p4-cache — Engineering Maturity Brief

A sales asset for buyers who ask "is this actually production-ready, or is it a hobby project with a marketing site?" The honest answer in numbers and citations.

TL;DR

p4-cache is a ~40K-line Rust workspace (six crates) that has been through three independent code audits and shipped remediation against all three. The workspace enforces compile-time defenses against entire classes of bugs (every unsafe operation requires explicit `unsafe { }` blocks; mutex guards across `.await` boundaries are a compile error). Workspace lints include `cast_sign_loss = "warn"`, `cast_possible_truncation = "warn"`, and `must_use_candidate = "warn"`. CI gates include `cargo audit` and `cargo deny check`.

This brief exists because most year-1 commercial software has zero audit trail; p4-cache has three. That fact is worth being explicit about.

By the numbers

Metric	Value	Note
Lines of Rust	~40,000	Across six workspace crates (see below)
Workspace crates	6	`p4cache` (daemon + tools), `p4cache-trigger` (per-call archive trigger), `shared` (types + protocol), `license` (signing/verification), `license-tool` (vendor CLI), `integrity-stamp` (release-artifact stamping)
Shipped release binaries	4	`p4-cache` (long-resident daemon), `p4cache-trigger` (static per-call archive trigger), `p4-cache-verify` (checkpoint/journal verifier), `p4-cache-manifest-summary` (upload-state reporter); plus the vendor-side `p4-cache-license` and `p4cache-integrity-stamp` build tools
Tests	378	`#[test]` + `#[tokio::test]`
Test files	80	Unit modules, integration tests, property tests, fuzz harnesses
Independent audits + review passes completed	4	`.audit/` (QA), `.codex-audit/` (revalidation), `.audit-verify/` (verifier binary),

		`REVIEW_FIXES.md` (2026-05-20 internal full-workspace code-review pass — 213 findings, all 21 HIGHs closed)
Open CRITICAL findings	0	"Data loss / RCE / auth bypass / secret leak / persisted state corruption / known-exploit CVE in direct dep"
Open HIGH engineering findings	0	All 21 HIGHs from the most recent (2026-05-20) review pass are closed in clusters 1–7 of `REVIEW_FIX_PLAN.md`. Prior HIGHs (`S-004` integrity-failed staging-path RAII, `S-verify-101` verifier argv hardening) remain closed; per-finding entries in the `*-REMEDIATION.md` trackers cite the fix commit and the pinning test.
Open HIGH supply-chain advisories	2	Both transitive dependencies awaiting upstream SDK bumps: `S-001` (`rustls-webpki 0.101.7` pinned by `aws-smithy-http-client`'s legacy `hyper-rustls 0.24` ladder), `S-003` (`quinn-proto 0.11.13` pinned by `tonic` in `google-cloud-gax`; HTTP/3 not used by the daemon). Both ignored in `rust/.cargo/audit.toml` with per-entry rationale.
`TODO`/`FIXME`/`HACK` in production code	2	Both `TODO` markers in `license/policy.rs`, flagging step-7+ user-agent / metadata propagation work; tracked, not silent.
`panic!`/`unimplemented!`/`todo!` in non-test paths	3	All bug-on-impossible-state, not unhandled cases: 2 in the storage backend contract test for post-delete `head_object` divergence, 1 in `license/load.rs` for an unparseable build-time `P4CACHE_BUILD_DATE` (build.rs ensures this never fires in shipped binaries).
`#[ignore]` tests	0	Every test runs every build
`unsafe` blocks in daemon	~58	Every one carrying a `// SAFETY:` comment; sites cover `dl_iterate_phdr` integrity self-hash, `clock_anchor` defense, license audit-log v2, SCM_RIGHTS fd-passing, and `SO_PEERCRED` plumbing

