2026-04-30 5-team audit

Full architectural audit conducted by 5 expert teams (Resolver, Kleppmann, Anthropic, OpenAI, Kelsey Hightower) with master synthesis by Petter Graff. Audit completed 2026-04-30 per MC #10357.

T6 — Petter Graff master synthesis

ALAI Holding AS — Chief Architect Master Synthesis

MC #10357 T6 | Petter Graff, Lead Architect

Date: 2026-04-30 | Status: FINAL

SECTION 1: EXECUTIVE SUMMARY (CEO read-first, 60 seconds)

What you have: A sophisticated single-machine AI orchestration system running on ANVIL (Mac Studio). The core — Claude Code as orchestrator, 40 sub-agents, 10,220 tracked tasks, 138 LaunchAgent daemons, Litestream backup, local Ollama fleet — is structurally sound. Better than most funded startups at similar stage. **The three risks that matter:**

1. **Legal exposure right now.** 84 signed corporate PDFs, NDAs, and insurance documents have NEVER been uploaded to cold storage. `legal-docs-azure-sync` has been failing silently since deployment. If you need to produce a document for Akershus funding review or any legal dispute, you are relying on local disk only.

2. **$129K/day burn with zero revenue.** 91% is Opus. The cost-tracker numbers need unit verification — if these are actual dollars (not micro-cents), you have 2-3 weeks of runway at this

rate before any individual account limit triggers. A runaway Opus loop at midnight runs for 8 hours before anyone sees it.

3. **Azure VM is an unprotected single point of failure.** BookStack (your operational wiki), Vaultwarden (every system password), Documenso, Grafana, Planka — none have ANVIL-side backup. If Azure deletes that VM tonight, the knowledge base and all credentials are gone.

What to do tomorrow morning: 1. Fix `legal-docs-azure-sync.sh` — add `--overwrite` flag (15-minute fix) 2. Verify cost-tracker units against actual Anthropic billing dashboard 3. Enable Azure VM-level backup policy from Azure portal (30-minute click job) 4. Run: `node ~/system/tools/cost-tracker.js summary today` and pin the number

The architecture is salvageable. The process discipline needs surgery, not the infrastructure.

SECTION 2: CURRENT-STATE ARCHITECTURE DIAGRAM

```
============================================================
LAYER A — ANTHROPIC ORCHESTRATION SURFACE
============================================================
```

CEO Prompt | v [John — Claude Code, Sonnet-4.6] | +-- UserPromptSubmit hooks (6 hooks, cache-defeating): | boot-enforcer.sh | validation-state-injector.sh <-- INJECTS mc.js list EVERY prompt | alai-hooks auto-verify (Kotlin CLI, 14MB binary) | alem-instruction-checker.sh | feasibility-check-advisory.sh | incident-response-mode.sh | +-- /mehanik (pre-dispatch gate, slash command) | -> ~/.claude/agents/mehanik.md [model: sonnet — OVERTIRED; should be haiku] | -> writes /tmp/mehanik-cleared-<MC_ID> | -> pre-dispatch-gate.sh validates 13-field marker | +-- Task tool dispatch (PreToolUse hook stack): lock-john-dispatch-cap.sh [cap=3; TOO LOW — should be 8] john-max-depth-gate.sh [ZAKON #28, trip-wires 1+2: CORRECT] pre-dispatch-gate.sh pre-action-da-gate.sh | v Sub-agent (isolated context, ~/.claude/agents/<name>.md) | 40 .md files — 22 doing real work, 8 umbrella personas (ROUTING TRAP), 5 reviewer duplicates, 7 mapped-but-missing (BROKEN DISPATCH) | v /task-postflight --> Proveo verify --> mc.js done alai-hooks evidence-gate enforces postflight marker

== = LAYER B — ORCHESTRATION KERNEL (~/system/kernel/) == =

pi-orchestrator.js [5,251 LOC MONOLITH] Status: IDLE since 2026-04-28 22:24Z (48+ hours) Blocked by: 3 tasks stuck in ready_for_review (#10038, #10039, #10043) Owns: 1,619 tasks (1,324 paused + 295 blocked) — NO PICKUP MECHANISM | +-- 30s poll loop -> getNextTask() [non-

atomic 3-statement claim — RACE CONDITION] +-- model selection (tier 1-5) -> Ollama (FORGE/ANVIL) or Claude API +-- quality gate (lint/test/semantic) +-- proof-of-work verification +-- HiveMind posting (fire-and-forget, NOT events.db) +-- cost tracking +-- YouTube RAG ingest [verified Ollama qwen3:8b, ZERO Anthropic cost]

cross-company-bus.js [6h cron] 85 matches today, 1 task created Enforces cross-company-routes.json (10 rules, no real-time hook)

chain-runner.js [YAML chains, lightly used] sprint-pipeline.js [DAG state in JSON column — cannot SQL-query] pipeline-controller.js [13-phase lifecycle]

DEAD COMPONENTS (marked): [DEAD] agent-orchestrator.js — archived 2026-03-21 (was never canonical) [DEAD] HelixSupport — merged into FlowForge; stale ref in cross-company-routes.json [DEAD] Proxima — merged into Lexicon; stale refs exist

==========
= COMPANY / AGENT ROUTING LAYER
==========
=

specialist-mapping.json — 28 agents mapped, 12 companies 17 ORPHAN agents (in ~/.claude/agents/ but NOT in mapping) 7 BROKEN entries (in mapping but .md file does NOT exist) Datavera: ZERO agents (ML/analytics domain UNROUTED)

Cross-company routing (bi-modal only): WRONG: tier-router.js — only "ollama" | "cc" | "human-queue" — no OpenAI, no Gemini OK: comms-responder.js — groq -> claude-api -> claude-cli -> ollama (only multi-vendor path) OK: rag-router.js — cache -> local-ollama -> local-enriched -> external (Claude fallback)

discover.js routing "<query>" — primary routing query tool Returns only indexed agents; 17 orphans are INVISIBLE to this tool

==========
= DATA LAYER
==========
=

mission-control.db [25.6 MB main + 23.3 MB WAL — needs checkpoint] 10,220 tasks | 14 concurrent writers | NO command authority boundary Non-atomic task claim (3 separate DML statements, no BEGIN TRANSACTION) Missing index: tasks(delegated_to) — hot query path unindexed 27 dangling FK references (PRAGMA foreign_keys NOT enforced) version column exists for optimistic locking — enforcement unverified

hivemind.db [60.1 MB main + 147 MB WAL — CRITICAL BLOAT] WAL = 245% of main file. Multi-second startup delay on crash restart.

flywheel.db [224.2 MB main + 264 MB WAL — HIGH BLOAT] WAL = 118% of main. Continuous Litestream replication of dirty WAL. flywheel.db.interactions has used_for_training column —

distillation HALF-BUILT

events.db [14.6 MB + 4.5 MB WAL — acceptable] Schema: idempotency_key UNIQUE, correlation_id, causation_id — CORRECTLY DESIGNED Problem: PRIMARY orchestration path BYPASSES this store (posts to HiveMind instead)

ingest-queue.sqlite.corrupt [~/system/state/ — evidence of past corruption, no RCA]

Litestream replication: 50+ databases -> Azure Blob (alaibackups0ebb.blob.core.windows.net) P0 critical: 1s sync, 7d retention azure-db-backup.sh: PARTIAL FAILURE (intermittent timeout on 400M+ blobs) RESTORE DRILL: NEVER PERFORMED

==========
= DAEMON FLEET (138 LaunchAgents)
==========
=

47 running with PIDs 69 calendar-OK (scheduled, idle) 4 HARD ERROR (exit 256 every run): [FAIL] com.alai.azure-db-backup — LightRAG blob upload timeout (intermittent) [FAIL] com.alai.apply-knowledge — needs investigation [FAIL] com.john.auto-verify-regression — MC #10173 closed but still failing [FAIL] com.john.infra-drift-detector — git rename limit (3178 files), one-liner fix

CRITICAL LEGAL FAILURE: [FAIL] com.john.legal-docs-azure-sync — exit 512, BlobAlreadyExists (missing --overwrite) 84/84 legal documents NEVER uploaded. Has NEVER worked.

NOT LOADED: com.alai.lightrag-auto-heal — plist exists, not bootstrapped com.john.rdap-audit-quarterly

FALSE ALARM FLOOD: com.john.ops-watchdog — docs.basicconsulting.no (302=live), vault.basicconsulting.no (302=live) flagged as DOWN 10+ consecutive checks. Real outage (lumiscare.no) buried in noise. lumiscare.no: fail 2500/2 — GENUINE outage, invisible due to noise.

MC #10173 FRAUD: Marked done 2026-04-30T03:55Z. 4 of 11 original failures STILL FAILING at 19:50Z.

==========
= VENDOR COUPLING
==========
=

Anthropic: 9/10 lock-in (Claude Code IS the orchestrator runtime) $129K/day (243 req), $368K/week (1,429 req) — 100% claude-cli 91% of requests are Opus (176 req today at ~$730 avg) UNIT VERIFICATION REQUIRED against actual billing

Local Ollama: 6/10 dependency (FORGE 10.0.0.2:11434 + ANVIL localhost:11434) 7 models on FORGE, 8 on ANVIL, ~143GB allocated MLX servers: :11435 gemma-4-26b, :11436 qwen3-32b, :11437 qwen3-coder-30b Fine-tuned: alaiml-task-v1, alaiml-email-v1, alaiml-tender-v1

Google Gemini: 1/10 (gemini-reviewer.md only, free tier) Groq: 2/10 (comms-responder fallback only) OpenAI: 0 requests ever — 0% integrated, 8/10 missed opportunity

==========

SECTION 3: TOP 10 PROBLEMS (Ranked by L×S/E)

Scoring: L = Likelihood (1-5), S = Severity (1-5), E = Effort-to-fix (1-5 hours) Higher score = worse and cheaper to fix = fix first.

P0 — THIS WEEK (Fix or accept the consequence)

#1 — legal-docs-azure-sync: 84 legal documents never uploaded

Score: L=5, S=5, E=1 → **25.0 Teams:** T5 (primary), also violates T1 ZAKON ARCHIVE FIRST **Evidence:** `legal-docs-azure-sync.log` shows `ERROR: BlobAlreadyExists` on all 84 files, plus ISBD source directory does not exist. Exit 512 every run. Has never succeeded. **Risk:** If Akershus funding review, investor due diligence, or legal dispute requires document production — reliance on ANVIL local disk only. If ANVIL fails without verified Azure backup, documents are gone. **Fix:** Add `--overwrite` flag to `az storage blob upload-batch` in `/Users/makinja/system/daemons/legal-docs-azure-sync.sh`. Fix ISBD path. Verify next run. Estimated: 15 minutes.

#2 — Cost units unverified: $129K/day could be catastrophic or harmless

Score: L=3, S=5, E=1 → **15.0 Teams:** T4 (primary), T5 (flagged) **Evidence:** `cost-tracker.js summary today` reports $129,209.12 today, $368,509.48 this week. These numbers come from ALAI's internal logger (claude-cli backend). It is NOT verified against Anthropic's billing dashboard. **Risk:** If actual dollars: 2-3 weeks until account-level limits trigger. No revenue to offset. If micro-

cents: the architecture is fine but the CEO is making decisions on phantom numbers. **Fix:** Login to console.anthropic.com, compare actual billing charges for this week to `cost-tracker.js` output. One manual verification, 10 minutes. Then add a monthly reconciliation task.

#3 — Azure VM: BookStack + Vaultwarden have NO backup

Score: L=2, S=5, E=2 → **5.0 Teams:** T5 (primary), T2 (cross-reference) **Evidence:** `litestream.yml` covers `~/system/databases/` on ANVIL, not `4.223.110.181`. `azure-db-backup.sh` backs up ANVIL Docker volumes, not the VM. `dr-sync.sh` syncs to FORGE (10.0.0.2), not Azure VM. **Risk:** Azure VM at `4.223.110.181` runs BookStack (operational wiki — 100% of your system documentation), Vaultwarden (every API key, every credential), Documenso, Grafana, Planka. VM deletion = total knowledge and credential loss. Vaultwarden contains the keys needed to rebuild everything. **Fix:** Enable Azure VM Backup policy from Azure portal (30-minute click job). OR add a nightly `mysqldump` + `sqlite3 .dump` pull from VM to ANVIL via SSH cron.

#4 — MC #10173 closed under false pretenses: 4 of 11 failures STILL FAILING

Score: L=5, S=3, E=2 → **7.5 Teams:** T5 (primary), T3 (process failure pattern) **Evidence:** Fleet snapshot at 19:50Z (16 hours after #10173 marked done) shows `calendar_err_256` on `auto-verify-regression`, `infra-drift-detector`, `apply-knowledge`. `legal-docs-azure-sync` still exit 512. **Risk:** This is primarily a process integrity problem. Hard Constraint #3 ("Builder cannot say done") was violated. The 4 remaining failures compound daily. infra-drift-detector has a one-liner fix (git config). **Fix:** Reopen MC #10173. Fix infra-drift-detector (add `git config diff.renameLimit 0`). Investigate apply-knowledge. Do NOT mark done until fleet snapshot confirms 0 errors on all 4.

