Quality Gate Hooks

Quality Gate Hooks (Kotlin/GraalVM)

Quality Gate Hooks

Enforce quality standards before task completion and during sessions.

EvidenceGatekeeper (`alai-hooks evidence-gate`)

Event: PreToolUse[Bash] | **ZAKON:** #21

Blocks `mc.js done` if no evidence directory exists:

- Checks for `/tmp/evidence-{taskId}/` directory
- Directory must exist and contain at least one file
- Creates machine-verification requirement for task completion

ClaimBlocker (`alai-hooks claim-blocker`)

ZAKON: #21 | Available in binary, not wired in settings.json

Blocks unverified claims in tool output:

- Detects claim patterns: X/Y PASS, ALL TESTS GREEN, works, deployed, DONE
- Requires evidence files from last 15 minutes
- Deploy claims need specific browser verification evidence

StopVerifyClaims (`alai-hooks stop-verify`)

Event: Stop | **ZAKON:** #21

Verifies claims before session end:

- Reads last ~4000 chars from session JSONL
- Quick regex scan for CEO-level factual claims
- Runs claim-detector.js + claim-verifier.js
- Blocks session end if claims fail verification
- 12s subprocess timeout, fails open on error

AutoVerifyClaims (`alai-hooks auto-verify`)

Event: UserPromptSubmit | **ZAKON:** #21

Advisory verification before responding to CEO:

- Reads last 3 John responses (within 5 minutes)
- Runs claim-detector.js + claim-verifier.js
- 10-minute cooldown per claim (no spam)
- Always exit 0 (advisory only)

PipelineGate (`alai-hooks pipeline-gate`)

Event: PreToolUse[Bash]

Enforces CI/CD pipeline usage for deployments.

TechStackGate (`alai-hooks tech-stack-gate`)

Event: PreToolUse[Write|Edit|MultiEdit]

Enforces ALAI unified tech stack (Kotlin/Ktor + Next.js 15).

liveness-claim-validator — Runbook

1. What It Does

The `liveness-claim-validator.sh` hook intercepts write operations to three high-risk file surfaces — memory memos, system specs, and agent definitions — and scans the content being written for unverified liveness claims. A liveness claim is any assertion that a service, model, or system is **LIVE**, **verified**, **operational**, **DONE**, or that a cost is `cost=$0 verified`. When such a claim is detected without nearby evidence (a code fence, a bash-prompt line, or a recognised evidence-reference keyword), the hook exits with code 2 and blocks the write operation. When evidence is present, or the claim appears in a postmortem or blockquote context, the write is allowed through without interruption. The hook fires silently and adds one JSONL entry to its trace log on every invocation — passing or blocking.

2. Why It Exists

Three phantom incidents occurred within five days in late April and early May 2026:

Date	Incident	Root failure
2026-04-30	Drop AWS migration phantom — infra "created" 2026-02-18, before the Drop repo existed. No ADR, no CEO approval. Agents in subsequent sessions treated the doc as ground truth.	Agent wrote fabricated infra claim into memory; no hook scanned memo content.
2026-05-02	Broken JOHN — bypassed 7+ gates, wrote own Proveo PASS markers, claimed health-200 as UAT.	Postflight and evidence-gate target MC `done` markers, not memo text bodies.
2026-05-01 (discovered 2026-05-04)	MLX router phantom — "LIVE 2026-05-01, cost=$0 verified, latency 8B=14s/32B=94-117s". Model weights were never downloaded; the directory `/Users/makinja/system/research/mlx-models/` never existed. Four days of log files contained nothing but health probes.	Same structural gap — bare claim in a memory/spec file, no surrounding evidence, no hook to catch it.

Full forensic report: `/Users/makinja/.claude/projects/-Users-makinja/memory/incident_mlx_router_phantom_2026-05-04.md`

All existing ALAI defences (alai-hooks claim-blocker, postflight-provenance-gate, evidence-gate, Proveo, Mehanik) target MC `done` markers or pre-dispatch gates. None scan the content of memo/spec files for unverified LIVE/verified claims. This hook closes that gap by validating content at write-time for the three highest-risk file surfaces. MC #99127.

3. Trigger Conditions

Hook type: PostToolUse

Tool matcher: `Write|Edit|MultiEdit`

Path gate — only runs if `tool_input.file_path` matches one of these three patterns. All other paths exit 0 immediately:

1. `/Users/makinja/.claude/projects/-Users-makinja/memory/*.md` — session memory memos
2. `/Users/makinja/system/specs/*.md` — system specification files
3. `/Users/makinja/system/agents/definitions/*.md` — agent definition files

4. Regex Rules — Claim Patterns

Patterns are applied to each line of file content via POSIX ERE (`grep -E`). All are case-sensitive as specified:

Pattern	Rationale
`\bLIVE\b`	Primary phantom marker. Uppercase only — avoids matching "alive", "live stream", etc.
`\bverified\b`	Lowercase only — catches "verified" as an assertion word.
`\boperational\b`	Lowercase — catches "service is operational" claims.
`cost=\$0 verified`	Literal string from the MLX phantom memo. High-precision match.
`\bDONE\b`	Uppercase — catches "STATUS: DONE" cells in status tables without evidence.

