resource upload and convert to zero only at `density_to_light`; density NaN is not converted to transparent zero density; `log_sensitivity` applies `10 ** value` before `nan_to_num`; representative selected consumed-field failures such as `Inf`, malformed rows, missing samples, wrong channel counts, length mismatches, and wrong wavelength axes fail before recipe publication/resource preparation. Include targeted evidence that old `parse_optional_float_allow_nan`, global non-finite literal substitution, malformed-row NaN insertion, finite-count JSON completeness gates, CSV/profile-folder substitution, direct-route selected print-profile payload loading, and direct-route neutral selected-entry lookup are absent from the spektrafilm selected-route path or recorded as named non-product bridges. Copied-bundle source/hash plus cheap inventory/smoke evidence is enough for Phase 2C when useful. Do not require `spektrafilm_validate_bundle.py`, and do not add a bundle/profile linter for copied spektrafilm resources, unused metadata, user-file hardening, or migration behavior. |
| 3 | Phase 3A-3C may use focused structural or fixture evidence, but Phase 3D requires an owner-run host artifact proving negative-direct CUDA writes non-pass-through destination pixels and a matched `external/spektrafilm` reference comparison using canonical scanner LUT semantics with DIR disabled and scanner post effects explicitly identity. The artifact names route, profiles, output transform, LUT resolution, auto-exposure method, Hanatos adaptation window/surface/spectral-blur state, disabled/enabled effects, rendered output/reference paths, and numeric comparison result. Phase 3D also includes targeted evidence for endpoint-inclusive C/M/Y scanner grids, normalized coordinates, `log10(XYZ)` reference semantics, PCHIP/Hermite interpolation, per-cell min/max clamping, and equivalence of any alternative GPU storage; plug-in load, pass-through/unchanged output, blocked-output diagnostics, launch-only structural checks, and skipped host rendering fail the Phase 3D gate. Also include targeted auto-exposure evidence for all typed methods or a matrix artifact proving the GPU meter dispatches each method, structural evidence that same-frame auto-exposure scale remains device-resident with no required GPU-to-CPU readback/synchronize before film exposure, proof that auto-exposure metering descriptors/scratch are based on the spektrafilm max-256 nearest-neighbor preview rather than full-frame bounds, proof from a partial render-window invocation that the GPU meter reads the full source image/base when auto exposure is enabled or blocks with a diagnostic when the host cannot provide that memory, proof that unselected `RgbToRawMethod` resources produce no descriptor/upload/admission/hash work, proof that accepted direct-route CUDA launches do not populate/copy/hash resource-request fields for inactive stages through broad `PipelineRunParams`, Phase 3D full active spatial DIR foundation evidence before print-route pixel acceptance, explicit validation-only status for any non-spatial or spatial-identity DIR fixture, explicit pre-Phase-8 scanner post-effect disposition evidence including `ScannerPostEffectsNotImplementedForPhase3` behavior when default non-identity post effects are not implemented, and `DirNotImplementedForPhase3` behavior for default-active DIR until Phase 3D lands. |
| 4 | Negative and positive print-route fixture command, or owner-run host artifact naming print profile, capture film profile, illuminant, dichroic set, C/M/Y values including `FilmJuicerMainCFilterShift`, preflash/normalization state, glare state, DIR state, and scanner post-effect disposition; print-route pixel acceptance consumes Phase 3D recipe-owned full