Stress / fallback shell harnesses	9	`p4_v2_read_stress.sh`, `p4_storage_benchmark.sh`, `p4_cache_read_stress.sh`, `p4_azure_nfs_fallback_test.sh`, `p4_commit_peak_hour_stress.sh`, `p4_read_stress_nocache.sh`, `p4_submit_storage_compare.sh`, `p4_gzip_parity.sh`, plus `compare_benchmarks.sh` for criterion baseline regression checks
`cargo fuzz` targets	2	Trigger request handler (`fuzz_handle_request`), Perforce checkpoint journal parser (`fuzz_checkpoint_parser`)
ADRs documenting major decisions	3	0001 panic strategy, 0002 content-hash schema (the legacy `FileEntry` hash evolution that informed v2's MD5 archive-content verification and the narrow `archive_catalog`/`upload_manifest` `redb` stores), 0003 peer-cred allowlists (all under `docs/ADR/`)
Direct Rust dependencies	~30	Reviewed; transitive advisories tracked in `rust/.cargo/audit.toml`
Rust edition	2024	Workspace-pinned
MSRV	1.88	Pinned in workspace `Cargo.toml`; let-chains in `if`/`while` are the current floor feature

Compile-time defenses

The workspace `[lints]` section enforces, *at compile time*:

Lint	Level	What it prevents
`unsafe_op_in_unsafe_fn`	deny	Rust 2024 idiom: every unsafe operation inside an `unsafe fn` body must sit inside an explicit `unsafe { ... }` block. Catches UB the outer signature used to hide.
`await_holding_lock`	deny	Holding a `MutexGuard` or `RwLockGuard` across an `.await` is a compile error. Prevents deadlock-under-load that would otherwise be a runtime bug.

`cast_sign_loss`	warn	Catches silent `i32 as u32` / `i64 as u64` / `isize as usize` sign-flip casts. Surviving same-width sign changes go through `.cast_unsigned()` as an explicit intent marker.
`cast_possible_truncation`	warn	Catches silent narrowing casts. Surviving sites use the `wrapping_as_*` helpers in `p4cache_shared` (intentional wrap) or `u32::try_from(x).expect("…")` (panic-on-bug at a boundary).
`must_use_candidate`	warn	Catches new public functions that should advertise their return value as `#[must_use]`. The `CacheConfig` accessor wall carries a local `#[allow(...)]` so the lint stays useful for new methods without spamming on obvious getters.

`panic = "unwind"` is set in both `[profile.release]` and `[profile.dev]` so the per-connection `catch_unwind` guard in the trigger listener can convert a handler panic into a structured ERROR log without killing the daemon. The daemon is unwind-safe (no FFI boundaries on the hot path; tokio + redb are unwind-safe; `Drop` impls release locks/fds correctly).

These aren't documentation. They're enforced. A change that violates any of them fails CI.

CI gates

`.github/workflows/ci.yml` includes:

- `cargo build --release` (workspace-wide)
- `cargo test`
- `cargo fmt --check`
- `cargo clippy -- -D warnings`
- `cargo audit` (advisory scan)
- `cargo deny check advisories bans licenses sources` (supply-chain policy)
- `cargo fuzz` smoke runs on both fuzz targets (~60 s each)
- Cross-build job: `windows-shared-build` builds lib + trigger + daemon binary on `windows-latest` and uploads the trigger artifact

A change that introduces a new transitive advisory, an unapproved license, a banned dependency, or a fuzz crash fails CI. The advisory ignore list in `rust/.cargo/audit.toml` is documented per-entry with the upstream tracker URL.

Audit history in detail

Audit pass 1 — full-workspace QA audit (.audit/)

Six phases: inventory, quality, architecture, security, performance, tech debt.

Inventory: 23K LoC at audit time, dep graph mapped, workspace topology documented.

Quality (40 findings): Mostly per-crate cleanups. Notable: 729 pedantic clippy warnings catalogued (most cosmetic); ~85 `unwrap`/`expect` sites in production paths reviewed (mostly bounds-guarded but flagged for hardening). One flaky test (pre-existing, lock-file collision).

Architecture (17 findings): Several large modules identified as refactoring targets. Storage backend duplication of `temp + fsync + rename + assert_regular + fsync_parent` plumbing identified.

Security (25 findings): Transitive vulns in `rustls-webpki`, `aws-lc-sys`, `quinn-proto`. A large number of `unsafe` blocks lacked `// SAFETY:` comments. No fuzz targets. No deny.toml.