#5 — WAL bloat: 411MB uncheckpointed across hivemind.db + flywheel.db

Score: L=4, S=3, E=1 → **12.0 Teams:** T2 (primary) **Evidence:** hivemind.db-wal = 147MB (245% of main), flywheel.db-wal = 264MB (118% of main). PRAGMA wal_checkpoint returned 0 pages checkpointed for mission-control.db. **Risk:** On unexpected restart, SQLite replays entire WAL before any operation. For hivemind.db, this means 147MB replay — multi-second to minute-level startup delay. Under full-disk condition, WAL growth can prevent writes to ALL SQLite databases sharing the volume. **Fix:** Add a 4-hour LaunchAgent running `PRAGMA wal_checkpoint(TRUNCATE)` on hivemind.db, flywheel.db, and mission-control.db. Create ~/system/tools/wal-checkpoint.js. Estimated: 2 hours.

P1 — THIS MONTH

#6 — ops-watchdog false alarms drowning real outages

Score: L=5, S=3, E=1 → **15.0 Teams:** T5 (primary) **Evidence:** `docs.basicconsulting.no` and `vault.basicconsulting.no` both return HTTP 302 (services live) but watchdog treats 302 as failure. Both show `fail 10/2`. Meanwhile `lumiscare.no` shows `fail 2500/2` — genuine 2500-check outage buried in noise. **Risk:** Watchdog loses credibility. Real production outages go undetected because every alert is assumed false positive. Lumiscare has been down for 2500 consecutive checks (2-minute interval = 83 hours of outage) with no escalation. **Fix:** Add `--location --max-redirs 1` to curl checks in ops-watchdog script. Accept 2xx/3xx as healthy. Investigate lumiscare.no DNS/CF issue separately. 1 hour.

#7 — 17 orphan agents + 7 broken mapping entries

Score: L=5, S=2, E=3 → **3.3 Teams:** T1 (primary), T3 (confirms) **Evidence:** 17 agents in `~/.claude/agents/` are invisible to `discover.js routing`. 7 mapping entries (`hadi-hariri`, `lee-robinson`, `james-bach`, `lisa-crispin`, `dorota-huizinga`, `maria-santos`, `resolver`) point to .md files that do not exist — any routing to these crashes at dispatch. **Risk:** John bypasses routing and dispatches manually. Programmatic routing (cross-company-bus, pi-orchestrator) cannot see 37% of agents. When pi-orchestrator resumes, it will route to broken entries and generate task failures. **Fix:** Add 17 orphans to specialist-mapping.json. Delete or stub-create 7 broken entries. Run `~/bin/agent-definitions-sync.sh`. Estimated: 3 hours.

#8 — Non-atomic task claim: latent double-claim race condition

Score: L=2, S=4, E=2 → **4.0 Teams:** T2 (primary) **Evidence:** The claim path in pi-orchestrator.js is 3 separate DML statements (lease_until update, status=in_progress, active_work insert) with no wrapping transaction. If process crashes between statements 1 and 2, state is inconsistent. With 14 concurrent writers to mission-control.db, the race window is real. **Risk:** With current pi-orchestrator idle this is latent not active. When it resumes at scale, double-claims can cause duplicate work, wasted Opus spend, and corrupted task state. **Fix:** Wrap the 3-statement claim in `db.transaction(() => { ... })()` using better-sqlite3's synchronous transaction API. Also add `db.pragma('foreign_keys = ON')` per connection. Add missing index

`idx_task_scheduling_next_eligible`. Estimated: 1-2 days.

#9 — Dispatch cap 3/session: too restrictive for legitimate workflows

Score: L=5, S=2, E=1 → **10.0 Teams:** T3 (primary), T4 (confirms) **Evidence:** `lock-john-dispatch-cap.sh` blocks 4th Task tool invocation per session. One legitimate task = build + Proveo + Skillforge = 3 dispatches. A second task in same session is immediately blocked. Today's 5-team audit required `[CEO_APPROVED]` override. **Risk:** John is forced to use `[CEO_APPROVED]` as a workaround for normal workflows. This defeats the purpose of ZAKON #28 tracking — legitimate overrides obscure actual drift events. Anthropic's own SDK design assumes multi-sub-agent parallel dispatch. **Fix:** Edit `~/.claude/hooks/lock-john-dispatch-cap.sh` line 77: change `MAX_DISPATCHES=3` to `MAX_DISPATCHES=8`. Also add skillforge, proveo to bootstrap-exempt list. Single-line change. 10 minutes.

#10 — validation-state-injector defeats prompt prefix cache every session

Score: L=5, S=2, E=2 → **5.0 Teams:** T3 (primary), T4 (cost context) **Evidence:** `validation-state-injector.sh` runs `mc.js list --owner john --priority H --json` on every UserPromptSubmit (30s TTL cache). Dynamic output changes whenever task state changes, invalidating Anthropic's automatic 5-min ephemeral cache from that breakpoint forward. On a 4-hour session with frequent task state changes: ~50% cache miss on content that should be static. **Risk:** Direct cost amplifier. T3 estimates ~$2.27/4-hour session in preventable input token charges. At daily usage, this compounds. More importantly, it is architecturally inverted: ambient state does not belong in the prompt prefix. **Fix:** Move `validation-state-injector.sh` from UserPromptSubmit to a `/status` slash command. CEO queries when they want it; it stops poisoning every prompt. 2 hours.

SECTION 4: WHAT'S MISSING (Gap Analysis)

These pieces do not exist and should:

G1 — Pre-dispatch task classifier hook

File: `~/.claude/hooks/pre-dispatch-classifier.sh` (does not exist) **What it does:** Takes task title +

description, returns {company, agent, tier} before Task tool spawn. Currently John picks agents by intuition or uses generic `builder.md` catch-all. **Why it matters:** John dispatches 95%+ work via Task tool directly. Without classification, ZAKON #27 (one product per session) and company boundary rules have no enforcement point. **Spec:** Call `discover.js routing "$TASK_TITLE"`, parse JSON, inject into task metadata. If no match, route to Resolver. Block dispatch if company boundary violated.

G2 — Cost spike real-time alert

File: `~/system/daemons/cost-spike-alert.sh` (does not exist) **What it does:** Monitors `cost-tracker.js summary today` every 30 minutes. If hourly burn exceeds configurable threshold (e.g., $10K/hour based on verified units), sends immediate Slack DM to CEO. **Why it matters:** $129K/day with no revenue. A runaway Opus loop from midnight to 8am is currently invisible for 8 hours. The daily digest at 08:00 is 8 hours too late.

G3 — 3-vendor consensus router for /prompt-forge

File: `~/system/tools/consensus-router.js` (does not exist) **What it does:** For architecture decisions only (not routine tasks): fan out to Claude-Opus + GPT-5 + Gemini-2.5-Pro. A 4th cheap model judges divergence. 3-way agreement = ship. Divergent = human queue. **Why it matters:** The `feedback_john_recursive_drift.md` failure pattern — where John chains 8 nested MCs from a single strategic prompt — stems from single-model overconfidence. A 3-way disagreement is a natural circuit breaker. **Cost:** ~$0.50-2.00 per consensus call. Cheap insurance.

G4 — Daily CEO fleet brief via Slack

File: `~/system/tools/morning-fleet-brief.js` + `com.alai.morning-fleet-brief.plist` (do not exist) **What it does:** At 09:00 daily: daemon errors, cost yesterday vs prior day, MC status, service health, backup health, lumiscare.no status — 8 lines in #exec Slack. Current cost-daily-report writes to a file no one checks. **Why it matters:** CEO visibility gap. Kelsey's T5 brief showed every observability piece EXISTS but is NOT surfaced. The aggregation glue is missing.

G5 — Restore drill for mission-control.db

Documentation: `/Users/makinja/system/config/litestream-restore.yml` exists but no drill has been run. **What it does:** Monthly: `litestream restore -config ~/system/config/litestream.yml -replica mc-abs /tmp/mc-restore-test.db` then verify row count matches live DB. **Why it matters:** T2's single most dangerous gap: "If Litestream's Azure SP credentials have rotated or SP permissions have changed, the replicas may be stale without any alert." Without a drill, you do not know if your backup works until you need it.

G6 — Atomic task claim with BEGIN IMMEDIATE

File: `/Users/makinja/system/kernel/pi-orchestrator.js` (needs modification, not creation) **What it does:** Wraps 3-statement task claim in `db.transaction()`. Adds FK enforcement. Adds missing indexes. **Why it matters:** Race condition is latent. With pi-orchestrator currently idle, there is a window to fix this before resumption.

G7 — openai.js adapter + gpt-4o-mini Batch for RAG ingest

File: `~/system/tools/adapters/openai.js` (does not exist) **What it does:** Mirror of groq.js structure. Register in adapters/index.js. Wire lightrag-bulk-upload.js to call OpenAI Batch API for embeddings refresh. **Why it matters:** gpt-4o-mini is 5.3x cheaper input / 6.7x cheaper output than Haiku. For high-volume RAG ingest, this is the single largest unactualized cost reduction available.

G8 — WAL checkpoint cron

File: `~/system/tools/wal-checkpoint.js` + `com.alai.wal-checkpoint.plist` (do not exist) **What it does:** Every 4 hours: PRAGMA wal_checkpoint(TRUNCATE) on hivemind.db, flywheel.db, mission-control.db. Already described in T2 recommendations.

SECTION 5: DAY-TO-DAY OPERATIONS PLAYBOOK

This section is operational ritual, not strategy. Every command is concrete and runnable.

MORNING RITUAL (5 minutes, after boot, EVERY day)

```
# 1. Boot system (mandatory)
bash ~/system/boot.sh
```

2. CEO emails first (ZAKON)

node ~/system/tools/email-inbox.js pending

3. Cost check — first number you see

node ~/system/tools/cost-tracker.js summary today

Red flag: today > $50K (if units are dollars). Investigate immediately.

4. H-priority tasks

node ~/system/tools/mc.js list --owner john --priority H --status open

5. Daemon health

python3 -c " import json; d=json.load(open('/Users/makinja/system/state/daemon-fleet-status.json'))['daemons'] errs=[k for k,v in d.items() if 'err' in v['state'] or 'exit_512' in v['state']] print('Fleet errors:', len(errs)); [print(' -', e) for e in errs] "

Expected: 4 or fewer (the known 4). New errors = investigate before any work.

6. Backup health

tail -3 ~/system/logs/azure-db-backup.out tail -3 ~/system/logs/dr-sync.log

Expected: "COMPLETE" and "DR Sync COMPLETE" within last 24h / 6h

7. pi-orchestrator status

tail -2 ~/system/logs/pi-orchestrator.log

Expected: "No eligible tasks" (idle is OK) or recent cycle. ERROR = immediate triage.

8. Legal docs sync check (until fixed)

tail -3 ~/system/logs/legal-docs-azure-sync.log

Expected: "COMPLETE" or "All blobs uploaded". Until fixed: "FAIL" = escalate.

Total: 5 minutes. If any red flag: create MC before starting any other work.

DURING WORK: Pre-Dispatch Protocol

Before EVERY Task tool dispatch:

```
# Step 1: Route query
node ~/system/tools/discover.js routing "<task description in plain language>"
```

Returns company + agent. Use the named specialist, NOT generic builder.md

Step 2: If routing returns nothing, use Resolver

Dispatch to resolver.md with task description. Let it classify.

Step 3: Check current dispatch count

If you have dispatched 2+ tasks this session: confirm you are

under cap.

Cap is 3 (current) — raise to 8 as per T3 recommendation.

Step 4: Mehanik clearance (H/BLOCKER only)

/mehanik — then wait for marker file before dispatch

Step 5: Dispatch with correct agent name

Task tool -> <named-expert>.md, NOT "builder"

Never dispatch to umbrella personas (codecraft.md, proveo.md, etc.) — they route ambiguously.

EVENING CHECK (3 minutes, end of work session)

```
# 1. Cost cumulative check
node ~/system/tools/cost-tracker.js summary today
```

Compare to morning. If increase > $30K during your session: check which tasks ran.

2. New daemon failures

python3 -c " import json; d=json.load(open('/Users/makinja/system/state/daemon-fleet-status.json'))['daemons'] errs=[k for k,v in d.items() if 'err' in v['state']] print('Errors:', len(errs)); [print(' -', e) for e in errs] "

3. Clear review queue

node ~/system/tools/mc.js list --status ready_for_review --json | python3 -c " import json,sys; tasks=json.load(sys.stdin) [print(f'#'+str(t['id']),t['title'][:60]) for t in tasks] "

Any tasks in ready_for_review owned by pi-orchestrator: these block pi-orch restart.

Tasks #10038, #10039, #10043 are current blockers.

4. Postflight compliance

node ~/system/tools/mc.js list --owner john --status done --json | tail -5

Confirm last 5 completed tasks have dod_evidence field populated.

WEEKLY REVIEW (30 minutes, every Monday)

```
# 1. Cost week-over-week
node ~/system/tools/cost-tracker.js summary week
```

Compare to prior week. Flag any >20% increase. Check model distribution — Opus >85% = review.

2. Full fleet error trending

Read ~/system/state/daemon-fleet-status.json summary block.

Errors should trend DOWN, not accumulate. Screenshot for comparison.

3. Azure backup verification

```
az storage blob list \ --account-name alaibackups0ebb \ --container-name system-db-backups \ --prefix "litestream/mission-control" \ --query "[?contains(name,'$(date +%Y-%m)')].{name:name,modified:properties.lastModified}" \ -o table
```

Expected: blobs from current week. None = Litestream silently stopped.

