Hooks Reference

Hooks Reference. Tier A orphan hooks wired in Claude settings PreToolUse:Bash for MC #101642: evidence-contract-validator.sh, git-author-guard.sh, mc-ready-gate.sh, pre-publish-claims-gate.sh, zakon-30-direct-probe-gate.sh. These hooks enforce evidence contracts, git-author safety, H-task ready wrapper, pre-publish claim checks, and ZAKON #30 direct machine probes. MC #101643 marker: GOTCHA stub generation and async auto-verify worker are documented for validator probes; evidence under /tmp/101643-evidence/.

- Security Hooks
  - Security Hooks (Kotlin/GraalVM)
- Quality Gate Hooks
  - Quality Gate Hooks (Kotlin/GraalVM)
  - liveness-claim-validator — Runbook
- Protocol Hooks
  - Protocol Hooks (Kotlin/GraalVM)
- GOTCHA Hooks
- Resource Hooks
- Obstacle & Diagnostic Hooks
- Email & Validation Hooks
- Hooks Architecture Overview
- John Drift-Prevention v3
- MC 101642 — Tier A Orphan Hooks Wiring
- Lesson E+B Hard Enforcement — PI Agent Plane (MC #103499)

Security Hooks

Security Hooks (Kotlin/GraalVM)

Security Hooks

All security hooks run as PreToolUse gates. Exit 2 = BLOCK, Exit 0 = ALLOW. Binary: `~/.claude/hooks/alai-hooks`

BashSecurityGate (`alai-hooks bash`)

Event: PreToolUse[Bash] | **ZAKON:** Multiple

Blocks dangerous shell commands:

- **NPM Audit Gate:** Blocks known malicious packages and dangerous flags
- **Destructive Commands:** DROP TABLE/DATABASE, DELETE without WHERE, dangerous git operations, recursive rm, chmod 777
- **Exfiltration Detection:** Blocks curl/wget to known exfil domains. Detects pipe-to-curl and DNS exfiltration
- **Shell Injection:** Blocks pipe to interpreter, eval, command substitution with dangerous commands
- **Inline SMTP:** Blocks inline email scripts (ZAKON #6)

WriteSecurityGate (`alai-hooks write`)

Event: PreToolUse[Write|Edit|MultiEdit]

Blocks writes to protected paths:

- ~/.ssh, ~/.gnupg, ~/.aws (credential theft)
- ~/Documents, ~/Desktop, ~/Downloads (security policy)
- Browser profiles, Keychains, Mail, Messages, Photos
- Advisory warning for secrets/API keys in file content

DeployGateZakon (`alai-hooks deploy-gate`)

Event: PreToolUse[Bash] | **ZAKON:** #2, #19

Blocks production deployments without CEO approval:

- `az containerapp update/create` blocked unless `/tmp/ceo-approved-deploy` exists
- `docker push` to production ACR blocked unless approved
- Strips heredoc content before pattern matching

BackendEditGuard (`alai-hooks backend-guard`)

Event: PreToolUse[Write|Edit|MultiEdit] | **ZAKON:** #20, #5

Prevents John from directly editing backend code:

- Detects .java, .kt, .go files in backend paths
- Skips subagent context (`/tmp/alai-subagent-context`)
- Warn mode (default) or strict mode (`/tmp/backend-edit-strict`)

HallucinationDetector (`alai-hooks hallucination`)

Event: PreToolUse[Write|Edit|MultiEdit] | **ZAKON:** #1

5-layer anti-hallucination defense:

1. **Known Wrong Facts:** Blocks known-incorrect values (wrong names, org numbers, API endpoints)
2. **Phantom Tools:** Blocks references to tools confirmed non-existent
3. **Wrong Ports:** Flags localhost ports not in known services map
4. **Phantom Endpoints:** Blocks known-invalid API endpoints for tracked services
5. **Phantom Paths:** Detects hardcoded file paths that don't exist on disk

Skips: ~/system/config/ files, /tmp paths, URLs, wildcards, template strings

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.

Date	Incident	Root failure
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.

Pattern	Rationale
`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`

Protocol Hooks

Protocol Hooks (Kotlin/GraalVM)

Protocol Hooks

Enforce organizational protocols and workflows.

LeadGuard (`alai-hooks lead-guard`)

Event: PreToolUse[Write|Edit|MultiEdit]

Protects lead/sales data integrity.

SessionStartContext (`claude-hooks session`)

Event: SessionStart

Injects session context on startup: current task, last session summary, system state, next steps.

