RockyGuard Library - Future Releases Roadmap

Tracking deferred items for v1.2.1, v1.3, and beyond

Document: RockyGuard_Future_Roadmap.txt
Library: RockyGuard C++17 License Library
Created: 2026-04-25
Status: LIVING DOCUMENT (updated as items land or are reprioritized)

Distribution: Internal. Sets scope expectations for v1.2.1 and v1.3 cycles.

This document consolidates every item that surfaced during the v1.2.0 QA cycle but was deliberately deferred to a future release. Each entry records (a) what the issue is, (b) where it was first surfaced, (c) why it was deferred from v1.2.0, and (d) the target release.

The first release after v1.2.0 ships is v1.2.1 (a point release scoped strictly to documentation and build-side polish, no breaking ABI). The release after that is v1.3 (a minor release where the first feature work, broader platform support, and ABI-affecting opt-in additions can land).

1. Scope and ground rules
2. v1.2.1 - Point Release (documentation and build polish)
2.1. Documentation polish (Phase 4.1 deferred items)
2.2. Build artifacts (Debug-CRT library)
2.3. Server-side limitation (log rotation)
3. v1.3 - Minor Release (features, platform expansion)
3.1. Build / packaging improvements
3.2. Continuous integration
3.3. Platform expansion
3.4. Performance regression
3.5. Non-blocking load API
3.6. Logging API
3.7. Upgrade / downgrade testing (Phase 8)
4. v1.4 and beyond - Speculative / unscheduled
5. Cross-reference: where each item was surfaced
6. Process improvements banked from v1.2.0

1. SCOPE AND GROUND RULES

v1.2.1 (point release)

- Strictly non-breaking: same public ABI, same shipped headers, same CMake imported targets.
- Doc fixes, clarifications, and additive build artifacts (e.g. a Debug-CRT-built library alongside the existing Release-CRT one).
- Should ship within ~2 weeks of v1.2.0 if the deferred-doc items are the bottleneck. If a v1.2.0 customer hits a real defect first, that defect is added to v1.2.1 scope immediately.

v1.3 (minor release)

- Allowed to add new public API (set_log_callback, find_package imported targets, etc.).
- Allowed to expand platform support (manylinux_2_28, macOS).
- Allowed to land the first automated CI gates and a performance regression suite.
- Becomes the first release where Phase 8 (upgrade / downgrade) testing has a target: customers with v1.2.0 deployed will be the upgrade source.

v1.4+ (speculative)

- Genuinely additive ideas not yet committed to any cycle. Listed here so they are not lost; not promised.

2. v1.2.1 - POINT RELEASE

2.1 Documentation polish (Phase 4.1 deferred items)

These items were surfaced in the Phase 4.1 customer onboarding dry-run and explicitly deferred from v1.2.0 to keep the ship window narrow. None blocks a customer who reaches their vendor license through the sales channel; together they polish the cold-read experience.

ID	Severity	Description
GA-001	HIGH	Customer_Documentati on.txt Section 2.2 "How you receive your vendor license" subsection: support / sales contact channel, expected delivery format, sample filename, troubleshooting (what to do if missing), and a one-paragraph description of what the file contains and how the library will use it.
GA-006	LOW	Customer_Documentati on.txt Section 3.1 (Static linking): add an explicit sentence "Static builds have no separate integrity file - the library's code is compiled directly into your executable, so rockyguard.sig is not needed." Mirror in Section 9.3.
GA-010	LOW	Pick one canonical form for license expiry dates (date-only "2027-12-31" used in API Reference Section 8 vs full ISO 8601 "2027-12-31T23:59:59Z" used in Customer_Documentati on.txt Section 4.3). Document tolerance explicitly.
GA-011	LOW	Customer_API_Referen ce.txt Section 2 "Namespace and Headers" omits export.h. Either add it to the list with a note that it carries the ROCKYGUARD_API export macro, or note that it is internal-only.
GA-012	LOW	Customer_Documentati on.txt: add a "What to ship to your end user" section with a per-variant checklist (Static vs Shared linking; what files go in the deployment directory; what is needed at runtime).

Source: qa/onboarding/RESULTS.md gap list; QA Report Section 4.2.7 forensic record.