4. Archive-first ledger sweep

```
cat /tmp/archive-first-scan-report-$(date +%Y%m%d).txt | head -20
```

Check candidate count. If growing: create MC to archive top-priority legal docs.

5. MC closure validation (ZAKON Hard Constraint #3)

```
node ~/system/tools/mc.js list --status done --owner john --json | \ python3 -c " import json,sys; tasks=json.load(sys.stdin) bad=[t for t in tasks if not t.get('dod_evidence') and not
```

t.get('validation_timestamp')] print('Undocumented closures:', len(bad)) [print(' -', t['id'], t['title'][:50]) for t in bad[:5]] "

Any closures without evidence: investigate. MC #10173 failure pattern.

6. ZAKON #28 override count

grep "\[CEO_APPROVED\]" ~/.claude/hooks/john-max-depth-gate.log 2>/dev/null | \ grep "$(date +%Y-%m)" | wc -l

If >2/week: escalate to CEO. Override cadence is a leading indicator of drift.

7. Archive-first top candidates

node ~/system/tools/discover.js "unarchived legal docs contracts NDA"

Pick top 3, archive to archive.alai.no (ZAKON ARCHIVE FIRST)

MONTHLY ARCHITECTURE SWEEP (30 minutes, first of each month)

```
# 1. Full restore drill (THIS IS MANDATORY — never skip)
litestream restore \
  -config ~/system/config/litestream.yml \
  -replica mc-abs \
  /tmp/mc-restore-test.db
sqlite3 /tmp/mc-restore-test.db "SELECT COUNT(*) FROM tasks;"
sqlite3 ~/system/databases/mission-control.db "SELECT COUNT(*) FROM tasks;"
```

These numbers should match (within ~10 rows for replication lag). If not: ALERT.

rm /tmp/mc-restore-test.db

2. Agent inventory audit

ls ~/.claude/agents/*.md | wc -l jq '.mappings | length' ~/system/agents/specialist-mapping.json

Expected after cleanup: 22 files, 19 mappings. If growing without justification: review.

3. RAG/HiveMind growth

node ~/system/tools/cost-tracker.js summary month sqlite3 ~/system/databases/hivemind.db "SELECT COUNT(*) FROM intel;"

Track growth rate. If >50K new entries/month: verify source diversity.

4. Azure VM backup verification (manual)

```
ssh alai-admin@4.223.110.181 " cd /opt && du -sh bookstack-data/ vaultwarden-data/ 2>/dev/null
echo 'Last modified:' && ls -lt /opt/ | head -5 "
```

Until VM-level backup is configured: this is your only assurance.

5. Cost model distribution audit

```
node ~/system/tools/cost-tracker.js task-summary | head -20
```

Identify top 5 cost-generating tasks. Are they justified? Opus where Haiku sufficient?

6. Orphaned temp cleanup

```
ls /tmp/alai-azure-backup-* 2>/dev/null | wc -l
```

If >10: rm -rf /tmp/alai-azure-backup-* (stalled backup runs, ~700MB each)

ls /tmp/mehanik-cleared-* 2>/dev/null | wc -l

Clean stale mehanik marker files from completed sessions.

7. Litestream WAL checkpoint verification

sqlite3 ~/system/databases/hivemind.db "PRAGMA wal_checkpoint(TRUNCATE);" sqlite3 ~/system/databases/flywheel.db "PRAGMA wal_checkpoint(TRUNCATE);" sqlite3 ~/system/databases/mission-control.db "PRAGMA wal_checkpoint(TRUNCATE);" ls -lh ~/system/databases/*.db-wal

WAL files should be <10MB after checkpoint. If still large: investigate who holds long transactions.

SECTION 6: PRIORITIZED ROADMAP (Next 4 Weeks)

WEEK 1: P0 — Survival (Legal + DR + Observability + Data Integrity)

| Item | File | Effort | Success Metric | |------|------|--------|----------------| | Fix legal-docs-azure-sync.sh (--overwrite + ISBD path) | `/Users/makinja/system/daemons/legal-docs-azure-sync.sh` | 30 min | Next cron run completes with 84/84 blobs uploaded, exit 0 | | Verify cost units against Anthropic billing dashboard | Manual (console.anthropic.com vs cost-tracker.js) | 30 min | Reconciled dollar amount documented in MC | | Enable Azure VM-level backup policy | Azure Portal (manual) | 30 min | Azure Backup vault shows VM in protected state | | WAL checkpoint cron (4h interval) | `~/system/tools/wal-checkpoint.js` + plist (create new) | 2h | hivemind.db-wal < 10MB after first run | | Reopen MC #10173: fix 4 remaining failures | `/Users/makinja/system/daemons/infra-drift-detector.sh` (git config renameLimit), investigate apply-knowledge + auto-verify-regression | 4h | fleet-status.json shows 0 errors for all 4 previously failing daemons | | Raise dispatch cap: 3 → 8 | `/Users/makinja/.claude/hooks/lock-john-dispatch-cap.sh` line 77 | 10 min | Session can dispatch 8 sub-agents without [CEO_APPROVED] | | Run first restore drill on mission-control.db | Manual (litestream restore command per playbook) | 30 min | Row count in /tmp restore matches live DB |

Week 1 total effort: ~8 hours

WEEK 2: P1 — Observability + Routing

| Item | File | Effort | Success Metric | |------|------|--------|----------------| | Morning fleet brief to Slack | `~/system/tools/morning-fleet-brief.js` + plist (new) | 2h | Daily 09:00 Slack message in #exec with 8-line brief | | Cost spike real-time alert | `~/system/daemons/cost-spike-alert.sh` + plist (new) | 2h | Alert fires when hourly burn > threshold; fires within 5min of trigger | | Fix ops-watchdog false alarms (--location flag) | `~/system/daemons/ops-watchdog.sh` | 30 min | docs.basicconsulting.no and vault.basicconsulting.no show green; lumiscare.no still red (correct) | | Repair specialist-mapping.json (17 orphans + 7 broken) | `~/system/agents/specialist-mapping.json` | 3h | `discover.js routing ""` returns results for all 40 agents; 0 broken entries | | Unblock pi-orchestrator: resolve tasks #10038, #10039, #10043 | Manual review of 3 tasks in ready_for_review | 2h | pi-orch log shows task pickup after unblock | | Clean cross-company-routes.json (remove HelixSupport, Proxima) | `~/system/config/cross-company-routes.json` | 30 min | No stale company references | | Bootstrap lightrag-auto-heal | Manual: launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/com.alai.lightrag-auto-heal.plist | 5 min | launchctl list shows com.alai.lightrag-auto-heal with PID |

Week 2 total effort: ~10 hours

WEEK 3: P2 — Efficiency + Agent Hygiene

| Item | File | Effort | Success Metric | |------|------|--------|----------------| | Move validation-state-injector to /status | `~/.claude/hooks/validation-state-injector.sh` → rename to skill; remove from UserPromptSubmit in settings.json | 2h | UserPromptSubmit hooks: 6 → 4; prompt cache hit rate increases | | Retune mehanik.md: sonnet → haiku | `~/.claude/agents/mehanik.md` line 3 | 5 min | Pre-dispatch gate uses Haiku model; gate logic unchanged | | Delete 8 umbrella persona files | `~/.claude/agents/` agentforge.md, codecraft.md, finverge.md, flowforge.md, proveo.md, securion.md, skybound.md, vizu.md | 2h | discover.js routing forces resolution to named specialists; test 10 routing queries | | Consolidate 3 reviewers → 1 | gemini-reviewer.md, redzo-reviewer.md, sentinel-validator.md → keep best, delete 2 | 1h | 1 review agent in ~/.claude/agents/; mapping updated | | Add openai.js adapter + OPENAI_API_KEY to Bitwarden | `~/system/tools/adapters/openai.js` (new) + register in adapters/index.js | 4h | comms-responder fallback chain includes OpenAI; confirmed via test call | | Wire lightrag-bulk-upload.js to gpt-4o-mini Batch | `~/system/tools/lightrag-bulk-upload.js` modification | 4h | Next bulk ingest run uses OpenAI Batch API; cost-tracker shows OpenAI spend | | Add FK enforcement to all DB connections | `pi-orchestrator.js`, `mc.js`, all tool scripts: add `db.pragma('foreign_keys = ON')` | 4h | PRAGMA foreign_key_check returns 0 violations on clean run | | Investigate builder.md + 4 persona ghosts with CEO | `~/.claude/agents/builder.md`, `alem-clone.md`, `dzevad-jahic.md`, `sylfest-lomheim.md` | 1h | Decision: delete or hard-scope each; documented in MC |

Week 3 total effort: ~18 hours

WEEK 4: P3 — Resilience + Architecture Foundation

| Item | File | Effort | Success Metric | |------|------|--------|----------------| | Atomic task claim (BEGIN IMMEDIATE) | `/Users/makinja/system/kernel/pi-orchestrator.js` getNextTask() function | 4h | 3-statement claim wrapped in db.transaction(); verified under load | | Add missing DB indexes | `pi-orchestrator.js` startup: idx_task_scheduling_next_eligible, idx_tasks_delegated_to | 2h | EXPLAIN QUERY PLAN on getNextTask() shows index use | | Monthly restore drill (recurring) | Manual (per Section 5 playbook) | 30 min | Row count verified; result documented in MC | | consensus-router.js POC for /prompt-forge | `~/system/tools/consensus-router.js` (new) | 8h | Claude-Opus + GPT-5 agree on a test architecture question; divergence creates human-queue task | | Fix Slack channel: dr-sync.sh → #exec | `~/system/tools/dr-sync.sh` | 5 min | No ERROR: Channel #ops not found in dr-sync log | | Add /tmp orphan cleanup to azure-db-backup.sh | `~/system/daemons/azure-db-backup.sh`: add trap cleanup EXIT | 1h | No /tmp/alai-azure-backup-* older than 24h | | Begin pi-orchestrator.js decomposition planning | Create BUILD-BLUEPRINT for 4-service split (task-classifier, model-router, agent-executor, task-scheduler) | 4h | Blueprint document; no code changes yet.

Code split is a 2-4 week effort AFTER blueprint approved | | Connect flywheel.db distillation → OpenAI Stored Completions | `~/system/tools/distillation-exporter.js` (new): query used_for_training=0, send to OpenAI | 8h | First batch of traces submitted; gpt-4o-mini fine-tune job ID recorded |

Week 4 total effort: ~28 hours

SECTION 7: ARCHITECT'S VERDICT

Is ALAI salvageable? Yes, unambiguously. The foundational investment — Litestream replication with proper tier classification, the events.db transactional outbox schema, the circuit breaker pattern in task_scheduling, the ZAKON enforcement hook architecture, the Ollama local fleet with fine-tuned domain models — represents genuinely sophisticated engineering for a zero-revenue company. Most funded startups at this stage do not have backup infrastructure this well-designed.

What John must change about himself (process):

The core failure pattern is documented in `feedback_john_recursive_drift.md` and proven again in MC #10173: **John closes tasks before verifying outcomes.** Not once, not twice — this is a structural habit. An agent reports work done, John marks done, CEO finds it was not done. The architectural response (Hard Constraint #3, Proveo validation layer, evidence-gate hook) exists and is correct. But the constraint fires on mc.js done — which means John is still the gate, and John keeps bypassing it. The solution is not more process. It is personal discipline: never mark a task done without looking at a concrete output artifact. A task is done when you can point to the thing, not when the agent says it is done.

The second habit: John dispatches to generic agents (builder.md) instead of named specialists, and to umbrella personas (codecraft.md) instead of the actual expert. This is route-by-habit, not route-by-classification. The 17 orphaned agents and the routing system's 95% bypass rate are symptoms of the same thing.

What must change about the system (technical):

Three high-leverage changes: (1) WAL checkpoint cron — 2 hours of work prevents multi-minute data loss on restart. (2) Atomic task claim — 1 day of work eliminates the race condition before pi-orchestrator resumes. (3) Validation-state-injector out of UserPromptSubmit — 2 hours of work stops burning money on cache invalidation every prompt. These three changes alone are net-negative cost (they save more than they cost).

The pi-orchestrator.js monolith is the largest technical debt. 5,251 lines, 14 subsystems, single point of failure. It is not broken today — it is idle. But when it resumes at scale, the non-atomic claim and the 14-writer stampede will create problems that are hard to diagnose in production. Begin the decomposition design in week 4, execute the split over the following month.

On vendor lock-in: Do not migrate away from Claude Code as orchestrator. The switching cost is

10x the risk premium at current scale. DO add OpenAI for RAG ingest (gpt-4o-mini Batch) — it is the single highest-ROI cost reduction available without touching orchestration. DO verify the $129K/day number against actual billing before making any vendor decisions. If those numbers are real dollars, ALAI has a 30-day runway problem, not an architecture problem. **TL;DR for CEO:** The architecture is solid and salvageable — fix three things this week (legal docs sync, Azure VM backup, WAL checkpoint), raise the dispatch cap from 3 to 8, verify the cost numbers against real billing, and establish the habit of never marking a task done without a concrete artifact to point at.