active spatial DIR foundation, while any DIR-disabled, spatial-identity, or non-spatial print fixture is validation-only isolation evidence; evidence that print-balance factors, filtered main/preflash illuminants, preflash raw, and print/scanner LUT descriptor hits are descriptor/cache-owned host derivations that do not rebuild during repeated-frame or same-frame multi-window render launch when identities are unchanged; proof that accepted print-route CUDA launches do not populate/copy/hash resource-request fields for inactive stages through broad `PipelineRunParams`; plus `GlareNotImplementedForPhase4` behavior where print glare has not yet landed, `ScannerPostEffectsNotImplementedForPhase4` behavior where other default non-identity scanner post effects have not yet landed, and scanner post effects pruned only when the fixture/control state explicitly makes them identity. |
5	DIR completion fixture or targeted reference-code check covering any remaining gamma defaults, Velvia/Provia overrides, negative and positive silver-density rules, `diffusion_size_um`, `diffusion_tail_um`, `diffusion_tail_weight`, Gaussian/exponential mixture behavior, disabled-DIR no resources/hash/kernel work, disabled-spatial-DIR no tail scratch/resource work, default-active spatial route through `DirCouplersRecipe`, and spatial scratch descriptor binding when enabled.
6	Exact optics PSF/impulse, visual-inspection, and admission fixture when Exact is exposed, covering spektrafilm optics behavior, descriptor construction, admission failure diagnostics, resource lifetime, overlap leases/failure, disabled/identity component no-work evidence for each exposed Exact component, and no downgrade to another backend; otherwise manifest records that Exact remains hidden/blocked through `ExactOpticsNotImplementedForPhase6`.
8	Scanner correction/glare/blur/unsharp route fixture or owner-run host artifact covering all four routes, reuse of the Phase 3D canonical scanner LUT contract without direct-route semantic divergence, direct-vs-print glare, positive direct black/white correction apply behavior, negative direct correction skip behavior, one print route with print exposure correction plus scanner XYZ correction, black/white ordering, `ScannerSpectralLutDescriptor`/color/post-effect identity separation and legacy cleanup, blur/unsharp placement, output color/CCTF not rebuilding `ScannerSpectralLutDescriptor` identity, and route-by-route repeated-frame descriptor/cache-hit evidence proving stable scanner LUT/post-effect resources do not rebuild or reupload when recipe/context/extent are unchanged.
9	Grain contract fixture covering `density_min` route bounds, print-media exclusion of `density_min`, spektrafilm statistical field/default semantics when exposed or consumed, `density_curves_layers` upload/binding, positive/negative layer semantics, malformed layer diagnostics, grain-off no static-noise upload, and Film-Juicer visual-grain handoff.
10	Final route matrix, final copied-bundle source/inventory evidence, cleanup symbol gate report, reviewed allowlists, and final bridge/resource ownership review.

Fixture routes. This is a menu for choosing small, deterministic cases when they are relevant to touched behavior; it is not a required matrix for every phase:
- catalog-visible bundled profile inventory
- exposed dichroic filter-set resource sweep
- neutral gray ramp
- encoded-input gray/color patch with input CCTF decoding

- auto-only, manual-only, and auto-plus-manual exposure cases across `center_weighted`, `average`, `median`, `partial`, `matrix`, `multi_zone`, and `highlight_weighted` method sentinels
- black/invalid-meter case
- UV/IR advanced filter controls and Hanatos adaptation window/surface/spectral-blur sentinels
- chromatic patch set
- direct negative scan
- direct positive scan
- negative print scan
- positive print scan
- print exposure/preflash patch
- print Y/M/C slider order and C/M/Y recipe packing
- `FilmJuicerMainCFilterShift=0` spektrafilm parity case and nonzero C-filter Film-Juicer extension case
- dichroic filter-set selection across reference/custom default, `durst_digital_light`, `thorlabs`, and `edmund_optics`
- missing neutral calibration
- positive-film profile
- diffusion/halation impulse
- exact optics PSF/impulse and visual-inspection evidence when Exact is exposed
- scanner black/white correction cases, including positive direct apply, negative direct skip, and print-route print-exposure-plus-scanner-XYZ correction
- grain/layer sanity cases
- glare route cases
- asymmetric RGB/CMY/channel-order sentinels

Validation evidence classes:
- `ReferenceCodeReview`: compare implementation stage order, formulas, channel ordering, clamps, defaults, and resource inputs against spektrafilm source for the behavior touched by the phase.
- `ReferenceNumeric`: targeted numeric comparison against spektrafilm reference output at an approximation/acceleration boundary or comparable-render milestone. Use DeltaE 2000 for comparable rendered outputs; record route, profiles, scanner settings, LUT resolution, output transform, enabled/disabled effects, and the reason the comparison is meaningful. Do not require broad phase-wide tolerance gates.
- `ReferenceFixture`: small fixtures generated from `external/spektrafilm`, including boundary cases for NaN/null/non-finite handling, channel ordering, clamps, and stage order. These fixtures are the preferred replacement for broad CPU-renderer debugging.
- `StageContract`: narrow C++/CUDA checks that consume explicit fixtures or subrecipe inputs at loader, recipe/digest, descriptor/hash, CUDA payload packing, upload/staging, or small-kernel boundaries. They are never OFX product renderers.
- `OwnerVisual`: owner manual visual validation for parity-capable renders, recording matched settings, artifact paths, expected behavior, observed result, and pass/fail.
- `Structural`: verify route selection, recipe fields, hashes, resource descriptors, prepared-frame views, and diagnostics without claiming pixel parity.

