# QoreStor Refcount Migration Script — Ultra-Technical Function Reference

## Global setup & invariants

- `set -euo pipefail`

- \*\*e\*\*: exit immediately on non-zero exit status (except in guarded contexts).

- \*\*u\*\*: treat unset variables as an error.

- \*\*pipefail\*\*: a pipeline fails if any command fails.

- `exec 3>&2`

- \*\*FD semantics\*\*: Duplicates current stderr (FD 2) onto FD 3. All debug uses FD 3 so it bypasses later redirection in `setup\_logging` and never pollutes command substitutions.

- \*\*Key globals (selected)\*\*

- Mode flags: `DRY\_RUN`, `SCAN\_ONLY`, `VERIFY\_CHECKSUM`, `TEST\_MODE`, `DEBUG\_MODE`.

- Layout: `ALT\_LAYOUT`, `REFCNT\_SUBPATH`, `R3\_JOURNAL\_PATH`.

- Paths: `CONFIG\_FILE`, `BACKUP\_DIR`, `LOG\_DIR`, `LOG\_FILE`, `MOUNTPOINT`, `NEW\_LINE`.

- Config toggles: `REFCNT\_OLD`, `REFCNT\_NEW`.

- Accounting: `SCAN\_FOUND`, `SCAN\_TOTAL\_BYTES`, `COPIED\_FILES`, `SUMMARY[]`.

- \*\*External commands used\*\*: `system`, `systemctl`, `awk`, `sed`, `grep`, `numfmt` (optional), `tail`, `tr`, `tee`, `stdbuf`, `rsync`, `du`, `df`, `mountpoint`, `find`, `mktemp`, `wc`, `cmp`, `diff`.

- \*\*Invariants\*\*

- Path computation uses `REFCNT\_SUBPATH` (either `.ocarina\_hidden/refcnt` or `refcnt`).

- All debug prints go to FD 3 via `debug\_\*`, never contaminating stdout.

- Config edits always via temp file; backup exists beforehand in live mode.

- Services are stopped before live copy; started after config edit (and optional verify).

## Execution order (why this order)

1. `parse\_args` → establish mode switches and capture free args.

2. `setup\_logging` → tee stdout/stderr early so logs include everything.

3. `detect\_layout\_once` → set `ALT\_LAYOUT`, `REFCNT\_SUBPATH`, `R3\_JOURNAL\_PATH` so all subsequent FS ops are correct.

4. Mode branches:

- \*\*SCAN\_ONLY\*\*: `scan\_refcnt\_sizes` → `print\_summary`.

- \*\*DRY\_RUN\*\*: `decide\_dryrun\_target` → prechecks → sim copy/verify → config preview → (optional) simulated start → `print\_summary`.

- \*\*LIVE\*\*: `setup\_mountpoint` → `confirm\_live\_run` → `make\_backup` → prechecks → stop → copy → (optional) verify → `apply\_changes` → start → `print\_summary`.

5. Any failure gates print summary and exit early to avoid partial state.

## Function-by-function (technical)

### `usage()`

\*\*Purpose\*\*: Print CLI usage and exit when requested.

\*\*Inputs\*\*: none.

\*\*Outputs\*\*: help to stdout.

\*\*Side effects\*\*: none.

\*\*Deps\*\*: none.

\*\*Errors\*\*: n/a.

\*\*Complexity\*\*: O(1).

\*\*Callers\*\*: `parse\_args` on `-h|--help` or unknown flag error.

\*\*Why here\*\*: User-facing guidance at argument parse-time.

### `banner(text, color=$CYAN)`

\*\*Purpose\*\*: Emphasize log sections with ANSI colors.

\*\*Inputs\*\*: `text`, optional `color`.

\*\*Outputs\*\*: stdout.

\*\*Side effects\*\*: none.

\*\*Complexity\*\*: O(1).

\*\*Callers\*\*: many.

\*\*Why\*\*: Readability and log segmentation.

### `decide\_dryrun\_target()`

\*\*Purpose\*\*: In dry-run, determine if a \*real\* destination is available to simulate copying/verify more realistically.

\*\*Inputs\*\*: `DRY\_RUN`.

\*\*Outputs\*\*: human prompts to stdout; debug to FD 3.

\*\*Side effects\*\*: sets `DRY\_HAS\_TARGET`, `DRY\_SKIP\_SERVICES`, potentially calls `setup\_mountpoint` to validate and form `NEW\_LINE`.