Report authored by: Petter Graff, Chief Architect, T6 Master Synthesis Input: T1 (Resolver), T2 (Kleppmann), T3 (Anthropic), T4 (OpenAI), T5 (Kelsey Hightower) All findings tool-verified by source teams. Zero memory assertions. Date: 2026-04-30

T1 — Resolver routing audit

Resolver - Cross-Company Routing Audit (MC #10357 T1)

Date: 2026-04-30 22:00 **Trigger:** Manual (CEO directive) **Auditor:** Resolver meta-company

---

1. CURRENT ROUTING TOPOLOGY

Agent-to-Company Mapping (specialist-mapping.json)

Total agents mapped: 28 agents across 12 companies

| Company | Agent Count | Agents | |---------|-------------|--------| | CodeCraft | 7 | martin-kleppmann, bruce-momjian, petter-graff, sentinel-developer, hadi-hariri, lee-robinson, sindre-sorhus | | Proveo | 7 | angie-jones, sentinel-tester, sentinel-validator, james-bach, lisa-crispin, dorota-huizinga, maria-santos | | AgentForge | 4 | chip-huyen, georgi-gerganov, claude-code-guide, anthropic-chief-architect, openai-chief-architect | | Securion | 2 | parisa-tabriz, sentinel-architect | | Vizu | 2 | brad-frost, lea-verou | | Skybound | 2 | paul-hudson, sentinel-ba | | FlowForge | 1 | kelsey-hightower | | Finverge | 1 | markos-zachariadis | | Lexicon | 1 | lexicon.md | | SkillForge | 1 | skillforge.md | | Resolver | 1 | resolver.md |

Companies with ZERO Agent Assignment

- **Datavera** - data/analytics/ml domain - NO AGENTS
- **HelixSupport** - merged into FlowForge (2026-03-31), references still in cross-company-routes.json
- **Proxima** - merged into Lexicon (2026-03-31)

Dispatch Surface Distribution

| Surface | Volume Today | Notes | |---------|--------------|-------| | Task Tool (John direct) | ~95% | All interactive work bypasses MC+pi-orch | | pi-orchestrator daemon | ~0% | Idle since 2026-04-28 (last completions: #10042, #10044) | | company-worker LaunchAgents | 2 active |

com.john.company-proveo, com.john.company-securion | | resolver-daemon (6h cron) | 4 runs today | Detected issues, created 0 tasks (all SKIP - duplicates exist) | | cross-company-bus | 4 runs today | 85 matches, 1 task created |

Evidence: pi-orchestrator.log shows continuous "No eligible tasks" since 2026-04-28 22:24:29Z. The 3 review-blocking tasks (#10038, #10039, #10043) are stuck in `ready_for_review` status with owner `pi-orchestrator`.

---

2. GAPS - WHERE ROUTING BREAKS

2.1 Orphan Agents (17 agents in ~/.claude/agents/ but NOT in specialist-mapping.json)

```
agentforge.md          codecraft.md       finverge.md        proveo.md
alem-clone.md          devils-advocate.md flowforge.md       redzo-reviewer.md
builder.md             dzevad-jahic.md    gemini-reviewer.md securion.md
mehanik.md             skybound.md        sylfest-lomheim.md validator.md
vizu.md
```

Impact: These agents can be invoked via Task tool but discover.js routing queries won't return them. John cannot programmatically route to them.

2.2 Companies Without Agent Coverage

| Company | Domain | Gap | |---------|--------|-----| | Datavera | data/analytics/ml | CRITICAL: 0 agents. ML work goes nowhere. | | Finverge | finance/payment | 1 agent only (markos-zachariadis) - thin coverage | | FlowForge | devops/infra | 1 agent only (kelsey-hightower) - thin coverage | | Lexicon | legal/docs | 1 agent only - thin coverage |

2.3 Dual-Store Problem

Agent definitions exist in TWO locations:

- `~/.claude/agents/` (40 files) - Claude Code reads from here
- `~/system/agents/definitions/` (57 files) - System scripts read from here

specialist-mapping.json only indexes 28 agents. Sync script exists (`~/bin/agent-definitions-sync.sh`) but mapping registry is stale.

2.4 Rules Without Enforcement Hooks

| Rule | Status | |------|--------| | cross-company-routes.json | 10 rules defined. Enforcement via cross-company-bus.js (6h cron). No real-time hook. | | ZAKON #7 (complex task >2h -> route through company) | NO HOOK. Policy only. John bypasses freely. | | tier-routing.json | Engine/model tiers defined. No enforcement - pi-orch is idle. |

---

3. PATTERN DETECTION (Last 7 Days)

HiveMind Intel by Agent (top 10)

| Agent | Entries | % | |-------|---------|---| | john | 3311 | 89.7% | | email-agent | 96 | 2.6% | | pi-orchestrator | 55 | 1.5% | | alem | 46 | 1.2% | | codecraft | 45 | 1.2% | | proveo | 34 | 0.9% | | agent-reporter | 30 | 0.8% | | daemon-fleet-watchdog | 22 | 0.6% | | validator | 21 | 0.6% | | system | 17 | 0.5% |

Pattern 1: John dominates HiveMind (90%). Companies barely log activity.

Mission Control Task Distribution

| Status | Count | |--------|-------| | done | 6,605 | | paused | 2,202 | | open | 891 | | blocked | 492 | | ready_for_review | 29 | | in_progress | 1 |

Pattern 2: 2,694 tasks stuck (paused+blocked) = 26% of active inventory.

Owner Distribution (open/blocked/paused)

| Owner | Count | Status | |-------|-------|--------| | pi-orchestrator | 1,619 | 1,324 paused + 295 blocked | | proveo | 390 | open | | autowork | 326 | paused | | edita | 186 | open | | john | 253 | 71 open + 118 blocked + 64 paused | | codecraft | 66 | open |

Pattern 3: pi-orchestrator owns 1,619 tasks but is IDLE. No pickup mechanism.

Top 3 Recurring Failures (from resolver.log)

1. **task_failures** - 16-20 tasks paused/blocked per 6h window (systemic) 2. **daemon-fleet-watchdog alerts** - 7+ daemons in failed state (infrastructure) 3. **orchestrator_error patterns** - routing/quality-based selection failures

---

4. WHERE DOES WHAT GO? (Canonical Routing Table)

Task Type -> Company -> Agent -> Surface

| Task Type | Company | Primary Agent | Surface | |----------|---------|---------------|---------| | **DEV/BUILD** | | | | | Backend API | CodeCraft | hadi-hariri (Kotlin/Ktor), lee-robinson (Next.js) | Task tool | | Database | CodeCraft | bruce-momjian (PostgreSQL) | Task tool | | Distributed systems | CodeCraft | martin-kleppmann | Task tool | | CLI tooling | CodeCraft | sindre-sorhus | Task tool | | **DESIGN** | | | | | Frontend/UI | Vizu | brad-frost (design systems), lea-verou (CSS) | Task tool | | **AUDIT/QA** | | | | | Test automation | Proveo | angie-jones, lisa-crispin | company-worker (12h) | | Exploratory testing | Proveo | james-bach, maria-santos | company-worker | | Performance testing | Proveo | dorota-huizinga | company-worker | | **SECURITY** | | | | | Security audit | Securion | parisa-tabriz | company-worker (12h) | | Architecture audit | Securion | sentinel-architect | company-worker | | **DEPLOY/OPS** | | | | | Kubernetes/CI/CD | FlowForge | kelsey-hightower | Task tool | | Incident/support | FlowForge | (no agent - gap) | Manual | | **ML/DATA** | | | | | ML pipeline | AgentForge | chip-huyen | Task tool | | Local LLM | AgentForge | georgi-gerganov | Task tool | | Analytics | Datavera | (NO AGENT - GAP) | BROKEN | | **LEGAL/DOCS** | | | | | Contracts/compliance | Lexicon | lexicon.md | Task tool | | Runbooks/knowledge | SkillForge | skillforge.md | Task tool | | **FINANCE** | | | | | Payments/fintech | Finverge | markos-zachariadis | Task tool | | **PRODUCT** | | | | | iOS/mobile | Skybound | paul-hudson | Task tool | | Business analysis | Skybound | sentinel-ba | Task tool | | **META** | | | | | Cross-company routing | Resolver | resolver.md | cron (6h) |

Query Mechanism

```
# Current: discover.js routing
node ~/system/tools/discover.js routing "<task description>"
```

Returns: Tools (6), Skills (0), Agents (2), MCP (0)

Problem: Only searches indexed sources. 17 orphan agents invisible.

---

5. DAY-TO-DAY ROUTING DEFICIT

What's Missing for Automatic Routing

Current State: John manually picks agents via Task tool. No pre-dispatch classifier. No enforcement of company boundaries.

Required Mechanism (3-Layer Fix)

Layer 1: Pre-Dispatch Classifier Hook

File: `~/.claude/hooks/pre-dispatch-classifier.sh` (does not exist) **Specification:**

```
# Input: task title + description (from Task tool invocation)
```

Output: { company: "X", agent: "Y", tier: "N" }

Action: Inject routing decision into task metadata before dispatch

Hooks to create:

- `~/.claude/hooks/pre-dispatch-classifier.sh` - classify task -> company
- `~/.claude/hooks/company-boundary-enforcer.sh` - block cross-company drift

Layer 2: Unified Agent Registry **Fix:** Merge specialist-mapping.json to include ALL 57 agents from ~/system/agents/definitions/ + 40 from ~/.claude/agents/. **Files to update:**

- `~/system/agents/specialist-mapping.json` - add 17 orphan agents
- `~/system/config/domain-to-company.json` - already correct (11 companies)

Layer 3: Slash Command for Routing Query **Proposal:** `/route ` -> Returns company + agent + reasoning **Implementation:**

- Add skill: `~/.claude/skills/route-task.md`
- Wire to: `node ~/system/tools/discover.js routing "$1"`

Files That Need Modification

| File | Action | |------|--------| | `~/system/agents/specialist-mapping.json` | Add 17 orphan agents | | `~/.claude/hooks/pre-dispatch-classifier.sh` | Create (does not exist) | | `~/system/config/cross-company-routes.json` | Remove HelixSupport references (merged) | | `~/.claude/skills/route-task.md` | Create /route slash command skill |

Verdict for John

Summary: The routing infrastructure exists but is BYPASSED. John dispatches 95%+ of work directly via Task tool, never touching pi-orchestrator, company-worker LaunchAgents, or cross-company-bus. The specialist-mapping.json is 39% incomplete (17/45 agents orphaned). Datavera has zero agent coverage - ML/analytics tasks have nowhere to go. Three review-blocking tasks (#10038, #10039, #10043) have stalled pi-orchestrator for 48+ hours. The 6h resolver-daemon cron detects patterns but creates duplicate tasks that get SKIP'd. To make routing automatic, John needs: (1) a pre-dispatch classifier hook that maps task->company before spawn, (2) specialist-mapping.json expanded to cover all 57 agents, and (3) a /route slash command for quick lookup. Until then, company routing is documentation, not enforcement.

Report generated: 2026-04-30T22:00:00Z **Next scheduled resolver-daemon run:** 2026-05-01T01:44:00Z

T2 — Kleppmann distributed systems audit

Distributed Systems & Data Layer Audit (MC #10357 T2 — Martin Kleppmann)

Auditor: Martin Kleppmann (DDIA) **Date:** 2026-04-30 **Scope:** ALAI orchestration data plane — kernel, tool scripts, SQLite stores **Method:** Read-only file inspection + sqlite3 queries against live databases

---

1. Event Flow & Consistency

Traced path: task intake ? execution ? completion

Canonical flow as implemented:

1. **Intake** — `mc.js add` inserts into `tasks` (mission-control.db). Status: `open`. No corresponding `task_scheduling` row yet. 2. **Poll** — `pi-orchestrator.js` (daemon, 30s default poll) executes a `getNextTask()` query: LEFT JOIN `tasks` with `task_scheduling`, filtering on `status IN ('open','in_progress')`, `dead_letter=0`, `next_eligible <= now`, `lease_until < now`. Priority ordering: H→M→L, then `updated_at ASC`. 3. **Claim** — `pi-orchestrator` sets `tasks.status = 'in_progress'`, upserts `task_scheduling.lease_until = datetime('now', '+N minutes')`, inserts `active_work(agent, task_id, session_id)`. These are three separate non-atomic DML statements (better-sqlite3 synchronous, but no BEGIN TRANSACTION wrapping the trio in code reviewed). 4. **Execute** — model selection (tier 1–5), agent spawn (Ollama or `claude` CLI subprocess). Execution happens outside the database entirely. 5. **Complete** — `mc.js done "outcome"` sets `tasks.status = 'done'`, clears `task_scheduling.lease_until`, removes `active_work` row. `task_history` append. 6. **Failure path** — after `MAX_TASK_RETRIES = 3`, `task_scheduling.dead_letter = 1`, `tasks.status = 'blocked'`, entry in `dead_letter_queue`.

