**Optimizor 1.0 — Concrete Action Catalog (UE 5.6) 13/08/25**

A chaptered, implementation-ready checklist of **exact actions** the plugin performs. Each action lists **What**, **How we decide**, **User controls**, and **Safety/Reporting**. Scope covers **Textures, Meshes, Materials**, plus **Cross-cutting** systems and **Preset Advice Bundles**.

**0) Cross‑Cutting Systems (applies to all actions)**

* **Scan Scope**  
  **What:** Enumerate assets by selection, include list, or project root; respect exclude lists and skip collections.  
  **How:** EditorAssetLibrary.list\_assets (fallback AssetRegistry), dedupe by path.  
  **Controls:** Dry run, Only selected, Include/Exclude CSV, Change cap.  
  **Safety:** Cancellable slow task; per-change cap; skip locked/read-only.
* **Change Application**  
  **What:** Write properties via safe setters with pre/post edit and package dirtying.  
  **How:** Central safe\_set(); optional auto-checkout; close editors before writes.  
  **Controls:** Save assets, Auto flush dirty, Auto checkout, Close editors, Verify after write.  
  **Safety:** Verify after save; retry post\_edit\_change; collect failures.
* **Collections & Reports**  
  **What:** Create Collections for *Changed*/*Skipped*; write TXT/CSV/JSON; stash revert snapshot.  
  **How:** ReportSvc; Saved/Optimizor/History/\*.json.  
  **Controls:** Report dir/basenames; include memory bytes; timestamped filenames.  
  **Safety:** Always dry-run first from UI; history list in Reports tab.
* **Preset Gatekeeping**  
  **What:** Each preset defines apply\_only keys and hard caps per action family.  
  **How:** JobRunner filters attempted edits; gate high‑risk mutations in Conservative mode.  
  **Controls:** apply\_only, max\_changes per run; per-target toggles.  
  **Safety:** Exceptions list (path/name tags), skip collection.

**1) Textures — Exact Actions**

1. **Compression Class Assignment**  
   **What:** Set CompressionSettings to TC\_Normalmap, TC\_Masks, TC\_Grayscale, TC\_Default, TC\_HDR for LUTs.  
   **How:** Filename heuristics + optional **material sampler inference** (looks at referencing MaterialExpressionTextureSample.sampler\_type).  
   **Controls:** Per‑type override (Color/Normal), inference on/off, referencer cap.  
   **Safety:** Gate via apply\_only: ["compression"]; verify after write.
2. **sRGB Flag Correction**  
   **What:** Toggle sRGB=true for color, false for normal/masks/linear.  
   **How:** Same heuristics+inference as above.  
   **Controls:** srgb\_color, srgb\_normal.  
   **Safety:** Report mismatches; never flip on LUTs/HDR.
3. **Mip Generation Policy**  
   **What:** Set MipGenSettings (TMGS\_FromTextureGroup, sharpen/blur variants, NoMipmaps for UI/LUT).  
   **How:** Group policy; UI and LUT force no mips; presets can override.  
   **Controls:** generate\_mips, explicit mip\_gen enum.  
   **Safety:** Gate via apply\_only: ["mip\_gen"].
4. **LOD Group Enforcement**  
   **What:** Assign LODGroup (World, WorldNormalMap, UI, ColorLookupTable).  
   **How:** Classification (UI/Normal/LUT) and project rules (e.g., Mobile\* LOD groups if present).  
   **Controls:** Optional group force per preset.  
   **Safety:** Off in Conservative mode unless preset explicitly allows.
5. **Never Stream for UI/LUT**  
   **What:** NeverStream=true for UI atlases, fonts, LUTs.  
   **How:** Path and tag hints (/UI/, sprite, icon, font, lut).  
   **Controls:** apply\_only: ["never\_stream"] via preset.  
   **Safety:** Skip large world-space decals unless preset says so.
6. **Virtual Texture Discipline**  
   **What:** Disable VirtualTextureStreaming for most regular textures; **respect Landscapes/UDIM**.  
   **How:** Path hints + RESPECT\_VT\_ON\_LANDSCAPES; per‑preset override.  
   **Controls:** virtual\_texture=true/false (semantic: allow vs force off).  
   **Safety:** Never disable VT on known landscape/UDIM assets.
7. **LOD Bias Lower Bound**  
   **What:** Enforce LOD Bias >= 1 on non‑UI, non‑small/detail textures when preset wants memory savings.  
   **How:** Name hints (detail, small dims), preset flag.  
   **Controls:** ENFORCE\_LOD\_BIAS\_1.  
   **Safety:** Gate with apply\_only: ["lod\_bias"].
8. **Size Caps (Advisory + Optional Reimport)**  
   **What:** Detect over‑budget resolution per channel type; optionally scale on import rule (advice only by default).  
   **How:** MaxSizeColor/Normal/Mask per preset.  
   **Controls:** Advice vs enforce; per‑project toggle.  
   **Safety:** Default to *advice only*; emit hotlist.
9. **Memory Impact Estimation**  
   **What:** Pre/Post byte estimate (format × res × mips).  
   **How:** Table lookup; report items and CSV.