\*\*Deps\*\*: interactive TTY for `read`.

\*\*Errors\*\*: none (logic switches).

\*\*Edge cases\*\*: Non-tty invocation will block; intended for interactive testing.

\*\*Complexity\*\*: O(1).

\*\*Callers\*\*: `main` dry-run branch.

\*\*Why here\*\*: Determines whether to simulate end-to-end (with a target) or plan-only (no target).

### `capture\_system\_info()`

\*\*Purpose\*\*: Print a stable subset of `system --show` for context.

\*\*Inputs\*\*: external `system` tool.

\*\*Outputs\*\*: stdout.

\*\*Side effects\*\*: none.

\*\*Deps\*\*: `system`, `grep`.

\*\*Errors\*\*: if `system` missing, warning and return 0.

\*\*Complexity\*\*: O(n) in lines of `system --show`.

\*\*Callers\*\*: `setup\_logging`.

\*\*Why\*\*: Early in logs to anchor environment details.

### `detect\_layout\_once()`

\*\*Purpose\*\*: One-shot detection of standard vs. alternate layout by reading config state and path existence.

\*\*Inputs\*\*: `CONFIG\_FILE`.

\*\*Outputs\*\*: debug to FD 3.

\*\*Side effects\*\*: sets \*\*globals\*\* `ALT\_LAYOUT`, `REFCNT\_SUBPATH`, `R3\_JOURNAL\_PATH`.

\*\*Deps\*\*: `grep`, `awk`, `sed`.

\*\*Errors\*\*: tolerant (uses `|| true`), returns 0 always.

\*\*Edge\*\*: If config says alt but path missing, remains standard.

\*\*Complexity\*\*: O(n) lines of config.

\*\*Callers\*\*: `main` early.

\*\*Why\*\*: Downstream FS functions depend on correct layout.

### `detect\_alt\_repo\_from\_config()`

\*\*Purpose\*\*: Pure helper to print `R3\_DISK\_JOURNAL\_PATH` when alt criteria satisfied.

\*\*Inputs\*\*: `CONFIG\_FILE`.

\*\*Outputs\*\*: prints path on success; exit 0/1.

\*\*Side effects\*\*: none.

\*\*Deps\*\*: `awk`.

\*\*Edge\*\*: Handles CRLF, quotes, export/no-export, `${...}` vs `$...`.

\*\*Complexity\*\*: O(n) lines.

\*\*Callers\*\*: helper retained.

\*\*Why\*\*: Diagnostic or optional split of concerns.

### `get\_repo\_location()`

\*\*Purpose\*\*: Emit the repository root dir according to layout.

\*\*Inputs\*\*: `ALT\_LAYOUT`, `R3\_JOURNAL\_PATH`, external `system`.

\*\*Outputs\*\*: repo path on stdout; errors on stderr.

\*\*Side effects\*\*: none (debug FD 3).

\*\*Deps\*\*: `system`, `awk`, `sed`.

\*\*Errors\*\*: returns 1 if tool missing or path invalid.

\*\*Edge\*\*: trailing spaces trimmed; ensures directory exists.

\*\*Complexity\*\*: O(n) lines of `system --show`.

\*\*Callers\*\*: `scan\_refcnt\_sizes`, `plan\_copy\_totals`, `copy\_all\_refcnt`, `verify\_all\_refcnt`.

\*\*Why\*\*: Single source of truth for repo root.

### `to\_bytes(s)` / `human\_bytes(b)`

\*\*Purpose\*\*: Bidirectional size conversions with robust fallbacks.

\*\*Inputs\*\*: `to\_bytes` accepts “123”, “123,456 bytes”, or IEC suffix. `human\_bytes` prints IEC.

\*\*Outputs\*\*: stdout integer / formatted string.

\*\*Deps\*\*: optional `numfmt`, otherwise `awk`.

\*\*Errors\*\*: tolerant → defaults to stripped digits or 0.

\*\*Complexity\*\*: O(1).

\*\*Callers\*\*: scanning/printing.

\*\*Why\*\*: Consistent size reporting independent of platform numfmt availability.

### `run\_with\_bar "$cmd" ...`

\*\*Purpose\*\*: Execute rsync (or similar) with a live progress bar (LIVE) or a simulated bar (DRY).

\*\*Inputs\*\*: command vector. Detects `-n/--dry-run`.

