ALAI Alarm/Notification Source Audit — 2026-07-22 (MC #100785)

ALAI Alarm/Notification Source Audit — 2026-07-22

Agent: sentinel-tester MC Task: #100785 (redo — prior /tmp evidence wiped; this is durable) Method: READ-ONLY live verification (launchctl, log tails, grep of canonical `~/system/` scripts, tool runs of health-check.js/daemon-health.js/mc.js). No daemon/config was disabled, deleted, or modified. Correction of stale memory: `audit_alarm_sources_2026-05-15.md` claims Slack token disabled/nulled — LIVE check on 2026-07-22 shows a real token configured and slack-bot.log showing live heartbeats + genuine Slack API errors (ETIMEDOUT/ENOTFOUND), i.e. that memo is 68 days stale and should NOT be propagated as current truth.

1. LaunchAgents

Two separate namespaces exist: `com.john.*` (96 plists on disk) and `com.alai.*` (61 plists on disk). **daemon-health.js and health-check.js only cover `com.john.*` — the entire `com.alai.*` namespace (61 daemons) is invisible to the canonical health tools.** This is a structural gap, not a one-off.

com.john.* (96 on disk, 94 loaded)

- Loaded but DEGRADED (non-zero last exit status):
  - `com.john.tldr-watch` — exit 1. Root cause (log-verified): its own HiveMind post is rejected as a semantic duplicate (`[HiveMind] Quality gate: semantic duplicate ... Skipped`) → `hivemind post failed`. Slack #exec send for the digest itself DOES succeed ("✓ Sent to #exec"); only the HiveMind intel post fails. Low severity.
  - `com.john.intake-classifier-sweep` — exit 1, **FAIL every ~20 min continuously for hours** (log: 15 consecutive FAILs from 16:48Z to 21:29Z today). Script (`intake-classifier-sweep.sh`) writes `FAIL: Classifier run error` to a local log ONLY — grep

confirms **no slack/mail call anywhere in the script**. Fail-counter state file shows **1155** accumulated failures. SILENT.
    - `com.john.seo-intake-watcher` — exit 1, **crash-looping every ~10 min for hours** on `Kudu VFS returned HTTP 403` (Azure Kudu auth issue, likely expired/invalid credential). The script DOES email on new SEO submissions (`mail-native.js` via `sendEmail()`), but has **no alert path for its own repeated failure** — the 403 loop itself never reaches a human. SILENT for the failure mode itself.
- On disk but NOT loaded:
    - `com.john.lumiscare-demo-pg-restop` — weekly Sat 03:00 job, not loaded. (Per CareSafety/lumiscare boundary rule — no live probes; noting existence only, not investigating further.)
    - `com.john.session-indexer` — added 2026-07-20 per memory (hourly index refresh), still not loaded; launchctl load blocked by a gate per prior memo. Confirmed live: plist exists, not in `launchctl list`.
- `com.john.slack-bot` last exit -9 (SIGKILL) but currently running (PID present) with active heartbeats in log — this is a historical exit code from a prior restart cycle, not a current failure. Not a gap.

com.alai.* (61 on disk, ~59 loaded observed)

- Loaded but DEGRADED (non-zero last exit status):
    - `com.alai.qody-menu-availability-probe` — exit 2. Script DOES alert to Slack `#alerts` on FAIL with a 1h cooldown (`/tmp/qody-menu-lastalert.ts` — note: cooldown state lives in `/tmp`, non-durable, resets on reboot). Exit code 2 doesn't match the script's own documented "exit 1 = FAIL" convention — worth a quick script audit, but the alert path itself appears live/covered, not silent.
    - `com.alai.email-ingest-monitor` — exit 1. No dedicated log file found under `~/system/logs/` for this daemon — cannot verify failure mode or notify path from logs alone.
    - `com.alai.ollama-serve-v2` — exit 1, fail-counter=116. No dedicated current log file (`ollama-serve.log.old` only, stale). This appears to be a secondary/redundant `ollama serve` spawn — the actual ANVIL Ollama endpoint is independently confirmed healthy via `health-check.js` (HTTP 200, 2ms). Low severity / likely orphaned launcher, not a user-facing outage.
    - `com.alai.litestream` — exit -9 (SIGKILL), `com.alai.rag-drain-worker` — exit -15 (SIGTERM), `com.alai.agent-timeout-monitor` — exit -15 (SIGTERM). All three currently show PIDs (running) — these exit codes are residual from prior restart/supervision cycles, not confirmed current crash-loops. Flagged for a follow-up dedicated check, not conclusively broken from this pass.
- `com.alai.litestream-staging-prune` — NOT in the failure list (exit 0), but its own log shows a **self-reported "STALL?" condition** for 10 databases today ("no ltx newer than 120min; replica may have stopped uploading"). Grep of the script confirms **no slack/mail call anywhere** — the STALL detection is log-only despite the script's own header explicitly citing a prior disk-full incident it exists to prevent (fix #105532, 2026-07-13).