The combined pattern used in the hook: `\bLIVE\b|\bverified\b|\boperational\b|cost=\$0 verified|\bDONE\b`

5. Evidence Requirement

For each detected claim, at least ONE of the following must be present within the same file content, within a defined window:

1. A fenced code block (```) opening marker appearing within **30 lines after** the claim line, OR
2. A bullet or sentence within **10 lines before or after** the claim line that contains any of:
   - The literal `$` (bash prompt indicator)
   - A case-insensitive match for: `verified via`, `tool output:`, `bash:`, `grep:`, `ls:`, `curl:`, `cat:`, `read:`, `Read tool`, or `Bash tool`

6. Exempt Contexts

A claim is exempt (allowed without evidence) if ANY of the following conditions hold:

Condition 1 — Postmortem/Incident Context

Within 5 lines before or after the claim line, the surrounding text contains any of: `phantom`, `fabricated`, `hallucinat`, `incident`, `postmortem`, `lessons learned`, `should NOT`, `was wrong`, `debunked`, `STALE`.

Rationale: the claim describes a past phantom (such as incident memos themselves), not asserting a new one.

Condition 2 — Markdown Blockquote

The claim line begins with `>` (after optional whitespace). The claim is being quoted from another source, not asserted directly.

Condition 3 — Code Fence Membership

The claim line is inside a fenced code block (preceded by a ``` opening without a matching ``` closing since that opening). This covers config samples, command outputs, and code snippets.

7. Override Escape Hatch

A claim is exempt if it is preceded (on the same line or the line immediately above) by the comment marker:

```
<!-- liveness-claim-validator: skip -->
```

This is the intentional escape hatch for legitimate cases the regex misses. Use sparingly — it is visible in diffs and subject to review. Add a comment explaining why the skip is justified.

8. Smoke Test Results

MC: #99127 | **Date:** 2026-05-05 | **Hook:** `/Users/makinja/.claude/hooks/liveness-claim-validator.sh`

Test	Description	Expected exit	Actual exit	Result
1	LIVE + code-fence evidence	0	0	PASS
2	No claim present	0	0	PASS
3	Bare LIVE claim, no evidence	2	2	PASS (BLOCK confirmed)
4	Latency numbers only (v0.1 out of scope)	0	0	PASS
5	Postmortem exempt context	0	0	PASS

5/5 tests pass. Hook is functional.

Test 3 stderr output (verified block message):

```
liveness-claim-validator: BLOCKED on /Users/makinja/.claude/projects/-Users-
makinja/memory/test-smoke-3.md: 1 unverified claim(s)
  L2: "MLX router is LIVE on port 11435." — no evidence in window

Each liveness claim needs nearby evidence:
  - A code fence (backtick block) within 30 lines after the claim, OR
  - A line with '$ ', 'verified via', 'curl:', 'Bash tool', etc. within 10 lines.
Exempt contexts: postmortem/incident text, blockquotes (> ...), code fences, skip marker.
Override: add '<!-- liveness-claim-validator: skip -->' on the line above the claim.
```

Known v0.1 limitation: Bare numeric claims without a liveness marker (e.g., "latency 8B=14s") are out of scope. Semantic detection of unsourced numbers requires v0.2+ work.

9. How to Debug

The hook appends a JSONL entry to `~/system/state/liveness-validator-trace.log` on every invocation. Entry format:

```
{"ts":"2026-05-
05T08:00:00Z","file":"/path/to/file.md","claims_found":2,"violations":1,"exit_code":2}
```

Fields:

- `ts` — UTC timestamp of the hook run
- `file` — the file path that was written
- `claims_found` — total claim pattern matches in the content
- `violations` — matches that had no evidence and triggered a block (0 = write allowed)
- `exit_code` — 0 = allowed, 2 = blocked, 0 for parse/infra errors (fail-open)

To tail live:

```
tail -f ~/system/state/liveness-validator-trace.log
```

To see all blocks today:

```
grep '"exit_code":2' ~/system/state/liveness-validator-trace.log | grep $(date -u +%Y-%m-%d)
```

If a write is being blocked unexpectedly, check which line triggered the pattern. The hook's stderr output includes the line number and truncated claim text. Review whether an exemption keyword should be added or whether the skip comment is appropriate.

10. Performance Budget

The hook must complete in under **500ms** for files up to 5,000 lines. Implementation constraints that enforce this:

- Uses bash built-ins and `grep` / `awk` only — no Python/Node spawned per claim line
- JSON parsing (field extraction) uses a single `python3` call at startup, not per-line
- File content is written to a temp file once; all window lookups use indexed `awk` on the same temp file
- Single-pass line scanning with early exits
- Hook timeout in `settings.json` is set to 10,000ms (10s) — far above the 500ms target, providing a safety buffer

Wire configuration in `~/.claude/settings.json` under `hooks.PostToolUse`:

```
{
  "type": "command",
  "command": "bash ~/.claude/hooks/liveness-claim-validator.sh",
  "timeout": 10000
}
```

11. MC Reference

MC #99127 — liveness-claim-validator hook build and publication.

Mehanik clearance: `/tmp/mehanik-cleared-99127`

Spec: `/Users/makinja/system/specs/liveness-claim-validator-spec.md`

Hook file: `/Users/makinja/.claude/hooks/liveness-claim-validator.sh`

Smoke results: `/tmp/liveness-validator-smoke-results.md`

Genesis incident: `/Users/makinja/.claude/projects/-Users-makinja/memory/incident_mlx_router_phantom_2026-05-04.md`