\*\*Outputs\*\*: progress bar on stdout; underlying command’s non-% lines reprinted; returns underlying exit code.

\*\*Side effects\*\*: spawns background tail/reader; temporary file; relaxes `errexit`/`pipefail` internally and restores.

\*\*Deps\*\*: `mktemp`, `stdbuf`, `tail --pid`, `tr`, `grep`.

\*\*Edge\*\*: If underlying command doesn’t print `%`, bar won’t advance (expects rsync `--info=progress2`).

\*\*Complexity\*\*: O(output lines).

\*\*Callers\*\*: `copy\_one\_refcnt` (LIVE), `copy\_all\_refcnt` (LIVE).

\*\*Why\*\*: User feedback for long transfers without contaminating stdout capture.

### `simulate\_bar()`

\*\*Purpose\*\*: Background simulated progress for dry-run UX.

\*\*Inputs\*\*: none.

\*\*Outputs\*\*: prints progress loop until killed by caller.

\*\*Side effects\*\*: infinite loop; caller must `kill`/`wait`.

\*\*Callers\*\*: DRY-RUN branches in `copy\_\*`.

\*\*Why\*\*: Perceived responsiveness during stat scans.

### `debug\_log`, `debug\_printf`

\*\*Purpose\*\*: Print debug messages \*\*to FD 3\*\* (original stderr).

\*\*Inputs\*\*: `DEBUG\_MODE`.

\*\*Outputs\*\*: FD 3 only.

\*\*Why\*\*: Never pollute stdout captured via `$(...)` given later tee redirection.

\*\*Callers\*\*: many.

### `parse\_args "$@"`

\*\*Purpose\*\*: Normalize flags into global booleans; collect free args → `PARSED\_ARGS`.

\*\*Inputs\*\*: CLI.

\*\*Outputs\*\*: globals mutated.

\*\*Errors\*\*: unknown `-\*)` prints error and exits 2. `--help` exits 0.

\*\*Edge\*\*: supports `--` to end options.

\*\*Callers\*\*: `main`.

### `setup\_mountpoint()`

\*\*Purpose\*\*: Prompt/validate `MOUNTPOINT` (path format + existence/mount checks by mode); compute `NEW\_LINE="export TGTSSDDIR=$MOUNTPOINT/ssd/"`; create `.../ssd` live only.

\*\*Inputs\*\*: `PARSED\_ARGS`, `DRY\_RUN`, `TEST\_MODE`, `SCAN\_ONLY`.

\*\*Outputs\*\*: to stdout; updates `SUMMARY`.

\*\*Side effects\*\*: sets `MOUNTPOINT`, `NEW\_LINE`; may `mkdir -p` live.

\*\*Deps\*\*: `mountpoint`.

\*\*Errors\*\*: loop until valid; prints color errors.

\*\*Complexity\*\*: O(1) prompt loop.

\*\*Callers\*\*: `decide\_dryrun\_target` (dry-run only if target), `main` (LIVE).

\*\*Why\*\*: Derive config edit line and target base; ensures safety constraints per mode.

### `setup\_logging()`

\*\*Purpose\*\*: Split stdout/stderr to the same logfile via separate `tee` processes; leave FD 3 untouched. Print run headers and system info.

\*\*Inputs\*\*: `TIMESTAMP`, `LOG\_DIR`.

\*\*Outputs\*\*: log file + banners.

\*\*Side effects\*\*: redirects FD 1/2 process-wide.

\*\*Deps\*\*: `mkdir`, `tee`, `stdbuf`.

\*\*Edge\*\*: Using `-o0 -e0` for unbuffered tee improves progress smoothness.

\*\*Callers\*\*: `main`.

\*\*Why\*\*: Capture complete run artifacts without contaminating stdout-only captures.

### `get\_system\_state()`, `get\_system\_reason()`

\*\*Purpose\*\*: Small extractors over `system --show`.

\*\*Outputs\*\*: strings; if missing, blank.

\*\*Deps\*\*: `system`, `awk`, `sed`.

\*\*Callers\*\*: service control.

### `verify\_ready\_to\_stop()`

\*\*Purpose\*\*: Snapshot states prior to issuing a stop (or dry-run narrative).

\*\*Outputs\*\*: banners + info; `SUMMARY` entries.

\*\*Deps\*\*: `systemctl`, `system`.