Performance (12 findings): 52 MiB default build; no benchmarks; double-pass disk I/O on restore (write-then-hash).

Tech debt (10 findings): Zero `TODO`/`FIXME`/`HACK` markers. Three `#[allow]`s, all justified. Four unused direct deps.

Audit pass 2 — codex re-audit (.codex-audit/)

Re-validation after pass-1 remediation landed. Deeper storage-SDK retry/perf review across S3, Azure, GCS. Doc and packaging drift review. Reproduced the automated gate spot-checks.

HIGH findings at the time of the pass — all now closed:

- `S-004`: Integrity-failed restores left rejected bytes at the live depot path. Closed by the staging-path RAII landing in the restore pipeline — the cache write only atomic-renames onto the live archive path after the content-hash check passes.
- `S-006`: cargo-audit gate was red on active transitive advisories. Closed by tightening `rust/.cargo/audit.toml` to per-advisory ignore entries with documented rationale, plus a `cargo update` for the items where an upstream fix existed (`rustls-webpki` to 0.103.13, `aws-lc-sys` to 0.40.0). The remaining transitive ignores are tracked in `S-001`/`S-003` under "open HIGH supply-chain advisories" above.

Plus MEDIUMs clustered into argv hardening, logging unification, resource limits, OsString-clean argv, live-shelf cap, and architecture lift — all closed or partially closed and tracked in `CODEX-REMEDIATION.md`.

Audit pass 3 — verify-binary audit (.audit-verify/)

Targeted verification audit of the `p4-cache-verify` checkpoint binary. Inventory complete; manual review of CLI argv handling, checkpoint/journal parser, source enumeration, state-dir lifecycle, backend exists-checking, verify runtime.

One HIGH finding (`S-verify-101`): depot path bytes flowed into `p4 verify` argv unfiltered. **Closed.** Two complementary defenses landed: `storage_row_from_fields` now rejects rows whose `depot_path` does not start with `//` (with a structured WARN), and a parser test (`skips_row_whose_depot_path_does_not_start_with_doubled_slash`) pins the validator. The argv-`---`-terminator approach was tried and reverted after `p4 verify` rejected it; the `//` prefix validator is the defense in the shipping code.

Plus MEDIUMs, including secondary-error-swallowing in `BackendSet::PrimaryAndSecondary`, OsString-vs-String argv parsing, sequential backend probes, and unbounded `live_shelves` HashSet on long-lived servers.

Review pass 4 — full-workspace internal code review (REVIEW_FIXES.md + REVIEW_FIX_PLAN.md, 2026-05-20)

Eight focused review agents ran in parallel across the workspace; collated output: **213 findings** (21 HIGH + 82 MED + 110 LOW), sequenced into a 10-cluster fix plan with atomic per-cluster commits. Severity is operator-impact: HIGH = data loss / privilege escalation / license bypass / daemon crash / audit-compliance gap.

HIGH-finding closure status (all 21 closed):