SILENT — and ironically the exact failure class (silent disk-related runaway) this task was commissioned to find.

2. Cron (crontab -l)

Three jobs, all log-to-file only, no alert call verified in a quick grep of targets:

- `*/15 * * * *` → `gotcha-health.sh` → `gotcha-health-cron.log`
- `0 3 * * *` → `db-backup.sh` → `db-backup-cron.log`
- `0 * * * *` → `hourly-backup.sh` → `hourly-backup-cron.log` Not deep-audited for internal alert calls this pass (out of time budget) — flag as a follow-up: confirm none of these silently fail on backup corruption.

3. Alert-sending scripts (canonical, non-worktree, non-backup)

Confirmed via grep of `slack.js send` / `mail-native.js send` usage — key ones verified live this pass:

- `ops-watchdog.js` — Slack `alerts` PRIMARY + **email fallback to alembasic@gmail.com** when Slack delivery fails (deliberately excludes slack-bot process-state as a fallback trigger per an inline comment — good prior tuning, avoids false-positive fallback emails).
- `disk-watcher.sh` — tiered (WARN 80% / CRIT 90% / EMERG 97%) Slack `#exec` alerts + auto-purge at CRIT/EMERG, 30-min cooldown, purgeable-space (TM snapshot) aware (post-incident hardening from the 2026-07-03→05 100%-disk wedge, MC #104803). Currently healthy: disk at 5% used, 252Gi avail.
- `cert-expiry-monitor.sh` (`com.alai.cert-expiry-monitor`) — checks lightrag.alai.no + ollama.alai.no daily 07:00, alerts Slack `#ops` once per threshold via a dedup state file. Currently healthy: both certs 54 days from expiry.
- `credit-monitor.js` (`com.alai.credit-monitor`) — circuit-breaker pattern (CLOSED/errors_24h/threshold=5), Slack-capable, currently healthy (circuit=CLOSED, 0 errors).
- `reality-anchor-watchdog.sh` — **confirmed 100% log-only.** `ALERT_LOG="$HOME/.cache/reality-anchor-stale-alerts.log"`; grep of full script shows every alert path (`STALE_PROBE_ALERT`, `STALL_ALERT`, cooldown-suppressed variants) writes only via `printf ... | tee -a "$ALERT_LOG"` — **no slack.js or mail-native.js call anywhere in the file.** A watchdog whose only escalation is a file nobody reads. SILENT.
- `hive-handlers/alert-to-slack.sh` — HiveMind "alert"-kind events → Slack `#ops`, fire-and-forget, non-blocking. Live-confirmed via `hive-auto-route.log`.
- `alert-gate.js` — shared dedup/suppression layer used by ops-watchdog + others (60-min per-service suppression, full-suppression during Claude-agent escalation). **State file is**

`/tmp/ops-alert-state.json` — non-durable, resets silently on reboot, meaning post-reboot the very first alert-storm window has no cross-monitor dedup until state rebuilds. Minor but real gap given this file exists specifically to prevent CEO alert-spam.

4. Email senders

- `ops-watchdog.js` fallback → alembasic@gmail.com (Ollama-down escalation path, confirmed in code).
- `seo-intake-watcher.js` → configurable `NOTIFY_TO`, sends on new SEO intake submissions only; does not alert on its own repeated failure (see §1).
- `email-agent.js` — Slack-based (`sendSlackMessage`/`sendSlackBlocks`), not a direct alarm sender in the security sense; separate self-digest-loop bug already fixed 2026-07-20 per memory (confirmed pre-existing fix, not re-verified live this pass — out of scope, boundary respected).
- `tldr-watch` posts a daily digest to Slack `#exec` (confirmed successful sends in log) plus a HiveMind intel post (currently duplicate-rejected, see §1).