\*\*Errors\*\*: none (informational).

\*\*Callers\*\*: all modes.

\*\*Why\*\*: Visibility/gating before disruptive operations.

### `wait\_for\_service\_stop(service="ocards")`

\*\*Purpose\*\*: Issue `systemctl stop`, then polling loop of `get\_system\_state` until “Stopped” or timeout.

\*\*Outputs\*\*: live-overwritten status lines; `SUMMARY`.

\*\*Side effects\*\*: stops service (LIVE).

\*\*Errors\*\*: returns 1 on failed stop or timeout.

\*\*Callers\*\*: dry-run (narrative only) and live.

\*\*Why\*\*: Ensure FS quiet before copy ensures consistency.

### `start\_services()`

\*\*Purpose\*\*: `systemctl start`, then loop until `System State=Operational Mode`, `Reason="Filesystem is fully operational for I/O."`, and `systemctl is-active == active` OR timeout.

\*\*Outputs\*\*: live-overwritten status lines; `SUMMARY`.

\*\*Errors\*\*: returns 1 on timeout/failure.

\*\*Callers\*\*: dry-run (simulated) and live.

\*\*Why\*\*: Ensure system returns healthy post-migration.

### `plan\_copy\_totals()`

\*\*Purpose\*\*: Count total files to consider under `<repo>/<id>/$REFCNT\_SUBPATH` using `find -type f`.

\*\*Inputs\*\*: repo from `get\_repo\_location`; `REFCNT\_SUBPATH`.

\*\*Outputs\*\*: integer to stdout.

\*\*Errors\*\*: returns 1 if repo resolution fails.

\*\*Complexity\*\*: O(number of files).

\*\*Callers\*\*: copy planning, previews.

\*\*Why\*\*: Preflight expectations for copy/verify.

### `safe\_rsync ...`

\*\*Purpose\*\*: Guard-rail — if `DRY\_RUN=true`, enforce presence of `-n/--dry-run` in args; otherwise delegate to `rsync`.

\*\*Outputs\*\*: passthrough.

\*\*Errors\*\*: returns `99` if policy violated.

\*\*Callers\*\*: all rsync-invocations that must respect dry-run.

\*\*Why\*\*: Prevent accidental writes during dry-run regressions.

### `scan\_refcnt\_sizes()`

\*\*Purpose\*\*: Per-`<id>` `du -sb` accumulation on `$REFCNT\_SUBPATH`; prints per-dir human sizes + total.

\*\*Inputs\*\*: repo, `REFCNT\_SUBPATH`.

\*\*Outputs\*\*: human-readable; updates `SUMMARY`, sets `SCAN\_FOUND`, `SCAN\_TOTAL\_BYTES`.

\*\*Errors\*\*: continues on missing refdirs; returns 0 even if 0 found (with message).

\*\*Complexity\*\*: O(number of dirs/files).

\*\*Callers\*\*: scan-only, live preview, dry-run contexts.

\*\*Why\*\*: Size planning & visibility.

### `check\_free\_space(target\_base, need\_bytes)`

\*\*Purpose\*\*: Ensure `df -PB1` available space ≥ needed bytes; creates target dir (LIVE).

\*\*Outputs\*\*: prints free/required; error on insufficient.

\*\*Errors\*\*: return 1 if insufficient; 0 if unknown (`df` missing).

\*\*Callers\*\*: `copy\_all\_refcnt` (LIVE and dry-run with target).

\*\*Why\*\*: Avoid mid-transfer ENOSPC.

### `rsync\_base\_flags()` / `rsync\_verify\_flags()`

\*\*Purpose\*\*: Emit canonical rsync flags via printed lines (consumed into arrays).

\*\*Base\*\*: `-aHAX --numeric-ids --sparse -W --human-readable --dirs`.

\*\*Verify\*\*: base plus `--checksum` iff `VERIFY\_CHECKSUM=true`.

\*\*Outputs\*\*: list to stdout.

\*\*Callers\*\*: `copy\_\*`, `verify\_\*`.

\*\*Why\*\*: Centralize rsync policy.

### `copy\_one\_refcnt(SRC, DST, base)`

\*\*Purpose\*\*: Copy a \*\*single\*\* `refcnt` subtree (handles callers passing a repo `<id>` dir instead of explicit `.../refcnt`).

\*\*Inputs\*\*: `REFCNT\_SUBPATH`, `DRY\_RUN`, `LOG\_FILE`.