- `Diagnostic`: intentionally malformed/missing inputs fail with the required failure class and identity fields.
- `Cleanup`: symbol/search gate proves old paths are deleted, blocked, or allowlisted.
- `EvidenceOnly`: old-output baselines may be recorded only as local deletion evidence when useful. They are not pass/fail targets, and old Film-Juicer saved-project behavior is never a validation target.

Do not use old Film-Juicer output or old saved-project behavior as an acceptance target. The normal target is spektrafilm reference behavior plus Host/OFX lifecycle correctness.

Grain visual matching is explicitly deferred to real grain-plate tuning. spektrafilm rendered grain is never the visual target for acceptance. Dust, scratches, and gate weave are also excluded from spektrafilm rendered-output parity. Glare is not excluded and should be matched through the scanner/print reference behavior for the active route.

Tooling Hooks

Expected tooling should stay phase-local and manifest-driven. This section is not permission to create a validation framework or a script per phase by default.
- `scripts/tidy_touched_files.ps1` runs targeted clang-tidy with the checked-in broad `.clang-tidy`.
- `scripts/format_touched_files.sh --check`, when present, checks clang-format for touched code. If a wrapper is unavailable, run the repository's direct clang-format command for the touched C++/CUDA files and record it.
- `scripts/spektrafilm_phase_manifest.py`, when present and useful, records phase evidence from explicit base/file manifests.
- `scripts/spektrafilm_cleanup_symbol_gates.sh`, when present and useful, verifies deleted/blocked/allowlisted old paths.
- `scripts/spektrafilm_cleanup_symbol_gates.sh --phase10-final`, when present and useful, is the final repo-wide Phase 10 cleanup selector; otherwise Phase 10 uses equivalent targeted cleanup checks.
- Phase-local validation scripts may exist only when a phase has touched behavior that needs repo-side evidence and a small command is clearer than owner-run host evidence. Prefer one existing bundle/cleanup/manifest command plus tiny focused checks over a script matrix.
- Script names are not part of the architecture. A phase must not pre-create schema, route, scanner, optics, grain, or final-sweep scripts merely because later rows mention those domains.

Touched files must come from explicit base/file manifests, not accidental dirty state.
Doc/script-only phases must still record why production-file tidy/build checks are not applicable.

Cleanup Gates

Cleanup runs every phase. Final cleanup is a verification sweep, not the place where major deletion first happens. The old OFX CPU renderer must already be blocked from product

rendering before final verification; any retained validation helper must already be narrow, fixture/subrecipe-driven, and isolated from product rendering.

For Phases 1A-9, search gates apply to touched files and explicitly named phase-owned symbols. Use repo-wide `src` scans only for Phase 10 or owner-approved broad sweeps. A phase-scoped gate passing does not approve untouched future-phase legacy code; it proves the phase did not leave untracked legacy behavior in the area it changed.

Phase 1A cleanup checks cover old profile identity, old catalog fallback/default entries in touched catalog/selection code, and CPU product-render cutoff/reference-fixture access. Route cleanup belongs to Phase 1B. Recipe/frame and prepared-frame bridge cleanup belongs to Phase 1C or the first owning resource phase.

A Phase 1A cleanup gate that scans touched files with a broad all-phase pattern set is not accepted unless the script also selects only Phase 1A-owned checks or the report separates Phase 1A-owned failures from future-phase symbols. Broad allowlists for unrelated `PrintBypass`, print-filter, DIR, scanner, optics, grain, or prepared-frame symbols in Phase 1A-touched files are not a substitute for phase/check selection.