2.2 Build artifacts (Debug-CRT library)

ID	Severity	Description
GA-013 (medium-term)	HIGH (was)	Ship a Debug-CRT-built library alongside the Release-CRT one so customers building a Debug consumer application can link without LNK2038 / LNK1319 / mismatch on _ITERATOR_DEBUG_LEVEL or RuntimeLibrary.

Implementation options under consideration:

Option A (additive, preferred):

```
- Add lib/static/rockyguard_mdd.lib (Debug CRT, /MDd)
- Add deps/lib/libssl_mdd.lib + libcrypto_mdd.lib
- Update the shipped CMake imported target (or the
  rockyguard-config.cmake landing in v1.3) to select the
  correct variant based on $<CONFIG>.
- No code change; only build-and-package.sh and CMake.
```

Option B (replace):

```
- Rebuild rockyguard.lib with /MT (static CRT) so the
  consumer's CRT choice no longer has to match.
- Drawback: forces rebuild of bundled OpenSSL with /MT, and
  static CRT has its own ergonomics issues (multiple CRT
  copies in the final binary). Not recommended unless A
  proves operationally hard.
```

The v1.2.0 short-term fix (CRT-variant caveat in Customer_Documentation Section 3.1) is sufficient to keep v1.2.0 customers unblocked: they can build in Release mode or request a Debug-CRT library out of band. The v1.2.1 work removes the friction.

Source: qa/onboarding/RESULTS.md GA-013; QA Report Section 4.2.8.

2.3 Server-side limitation (log rotation)

Phase	Description
5.4.3	Floating-server log file grows without bound. A long-running server eventually fills the disk. Add size-based rotation: rotate at e.g. 100 MB, keep last N (5) files. Same defaults on Windows and Linux.

Source: qa/package_audit/RESULTS.md Phase 5.4.3.

2.4 Linux dry-run findings deferred to v1.2.1

The Phase 4.1 Linux full Stages 0-5 dry-run (2026-04-26) surfaced one BLOCKER (already fixed in v1.2.0) and a handful of smaller findings deferred here:

Item	Description
LX-F3	Customer_API_Reference Sec 7.5 promises compute_fingerprint() emits a stderr "Hardware component <name> is unavailable" warning at every collect, but rg_fingerprint -v on a host with empty Disk serial prints no such warning to stderr (license_create does, so the library code is correct; rg_fingerprint suppresses or never re-routes the library's stderr). Fix: either un-suppress the library stderr in rg_fingerprint, or update the doc to say "warning is emitted by the library but rg_fingerprint may consume it".
LX-F5	LicenseVerifier constructor throws std::runtime_error(" Failed to parse public key PEM") on a malformed PEM. The class is documented as "never throws unhandled exceptions" with respect to load(), but the constructor's behavior is undocumented. A customer who passes a bad PEM and does not wrap construction in try/catch sees process abort. Fix: either catch in the constructor and defer the failure to the first load() call (returning MalformedFile), or document the constructor explicitly as throwing and recommend wrapping.
LX-F6	The CLI tools' "Library not initialized. Call rockyguard::init()..." error string references a vendor-only API (init()) the customer never sees in the customer headers. Customer_Documentation Sec 10.3.1 documents the cause as "missing --vendor-license" with the right resolution, but the leaked init() reference in the error text is confusing. Fix: change the CLI's error string to "Vendor license missing. Re-run with --vendor-license vendor_license.json."
LX-F7	Minor inconsistencies in the docs: README.txt CONTENTS omits version.h while Customer_Documentation Sec 2.1 includes it; rg_fingerprint listed in two slightly different positions in the package contents tables. Fix: align both sources.
LX-F8	README.txt Linux side mentions storage anchor at $HOME/.lck_svc_<tag> but Customer_Documentation Sec 9.2 only refers to "multiple hidden locations" without naming the per-license anchor convention. Fix: name the anchor file convention in Sec 9.2 for both Windows (registry) and Linux (file path).

Source: qa/onboarding/RESULTS.md "Linux Full Phase 4.1 Re-Run".