Events durability: `events.db` has a proper schema with `idempotency_key UNIQUE`, `correlation_id`, `causation_id`, `publisher`, `retry_count`, `dead_letter` table — this is a full

transactional outbox pattern. **However, it is underused.** `pi-orchestrator.js` does not write to `events.db` for task transitions; it uses `postHiveMind()` (fire-and-forget to HiveMind daemon) instead. `chain-runner.js` uses `bus.emit()` (in-process event bus, not durable). The durable `events.db` store exists but the primary orchestration path bypasses it. **Source of truth fragmentation (confirmed by query):**

```
tasks:           10,220 rows  (primary state machine)
task_scheduling:  4,211 rows  (covers 41% of tasks — not all tasks have scheduling entries)
active_work:          2 rows  (in-memory live view — only current active agents)
task_history:    36,865 rows  (audit log — orphan FK violations detected, see §2)
```

State is split across four tables plus `active_work`. A task can be `status=in_progress` without a `task_scheduling` row (confirmed: 1 task in this state). The `active_work` table has a stale row (`skillforge|10342|ready_for_review`) — the task is no longer `in_progress` but the `active_work` entry was not cleared. **This is a consistency bug.**

Replayability: The `events.db` schema supports replay (`status`, `retry_count`, `idempotency_key`), but since orchestration transitions are not written there, an event replay would be incomplete. `task_history` provides an immutable audit trail but lacks payload to reconstruct state from scratch.

---

2. Data Layer Health

Schema summary (mission-control.db)

tasks table — 38 columns including `version INTEGER DEFAULT 0` (optimistic locking field exists but usage requires code audit to confirm enforcement), `quality_signals TEXT DEFAULT '{}'`, `dod_evidence`, `builder_model`, `validator_model`, `validation_timestamp`. Schema has grown organically via ALTER TABLE (evidenced by column ordering and `review_requeue_count` trailing addition).

```
-- Indexes:
CREATE INDEX idx_tasks_status_owner ON tasks(status, owner);
CREATE INDEX idx_tasks_status_priority ON tasks(status, priority, created_at);
```

Only two indexes on 10,220 rows. The primary poll query uses `status`, `priority`, `updated_at` — the existing index on `(status, priority, created_at)` covers this well. No index on `delegated_to` despite the primary `getNextTask()` path filtering on `delegated_to = 'pi-orchestrator'` first. This is a **missing index**. **task_scheduling table:**

```
CREATE TABLE task_scheduling (
  task_id INTEGER PRIMARY KEY REFERENCES "tasks"(id),
  cb_state TEXT DEFAULT 'closed' CHECK(cb_state IN ('closed', 'open', 'half_open')),
```

```
  attempt_count INTEGER DEFAULT 0,
  last_attempt TEXT, next_eligible TEXT, lease_until TEXT,
  last_outcome TEXT, last_error TEXT, dead_letter INTEGER DEFAULT 0
);
```

No index on `next_eligible` or `lease_until`, both used in hot query path. No index on `dead_letter`. **active_work table:**

```
CREATE TABLE active_work (
  agent TEXT PRIMARY KEY,
  task_id INTEGER DEFAULT NULL,
  FOREIGN KEY (task_id) REFERENCES "tasks"(id)
);
```

`agent TEXT PRIMARY KEY` — means only one task per agent name. This is correct for the current design but implies no parallelism per named agent. **Row counts (actuals):**

| Table | Count | |---|---| | tasks | 10,220 | | tasks (open) | 890 | | tasks (in_progress) | 2 | | tasks (done) | 6,605 | | tasks (paused) | 2,202 | | tasks (blocked) | 492 | | tasks (ready_for_review) | 29 | | task_scheduling | 4,211 | | active_work | 2 | | task_history | ~36,865 | | outbox | 3,262 total, 0 unprocessed | | dead_letter_queue | 22 | | task_scheduling.dead_letter=1 | 52 | | events.db total | 3,710 | | events.db dead | 19 |

Foreign key violations (PRAGMA foreign_key_check):

```
task_history → tasks: 19 orphan rows (task_ids: 962, 982, 983, 984, 1770-1772, 7436-7438,
7446, 14430-14431, 15883, 33500, 36110-36111, 36127, 36133)
task_scheduling → tasks: 1 orphan row (task_id: 9351)
task_metrics → tasks: 7 orphan rows (10053-10054, 10056, 10058-10059, 10061, 10309)
```

Total: **27 dangling foreign key references**. SQLite enforces FK constraints only when `PRAGMA foreign_keys = ON` is set per connection. `pi-orchestrator.js` uses `busy_timeout = 3000` but does not set `PRAGMA foreign_keys = ON` — FK integrity is not enforced at runtime. **WAL state:**

```
mission-control.db      25.6 MB  (main)
mission-control.db-wal  23.3 MB  (WAL = 91% of main — WAL checkpoint is NOT running)
hivemind.db             60.1 MB  (main)
hivemind.db-wal        147.0 MB  (WAL = 245% of main — severely bloated)
flywheel.db            224.2 MB  (main)
flywheel.db-wal        263.6 MB  (WAL = 118% of main)
events.db               14.6 MB  (main)
events.db-wal            4.5 MB  (WAL = 31% — acceptable)
```

`PRAGMA wal_checkpoint` returned `0|1|0` for mission-control.db, confirming the checkpoint ran 0 pages — the WAL has not been checkpointed recently. hivemind.db WAL at 147MB is a **P1 operational risk**: if the process crashes, SQLite must replay the entire WAL on next open, causing

multi-second startup delays. Large WALs also inflate Litestream replication bandwidth.

Last vacuum: No `VACUUM` or `ANALYZE` evidence found. The db has `freelist_count = 0` (no free pages), which is normal but may indicate vacuum has never run or ran recently on a growing dataset. **Growth rate:**

```
Week 2026-W09: 798 tasks/week
Week 2026-W13: 1354 tasks/week (peak)
Week 2026-W16: 1145 tasks/week
Week 2026-W17: 790 tasks/week
```

At ~900 tasks/week, mission-control.db will reach 50,000 tasks in ~4 months. At current page size (4096 bytes), the database will reach ~500MB without periodic archival of done/blocked tasks.
---

3. Concurrency & Ordering

Multiple writers

The following processes can write to mission-control.db concurrently:

- `pi-orchestrator.js` daemon (primary writer — poll loop + task state transitions)
- `mc.js` CLI (human-invoked — add, done, block, assign, start)
- `sprint-pipeline.js` (calls `mc.js start` via `execSync` — indirect writer)
- `pipeline-controller.js` (reads MC DB directly readonly + indirect via mc.js)
- `cross-company-bus.js` (creates tasks via `mc.js add`)
- `chain-runner.js` (updates tasks via MC script calls)
- Multiple Claude agent subprocesses calling `mc.js` (spawned by pi-orchestrator)

Concurrency mechanism: better-sqlite3 is **synchronous** — all reads/writes block the Node.js event loop. With WAL mode enabled (`PRAGMA journal_mode = WAL`), readers do not block writers and writers do not block readers. Multiple writers still serialise through SQLite's single-writer lock. `pi-orchestrator.js` sets `busy_timeout = 3000` ms — meaning it will wait up to 3s for the lock, then throw. This is acceptable for a single-machine deployment. **Lease mechanism analysis:**
The lease is implemented in `task_scheduling.lease_until`. The `getNextTask()` query filters `lease_until < datetime('now')` before claiming. Claim path:

```
// From getNextTask() — three separate statements, not in a single transaction:
1. UPDATE task_scheduling SET lease_until = datetime('now', '+N min')
2. UPDATE tasks SET status = 'in_progress'
3. INSERT OR REPLACE INTO active_work(agent, task_id, ...)
```

Critical bug: These three writes are not wrapped in a single SQLite transaction. If the process crashes between statements 1 and 2, `task_scheduling.lease_until` is set but `tasks.status`

remains `open`, and `active_work` has no entry. The task appears "leased" to a dead worker but also appears available to the next poller once the lease expires — which is the correct recovery. **However**, between crash and lease expiry, `getNextTask()` will skip this task unnecessarily, creating transient unavailability without notice. **Confirmed stuck leases (from query):** 3,254 of 3,255 `task_scheduling` rows with a `lease_until` value have an *expired* lease (`lease_until < datetime('now')`). This means nearly all leases are in the past — which is normal if tasks completed (the lease was not cleared to NULL on completion, just left expired). But it also means the lease mechanism provides **zero protection** against concurrent double-claiming, because any new poller will see `lease_until < now` as valid to claim again. This is by design (lease expiry = re-eligibility), but implies: **if two pi-orchestrator instances ran simultaneously, both could claim the same task** because the claim is not atomic (no SELECT...FOR UPDATE, SQLite has no such construct, and no BEGIN IMMEDIATE around the claim trio). **Reconcile mechanism:** `reconcileWorkerCounters()` (pi-orchestrator.js line ~3248) runs every 30s to detect drift between `activeTasks` Map and `activeWorkers` counters. This addresses the in-process ghost slot problem but not the cross-process double-claim problem. **On daemon crash mid-task:** The task remains `status=in_progress` with an expired lease. The next orchestrator cycle (after restart) will pick it up from `getNextTask()` because `lease_until < now`. `attempt_count` is incremented. After 3 retries, it goes to DLQ. This recovery path is correct but relies on the orchestrator restarting — if it does not restart (manual intervention needed), the task is orphaned in `in_progress` indefinitely. Confirmed: MC task #10330 is currently `in_progress` with `active_work` row showing `john` as agent — this appears to be a legitimate in-progress task, not a ghost. **Event log:** `events.db` has a `dead_letter` table with 19 entries — these are events that exhausted retries. The events schema includes `correlation_id` and `causation_id` for lineage tracing. However, as noted in §1, the primary orchestration path does not write to this store.

---

4. Backup & Recovery

What exists

Litestream (`/Users/makinja/system/config/litestream.yml`): Comprehensive — 50+ databases replicated to Azure Blob Storage (`alaibackups0ebb.blob.core.windows.net`, container `system-db-backups`). Tiering:

- **P0-critical** (mission-control.db, hivemind.db, costs.db, events.db, durable-runner.db, knowledge.db, email-inbox.db, alem-directives.db): `sync-interval: 1s`, `retention: 168h` (7 days)
- **P0-financial** (fiken.db, invoices.db, contracts.db, leads.db): `sync-interval: 1s`, `retention: 720h` (30 days)
- **P1** (agent-routing.db, bee-index.db, contacts.db, etc.): `sync-interval: 1s`, `retention: 72h`
- **P2** (prompt-cache.db, baikal-caldav.db, etc.): `sync-interval: 10s`, `retention: 24h`

flywheel.db (224MB main + 263MB WAL): `sync-interval: 30s` — reasonable given size, but WAL bloat means Litestream is continuously replicating a large dirty WAL. **azure-db-backup.sh** (`com.alai.azure-db-backup`, every 4h): Independent Postgres + Docker volume + Qdrant backup

to Azure Blob. This covers the cloud products (Bilko, Drop) but is reported as silently failing (daemon-fleet watchdog MC #10173). The error log shows: `rm: /tmp/az-backup-50926: Directory not empty` — non-fatal cleanup failure but indicative of script state pollution. **Point-in-time restore:** Litestream `.ltx` format supports point-in-time restore within the retention window. `litestream.yml.bak-20260425` and restore config at `litestream-restore.yml` confirm the restore procedure has been documented. **Manual snapshots:** `mission-control.db.bak-20260306-175717` (4.5MB) and `mission-control.db.bak-before-7082` (21.6MB) exist on disk — evidence of manual before/after snapshots. The 2026-03-06 snapshot is ~7 weeks old, covering a period when the database was ~1/5 current size. These are not automated. **RPO assessment:**

- mission-control.db: RPO = ~1s (Litestream sync-interval) + Azure write latency (~100ms). **Effective RPO: ~2s**. This is excellent for a SQLite-based system.
- Litestream replication of a 23MB WAL: the WAL is being replicated continuously, but a large un-checkpointed WAL means recovery requires replaying that WAL — **estimated recovery time: 5–15 seconds** on a healthy machine.

RTO assessment:

- If mission-control.db corrupts NOW: restore from Litestream latest snapshot + WAL segments. Litestream restore to local path, then restart pi-orchestrator. Assuming Azure access is available: **RTO ~2–5 minutes**.
- If ANVIL (the Mac Studio) dies: restore requires a new machine, reinstall Homebrew + Node.js + Litestream, then `litestream restore`. **RTO ~30–60 minutes** for full service restoration.

What is lost on corruption NOW:

- `active_work`: 2 current rows — 2 in-flight tasks would need manual status check/reset
- Any `task_history`, `outbox`, or `approval_tokens` rows written in the last ~2 seconds
- The 23.3MB WAL (uncommitted pages) — Litestream replicates WAL pages, so this is covered

Untested restore: No evidence of a restore drill in logs or memory files. `litestream-restore.yml` exists as documentation but there is a note in MEMORY.md that `azure-db-backup` was silently failing. If Litestream's Azure SP credentials (`AZURE_CLIENT_ID`) have rotated or the SP permissions have changed, the replicas may be stale without any alert. This is the **single most dangerous gap** in the backup posture. **ingest-queue.sqlite.corrupt** in `~/system/state/`: A 6.4MB corrupt SQLite file alongside a backup — evidence of a past corruption event. No RCA documented in accessible files.

---

5. Evolution Risk

The 5,251-line pi-orchestrator.js problem

`pi-orchestrator.js` is a 5,251-line monolith that owns: semantic classification, model selection, prompt construction (with 4 RAG sources), worker pool management, quality gating (lint, test,

semantic), proof-of-work verification, routing token writing, DLQ writing, HiveMind posting, cost tracking, and the main daemon loop. It has at least 14 conceptually independent subsystems co-located in one file.

Concrete risks:

1. **Single process, single point of failure.** If pi-orchestrator crashes mid-task, all in-flight work is abandoned. The lease mechanism provides recovery but with a gap of `lease_until` duration (typically 5–30 minutes) before re-eligibility.