Search gates should reject:
- old integer profile selection
- raw `PrintBypass`, `NegativeOnly`, `Print`, or `ScannerMedium` values used as route identity past the UI adapter or across the recipe, frame-request, descriptor, CUDA payload, or hash boundary
- old profile containers and index APIs
- global non-finite literal sanitizers, ungated `parse_optional_float_allow_nan`-style helpers, JSON-null-to-NaN substitution outside schema-approved fields, malformed-row NaN insertion, finite-count JSON completeness gates, or CSV/profile-folder substitution used to accept `Inf` or unusable selected-profile arrays in the spektrafilm loader path
- old folder/CSV paper substitution
- old neutral DB names under `Resources/profiles`
- old print filter units
- old dichroic-set code that selects 170-step wheel behavior instead of spektrafilm filter resources
- internal Y/M/C storage that bypasses the recipe-boundary conversion to C/M/Y CC units
- old normalized Y/M/C storage crossing the recipe boundary
- old per-set neutral DB default/substitution semantics
- retained `agx-emulsion parity` or old Film-Juicer parity comments in touched colour-science/render code unless the phase note proves the named behavior still matches current spektrafilm semantics or rewrites the comment to spektrafilm authority
- print-filter compatibility branches
- old shared exposure normalization
- old baseline names
- old DIR profile fields

- old `Couplers::define_params`, `Couplers::fetch_runtime`, `Couplers::Runtime`, or `JUICER_ENABLE_COUPLERS` stub paths used as the production DIR behavior contract instead of `DirCouplersRecipe`
- old halation `strength/size_um/scattering_strength/scattering_size_um` runtime path
- glare-compensation removal in the spektrafilm path
- scanner/enlarger density-bound recomputation outside `DensityBoundsRecipe`
- production CPU render entry points
- CPU pixel reference harnesses reachable from OFX/product render at any time
- old OFX CPU renderer or validation helpers reachable from OFX product render, product route matrices, production auto-exposure, product source/destination pixel writes, or any fallback path
- production CPU source-pixel reads for auto-exposure metering or other output-affecting render analysis
- public render-path `ensure_*` chains outside `PreparedCudaFrame` and context-owned resource-manager internals
- new or expanded `ResourceKind` / `ResourceKindContract` / `kResourceKindContract` entries that add schema interpretation, route/default policy, fallback behavior, pressure policy, or compatibility semantics instead of minimal mechanical admission/retire hooks
- private LUT/resource fallback paths used as normal spektrafilm product behavior instead of hard admission failure or an explicitly named output-identical descriptor-local strategy
- untracked `Root::prepare_cuda_frame` bridges that accept `WorkingState` directly as the normal spektrafilm preparation input
- new public effect-specific `PreparedCudaFrame::prepare_*` or public render-path `ensure_*` / `command_ensure_*` methods for spektrafilm resource families
- untracked prepared-frame bridge APIs that accept legacy render payloads
- untracked public `PreparedCudaFrame::prepare_*` methods that perform resource discovery, upload, allocation, schema interpretation, hash construction, or old-state translation after the frame-preparation boundary
- untracked `WorkingStateCorePayload` member-by-member mirrors in production CUDA payload/resource packing
- new CUDA preparation signatures that pass broad `WorkingState`, `Print::Runtime`, or `Print::Params` instead of narrow descriptors
- CUDA payload/resource packers that consume scattered legacy state instead of `RenderRecipe` or named subrecipes for touched spektrafilm behavior
- mutable `JuicerProcessor` side-channel fields or setters consumed as the real render contract after the corresponding value is required to flow through `FrameRequest`, `RenderRecipe`, or a named subrecipe
- accepted spektrafilm pixel behavior whose real recipe contract is still a flat expansion of `WorkingState` / `JuicerState.*` rather than focused recipe/domain ownership
- accepted spektrafilm pixel behavior whose `FrameRequest` contract is still implemented by copying broad fields into mutable `JuicerProcessor` members and consuming those members as policy
- spektrafilm durable resources without a resource-family descriptor/hash owned by the implementing phase

- spektrafilm scratch/staging resources that bypass `FrameScratchDescriptor` and prepared-frame ownership
- Exact optics policy hidden inside scanner post-effect, visual-grain, gate-mask/static-noise, or broad `OpticsKernelView` / workspace-request ownership instead of `SpatialOptics` descriptors
- unconditional static grain/defect asset uploads when visual grain, dust, scratches, gate weave, and other static-noise consumers are disabled
- raw CUDA allocation ownership introduced in render code after frame preparation begins
- steady-state per-frame raw `cudaMalloc`/`cudaFree`, `cudaMallocHost`/`cudaFreeHost`, CUDA event create/destroy, or equivalent transient staging churn for accepted CUDA pixel routes when the same descriptor could be admitted, leased, retained, or justified as a bounded always-present diagnostic family
- ad hoc `pixelSizeUm` derivation outside `FrameRequest` construction or descriptor preparation from `FrameRequest`
- silent exact-to-fast downgrade
- first-catalog-entry substitution
- fake required profile defaults