SubagentStartContext

Events: `claude-hooks subagent` + `alai-hooks subagent`

Injects context into spawned subagents: core protocol, active task, file ownership, anti-hallucination rules.

Shell Scripts

Script	Event	Purpose
boot-enforcer.sh	UserPromptSubmit	Enforce boot.sh first
user-message-logger.sh	UserPromptSubmit	Log user messages

Script	Event	Purpose
session-cleanup.sh	Stop	Clean temp files
session-ledger.sh	Stop, PreCompact	Record session summary
context-bundle-logger.sh	PostToolUse (async)	Log tool usage
worktree-create.sh	WorktreeCreate	Configure new worktrees

GOTCHA Hooks

Resource Hooks

Obstacle & Diagnostic Hooks

Email & Validation Hooks

Hooks Architecture Overview

Hooks Architecture Overview

Last updated: 2026-04-05 **Status:** All hooks migrated to Kotlin/GraalVM native binaries

Architecture

ALAI uses two GraalVM native binaries for hook enforcement:

Binary	Purpose	Size
`alai-hooks`	Business logic hooks (security, quality, ZAKON enforcement)	14 MB
`claude-hooks`	Session/subagent context + post-tool logging	~14 MB

Both are **Mach-O 64-bit arm64** executables compiled from Kotlin via GraalVM `native-image`. Zero JVM startup cost (~2-5ms per invocation).

Hook Events

Event	When	Hooks
PreToolUse[Bash]	Before any shell command	bash, evidence-gate, pipeline-gate, deploy-gate
PreToolUse[Write/Edit]	Before file writes	write, tech-stack-gate, lead-guard, backend-guard, hallucination
PostToolUse[.*]	After any tool	claude-hooks post, context-bundle-logger
Stop	Session end	session-cleanup, session-ledger, stop-verify
PreCompact	Before context compaction	session-ledger
UserPromptSubmit	Before processing user message	boot-enforcer, user-message-logger, auto-verify
WorktreeCreate	Git worktree creation	worktree-create

Event	When	Hooks
SessionStart	New session	claude-hooks session
SubagentStart	New subagent	claude-hooks subagent, alai-hooks subagent

Language Breakdown (Post-Migration 2026-04-05)

Language	Count	Usage
Kotlin/GraalVM	15	All security + quality hooks
Shell	6	Session management, boot, logging
Python	0	Fully migrated (was 9 hooks)

Migration History

- **2026-04-01:** Initial Kotlin binary (`alai-hooks`) with 8 hooks
- **2026-04-05:** Migrated remaining 6 Python hooks to Kotlin:
  - `deploy-gate-zakon.py` to `alai-hooks deploy-gate`
  - `backend-edit-guard.py` to `alai-hooks backend-guard`
  - `hallucination-detector.py` to `alai-hooks hallucination` (NEW, was not wired)
  - `claim-blocker.py` to `alai-hooks claim-blocker`
  - `stop_verify_claims.py` to `alai-hooks stop-verify`
  - `auto_verify_claims.py` to `alai-hooks auto-verify`
- **2026-04-05:** Python dispatchers/, lib/, backup/ archived
- **2026-04-05:** hook-daemon.py retired (socket-based Python dispatcher)

Source Code

```
~/system/tools/alai-hooks/
  build.gradle.kts            # Kotlin 2.2.0 + GraalVM native-image
  src/main/kotlin/com/alai/hooks/
    Main.kt                   # Dispatch router
    JsonParser.kt             # Minimal JSON parser (no reflection)
    BashSecurityGate.kt       # NPM audit, destructive commands, exfil, SMTP
    WriteSecurityGate.kt      # Protected paths, secrets detection
```

```
EvidenceGatekeeper.kt     # ZAKON #21: evidence before task completion
TechStackGate.kt          # ALAI tech stack enforcement
PipelineGate.kt           # CI/CD pipeline gate
LeadGuard.kt              # Lead/sales data protection
DeployGateZakon.kt        # ZAKON #2/#19: CEO approval for deploys
BackendEditGuard.kt       # ZAKON #20/#5: orchestrator vs executor
HallucinationDetector.kt  # Anti-hallucination (facts, ports, paths, tools)
ClaimBlocker.kt           # ZAKON #21: evidence for claims
StopVerifyClaims.kt       # Session-end claim verification
AutoVerifyClaims.kt       # Pre-prompt claim verification
SessionStartContext.kt    # Session context injection
SubagentStartContext.kt   # Subagent context injection
PostToolLogger.kt         # Post-tool audit logging
RoleDetector.kt           # Agent role detection
PermissionEnforcer.kt     # Permission enforcement
YamlPermissionLoader.kt   # YAML config loader
```

Building

```
cd ~/system/tools/alai-hooks
gradle nativeCompile
cp build/native/nativeCompile/alai-hooks ~/.claude/hooks/alai-hooks
```

Requires: GraalVM CE 25+, Kotlin 2.2.0, Gradle.