2. **In-process worker pool counters** (`activeWorkers` object) are process-local. These are not persisted to the database. On restart, `activeWorkers` resets to zero even if tasks are still `status=in_progress` in the DB — the orchestrator will over-claim slots until `reconcileWorkerCounters()` detects the drift (up to 30s).

3. **No command authority boundary.** `chain-runner.js`, `sprint-pipeline.js`, `pipeline-controller.js`, `cross-company-bus.js`, and `mc.js` all write directly to mission-control.db. There is no single command handler. A schema migration or constraint change in `tasks` requires auditing all 14+ writers. The 38-column `tasks` table with columns added via `ALTER TABLE` is the result.

4. **`task_scheduling` not populated for all tasks.** 4,211 scheduling rows for 10,220 tasks means 58% of tasks have no scheduling metadata. `getNextTask()` handles this with `LEFT JOIN ... COALESCE(ts.cb_state, 'closed')` — safe but means circuit breaker and lease state is silently absent for most tasks.

5. **`sprint-pipeline.js` DAG state in JSON column.** `sprints.dag_json TEXT` stores the full DAG serialisation inside a SQLite text column. This means querying or filtering on DAG phase status requires deserialisation in application code, cannot use SQL indexes, and is vulnerable to partial writes leaving malformed JSON.

Modernisation recommendations

Recommendation 1 — Immediate: WAL checkpointing cron (hours) Run `PRAGMA wal_checkpoint(TRUNCATE)` on hivemind.db, flywheel.db, and mission-control.db every 4 hours via LaunchAgent. The 147MB hivemind WAL is a live production risk. Cost: 2 hours, zero architectural change. **Recommendation 2 — Short term: Atomic claim transaction (1–2 days)** Wrap the task claim (lease_until update + status=in_progress + active_work insert) in a single `BEGIN IMMEDIATE ... COMMIT`. This is trivially achievable with better-sqlite3's `db.transaction()` API and eliminates the three-statement race window. Also add `CREATE INDEX IF NOT EXISTS idx_task_scheduling_next_eligible ON task_scheduling(next_eligible, dead_letter)` and `CREATE INDEX ON tasks(delegated_to)` to fix the missing indexes in the hot query path.
Recommendation 3 — Short term: FK enforcement (1 day) Add `db.pragma('foreign_keys = ON')` to every better-sqlite3 connection open in pi-orchestrator.js, mc.js, and all tool scripts. Then fix the 27 existing orphan rows (most are safe to delete — they reference tasks that were hard-deleted, violating the soft-delete principle). This prevents a class of data inconsistency bugs at zero performance cost. **Recommendation 4 — Medium term: Extract command boundary (1–2 weeks)** Move all task state mutations (open→in_progress, in_progress→done, etc.) behind `mc.js` as the **single writer**. All tools that currently open mission-control.db directly as a writer (sprint-

pipeline.js, pipeline-controller.js, cross-company-bus.js) should call `mc.js` via IPC or HTTP instead. This eliminates the 14-writer problem and makes schema evolution safe. The `outbox` table in mission-control.db is already structurally correct for this — route all mutations through it with `processed=0`, and have mc.js process the outbox as the canonical writer. **Recommendation 5 — Medium term: Decompose pi-orchestrator.js (2–4 weeks)** Split into four services:

- `task-classifier.js` — semantic classification + caching (stateless, ~200 LoC)
- `model-router.js` — model selection + fleet health + routing tokens (~300 LoC)
- `agent-executor.js` — subprocess management + quality gate + proof-of-work (~500 LoC)
- `task-scheduler.js` — the daemon loop + lease management + DLQ + concurrency counters (~300 LoC)

These communicate through the existing `events.db` store (which is already correctly designed for this). This eliminates the single-process SPOF and enables independent restart of each component. **Recommendation 6 — Long term: Event-sourced task store (4–8 weeks)** Replace the multi-table task state machine with an event-sourced design: one `task_events` table (taskId, eventType, payload, timestamp, actor) as the source of truth; `tasks` becomes a materialised view rebuilt from events. This enables full replay, audit trail, and time-travel queries. The existing `task_history` table is a partial implementation of this pattern. The `events.db` schema (with `idempotency_key`, `correlation_id`, `causation_id`) is the correct foundation — extend it to cover task lifecycle events and wire pi-orchestrator to write there instead of posting to HiveMind. **CDC option:** An alternative to event sourcing is Litestream-level CDC: Litestream already replicates WAL pages; a consumer reading those pages could reconstruct a change log. This requires custom Litestream consumer code but avoids changing the write path.

Top 3 Architectural Risks

RISK 1 — Unverified Litestream backup integrity (CRITICAL) Litestream is configured for 1s RPO on mission-control.db, but the azure-db-backup daemon is known-silently-failing (MC #10173). If the Litestream LaunchAgent has also silently stopped (daemon watchdog found 11 failures), the Azure replicas may be hours or days stale. A corruption event today could result in data loss far exceeding the stated 2s RPO. **Action needed: verify last successful Litestream replication timestamp immediately.** Run `litestream snapshots -config ~/system/config/litestream.yml` and check dates. Then run a restore drill to a temp path. **RISK 2 — Non-atomic task claim + 14 concurrent writers (HIGH)** The three-statement claim (lease/status/active_work) is non-atomic. With 14 identified writers accessing mission-control.db without a command authority boundary, there is no structural guarantee against double-claims, lost updates, or constraint bypass. The `version` column in `tasks` exists for optimistic locking but its enforcement is not verified — if writers do not check `WHERE version = ?` in their UPDATE statements, it provides no protection. **RISK 3 — 147MB hivemind.db WAL + 264MB flywheel.db WAL (HIGH)** Both WALs exceed their main database file sizes. WAL mode accumulates pages when readers hold long-running transactions or when `wal_autocheckpoint` threshold is never reached due to write volume. These databases cannot benefit from Litestream's efficient snapshot mechanism until checkpointed. On an unexpected restart, SQLite must replay the entire WAL — for hivemind.db this means replaying 147MB before any operation can proceed, causing multi-second to minute-level startup delays.

Under extreme conditions (full disk), the WAL files could prevent writes to all SQLite databases sharing the same volume.

Top 3 Strengths

STRENGTH 1 — Litestream replication architecture The investment in Litestream with proper tier classification (P0/P1/P2), 7-day retention for critical stores, and Azure Blob backend is genuinely sophisticated for a single-machine SQLite deployment. This puts ALAI's backup posture ahead of many funded startups. The `litestream-restore.yml` and SP-based auth show operational maturity. **STRENGTH 2 — Circuit breaker + dead-letter queue pattern** `task_scheduling.cb_state IN ('closed', 'open', 'half_open')` with `attempt_count`, `next_eligible` backoff, and `dead_letter` flag implements a textbook circuit breaker. The DLQ path (auto-block + Slack alert after 3 retries) prevents runaway retry storms. 52 entries in DLQ with max 5 attempts shows the mechanism is working in production. **STRENGTH 3 — events.db transactional outbox schema** The `events.db` schema — with `idempotency_key UNIQUE`, `correlation_id`, `causation_id`, `status` state machine, `dead_letter` table, and proper indexes — is correctly designed for at-least-once event delivery with deduplication. The schema supports everything needed for an event-sourced architecture. The gap is that the primary orchestration path does not yet write to it; when it does, this becomes a genuine durable, replayable event log.

Appendix: Key File Paths

- `/Users/makinja/system/kernel/pi-orchestrator.js` — 5,251 LoC main daemon
- `/Users/makinja/system/kernel/cross-company-bus.js` — lateral event routing
- `/Users/makinja/system/tools/chain-runner.js` — DAG chain execution
- `/Users/makinja/system/tools/sprint-pipeline.js` — sprint coordination
- `/Users/makinja/system/tools/pipeline-controller.js` — 13-phase lifecycle
- `/Users/makinja/system/databases/mission-control.db` — 25.6MB, 10,220 tasks
- `/Users/makinja/system/databases/mission-control.db-wal` — 23.3MB (needs checkpoint)
- `/Users/makinja/system/databases/hivemind.db-wal` — 147MB (CRITICAL — needs checkpoint)
- `/Users/makinja/system/databases/flywheel.db-wal` — 264MB (needs checkpoint)
- `/Users/makinja/system/databases/events.db` — 3,710 events, 19 dead
- `/Users/makinja/system/config/litestream.yml` — 741 lines, 50+ DBs replicated
- `/Users/makinja/system/state/ingest-queue.sqlite.corrupt` — evidence of past corruption

T3 — Anthropic agent layer audit

Anthropic SDK + Agent Loop Audit (MC #10357 T3)

Lens: Anthropic Chief AI Architect (Krieger / Kaplan / Cherny / MCP-team composite). **Layer scope:** Claude Code Task-tool sub-agent dispatch surface, hooks, prompt caching, tier routing, ZAKON #28 max-depth, dispatch cap. **Date:** 2026-04-30 22:01 CEST. **Method:** Read-only filesystem inspection. Tool-first per ZAKON NULA.

> **Important scope correction up front:** the CEO brief listed > `~/system/tools/agent-orchestrator.js` as canonical source. That file does not > exist at that path. The only `agent-orchestrator.js` on disk lives at > `~/system/archive/orphan-orchestrators-2026-03-21/agent-orchestrator.js` — > archived 5+ weeks ago. The actual orchestration surface today is: > 1. Claude Code Task tool (Anthropic-canonical, the dominant path), > 2. `~/system/tools/chain-runner.js` (YAML chains, lightly used), > 3. `~/system/kernel/pi-orchestrator.js` (Ollama DAG, separate process). > This doc audits #1 (the Anthropic layer); #2 and #3 fall under T2/T4.

---

1. Agent Loop Coherence

What's there

Today's dispatch path:

```
CEO prompt
  -> John (orchestrator, Sonnet by default)
       -> SlashCommand /mehanik (PreToolUse: pre-dispatch-gate.sh validates 13-field marker)
            -> writes /tmp/mehanik-cleared-<MC_ID>
       -> Task tool dispatch (subagent_type=<specialist>)
            -> PreToolUse hook stack: lock-john-dispatch-cap, claude-hooks pre,
               pre-action-da-gate, pre-dispatch-gate, john-max-depth-gate
            -> Claude Code spawns isolated sub-agent process from
               ~/.claude/agents/<name>.md frontmatter (model + tools)
            -> sub-agent runs its own tool loop; only final assistant message
```

```
            returns to John
        -> /task-postflight (Proveo verification, postflight marker)
        -> mc.js done (state injector + alai-hooks evidence-gate enforce marker)
        -> CEO report
```

Anthropic-canonical or not? — verdict per piece

| Component | Verdict | Note | |---|---|---| | Task tool sub-agent dispatch | **Canonical** | Exactly the SDK pattern: isolated context, frontmatter declares model + tools. Result, not reasoning, returns to parent. | | `~/.claude/agents/*.md` with `name:`, `model:`, `tools:`, `description:` frontmatter | **Canonical** | Matches Claude Code documented sub-agent schema. 40 of 40 files have frontmatter. | | PreToolUse hook on `Task\|Agent` matcher | **Canonical primitive, custom payload** | Hooks are an Anthropic primitive; the policies enforced on top (Mehanik clearance, dispatch cap, max-depth) are ALAI-specific. This is the supported pattern. | | Mehanik gate as pre-dispatch deterministic check | **Custom but Anthropic-blessed shape** | Mehanik itself is a sub-agent (`~/.claude/agents/mehanik.md`, model: sonnet, tools: Read+Bash) bootstrap-exempt from cap and depth hooks. Writes a marker file consumed by `pre-dispatch-gate.sh`. Anthropic's own pattern: "use a small judge sub-agent + a deterministic hook to enforce." This is *exactly* what Krieger has demoed publicly — Mehanik is well-shaped. | | 13-field marker schema in `/tmp/mehanik-cleared-*` | **Custom, sensible** | Deterministic state hand-off. Field validation in pre-dispatch-gate.sh lines 73-79 is the right shape. | | ZAKON #28 trip-wire 1 (depth ≥ 3 blocks) | **Custom, defensible** | Anthropic does not ship a depth guard. But for a 1-CEO orchestrator with no parallel team, it matches the Boris-Cherny stated principle: "agents are loops; chains-of-chains are workflows pretending to be agents." | | ZAKON #28 trip-wire 2 (emergent-spawn budget approved+3) | **Custom, sensible** | Direct response to MC #10043 drift. Budget-against-Mehanik clearance is a clean primitive. | | Dispatch cap 3/session | **Anthropic would say: too low** | See Section 4. | | Boot enforcer / validation-state-injector on `UserPromptSubmit` | **Canonical primitive, cost-amplifying** | UserPromptSubmit injection is supported. But `validation-state-injector.sh` runs `mc.js list --owner john --priority H --json` on every cache miss (30s TTL) and prepends warnings to every prompt. See Section 3. |

What's diverged from Anthropic-canonical

1. **Sub-agent definitions split across two stores.** `~/.claude/agents/*.md` (40 files) is what Claude Code actually reads. `~/system/agents/definitions/` (129 files) is a parallel store. Memo `feedback_agent_definitions_dual_store.md` documents this with a sync script. **Anthropic's recommendation: one source of truth.** The dual store is a tax. 2. **Specialist-mapping JSON drift.** 40 FS agents vs 30 mapped. 17 FS-only files (the personas: agentforge, codecraft, finverge, vizu, skybound, mehanik, builder, validator, devils-advocate, alem-clone, dzevad-jahic, sylfest-lomheim, gemini-reviewer, redzo-reviewer, proveo, securion, flowforge) have no mapping entry. 7 mapped names point to FS files that don't exist (dorota-huizinga, hadi-hariri, james-bach, lee-robinson, lisa-crispin, maria-santos, resolver). **The mapping JSON has structural rot.** 3. **No prompt-cache breakpoints declared anywhere.** Sub-agent .md files do not place explicit cache markers. With CLAUDE.md ~44 lines plus orchestration-surface.md (89 lines) plus rule-

pointers, the system prompt is short enough that Claude Code's automatic 5-min ephemeral cache should hit on stable content — but with 6 UserPromptSubmit hooks injecting dynamic content (boot-enforcer, alem-instruction-checker, validation-state-injector, alai-hooks auto-verify, feasibility-check-advisory, incident-response-mode), **cache invalidation is happening every prompt**. Section 3 quantifies. 4. **Mehanik is sonnet but does pure deterministic verification.** `~/.claude/agents/mehanik.md` line 3: `model: sonnet`. Mehanik's job is to grep blueprint files, count constraints, and write a marker. **This is Haiku work**, not Sonnet. See Section 5.