Search gates should allow only with reviewed allowlists:
- user-facing labels such as `Durst Digital Light`, `Thorlabs`, and `Edmund Optics`
- typed `DichroicFilterSet` values for spektrafilm resources
- Y/M/C UI parameter names when converted into recipe-owned C/M/Y CC values
- validation-only helpers and intentionally retained host diagnostics
- narrow validation-only helper code with no product render call site and no parity authority over `external/spektrafilm`
- named temporary prepared-frame bridges listed in this plan or the phase manifest, with exact allowed call sites and removal phase
- `PreparedCudaFrame` view/lease accessors that expose already-prepared resources without doing preparation work
- context-owned resource-manager internals whose `ensure_*` names are private implementation details behind `PreparedCudaFrame`

Allowlist entries must be narrow and include owner, reason, phase or bridge/default name, source-adjacent marker status for editable code, and removal phase when the symbol is temporary. Do not use broad compatibility-language allowlists to hide real old paths.

Temporary allowlist entries must use the same bridge name recorded in the phase manifest and source-adjacent marker, and include `SF_TEMP_BRIDGE` or `SpektrafilmBridge` so they are searchable. A cleanup gate passing because of an unnamed allowlist or manifest-only bridge name is not accepted.

Readiness Standard

The port is ready when:
- all four routes render through CUDA

- Phase 3D produced the first matched, non-pass-through negative-direct host/reference parity artifact using canonical scanner LUT semantics; later phases did not proceed on structural-only or pass-through rendering evidence
- spektrafilm behavior is the only production behavior
- no intermediate blocked-output release exception is being used as final readiness evidence; pre-readiness builds are internal-only unless owner-approved as blocked-output releases, and no build may render old Film-Juicer/agx pixels under spektrafilm profile keys
- positive print scan follows the normal print route
- optional optics default to off
- spektrafilm `settings.preview_mode` remains unsupported, no Film-Juicer Preview optics backend exists in this port, and no host preview/draft hint changes output
- no Fast optics backend, Fast UI, Fast resources, or Exact-to-Fast fallback is part of this port
- exact optics matches spektrafilm optics behavior or fails clearly when requested
- Film-Juicer grain remains the production visual grain implementation
- host CPU code is limited to loading, recipes, minimal consumed-field checks, hashing, diagnostics, and resource prep
- any retained CPU validation helper is narrow, fixture/subrecipe-driven, isolated from product rendering, and subordinate to `external/spektrafilm`; no broad C++ CPU pixel reference renderer, production CPU render backend, fallback, or host-visible CPU render route remains
- owner-supplied copied-bundle source/inventory evidence covers catalog-visible bundled profiles and exposed required resources
- final scanner/enlarger ranges are owned by `DensityBoundsRecipe`
- all durable GPU resources are context-owned
- all mutable frame scratch is prepared-frame owned
- `WorkingState` is a small immutable recipe envelope, and meaningful recipe behavior lives in focused recipe/domain files rather than in a flat `JuicerState.*` expansion
- no untracked prepared-frame bridges remain
- normal Release builds do not pay for heavy diagnostics or validation-only instrumentation
- every phase has cleanup, build evidence or owner-run build reason, clang-format and clang-tidy evidence for touched code, relevant automated route/fixture/structural checks, owner manual visual validation once parity renders are possible, and cleanup-gate evidence sized to the phase
- once rendering works, the final owner output-validation check has a complete artifact package and is either passed or explicitly recorded as pending owner-run validation
- Phase 10 confirms earlier cleanup rather than performing a large first-time deletion sweep; missed major cleanup fails readiness and returns to the owning earlier phase or owner-approved cleanup phase

Provenance Appendix

- This plan is the active spektrafilm port contract for this workspace.
- `.tmp/spektrafilm-diff/film-juicer-port-map.md` is unavailable in this workspace.

- The available local references are `external/spektrafilm`, `.tmp/spektrafilm-diff/spektrafilm-agx-emulsion-pipeline-diff-map.md`, and the Film-Juicer source tree.
- Missing historical notes do not weaken the normative plan requirements.