John Drift-Prevention v3

John Drift-Prevention v3 (2026-05-02/03)

Genesis

On 2026-05-02, CEO asked "ovi hooks u kojem jeziku" (a QUESTION). John amplified this into a Kotlin port chain, opening MC #10586 and emergent MC #10589 before CEO intervened with "otisao si u rabbit hole again." This occurred despite v2 hooks (MC #10570) being live and registered. The root cause is structural: v2 gates fire at tool-invocation time (PreToolUse/PostToolUse), but the LLM's planning trajectory commits before the first gate fires. Both drift incidents (Kotlin rabbit-hole + AWS phantom earlier same day) unfolded across multiple turns, each individually compliant with v2 per-turn rules.

Context: feedback_john_kotlin_rabbit_hole_2026-05-02.md

v2 ? v3 Progression

v2 (MC #10570) provided:

- Per-turn MC counter (one-ceo-turn-mc-cap.sh) — caps mc.js add per turn
- CEO_APPROVED token rate limit — blocks John self-issuing overrides
- Turn boundary reset (mc-turn-reset.sh) — clears MC counter on UserPromptSubmit
- Depth/emergent spawn gates (john-max-depth-gate.sh) — 3 trip-wires on Task/Agent dispatch

v2 gaps:

- No intent classification before planning — QUESTION/CRITIQUE turns treated same as DIRECTIVE
- MC counter reset did NOT clear CEO_APPROVED token counter — re-issuance mechanically impossible
- No cap on Task/Agent dispatch — Kotlin incident dispatched codecraft + sentinel-tester in same turn, both uncapped

- Per-turn enforcement only — multi-turn accumulation (MC #10586 → #10589 across two turns) not detected

v3 adds:

- **Rank 1:** Fix CEO_APPROVED counter reset — now resets on every UserPromptSubmit (unconditional)
- **Rank 2:** Intent classifier at UserPromptSubmit — emits STOP reminder to LLM context before planning on QUESTION/CRITIQUE turns; blocks mc.js add unless [CEO_APPROVED]
- **Rank 3:** Task/Agent dispatch cap — per-turn limit derived from Mehanik approved_subtask_count (fallback: 1)

Three Hooks

Rank 1: mc-turn-reset.sh (UPDATED)

Path: `~/.claude/hooks/mc-turn-reset.sh`
Event: UserPromptSubmit (fires on every new CEO message)
What it does: Resets three counters to zero on every new CEO turn:

- `/tmp/john-mc-turn-counter.json` (per-turn MC count)
- `/tmp/ceo-approved-token-uses-${SESSION_ID}.count` (CEO_APPROVED token uses — v3 FIX)
- `/tmp/john-dispatch-turn-counter.json` (Task/Agent dispatch count — v3 NEW)

Sample output: Silent (exit 0). Counters visible via `cat /tmp/john-mc-turn-counter.json` → `{"count":0, "turn_id":"...", "ts":"..."}`

Override: None. This is a mandatory reset per documented contract ("token valid for one turn").

Rank 2: ceo-intent-classifier.sh (NEW) + pre-mc-add-gate.sh (UPDATED)

Paths:

- `~/.claude/hooks/ceo-intent-classifier.sh` (NEW, UserPromptSubmit)
- `~/.claude/hooks/pre-mc-add-gate.sh` (UPDATED, PreToolUse Bash)

Event: UserPromptSubmit (classifier) + PreToolUse Bash when tool=mc.js (gate)
What it does: Classifies CEO message intent into five classes:

- DIRECTIVE — action request (default, fail-open)
- QUESTION — inquiry (ends with ?, starts with Bosnian/English question word)
- CRITIQUE — rhetorical negative ("to ne radi", "broken", "nije ok")

- CONFIRM — short affirmative ("da", "ok", "yes")
- REISSUE — contains [CEO_APPROVED]

Writes result to `/tmp/ceo-intent-${SESSION_ID}.json`. On QUESTION or CRITIQUE, emits STOP system reminder to LLM context before planning begins. `pre-mc-add-gate.sh` blocks mc.js add on QUESTION/CRITIQUE unless [CEO_APPROVED] is in the command.