2.5 Documentation enhancements (deferred from doc-structure review)

A doc-structure review against a generic licensing-product outline (commissioned 2026-04-26 during the v1.2.0 ship window) identified five doc gaps. Two landed in v1.2.0 (Customer_Documentation Sec 9.6 "Threat Model" and Sec 10.1 "Error Reference Table"); the remaining three are deferred here:

Item	Description
Glossary	Customer_Documentation: add a glossary of terms (vendor license, end-user license, fingerprint, time anchor, grace period, node-locked vs floating, Basic vs Premium tier, etc.). Currently terms are defined inline scattered across sections; consolidating helps a customer who reads one section in isolation.
Manual / reference split tightening	Customer_Documentation currently mixes integration tutorial (Sec 1-3) with CLI reference (Sec 7) and license-format reference (Sec 8). The cleaner split (task-oriented manual + technical reference) would move CLI / format reference material out of the manual and into the API Reference's appendix. Mostly editorial; no new content.
Production checklist (consolidation)	Production-relevant guidance is currently spread across: Customer_API_Reference Sec 2 (CRT caveat), Sec 6.2 (synchronous network I/O / threading), Sec 8.1 (TLS layered defenses); Customer_Documentation Sec 3.2 (deployment files), Sec 9 (anti-tampering). A single "Production Checklist" page that gathers these into a copy-paste preflight list would help customers shipping to end users.

Source: doc-structure review 2026-04-26 (in-session triage; full analysis in the chat transcript that produced this commit).

Items rejected from the same review (out of scope for any RockyGuard release): multi-language SDK framing (we are C++17 only), rk.activate() / vendor-portal flow (we don't have online activation or a portal), cloud / hybrid mode (we don't offer SaaS licensing), telemetry hooks (we collect nothing), migration guides (v1.2.0 is the inaugural release).

3. v1.3 - MINOR RELEASE

3.1 Build / packaging improvements

Item	Description
rockyguard-config.cmake imported target	Ship a CMake config package so customers can call find_package(rockyguard CONFIG REQUIRED) and link against rockyguard::rockyguard, with all transitive dependencies (OpenSSL, Windows system libraries, the right CRT variant) handled by the imported target. Eliminates the GA-002 class of defect (forgotten linked libraries) by construction.
Single-zip multi-platform layout	Optional: collapse the two per-platform zips back into one if the on-disk layout becomes uniform enough to make this clean. v1.2.0 chose two zips because per-platform README content differed. Re-evaluate after rockyguard-config.cmake unifies the integration story.

Source: Phase 4.1 RESULTS.md follow-up actions; QA Report Section 4.2.7 forensic record (long-term recommendation).

3.2 Continuous integration

Item	Description
Phase 0 CI pipeline	Automated gates for: full test suite (Windows + Linux), build_and_package.sh dry-run, customer-use zip extraction and consumer-build sanity. The QA Plan reserves Phase 0 for CI; v1.2.0 ran with no CI.
Phase 4.1 dry-run replication CI job	A CI job that reproduces a stripped-down version of the Phase 4.1 customer dry-run on every build: extract the freshly-packaged zip into an isolated temp dir, build the example consumer project, run it, assert "Licensed". Would have caught GA-002 (the missing libssl in the link line) in the development cycle, before the customer onboarding dry-run was ever needed.
MANIFEST.txt tag check	Every tag should write its zip SHAs into archives/MANIFEST.txt and the CI should refuse to tag if the working tree zip differs from MANIFEST.

Source: Phase 4.1 RESULTS.md (CI item), QA Plan Phase 0.

3.3 Platform expansion

Item	Description
manylinux_2_28 build	Linux x86_64 with glibc < 2.34 (Ubuntu 20.04, RHEL 8, Amazon Linux 2, Debian 10). Build the library inside a manylinux_2_28 container so the resulting libc dependency is glibc 2.28 or newer, broadening compatibility back ~4 years.
macOS arm64 + macOS x86_64	Library-side support code already exists (IOKit fingerprint, Mach-O paths) but is untested in v1.2.0. Need: IOKit fingerprint validation across hardware, Mach-O integrity-check verification, code-signing pipeline for the shipped .dylib, signed-tag verification for downloads.
Graceful "platform too old" detection	Instead of relying on ld.so to fail at load with "GLIBC_2.34 not found", emit a clear startup warning if the host glibc is older than the library's minimum and direct the customer to the manylinux_2_28 build.

Source: QA Report Section 5.1; QA Plan Phase 3.