Verdict — Loop coherence: *Anthropic-canonical core, with custom enforcement that fits the SDK seams correctly. The shape is right. The drift is in inventory hygiene and tier choices.*

---

2. Sub-Agent Inventory Health

Counts (tool-verified)

```
ls ~/.claude/agents/*.md | wc -l                    → 40
jq '.mappings | length' specialist-mapping.json     → 30
ls ~/system/agents/definitions/ | wc -l             → 129
```

Tier breakdown across 40 FS agents

Tier	Count	Files
opus	4	anthropic-chief-architect, dzevad-jahic, openai-chief-architect, sylfest-lomheim
sonnet (incl. claude-sonnet-4-5)	30	builder, mehanik, all PI personas (codecraft/proveo/vizu/...), all expert personas except 4 above
haiku	6	claude-code-guide, devils-advocate, gemini-reviewer, redzo-reviewer, sentinel-validator, validator
no model field	0	(clean)

Anthropic-recommended tier split for orchestrator-heavy single-user system: ~50% Haiku (reads/triage/grep), ~40% Sonnet (default work), ~10% Opus (planning + novel architecture). **Current: 15% / 75% / 10%.** Sonnet is over-loaded.

Anti-pattern findings

AP-A: Persona agents acting as builders without strict scope.

- `builder.md` (321 lines, sonnet) is a generic "build this" catch-all with full Read/Write/Bash/Edit. *Risk:* John reflexively dispatches "builder" instead of resolving via `discover.js routing` to a domain specialist (kelsey-hightower for K8s, bruce-momjian for Postgres, etc.). The builder file should either be deleted (force routing through specialists) or hard-scoped to "non-domain trivial scaffolding only".
- `proveo.md`, `securion.md`, `vizu.md`, `codecraft.md`, `flowforge.md`, `finverge.md`, `agentforge.md`, `skybound.md` are **company-level umbrella personas**, not agents.

> Each company already has named experts (kelsey for FlowForge, brad-frost+lea-verou for Vizu, etc.). The umbrella personas are a routing trap: ambiguous which specialist actually runs.

AP-B: Mapped-but-missing agents. 7 specialist-mapping entries (`hadi-hariri`, `lee-robinson`, `james-bach`, `lisa-crispin`, `dorota-huizinga`, `maria-santos`, `resolver`) point to .md files that **do not exist in `~/.claude/agents/`**. Any dispatch routed to these via `discover.js routing` will crash at sub-agent spawn time. **This is a live broken edge. AP-C: Validator duplication.** Three validators: `validator.md` (haiku), `sentinel-validator.md` (haiku), `proveo.md` (sonnet umbrella). Plus `gemini-reviewer.md` and `redzo-reviewer.md`. Anthropic recommendation: one validator role with domain-routed sub-flavors. Five competing files = unclear authority. **AP-D: Two `*-chief-architect` personas (anthropic + openai) both opus.** Useful for vendor-diverse architecture review, but each dispatch costs Opus rates. Make sure they are summoned with `[CEO_APPROVED]` only, not by reflex.

Inventory verdict

| | Count | |---|---:| | Healthy specialist agents (named expert, mapped, used) | ~17 | | Persona umbrellas (delete or hard-scope) | 8 | | Validator/reviewer overlap (consolidate) | 5 | | Bootstrap gates (mehanik, validator, devils-advocate) | 3 | | Mapped-but-missing (broken on dispatch) | 7 | | Total FS agents | 40 |

Pruning target: 40 → 22. Delete or merge 18 files. See Section 5.

---

3. Prompt Caching & Token Economy

What lands in the system prompt every John prompt

Inspected hook config (`~/.claude/settings.json`) — UserPromptSubmit fires 6 hooks:

```
UserPromptSubmit:
  -> incident-response-mode.sh          (state-dependent, mostly inert)
  -> boot-enforcer.sh                   (inserts 8-hour banner if no boot flag)
  -> user-message-logger.sh             (logging, no injection)
  -> alai-hooks auto-verify             (Kotlin CLI, ~14MB binary, injects verification)
  -> alem-instruction-checker.sh        (CEO directive parser, injects when triggered)
  -> feasibility-check-advisory.sh      (ZAKON FEASIBILITY, injects if conflicts)
  -> validation-state-injector.sh       (mc.js list output cached 30s)
```

The cache-amplification problem: every UserPromptSubmit hook that prints to stdout becomes

part of the conversation. Even if Claude Code's automatic 5-min ephemeral cache were active on the static prefix (CLAUDE.md + tool schemas + sub-agent registry), **a hook-injected line that varies per-call evicts cache from that breakpoint forward.**

`validation-state-injector.sh` cost analysis

Lines 50-110 — runs `node ~/system/tools/mc.js list --owner john --priority H --status open --json`, parses, prints WARNING lines per H-task missing postflight. Cached 30s in `~/system/state/validation-state-cache.txt`.

- 30s TTL is sensible.
- BUT: the output text changes whenever tasks change state, postflight markers appear, tasks close. **Each change = cache breakpoint invalidation downstream.**
- Estimated frequency: 5–20 invalidations per hour during active session.

Anthropic recommendation: UserPromptSubmit hooks that inject dynamic state should **append after** the stable system prompt and **not** be read as part of the cached prefix. The current architecture treats them as additional system context, defeating prefix cache.

`mc.js list` injection on every prompt — confirmed cost amplifier

Yes, `validation-state-injector.sh` runs `mc.js list --owner john --priority H --status open --json` (or hits cache). Even at 30s cache, on a 4-hour session = ~480 prompt submissions × full output included = constant cache invalidation, plus ~30 mc.js invocations (cache miss path).

Quantified: assume CLAUDE.md + tool schemas = ~3500 cached tokens; each invalidation forces re-tokenization. At Sonnet pricing $3/MTok input / $0.30/MTok cache-read, the gap = $2.70 per 1M tokens × N invalidations. At 480 prompts × 3500 tokens of *should-have-been-cached* prefix = 1.68M tokens. If 50% miss cache due to injector churn = ~$2.27 saved per 4-hour session if injector becomes append-only (or moves to a separate context window).

YouTube RAG ingestion — verified Ollama, NOT Anthropic

`~/system/kernel/pi-orchestrator.js`:

- Line 4796: `'[IDLE] YouTube batch started (PID ' + child.pid + '), model=qwen3:8b-q8_0 on FORGE'`
- 71 references to OLLAMA / 11434 / 10.0.0.2 (FORGE box) vs 2 references to "anthropic"
- The 2 anthropic references are imports/strings, not API calls

Verdict: pi-orch idle YouTube ingestion is on local Ollama (FORGE 10.0.0.2:11434, qwen3:8b-q8_0). ZERO Anthropic API spend on that path. Not a runaway cost. CEO

concern resolved.

Caching verdict

| Component | Cache health | Anthropic verdict | |---|---|---| | CLAUDE.md (44 lines) | Should hit auto-cache | Good — short, stable | | orchestration-surface.md (89 lines) | Loaded on demand | Good — not in every prompt | | sub-agent .md frontmatter | Per-dispatch, isolated | Good — separate context | | UserPromptSubmit injectors | **Cache-defeating** | **Fix** | | YouTube RAG | Local Ollama, no API cost | Good | | mc.js list as ambient state | Injected per prompt | **Move to /status command, not auto-injected** |

---

4. Max-Depth ZAKON #28 + Dispatch Cap

Trip-wire summary (tool-verified from hook source)

| Wire | File | Behavior | Bypass | |---|---|---|---| | Dispatch cap | `lock-john-dispatch-cap.sh` | Block 4th `Task\|Agent` per session | `[CEO_APPROVED]` token in prompt | | Trip-wire 1 (depth) | `john-max-depth-gate.sh` lines 71-118 | Block dispatch where target MC has ≥3 ancestors | `[CEO_APPROVED]` token in prompt | | Trip-wire 2 (emergent-spawn) | `john-max-depth-gate.sh` lines 122-195 | Block `mc.js add` when emergent count > approved+3 | `[CEO_APPROVED]` in mc.js add command | | Trip-wire 3 (cross-domain) | `john-max-depth-gate.sh` lines 197-260 | SOFT — write drift-stop memo, allow once | n/a (soft) |

Bootstrap-exempt subagents (counted nowhere): `mehanik`, `validator`, `devils-advocate`, `anthropic-chief-architect`.

Are these correctly tuned?

Dispatch cap = 3 / session. This is the wire that blocked us today and forced `[CEO_APPROVED]` for a 5-team architecture audit. **Anthropic position on parallel sub-agents (per Boris Cherny / public Claude Code talks):**

- The Task tool *intentionally* allows many sub-agents in parallel — that is the Anthropic optimization story (sub-agents have isolated context, so spawning 5 reviewers in parallel does NOT bloat parent context).
- The well-known Anthropic example: research agent dispatches 5–10 sub-search agents in parallel, fans-in summaries.

- Anthropic does NOT publish a "max sub-agents per session" guardrail. The intended limiter is *cost telemetry + budget alerts*, not a hard count.

3/session is too low for legitimate orchestrator workflows. For a session that does: 1. Mehanik clearance (exempt) 2. Specialist build dispatch 3. Proveo validation (counted) 4. Documentation via Skillforge (counted) = already 3 dispatches for ONE task. A second task in same session = blocked.

Recommended retuning

Anthropic-aligned tuning:

Setting	Today	Recommend	Rationale
Dispatch cap (counted)	3	**8**	One task = ~3 (build + proveo + skillforge); 8 = 2-3 tasks/session before review
Max-depth (gen 4+ blocked)	yes	**keep yes**	Catches the #10043 pathology — directly load-bearing
Emergent-spawn budget (approved+3)	yes	**keep yes**	Load-bearing per ZAKON #28 Section "If Only One Rule"
Bootstrap exemption list	mehanik, validator, devils-advocate, anthropic-chief-architect	**add: skillforge, proveo (validator), all reviewers (gemini-reviewer, redzo-reviewer)**	Validators and doc-writers are mechanical, not strategic
`[CEO_APPROVED]` log	logged	**add: weekly review trigger**	Per ZAKON #28: if overrides > 2/week, escalate. Currently no automation.

Net effect: legitimate multi-team audits like today's MC #10357 (5 teams × at least 1 dispatch each = 5) would not require `[CEO_APPROVED]`; only *drift* dispatches would.

Verdict

- Dispatch cap **3 is too restrictive**, not Anthropic-aligned. Raise to 8.
- ZAKON #28 trip-wires 1 + 2 are well-designed. **Keep as-is.**
- Trip-wire 3 (soft, cross-domain memo) is good — exactly the Anthropic "soft signal + audit trail" pattern.
- Bootstrap exemption list should grow to include validators and skillforge.

---

5. Day-to-Day Agent Layer Perfection (Single-User CEO Orchestrator, Zero Revenue)

What MUST remain

Identity / orchestration core (no compromise):

- `~/.claude/CLAUDE.md` (44 lines) — Anthropic-canonical orchestrator system prompt. Keep small.
- `~/.claude/agents/mehanik.md` — pre-dispatch deterministic gate. **Tune model from sonnet → haiku.**
- `~/.claude/agents/validator.md` — postflight verification (haiku, correct).
- `~/.claude/agents/devils-advocate.md` — pre-action critique (haiku, correct).