\*\*Normalization\*\*: If `SRC` isn’t ending in `/refcnt` but `SRC/$REFCNT\_SUBPATH` exists, rewrite `SRC`/`DST` to the subtree.

\*\*Dry-run\*\*: `safe\_rsync` with `-n --stats` into tmpfile; `simulate\_bar`; parse “Number of regular files transferred” → report “would be transferred”.

\*\*Live\*\*: run `run\_with\_bar rsync` with `--info=progress2`; print final stats.

\*\*Outputs\*\*: human lines; `SUMMARY`; increments `COPIED\_FILES`.

\*\*Errors\*\*: live path returns rsync RC on failure; writes errors.

\*\*Complexity\*\*: O(files under subtree).

\*\*Callers\*\*: Not used by bulk path (bulk uses `copy\_all\_refcnt`), but correct independently.

\*\*Why\*\*: Reusable primitive, safe normalization.

### `verify\_one\_refcnt(SRC, DST)`

\*\*Purpose\*\*: Rsync-based comparison (size/time or checksum).

\*\*Dry-run\*\*: `-n --stats` to report would-compare counts.

\*\*Live\*\*: Run rsync and interpret exit code + output: \*\*no output\*\* means identical; any output indicates deltas (treated as mismatch).

\*\*Outputs\*\*: banners + `SUMMARY`.

\*\*Errors\*\*: non-zero RC or non-empty output → failure.

\*\*Callers\*\*: Not in main bulk flow; `verify\_all\_refcnt` preferred.

\*\*Why\*\*: Reusable primitive.

### `copy\_all\_refcnt()`

\*\*Purpose\*\*: Bulk copy of all refcnt \*\*files\*\* across all `<id>` dirs.

\*\*Flow\*\*:

1) `planned\_files=$(plan\_copy\_totals)` → print/summary.

2) `repo=$(get\_repo\_location)` guard.

3) If dry-run without target → plan-only (no `df`, no rsync).

4) Compute `target\_base` from `NEW\_LINE` (strip `export TGTSSDDIR=` and trailing `/`).

5) `check\_free\_space` using `SCAN\_TOTAL\_BYTES`.

6) Build `filelist` with `find . -type f -path "\*/${REFCNT\_SUBPATH}/\*"` under repo root.

7) DRY: `safe\_rsync --files-from ... -n --stats` with simulated bar → print stats and totals.

8) LIVE: `run\_with\_bar rsync --files-from ... --info=progress2` → “Rsync completed” + summary; set `COPIED\_FILES`.

\*\*Outputs\*\*: detailed logs; `SUMMARY`.

\*\*Errors\*\*: aborts on guards or rsync failure.

\*\*Complexity\*\*: O(files).

\*\*Why\*\*: Efficient bulk path; parent dirs created implicitly by file copies; `--dirs` retained.

### `verify\_all\_refcnt()`

\*\*Purpose\*\*: Bulk \*\*per-`<id>`\*\* verification of `SRC=$repo/<id>/$REFCNT\_SUBPATH` vs `DST=$target\_base/<id>/$REFCNT\_SUBPATH`.

\*\*Flow\*\*:

- DRY: `safe\_rsync -n --stats $(rsync\_verify\_flags)` per dir → report “would be compared” and totals.

- LIVE: ensure `DST` exists; run `safe\_rsync --stats $(rsync\_verify\_flags)`; treat \*\*non-empty output\*\* or non-zero RC as mismatch/failure.

\*\*Outputs\*\*: per-dir results; totals; `SUMMARY`.

\*\*Errors\*\*: fail fast on first mismatch; returns 1.

\*\*Complexity\*\*: O(files).

\*\*Why\*\*: Strong validation immediately after copy (optionally checksum).

### `config\_preview\_live()`

\*\*Purpose\*\*: For LIVE preview phase: show intended insert + refcnt toggle change (informational only).

\*\*Note\*\*: The actual replace-or-insert logic is in `apply\_changes` / `dry\_run\_preview`.

\*\*Outputs\*\*: preview banners; `SUMMARY`.

\*\*Callers\*\*: `confirm\_live\_run`.

\*\*Why\*\*: Let operator confirm config impact before live run.

### `dry\_run\_preview()`

\*\*Purpose\*\*: Render a \*\*temp transformed config\*\* (using awk \*\*replace-or-insert\*\* logic) and show a colorized diff.