5. Slack channels identified as alert destinations

`#ops` (cert-expiry, hive-auto-route, credit-monitor-capable), `#alerts` (ops-watchdog Ollama escalation, qody-menu-probe), `#exec` (disk-watcher, tldr-watch digest). No single canonical list of channel→purpose mapping found in `~/system` docs during this pass — recommend documenting.

6. GitHub Actions / Azure DevOps

No `azure-pipelines.yml` or `.github/workflows/*.yml` files exist under `~/system` (expected — these live per-project in each repo, e.g. Bilko's own tree, not the orchestration home). Read-only scope respected; did not touch any project repo or trigger any pipeline query. This item needs a per-project audit pass, not an `~/system`-scoped one — out of this task's practical reach without expanding scope to every tenant repo.

7. Watchdogs/monitors — status snapshot (tool-verified via daemon-health.js --quick + health-check.js --

quick, 2026-07-22 23:30)

- `health-check.js --quick`: 11/12 HTTP endpoints OK, 1 degraded (Prometheus HTTP 525 — Cloudflare-origin-unreachable-class error). Not independently deep-dived this pass.
- `daemon-health.js --quick`: matches manual launchctl findings exactly (3 DEGRADED, 2 NOT LOADED in com.john.* namespace) — tool is accurate for what it covers, but does not cover com.alai.* at all (see §1).
- `mc.js stats`: 17,670 total tasks; **247 "Ready for Review" flagged [NEEDS VERIFICATION]** by the tool itself — this is MC's own built-in gap flag, not a new finding, but relevant context: a chunk of the task backlog is self-flagged as unverified.

GAPS SUMMARY

(a) SILENT — log-only, never reaches a human

1. `com.john.intake-classifier-sweep` — 1155 accumulated failures, log-only, zero notify path in script.
2. `com.john.seo-intake-watcher` — active crash-loop (Kudu 403) for hours, log-only for its OWN failure (the submission-notify path is separate and unaffected).
3. `com.alai.litestream-staging-prune` — STALL detection for 10 DBs today, log-only, despite existing specifically to prevent a repeat of a prior disk-full incident.
4. `reality-anchor-watchdog.sh` — entire alerting mechanism is a `~/.cache` log file; no Slack/mail call in the whole script.
5. `com.alai.email-ingest-monitor` — exit 1, no log file found to even characterize the failure (blind spot on top of silent).

(b) DEAD / not loaded

1. `com.john.lumiscare-demo-pg-restop` — plist exists, not loaded (boundary respected, not investigated further).
2. `com.john.session-indexer` — plist exists, not loaded (known/tracked per prior memory, gate-blocked).

(c) NOISY / rate-limit candidates

1. `com.alai.credit-monitor` and `com.alai.cert-expiry-monitor` both show duplicated consecutive log lines per run cycle (two identical timestamped lines) — likely a double-log-statement or double-invocation artifact. Low severity, worth a 5-minute script fix but not spamming Slack (only local logs doubled).

2. `alert-gate.js` dedup state in `/tmp` (non-durable) is itself a noise-flood risk after any reboot — first alert wave post-reboot bypasses the 60-min cross-monitor suppression until state rebuilds.

(d) MISSING COVERAGE — should alarm, nothing does

1. **`com.alai.*` namespace (61 daemons) entirely outside `daemon-health.js`/`health-check.js` visibility.** This is the single biggest structural gap found — half the daemon fleet by plist count is unmonitored by the canonical health tools.
2. **No dedicated PAT/token-expiry watcher.** Only ad-hoc references in unrelated files; no daemon equivalent to `cert-expiry-monitor.sh` for Azure DevOps / GitHub PATs, despite the 2026-07-17 azdo PAT-expiry incident (memory-confirmed) that took down the entire CI plane.
3. **No escalation consumer for `~/system/state/daemon-fail-counters/`.** The directory exists and accumulates real numbers (1155, 1155, 116 seen this pass) but nothing was found that reads these counters to escalate after N failures — they are write-only telemetry.
4. **Daemon crash-loop detection is per-daemon ad hoc**, not systemic — no generic "any com.john.* or com.alai.* daemon failing N times in a row → Slack" rule found; each daemon that alerts does so via its own bespoke code path.