Sample output (stderr):

```
[ceo-intent-classifier] intent=QUESTION
SYSTEM REMINDER [intent-gate/QUESTION]: CEO message classified as QUESTION.
  MANDATORY RESPONSE PATTERN:
  1. Answer the question or acknowledge the critique in plain text.
  2. Do NOT open MCs, dispatch agents, or run Mehanik in this turn.
  3. If action seems warranted, surface a ONE-LINE option menu and STOP.
  4. Wait for explicit CEO DIRECTIVE before any mc.js add or Task/Agent dispatch.
```

Override: Include `[CEO_APPROVED]` in the mc.js add command or Task/Agent dispatch prompt.

Rank 3: one-ceo-turn-dispatch-cap.sh (NEW)

Path: `~/.claude/hooks/one-ceo-turn-dispatch-cap.sh`
Event: PreToolUse Task|Agent
What it does: Caps Task/Agent dispatches per CEO turn. Cap derived from most recent Mehanik marker file (`/tmp/mehanik-cleared-*`) field `approved_subtask_count`. If no marker: cap=1. Reads and increments `/tmp/john-dispatch-turn-counter.json`. Blocks on count > cap unless [CEO_APPROVED] in prompt.

Sample output (stderr):

```
BLOCKED [one-ceo-turn-dispatch-cap]: dispatch #2 in this CEO turn (cap=1 from Mehanik marker).
  Multi-dispatch from a single ambiguous CEO turn = drift signal (Kotlin incident 2026-05-02).
  If CEO explicitly authorized N agents: include [CEO_APPROVED] in dispatch prompt.
```

Override: Include `[CEO_APPROVED]` in Task/Agent dispatch prompt, OR obtain Mehanik clearance with multi-agent plan (marker will contain approved_subtask_count > 1).

Live Evidence

Proveo validation: /tmp/v3-proveo-validation.md (2026-05-02)

Suite scores:

- Suite A (Rank 1 — mc-turn-reset.sh): 4/4 PASS

- Suite B (Rank 2 — intent classifier + pre-mc-add-gate): 9/10 PASS (1 NOTE: "a" correctly defaults to DIRECTIVE per spec — not a defect)
- Suite C (Rank 3 — dispatch cap): 5/5 PASS
- Suite D (Deviations): 3/3 ACCEPT (all low-risk broadening of patterns)
- Suite E (E2E regression): 4/4 PASS

Verdict: PASS. No blocking defects. All 3 ranked recommendations correctly implemented and verified with real hook execution.

Override Tokens Table

Token	Usage	Scope	Reset
`[CEO_APPROVED]`	Override any single gate (MC cap, dispatch cap, intent gate, emergent spawn)	One turn	Every UserPromptSubmit (mc-turn-reset.sh)
`[CEO_APPROVED_MULTIPLE_MC]`	Explicitly authorize multiple MC adds in one turn (bypasses one-ceo-turn-mc-cap.sh)	One turn	Every UserPromptSubmit

Note: v3 Rank 1 fix ensures both tokens reset unconditionally on new CEO turn, making re-issuance mechanically possible (was structurally broken in v2).

Known Limitations

- **Intent classifier keyword list:** Initial set covers common Bosnian + English patterns. False-negatives possible on idiomatic phrasing. Fails open to DIRECTIVE (no regression). Tune iteratively from live incidents.
- **Rank 2 john-max-depth-gate.sh intent integration:** Spec section T1-A4 describes piggybacking intent check onto depth-gate Branch A. NOT implemented in v3. Explicitly ranked below top-3 (optional enhancement). Core Rank 1/2/3 fully functional.
- **Drift-pressure score (T2-A1):** Deferred to v3.1. Formula weights unvalidated; shipping guesses would cause false WARN floods.
- **Trip-wire 3 elevation (T2-B3):** Deferred. Root cause fix (intent classifier) is upstream; elevating downstream symptom adds CEO friction without addressing cause.

References

- **v3 spec:** `~/system/specs/john-drift-prevention-v3-spec.md`
- **Parent ZAKON:** `~/system/rules/zakon-28-max-depth-boundary.md`

- **v2 implementation:** MC #10570 (2026-05-02)
- **Genesis incident:** MC #10586 + #10589 (closed parked), feedback_john_kotlin_rabbit_hole_2026-05-02.md

---

Last updated: 2026-05-03. Owner: Skillforge. Status: Live in production.