- **Cluster 1 — License + integrity hardening (10 HIGH, all closed across 1a/b/c/d).** Self-hash check is now hard-fail at startup with a `dl_iterate_phdr` walk over every `PT_LOAD`/`PF_X` segment in `p_vaddr` order, defeating split-text patches that single-section hashing missed. `P4CACHE_LICENSE` env override is gated on root-owned-0600 + symlink rejection + audit-logged. New `clock_anchor` module persists wall-clock-at-load and refuses startup on backward motion above the 1-hour NTP tolerance. License audit log bumped to v2: monotonic `seq` numbers, RFC 8785 canonical HMAC input via `serde_jcs`, cross-restart anchor file (`license-audit.anchor.json`) for truncation detection. Algorithm pinning between `vendor_key_id` and `signature_alg` with constant-time `key_id` comparison; explicit-length-checked signature parsing; past-`expires_at` and ≥ 1 EiB capacity rejected at parse time. Developer-build license writes plaintext unchained audit lines with a prominent header so dev diagnostics don't masquerade as production audit history.
- **Cluster 2 — Privilege boundary (2 HIGH, all closed across 2/2b/2c).** `peer_auth` refuses `MSG_CTRUNC`/`MSG_TRUNC` and validates `cmsg_len` + 8-byte cmsg-buf alignment; stale-socket cleanup gated behind `S_ISSOCK`; bind→publish uses `hard_link` (atomic EEXIST) instead of `rename`.
- **Cluster 3 — Config reload (4 HIGH, all closed).** `LiveConfig::apply_from` audit: fields whose consumers capture a value at startup are restart-required; fields backed by atomic loads are hot-reloadable. `restart_required_diff` expanded to cover pid_file, log_file, metrics_file, every ES/PG sink connection field, `access_spool_dir`, and both backend blocks via `backend_configs_eq` with `Redacted::expose`. Rotation task split so failed-uploads age-rotation runs independently of main-log size-rotation.
- **Cluster 4 — Restore + manifest (1 HIGH, closed in 4).** Restore parent-fsync moved *after* the manifest state transition (closes the disk-leak HIGH where a crash between rename and the state update orphaned manifest state). Semaphore acquire-Err path explicit; manifest iteration errors propagated; summary counters use `saturating_add`.
- **Cluster 5 — Storage (1 HIGH, closed in 5).** ETag-pinning on ranged downloads closes a mid-download object-replacement bug. The backend is a single `ObjectStoreBackend` wrapper over the Apache Arrow `object_store` crate.
- **Cluster 6 — Checkpoint verifier (2 HIGH, closed in 6).** Parser skip-and-warn on schema mismatch + per-line parse failure (closes a false-positive abort HIGH); `shelf_delete_key` anchored on `EXPECTED_REV_FIELD_COUNT` + index 15 (closes wrong-shelf-removal on malformed `@dv@` rows). Plus wallclock cap on the `p4 changes`/`p4 info` subprocesses, verify `.lock` opened with `O_NOFOLLOW`, NFS exists-checker `..` substring rejection.
- **Cluster 7 — Audit log + log_rotate + metrics (1 HIGH, closed in 7).** `note_failure` seeds `disabled_retry_at` on the threshold-trigger path so the immediate-retry-after-disable contract bug is closed; `manifest_counter_drift_total` mirrored locally so backward jumps surface; metrics export floors at 1 s on pathological 0/0.

Remaining work (no HIGH; all MED + LOW): cluster residuals 4c (5 items), 5c (4 items), 6c (~10 items), 7b (~7 items); plus cluster 9 (config + main remaining — 9 MED + 8 LOW) and cluster 10 (LOW cleanup tail — ~35 items). Per-cluster commit log + fix-plan ground rules live in `REVIEW_FIX_PLAN.md`; "build + `cargo test -p`

`p4cache` after every cluster, no red tree" is the standing rule and every closure commit in the log cites a pass count.

Remediation pattern

Each audit pass produces a synthesis report. Each report's findings get tracked in a `*-REMEDIATION.md` file at the repo root. The CHANGELOG records what landed:

> *The QA audit at `.audit/6-synthesis/REPORT.md` was remediated against this branch. Per-finding tracker: `QA-REMEDIATION.md`. Highlights: supply chain (cargo update + deny.toml + CI gate); build size (32.7 MiB default, 9.2 MiB NFS-only); criterion benchmarks; SAFETY comment pass; storage trait contract test; fuzz harness; property tests; ADRs under `docs/ADR/`.*

This is not unusual for a mature commercial product. It *is* unusual for a year-1 product.

What the audits did not find

Documented because their absence is informative:

- **No data-loss CRITICAL.** No bug, in any audit, where p4-cache could destroy customer depot data without operator action.
- **No RCE.** No bug, in any audit, where untrusted input becomes code execution.
- **No auth bypass.** The peer-credential allowlist mechanism (ADR-0003) was reviewed across both subsequent audits without bypass findings.
- **No secret leak.** `Redacted<T>` wrapper applied to all credential types; verified in audits 2 and 3.
- **No persisted state corruption path.** The two narrow `redb` stores — `archive_catalog` (a local digest cache of `(depot_path, lbr_rev) -> {md5, gzipped}`) and `upload_manifest` (the durable Dirty/Uploading/Clean async-upload state machine) — use redb's single-writer single-tx model, atomic at the storage layer; their record codecs are forward- and backward-tested. db.storage stays the replicated, authoritative digest source of truth.