3.4 Performance regression

Item	Description
Performance budget document	Establish per-API target wall-clock budgets (license verify under 50 ms, fingerprint computation under 100 ms, floating checkout under X ms, etc.). Without this document there is nothing to regress against.
Automated benchmark harness	Once a budget exists: a Google Benchmark or similar harness that runs the budgeted APIs and compares against a baseline checked into the repo. Failing budgets break the build.

Source: QA Report Section 5.1; QA Plan Phase 7.

3.5 Non-blocking load API

Item	Description
LicenseVerifier:: load_async()	New public API: std::future<LicenseR esult> load_async(const std::string& license_file_path). Same semantics as load() but returns immediately; the future resolves once the license file is parsed and the anti-tampering pipeline (including the synchronous online time-anchor check, which is the latency wild card in load()) has finished. Lets GUI / desktop hosts call into the library from the UI thread without freezing the application's first-run startup for the tens of seconds that a flaky network worst case takes. v1.2.0 ships only the synchronous load() with documentation telling GUI integrators to wrap it in std::async themselves; v1.3 ships the wrapper.
load_async() cancellation	Optional second-pass design: a cancellation token so a host that decides to abort startup can cancel the in-flight HTTPS attempts without orphaning the pool sockets. Could be deferred to v1.4 if the basic future-returning form is enough.

Source: Phase 6.1 reviewer findings 2026-04-25 (round 4: synchronous-network-I/O-in-load concern).

3.6 Logging API

Item	Description
set_log_callback(...)	New public API: void set_log_callback(std: :function<void(LogLevel, const std::string&)>). Replaces the ad-hoc std::cerr writes scattered across the library (integrity_check, hardware_fingerprint, license_generator, license_verifier, server_logger, vendor_license, license, time_anchor). When no callback is set, default to stderr (current behavior, backward compatible). When a callback is set, route every diagnostic through it with a severity level (DEBUG / INFO / WARNING / ERROR).
Severity filtering	Default cut-off WARNING. Customers can lower to DEBUG to capture the verbose paths or raise to ERROR to silence informational warnings.
Thread safety	Callback is invoked from arbitrary library threads (especially on the floating-server worker pool). Document the threading contract clearly: callback must be re-entrant or the customer must serialize.

Rationale: a Phase 6.1 reviewer correctly observed that hardcoded stderr output in a C++ library pollutes GUI / daemon hosts. The v1.2.0 fix was documentation-only (Customer_API_Reference Section 2.1 "Diagnostic output"); the v1.3 fix is the real one. This is the recommended approach over alternatives because:

- It is fully additive: no existing customer breaks.
- It does not change the meaning of LicenseResult or LicenseStatus.
- It supports both "drop into existing logger" and "silence everything" use cases without forcing a process-level redirect.

Source: Phase 6.1 reviewer findings (2026-04-25 batch).

3.7 Upgrade / downgrade testing (Phase 8)

Item	Description
First end-to-end Phase 8 execution	v1.2.0 is the inaugural shipping release, so Phase 8 was N/A. Once v1.2.0 customers exist, v1.3 (or v1.2.1, whichever ships next) becomes the first release with a real upgrade target. Validate: legacy time-anchor read fallback, legacy generation-counter HMAC fallback, license-format compatibility from a v1.2.0 vendor against a v1.3 verifier, and (if applicable) a documented downgrade path.

Source: QA Report Section 5.2; QA Plan Phase 8.

4. v1.4 AND BEYOND - SPECULATIVE / UNSCHEDULED

These items are not committed to any cycle. They are recorded so they are not lost between roadmap reviews.

Item	Notes
LicenseResult.warnings bitmask field (additive non-breaking extension)	An alternative / complement to set_log_callback: surface the conditions that today print to stderr as a structured warning bitmask in the LicenseResult struct. Customers who want machine-readable diagnostics rather than a callback get them through the return value. Requires care to land non-breaking (default-constructed struct stays zero-initialized).
Default stderr suppression	After set_log_callback ships in v1.3, consider flipping the v1.4 default to "silent unless a callback is set". This is technically a behavior change so it does belong in a minor-version bump.
Code signing / notarization for shipped artifacts	Sign the Windows .lib / .dll and the macOS .dylib at the artifact level. Requires a code-signing certificate and (for macOS) Apple Developer ID notarization.
Hardened cert pinning for the time-anchor HTTPS pool	The current pool is 12 hosts with TLS verification via the system cert store. v1.4+ could pin a fixed CA bundle so the library does not depend on the host's cert store at all (closes the misconfigured-cert-store edge case more tightly).
First-class fingerprint TPM / TEE binding	Optional component on platforms with a TPM or secure enclave. Would replace one of the four fingerprint slots with a TPM-derived value, much harder for an attacker to forge or move. Significant new code; deserves its own design doc.
Legal-entity placeholder resolution	Once the [Legal Entity TBD] / [Jurisdiction] / [Address] decisions are made (Phase 6.2 of v1.2.0), propagate through every doc, header, and shipped artifact. Treated as a release-mechanics item, not a feature.