**2) Static Meshes — Exact Actions**

1. **Nanite Enable/Disable**  
   **What:** Toggle bNaniteEnabled and set fallback ScreenSize.  
   **How:** Preset target, triangle count thresholds, blacklist (foliage/physics‑heavy).  
   **Controls:** nanite.enable, nanite.fallback\_screensize.  
   **Safety:** Skip if per‑asset tag \_keep/\_\_nofix present.
2. **LOD Generation (Triangles/Screen Curve)**  
   **What:** Generate or re‑tune LODs via IMeshReduction; set % triangles and ScreenSize per LOD.  
   **How:** Strategy: GroupFirst (enforce LODGroup percents) → cap by triangle ceilings → apply screens curve.  
   **Controls:** lods.strategy, caps per LOD, screens array, group name.  
   **Safety:** Dry‑run diff table; preserve existing LODs if better.
3. **LOD Group Enforcement**  
   **What:** Apply project LODGroups (e.g., Mobile\*) when present.  
   **How:** Lookup UStaticMesh::LODGroup.  
   **Controls:** lods.group with enforce=true/false.  
   **Safety:** Clamp instead of overwrite when authoring intent is stricter.
4. **Lightmap UVs (Auto‑Generate)**  
   **What:** Generate missing lightmap UVs; set Min/Max Lightmap Resolution.  
   **How:** Run built‑in charting; overlap detection.  
   **Controls:** lightmap\_uvs.generate, min\_res, max\_res.  
   **Safety:** Skip meshes with manual channel tags; report overlaps.
5. **Collision Simplification**  
   **What:** Create simple convex if missing; remove redundant complex-as-simple when Nanite.  
   **How:** Analyze BodySetup primitives count.  
   **Controls:** collisions.auto\_simple=true.  
   **Safety:** Respect physics interaction tags (VR props keep detailed colliders).
6. **Optional Merge Actors (Editor‑Only)**  
   **What:** Batch merge by cluster to reduce draw calls for baked sets.  
   **How:** Editor utility call; keep source assets if Conservative.  
   **Controls:** merge\_actors.enabled.  
   **Safety:** Only in dry‑run unless explicitly enabled.

**Reporting:** per‑mesh tri counts per LOD, Nanite stats, UV/channel summary; Collections for *Changed*, *Skipped*.

**3) Skeletal Meshes — Exact Actions (Phase 2.5)**

1. **LOD Generation / Reduction**  
   **What:** Generate LODs via IMeshReduction for Skeletal; set ScreenSize curve.  
   **How:** Triangle thresholds per preset; preserve imported LODs.  
   **Controls:** skel.lods.strategy, caps, screens.  
   **Safety:** Never touch cloth-painted sections by default.
2. **Animation Compression Policy**  
   **What:** Set compression schemes (ACL/Default) per preset.  
   **How:** Apply to AnimSequences in scope; skip cinematic takes.  
   **Controls:** anim.compression=preset.  
   **Safety:** Dry‑run delta size + max error snapshot.

**4) Materials & Instances — Exact Actions**

1. **DX/GL Normal Convention Audit**  
   **What:** Flag projects mixing OGL vs DX normal maps; identify suspect materials.  
   **How:** Scan texture names and sampler types; compare green channel assumptions.  
   **Controls:** Report‑only (advice); optional auto‑flip mask outputs in MI (off by default).  
   **Safety:** No auto rewiring by default.
2. **Static Switch Cleanup**  
   **What:** Remove unused static branches; collapse constants.  
   **How:** Graph analysis in C++; optional Python material traversal.  
   **Controls:** static\_switch\_cleanup=true/false.  
   **Safety:** Save backup and diff summary.
3. **Sampler Consolidation**  
   **What:** Detect duplicate samplers (same texture/settings) and suggest reuse.  
   **How:** Hash sampler states.  
   **Controls:** Report‑first; auto‑fix optional.  
   **Safety:** Only within MI where safe.
4. **Packed Map Hints**  
   **What:** Detect \_orm/\_rma usage; advise BC5/BC4 opportunities and channel wiring guidance.  
   **How:** Cross‑check texture names + material pins.  
   **Controls:** Advice bundle per preset.