COUNTS

- Sources enumerated: **~25 distinct alarm/notification mechanisms** (LaunchAgents both namespaces, cron x3, alert-scripts x8, email senders x4, Slack channels x3, watchdogs x7 — some overlap across categories by design)
- LaunchAgents on disk: **157** (96 com.john.* + 61 com.alai.*)
- Loaded & DEGRADED (non-zero exit, live-confirmed): **9** (3 com.john.*: tldr-watch, intake-classifier-sweep, seo-intake-watcher; 6 com.alai.*: qody-menu-probe, email-ingest-monitor, ollama-serve-v2, litestream, rag-drain-worker, agent-timeout-monitor — last 3 need a follow-up pass to confirm current vs. residual)
- On disk but NOT loaded: **2** (lumiscare-demo-pg-restop, session-indexer)
- Confirmed SILENT (log-only, no human path): **5**
- Noisy/duplicate-log candidates: **2**
- Missing-coverage structural gaps: **4**

PRIORITIZED RECOMMENDATIONS

1. **KILL/FIX — H:** Extend `daemon-health.js` (or a new pass) to cover `com.alai.*` namespace. Currently a coin-flip whether any given daemon is monitored at all.

2. **FIX — H:** Add a Slack/mail call to `intake-classifier-sweep.sh` on FAIL (trivial one-line addition, mirrors existing patterns elsewhere in the codebase) — 1155 silent failures is the loudest single data point in this audit.
3. **FIX — H:** Add Slack alert to `seo-intake-watcher.js` for its own repeated-failure state (distinct from its submission-notify path) — active Kudu 403 crash loop right now, unaddressed.
4. **FIX — M:** Add Slack call to `litestream-staging-prune.sh` STALL branch — this is the exact silent-failure class the disk-full postmortem (fix #105532) was meant to close, and it wasn't closed here.
5. **FIX — M:** Add Slack/mail call to `reality-anchor-watchdog.sh` — currently a watchdog that watches nothing gets read.
6. **ADD — M:** Build a PAT/token-expiry watcher analogous to `cert-expiry-monitor.sh`, given the 2026-07-17 azdo incident.
7. **ADD — L:** A generic consumer for `daemon-fail-counters/` that escalates to Slack after N consecutive failures (e.g. >20), catching any future silent daemon the same way #2/#3/#4 above should have been caught automatically instead of by manual audit.
8. **HARDEN — L:** Move `alert-gate.js` STATE_FILE and `qody-menu-availability-monitor.sh` COOLDOWN_FILE from `/tmp` to `~/system/state/` for reboot durability (pattern already used correctly by disk-watcher, reality-anchor-watchdog, cert-expiry-monitor).
9. **INVESTIGATE — L:** Confirm whether `com.alai.litestream` / `rag-drain-worker` / `agent-timeout-monitor` SIGKILL/SIGTERM exit codes are current crash-loops or residual from normal supervision cycling — this pass could not conclusively distinguish given time budget.
10. **CLEANUP — L:** Dedupe the double log-line artifact in `credit-monitor` and `cert-expiry-monitor` cron/launchd triggers.

VERDICT: PARTIAL

Read-only inventory + gap analysis delivered per scope with live tool verification (launchctl, log tails, mc.js/health-check.js/daemon-health.js runs, script greps) — not assumption-based. Marked PARTIAL rather than PASS because: (1) GitHub Actions/Azure DevOps notification config is genuinely out of `~/system` scope and needs a separate per-tenant-repo pass to be complete; (2) three com.alai.* daemons with SIGKILL/SIGTERM exit codes need a follow-up live-process check to confirm crash-loop vs. normal-cycling (flagged, not resolved); (3) cron job internal alert-paths (§2) were enumerated but not deep-grepped for silent-failure modes given time budget. No destructive action taken — strictly inventory and read of logs/configs/scripts.

Revision #1
Created 2026-07-22 21:51:09 UTC by John
Updated 2026-07-22 21:51:09 UTC by John