5. CROSS-REFERENCE: WHERE EACH ITEM WAS SURFACED

For every roadmap entry above, the source-of-truth reference where the item was first identified (so a future reviewer can read the original finding in context):

Roadmap entry	First surfaced in
GA-001 vendor-license onboarding section	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-001)
GA-006 static-build .sig clarification	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-006)
GA-010 date-format consistency	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-010)
GA-011 export.h missing from headers list	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-011)
GA-012 ship-to-end-user checklist	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-012)
GA-013 medium-term Debug-CRT library	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-013) ; QA Report Section 4.2.8
Phase 5.4.3 log rotation	qa/package_audit/ RESULTS.md (Phase 5.4.3)
rockyguard-config.cmake imported target	qa/onboarding/RESULTS.md follow-up actions ; QA Report Section 4.2.7
Phase 0 CI pipeline	QA Plan Section 0 ; QA Report Section 5.1
Phase 4.1 dry-run replication CI job	qa/onboarding/RESULTS.md follow-up actions
manylinux_2_28 build	QA Report Section 5.1 ; QA Plan Phase 3
macOS arm64 + x86_64	QA Report Section 5.1 ; QA Plan Phase 3
Performance budget + harness	QA Report Section 5.1 ; QA Plan Phase 7
LicenseVerifier:: load_async()	Phase 6.1 reviewer findings 2026-04-25 (round 4: synchronous network I/O in load)
set_log_callback	Phase 6.1 reviewer findings 2026-04-25 (5th finding, stderr in library)
Phase 8 first execution	QA Report Section 5.2 ; QA Plan Phase 8
GA-008 legal entity placeholder resolution	qa/onboarding/RESULTS.md (Phase 4.1 Run 1, GA-008) ; Phase 6.2

6. PROCESS IMPROVEMENTS BANKED FROM v1.2.0

The v1.2.0 cycle surfaced several "don't do that again" lessons. They are not features and do not have a release target; they are operational norms that future cycles inherit.

- Always re-run bash build_and_package.sh AFTER every in-session doc fix and verify against the packaged zip - NOT against the source tree. Discovered as the meta-finding from the Phase 4.1 delta re-run (DR2-001): the source had the fix but the shipped zip predated the rebuild, so a fresh customer would have seen stale docs.
- Phase 4.1 fixture generation (vendor license + sample end-user license) MUST run in a separate holding directory and be copied into the sub-agent's cwd at handoff. Running fixture generation inside the sub-agent's cwd leaks state files (.lck_cache_, .rg_gencount_) and contaminates the cold-read environment.
- Bash scripts: do not combine set -o pipefail with grep | head -1. SIGPIPE from head closing early propagates back as a failure under pipefail. Use awk '/pattern/ {print; exit}' or grep -m 1.
- Reviewer findings of the form "this is a code defect" should be verified against committed source before agreeing. The Phase 4.1 reviewer mistakenly believed the in-session GA-002 / GA-013 / etc. fixes were missing because they ran git status on a clean tree (which only shows uncommitted changes); the fixes were in git log. A 30-second grep -n on the committed file disposes of this class of confusion.
- When a doc fix lands AFTER Phase 4.1 sign-off (e.g., the Phase 6.1 fixes that landed on 2026-04-25 after the Phase 4.1 final run on 2026-04-24), evaluate whether the change touches any path the Phase 4.1 sub-agents walked. If it does not (e.g., adding a paragraph that explains existing behavior rather than changing a procedure), no Phase 4.1 re-run is required. The Phase 4.1 Report accurately reports what was tested; the new zip SHA supersedes the tested SHA in archives/MANIFEST.txt without invalidating the prior verdict.

End of roadmap.