MC 101642 — Tier A Orphan Hooks Wiring

MC 101642 — Tier A Orphan Hooks Wiring

This page documents the Tier A orphan hooks wired into `~/.claude/settings.json` under `PreToolUse:Bash` for MC #101642.

Tier A hook names

- `evidence-contract-validator.sh`
- `git-author-guard.sh`
- `mc-ready-gate.sh`
- `pre-publish-claims-gate.sh`
- `zakon-30-direct-probe-gate.sh`

Purpose

- `evidence-contract-validator.sh` validates verdict JSON evidence contracts before H-task readiness.
- `git-author-guard.sh` prevents unsafe/incorrect git author operations.
- `mc-ready-gate.sh` blocks direct H/BLOCKER `mc.js ready` bypass and requires wrapper evidence.
- `pre-publish-claims-gate.sh` checks claim publishing before outward-facing completion claims.
- `zakon-30-direct-probe-gate.sh` requires direct machine-probe evidence for H/BLOCKER readiness.

Validation evidence

Evidence packet: `/tmp/101642-evidence/`

- `/tmp/101642-evidence/verdict.json`
- `/tmp/101642-evidence/report.md`
- `/tmp/bash-output-101642-direct-probe.txt`

Review blocker fixed: the Hooks Reference BookStack surface now includes the Tier A marker and all five hook names.

Lesson E+B Hard Enforcement — PI Agent Plane (MC #103499)

Lesson E + B Hard Enforcement — PI Agent Plane (MC #103499)

Date: 2026-06-12 **Owner:** John (orchestrator) **Status:** Implemented + Proveo-verified

Problem

Lesson E (credential-pair validation) and Lesson B (verify-by-live-outcome) were hard-enforced on the **Claude Code (CC) plane** via `~/.claude/hooks/cred-pair-gate.sh` (PreToolUse Bash hook in `settings.json`). But the **PI agent plane** (`agent-runner.js` / `durable-runner.js`) is a separate Node runtime that does **not** read `~/.claude/settings.json`, so E/B were only soft advisories there.

Fix — Hard enforcement at the safe convergence point

Two enforcement surfaces added to the PI plane:

1. **mc.js done/ready gate** — `~/system/tools/mc.js` (~line 2192-2214). Calls `lesson-e-b-validator.js check(taskId, task, outcomeMsg, forceFlag)`. For deploy/auth/oauth/integration/secret-rotation tasks lacking live-outcome (B) **and** cred-pair (E) evidence, the gate pushes to `blocks[]` when `enforcement.json.lesson_e_b == "block"`. **Fail-open:** any validator require/internal error is non-blocking. **Scoped:** only risk categories. **--force** bypass writes an audit row to `/tmp/mc-forced-completions.log` (and now routes to CEO approval queue under Reality-Anchor P1.1).
2. **Dispatch brief injection** — `~/system/tools/agent-runner.js` injects B+E mandatory clauses into every agent contract/brief, so every Ollama PI agent receives them.

Config

- `~/.claude/hooks/config/enforcement.json` → `lesson_e_b: "block"` (flip from `warn`).
- Revert is one line (`block` → `warn`) if a false-positive ever blocks a legitimate task.

Verification (machine evidence)

Functional gate test `/tmp/evidence-103499/06-block-allow-test.txt`:

Case	Input	Result
BLOCK-CASE	deploy/oauth task, no evidence	`ok:false` (blocked) ✅
ALLOW (docs)	docs task	`skipped:true` (out of scope) ✅
ALLOW (deploy)	deploy + curl-200 + cred-pair	`ok:true` (passes) ✅

Proveo independent verification: PASS — mesh thread `mesh-thr-d2685520-47ea-4923-98b8-f662cf85acc2` (eval agent read all 7 evidence files). Materialized evidence: `/tmp/alai/p2p-pairing-evidence/103499-mesh-thr-d2685520-47ea-4923-98b8-f662cf85acc2.json`.

Enforcement matrix (post-change)

Plane	Surface	Mode
CC (Claude Code)	`cred-pair-gate.sh` PreToolUse hook	block
PI `mc.js` done-gate	`lesson-e-b-validator.js`	block (fail-open, scoped, --force→CEO queue)
PI agent brief	B+E clauses in `agent-runner.js`	mandatory per dispatch

Ref memo: `feedback_generalizable_corrections_2026-06-12`.