\*\*Logic (awk)\*\*:

- Replace first `export TGTSSDDIR=...` with `$preview\_line`, discarding any duplicates.

- Else insert after `export TGTDIR`, else append at end.

- Replace `PLATFORM\_DS\_REFCNTS\_ON\_SSD=0` with `=1`; ensure `=1` exists exactly once (no duplicates).

\*\*Outputs\*\*: `[DRY RUN]` preview and colored diff; `SUMMARY`.

\*\*Callers\*\*: dry-run branch.

\*\*Why\*\*: Safe visualization before live changes.

### `make\_backup()`

\*\*Purpose\*\*: Create timestamped backup of `CONFIG\_FILE` into `BACKUP\_DIR`, preserving mode/owner (`cp -p`).

\*\*Outputs\*\*: path; banners; `SUMMARY`.

\*\*Why\*\*: Rollback safety.

### `apply\_changes()`

\*\*Purpose\*\*: Apply \*\*the same\*\* awk transform as preview, but write to `${CONFIG\_FILE}.tmp`, compare with `cmp`, and `mv` into place when different.

\*\*Diff\*\*: show colorized diff vs `BACKUP\_FILE`.

\*\*Dedupe\*\*: ensures single `TGTSSDDIR` and single `PLATFORM\_DS\_REFCNTS\_ON\_SSD=1`.

\*\*Errors\*\*: If no changes, print “already up-to-date”.

\*\*Why\*\*: Deterministic, idempotent config update with audit.

### `print\_summary()`

\*\*Purpose\*\*: Emit mode, checksum setting, dry-run target availability, and all accumulated `SUMMARY[]` lines; finish with log file path.

\*\*Outputs\*\*: human-readable.

\*\*Why\*\*: Single consolidated end-of-run artifact.

### `confirm\_live\_run()`

\*\*Purpose\*\*: Live \*\*gating\*\* step: show preview banners, scan totals, planned files, and config preview; require explicit `yes` to proceed.

\*\*Outputs\*\*: interactive prompt; `SUMMARY`.

\*\*Why\*\*: Human-in-the-loop confirmation reduces risk.

### `main "$@"`

\*\*Purpose\*\*: Orchestrator for all modes with strict failure gates and clear summaries.

\*\*Key guards\*\*:

- Missing `CONFIG\_FILE` after confirmation → abort.

- Stop failure → abort.

- Copy failure → abort before config.

- Verify failure (when enabled) → abort before config.

- Start failure → abort.

\*\*Why\*\*: Ensure \*\*no partial migration\*\* and clear operator feedback.

## Safety & correctness notes

- \*\*FD discipline\*\*: Debug to FD 3, logs split to FD 1/2 ensures `repo="$(get\_repo\_location)"` is clean even with logging active.

- \*\*Rsync enforcement\*\*: `safe\_rsync` prevents accidental writes during dry-run regressions.

- \*\*Deterministic config edit\*\*: replace-or-insert logic guarantees \*\*one\*\* `TGTSSDDIR` and \*\*one\*\* `PLATFORM\_DS\_REFCNTS\_ON\_SSD=1`.

- \*\*Progress UX\*\*: `run\_with\_bar` isolates fancy output from correctness (bar derived from `%` scan lines; summary reprinted sans `%`).

- \*\*Service gating\*\*: stop before copy, verify, then config, then start — preserves consistency.

## Quick test checklist

- \*\*Resolve repo\*\*

`DEBUG\_MODE=true repo="$(get\_repo\_location)" && printf '%s\n' "$repo"` → should print path only.

- \*\*Scan-only\*\*

`./script --scan-only` → per-dir lines + total; summary present.

- \*\*Dry-run plan-only\*\*

`./script --dry-run` (answer “no”) → plan-only copy/verify counts; config preview diff.

- \*\*Dry-run with target\*\*

`./script --dry-run /mnt/foo` (dir exists) → simulated copy/verify with counts; config preview with real line.

- \*\*Live smoke (test mount, no service impact)\*\*

`TEST\_MODE=true ./script --verify /mnt/foo` → will still enforce confirmations; use test env.

- \*\*Config transform dry-run\*\*

Run `dry\_run\_preview` and confirm diff shows a single `TGTSSDDIR=...` and `PLATFORM\_DS\_REFCNTS\_ON\_SSD=1`.