The closest the audits came to CRITICAL was `S-001`/`S-002`/`S-003` in pass 1 — transitive vulns in `rustls-webpki`/`aws-lc-sys`/`quinn-proto`. None exploited under the threat model (all required attacker-on-the-wire). All resolved by `cargo update`.

What the audits are still working on

Honest list, current to the latest review-pass execution log:

High-priority, bounded: *None.* All 21 HIGHs from the 2026-05-20 internal review pass are closed in clusters 1–7 of `REVIEW_FIX_PLAN.md`. Prior-audit HIGHs (`S-004`, `Q-p4cache-006`, `S-verify-101`) also closed and pinned by regression tests.

Supply-chain HIGH (deferred to upstream):

- `S-001` — `rustls-webpki 0.101.7` remains as a transitive pinned by `aws-smithy-http-client`'s legacy `hyper-rustls 0.24` ladder; no fix exists in the 0.101 line. Tracked in `rust/.cargo/audit.toml` with rationale; cargo-deny does not raise this one under v2 settings.

- `S-003` — `quinn-proto 0.11.13` pinned by `tonic` in `google-cloud-gax`. HTTP/3 is not used by the daemon. Tracked the same way.

Residual MED + LOW (from `REVIEW_FIX_PLAN.md`):

- Cluster 5c (storage residual): async cleanup of temp files + expected-size check, per-chunk `to_vec` avoidance for multi-GB downloads.
- Cluster 6c (checkpoint verifier residual): parser send-channel root-cause, runtime flush-on-error guard, runtime tracker fixes, checkpoint canonicalize bail, `prune_orphan_source_states` valid-filenames, shelf/CRLF validation, `progress.rs` rotation race + writer exit, runtime drop on submit error, sources strict suffix.
- Cluster 7b (access-log residual): `MSG_TRUNC` recv handling, spool re-validate on read, ES index allowlist, chmod-failure surfacing, 32-bit batch panic floor, Postgres `.expect` → `Transient` classification.
- Cluster 9 (config + main remaining — 9 MED + 8 LOW), cluster 10 (LOW cleanup tail — ~35 items): not started.

Low-priority, hygiene:

- Remaining `cast_possible_truncation` warning sites — most are FFI-shaped (`socklen_t = u32`); replace with explicit `try_from` or `wrapping_as_*` helpers as touched.
- Duplicate crate versions in the dep graph — natural consequence of the cloud SDK matrix exposed through `object_store`; not a bug.

Recently landed engineering hygiene:

- Shared `download_parallel_to_temp` driver in `storage/mod.rs` — single driver for the parallel range-GET path.
- Shared `pwrite_chunk_at` and `plan_ranges` helpers.
- `write_atomic_with` helper in `checkpoint_verify/state.rs` — consolidates the temp/rename/parent-fsync skeleton shared between `write_json_atomic` and `write_specs_atomic`.

The remediation pattern from prior audits shows these typically close within weeks of someone picking them up. None block customer deployment; all are visible and tracked.

How to read this brief

If you're a *technical evaluator*, every claim here is verifiable in the source repo. The audit synthesis reports are in `.audit/6-synthesis/REPORT.md`, `.codex-audit/6-synthesis/REPORT.md`, and `.audit-verify/6-synthesis/REPORT.md` respectively. Per-phase findings are in the numbered subdirectories. ADRs are in `docs/ADR/`. The lint policy is in `rust/Cargo.toml` under `[workspace.lints]`. CI is `.github/workflows/ci.yml`.

If you're a *commercial buyer*, the headline is: **this is software built by someone who treats it like software people pay for, not like a side project.** The audit infrastructure is the evidence. The remediation track record is the evidence. The compile-time defenses are the evidence.

Most products at this stage don't have this story to tell. The ones that do are the ones their customers stop worrying about.

Document control

- Version 1.0
- Maintained against repo HEAD
- Numbers verified by source grep at the date of last update
- Audit report citations link to the actual files in the source tree