**5) Levels/World — Advisory Checks (Report‑Only v1)**

* **Foliage Cull & LOD Budget**  
  **What:** Report foliage types with missing cull distances or LOD ranges off budget.  
  **How:** Inspect FoliageType.  
  **Controls:** Thresholds per preset.
* **HLOD / World Partition Tiles**  
  **What:** Report maps without HLOD cluster assets; budget draw calls.  
  **How:** Query HLOD setup; emit steps to generate.
* **Lighting Pipeline Mismatch**  
  **What:** Detect incompatible shadow quality vs platform preset.  
  **How:** Project Settings snapshot + preset advice.

*(All world checks are non‑mutating in v1; they attach advice text to the report.)*

**6) Preset Advice Bundles (auto‑attached to reports)**

Each preset ships an **Advice.md** snippet injected into the report:

* **PC High**: prefers BC7 for color, keep mips sharpen ≤ 0.25; use Nanite for static props; maintain LODBias=0 unless VRAM bound.
* **Console Balanced**: cap color at 2k, normals/masks at 1k; enable Nanite broadly; prefer BC5 for normals (no alpha); avoid VT for small props.
* **Mobile Low**: downscale aggressively; enforce Mobile\* LODGroups; avoid VT; atlas UI into non‑PoT allowed; disable per‑pixel normal where costly.
* **VR**: favor stability (low shimmer); mipgen Sharpen ≤ 0.25; collisions simplified only for non‑interactive; ensure 90/120 Hz budgets.
* **Cinematic**: BC7 for hero color; BC5 for normals; VT allowed on UDIM/heroes; keep LODBias=0; no auto merge.
* **UI Crisp**: No mips; NeverStream; clamp addressing; linear color for grayscale masks.

**7) Action→Code Map (where each action lives)**

* **Textures**: Python module texture\_doctor.py (classification, inference, safe writes, reporting). Entry texture\_doctor\_entry.py bridges C++ JSON to Python.
* **Meshes**: C++ MeshDoctor subsystem (new) using IMeshReduction, NaniteTools, MeshUtilities. Report via ReportSvc.
* **Materials**: Mixed C++/Python traversal; C++ graph utilities; Python for CSV/JSON enrichment.
* **World Checks**: C++ Editor utility functions (read‑only) producing advice blocks.

**8) UI/UX Hooks (SOptimizorTab)**

* Tabs: **Textures | Meshes | Materials | Reports**.
* For each action group: checkboxes map to apply\_only keys; advanced drawers per family.
* Footer: **Scan** (forces dry‑run) and **Auto‑Fix** (honors safety gates).
* One‑click **Create Collections** and **Open Report Folder**.

**9) Milestones (Macro → Micro)**

1. **M1 — Texture Foundation**: finalize schema; fix current TD bugs; robust reporting; Collections; revert snapshot; preset advice injection.
2. **M2 — Static Meshes**: Nanite toggles; LOD generation; lightmap UVs; collision simplifier; CSV stats.
3. **M2.5 — Skeletal/Anim**: Skeletal LODs; animation compression policy.
4. **M3 — Materials**: DX/GL audit; static switch cleanup; sampler consolidation.
5. **M4 — World Checks & Polish**: non‑mutating map audits; history UI; headless CLI.

**10) Known Issues in Current Prototype (quick fixes)**

* **Duplicate functions in texture\_doctor.py**: two gather\_textures() defs — keep the second (selection‑aware), delete the first.
* **STextureDoctorPanel.cpp typos**: duplicated Json var; WriteTextToSavedFile() uses undefined Abs (should be Path).
* **Enum plumbing**: ensure C++→Python enums are passed as ints and resolved via \_enum\_from\_int(); keep BOM‑tolerant JSON I/O.
* **ACTIVE\_PRESET label**: unify name across C++ and Python (use preset).

This catalog is the **source of truth** for implementation and QA. Next, we can take each chapter and design the exact UI controls and the code pathways (C++/Python) that perform the actions.