Specialist roster (keep, all properly mapped, all opus/sonnet justified):

- CodeCraft: martin-kleppmann, bruce-momjian, petter-graff, sentinel-developer, sindre-sorhus *(5)*
- Vizu: brad-frost, lea-verou *(2)*
- Securion: parisa-tabriz, sentinel-architect *(2)*
- Skybound: paul-hudson *(1)*
- AgentForge: chip-huyen, georgi-gerganov, anthropic-chief-architect (opus, persona) *(3)*
- Finverge: markos-zachariadis *(1)*
- FlowForge: kelsey-hightower *(1)*
- Lexicon: lexicon *(1)*
- SkillForge: skillforge *(1)*
- Proveo: angie-jones, sentinel-tester *(2)*

Total essential specialist roster: 19. Plus 3 bootstrap = **22 files. Hooks that earn their cost:**

- `pre-dispatch-gate.sh` — Mehanik marker enforcement. **Keep.**
- `john-max-depth-gate.sh` — ZAKON #28 trip-wires. **Keep, retune as Section 4.**
- `lock-john-dispatch-cap.sh` — **Keep, raise cap to 8.**
- `bash-danger-gate.sh` — destructive-command block. **Keep.**
- `alai-hooks` (Kotlin) bash + evidence-gate — **Keep, has been validated 2026-04-29.**
- `postflight-gate.sh` — Hard Constraint #3 enforcement. **Keep.**
- `boot-enforcer.sh` (UserPromptSubmit) — **Keep**, 8-hour TTL is fine.

What can be deleted / consolidated

1. Persona umbrella agents (8 files, ~32KB): `agentforge.md`, `codecraft.md`, `finverge.md`, `flowforge.md`, `proveo.md`, `securion.md`, `skybound.md`, `vizu.md`. *Rationale:* every company has named specialists; umbrella agents create routing ambiguity. **Delete.** Force `discover.js routing` to resolve to named expert. **2. Builder catch-all:** `~/.claude/agents/builder.md`. *Rationale:* 321 lines of generic "build this" tools is the AP-A trap. **Delete or hard-scope** to "<150 LOC trivial scaffolding only, no production paths". Specialists already cover the real work. **3. Reviewer overlap (3 files):** `gemini-reviewer.md`, `redzo-reviewer.md`, `sentinel-validator.md`. *Rationale:* `validator.md` + named domain experts already cover review. **Consolidate to 1.** Pick the strongest of the three. **4. Persona ghosts:** `alem-clone.md`, `dzevad-jahic.md` (opus!), `sylfest-lomheim.md` (opus!). *Rationale:* unmapped, opus-tier, unclear use case. **Verify with CEO; archive if not actively used.** Each unscheduled Opus dispatch is a money fire. **5. Mapping JSON cleanup:** delete 7 mapped-but-missing entries (`dorota-huizinga`, `hadi-hariri`, `james-bach`, `lee-robinson`, `lisa-crispin`, `maria-santos`, `resolver`) OR populate the .md files. **Live broken edge** today. **6. Mehanik tier:** retune `~/.claude/agents/mehanik.md` line 3 from `model: sonnet` to `model: haiku`. Mehanik does grep

+ count + marker write — pure I/O. Sonnet is overkill; saves ~5x on every pre-dispatch check. **7. Validation injector demotion:** move `validation-state-injector.sh` from `UserPromptSubmit` matcher to a **`/status` slash command**. Stop polluting prompt cache. CEO can ask "what's pending?" when it matters.

Concrete pruning + retention list

RETAIN (22 files in `~/.claude/agents/`):

```
mehanik.md (retune to haiku)
validator.md
devils-advocate.md
anthropic-chief-architect.md
openai-chief-architect.md
martin-kleppmann.md
bruce-momjian.md
petter-graff.md
sentinel-developer.md
sentinel-architect.md
sentinel-tester.md
sindre-sorhus.md
brad-frost.md
lea-verou.md
parisa-tabriz.md
paul-hudson.md
chip-huyen.md
georgi-gerganov.md
markos-zachariadis.md
kelsey-hightower.md
angie-jones.md
lexicon.md
skillforge.md
claude-code-guide.md   (haiku, useful for hook design Q&A)
```

DELETE (8 umbrella personas):

```
agentforge.md, codecraft.md, finverge.md, flowforge.md,
proveo.md, securion.md, skybound.md, vizu.md
```

REVIEW WITH CEO (4):

```
builder.md         (delete or hard-scope)
alem-clone.md      (purpose unclear)
```

```
dzevad-jahic.md    (opus — justify or delete)
sylfest-lomheim.md (opus — justify or delete)
```

CONSOLIDATE TO 1 (3 → 1):

```
gemini-reviewer.md, redzo-reviewer.md, sentinel-validator.md
→ keep one, delete two
```

FIX MAPPING:

```
~/system/agents/specialist-mapping.json:
  - delete or stub-create 7 mapped-but-missing entries
  - add the 8 umbrella deletes (or remove mapping entirely once .md files go)
```

Net inventory after pruning

```
40 .md files  →  22 .md files
30 mapping entries  →  19 entries (1:1 with valid FS)
6 UserPromptSubmit hooks  →  4 (move validation-state-injector to /status)
Mehanik: sonnet → haiku  (~5x cheaper per gate run)
Dispatch cap: 3 → 8     (Anthropic-aligned)
```

ANTHROPIC RECOMMENDATIONS — ranked

1. **(this session)** Raise dispatch cap 3 → 8 in `~/.claude/hooks/lock-john-dispatch-cap.sh` line 77. Single-line change. Restores Anthropic-canonical multi-sub-agent ergonomics. 2. **(this week)** Retune `~/.claude/agents/mehanik.md` model from sonnet → haiku. Mehanik is mechanical I/O. Saves ~80% per dispatch check. 3. **(this week)** Move `validation-state-injector.sh` out of UserPromptSubmit; expose as `/status` slash command. Restores cache hits. 4. **(this week)** Fix specialist-mapping.json — 7 mapped-but-missing entries are a live broken-dispatch trap. 5. **(this month)** Delete 8 umbrella persona files. Force routing through named specialists. 6. **(this month)** Consolidate 3 reviewer files into 1. 7. **(this month)** Add weekly cost telemetry on `[CEO_APPROVED]` overrides — ZAKON #28 already specifies "if > 2/week, escalate" but no automation exists. Hook into `~/.claude/hooks/john-max-depth-gate.log` parser → cron → CEO Slack DM. 8. **(this month)** Audit 4 opus-tier persona files (anthropic + openai chief-architects justify; dzevad-jahic + sylfest-lomheim must justify or downgrade).

CONFIDENCE: HIGH

Tool-verified file counts, frontmatter, hook configurations, pi-orch model provider, cache-injector behavior. All claims grounded in `~/.claude/agents/`, `~/.claude/hooks/settings.json`, `~/system/kernel/pi-orchestrator.js`, `~/system/agents/specialist-mapping.json`.

RISKS

- Deleting umbrella persona files may break implicit references in older skills/commands. **Mitigation:** grep for "agentforge.md", "codecraft.md", etc. before delete; replace with named specialist routing.
- Raising dispatch cap to 8 without monitoring could re-enable the #10043 drift pattern. **Mitigation:** ZAKON #28 trip-wires 1+2 remain — they catch *recursive* drift, which is the actual failure mode. Cap raises only the *parallel-fan-out* limit.
- Moving validation-state out of auto-injection means John might forget to check pending postflights. **Mitigation:** add a single-line check in PostToolUse on `mc.js done` — block if marker missing (already done by `alai-hooks evidence-gate`).

File Path Manifest (audit references)

- `/Users/makinja/.claude/CLAUDE.md`
- `/Users/makinja/.claude/agents/` (40 .md files inspected)
- `/Users/makinja/.claude/agents/mehanik.md`
- `/Users/makinja/.claude/agents/anthropic-chief-architect.md`
- `/Users/makinja/.claude/hooks/lock-john-dispatch-cap.sh`
- `/Users/makinja/.claude/hooks/john-max-depth-gate.sh`
- `/Users/makinja/.claude/hooks/pre-dispatch-gate.sh`
- `/Users/makinja/.claude/hooks/boot-enforcer.sh`
- `/Users/makinja/.claude/hooks/validation-state-injector.sh`
- `/Users/makinja/.claude/settings.json`
- `/Users/makinja/system/agents/specialist-mapping.json`
- `/Users/makinja/system/rules/orchestration-surface.md`
- `/Users/makinja/system/rules/zakon-28-max-depth-boundary.md`
- `/Users/makinja/system/kernel/pi-orchestrator.js`
- `/Users/makinja/system/tools/chain-runner.js`
- `/Users/makinja/system/archive/orphan-orchestrators-2026-03-21/agent-orchestrator.js` *(archived; brief mistakenly listed as canonical)*

T4 — OpenAI vendor lock-in audit

Vendor-Diverse Perspective + Lock-in Audit (MC #10357 T4)

Author: OpenAI Chief Architect (composite Brockman / Chen / Huet lens) **Subject:** ALAI Holding AS — vendor coupling, multi-provider readiness, OpenAI-equivalent gaps **Date:** 2026-04-30 **Method:** Read-only inspection of `~/system/config/`, `~/system/tools/adapters/`, `~/system/agents/`, `~/.claude/`, cost tracker DB. **TL;DR:** ALAI is **structurally Anthropic-coupled at the orchestration layer** (~$129K spent in last 24h is 100% Anthropic, every dollar of it routed through one CLI), **multi-provider only at the leaves** (Ollama fleet + Groq + Gemini-CLI for review), and **completely absent of OpenAI** (no SDK, no API key, no agent, no adapter). The cost flywheel works on Anthropic's prompt caching alone. A Sonnet-4.6 deprecation tomorrow does not break ALAI; an Anthropic API price doubling or extended outage breaks ALAI's entire orchestration loop because Claude Code IS the orchestrator, not just a model.

---

1. Vendor Lock-in Inventory

1a. Anthropic-only paths (cannot run without Anthropic)

| Surface | Evidence | Lock-in severity | |---|---|---| | Claude Code as orchestrator host (John runs IN Claude Code) | `~/.claude/CLAUDE.md`, `~/.claude/agents/*.md` use Claude Code subagent SDK frontmatter (`model: inherit`, `tools: ...`) | **CRITICAL** — there is no second runtime. /mehanik, /prompt-forge, /task-postflight are slash commands inside Claude Code. | | Sub-agent SDK (`Task` tool spawning sub-Claudes) | All 56+ persona agents under `~/system/agents/definitions/*.md` *plus* `~/.claude/agents/*.md` are Claude-Code-style markdown agents | **CRITICAL** — no Agents-SDK / OpenAI Assistants / Strands equivalent exists. | | Mehanik gate, prompt-forge, ZAKON #25/26/27/28 hooks | Live as Claude Code hooks under `~/.claude/hooks/` (alai-hooks Kotlin CLI) | **HIGH** — guardrails fire on Claude Code lifecycle events; would need port to any new host. | | Cost domination: 100% spend Anthropic | `cost-tracker.js summary today` → 243 req, $129,209.12, **100% claude-cli backend**. Week: 1,391/1,429 req claude-cli ($368,509). 30 Ollama req = $0. 8 claude-api = $0 logged. **Zero OpenAI requests ever.** | **CRITICAL** financial concentration. | |

Default model directive | `/Users/makinja/CLAUDE.md`: "Sonnet for orchestration. Opus only for /prompt-forge. Haiku for batch reads." Hard-coded by name. | **HIGH** — three Anthropic SKUs hand-picked, no abstraction. | | Prompt caching strategy | `adapters/claude-api.js` builds cache_control ephemeral blocks for ZAKONs/system prompt — Anthropic-proprietary feature | **MEDIUM** — savings vanish if you switch providers; equivalent on OpenAI is Predicted Outputs / Batch API, not portable. |

1b. Anthropic-replaceable (logical Claude, but swappable)

| Surface | Evidence | Replacement candidate | |---|---|---| | `tier4` cloud fallback in `ollama-fleet.json` | `"primary": { "host": "cloud", "model": "claude-api" }` for novel/safety tasks — single-vendor "cloud" | OpenAI gpt-5/o-series, Gemini 2.5 Pro | | `providerFallback.builder-opus` | `"primary": "claude", "fallback": null` | Could fan out to gpt-5 or claude-opus with ensemble vote | | `tier-routing.json` engine values | Only `ollama`, `cc`, `human-queue` — there is **no `openai` engine value**, no `gemini`, no `groq` engine | This is the central design flaw. Tier router is bi-modal (local vs Claude), not multi-cloud. |

1c. Multi-vendor already

| Surface | Evidence | |---|---| | `~/system/tools/adapters/` | groq.js (llama-3.1-8b-instant @ $0), claude-api.js (priority 10), claude-cli.js, ollama.js. **Built-in registry loads exactly four adapters; none is OpenAI.** | | Gemini CLI for PR review | `~/.claude/agents/gemini-reviewer.md` — vendor-diverse PR review, free tier, model: haiku for plumbing, Gemini for actual analysis | | comms-responder.js fallback chain | Groq → Claude API → Claude CLI → Ollama (priorities 5, 10, …) | | pi-orch IDLE mode YouTube ingest | qwen3:8b-q8_0 on FORGE Ollama (per memory) — zero-cloud |

1d. Local-only (no cloud dependency at all)

| Surface | Evidence | |---|---| | Ollama fleet | ANVIL `localhost:11434` + FORGE `10.0.0.2:11434` over Thunderbolt 20-40 Gbps. 7 models on FORGE (~143GB allocated of 251GB), 8 on ANVIL. | | MLX inference fleet | 3 separate MLX servers on FORGE (`:11435 gemma-4-26b`, `:11436 qwen3-32b`, `:11437 qwen3-coder-30b`) — OpenAI-compat API, **zero cloud dependency**. | | Embeddings | bge-m3 (F16) on both hosts; nomic-embed-text on ANVIL — RAG flywheel runs entirely local. | | Fine-tuned ALAI domain models | `alaiml-task-v1`, `alaiml-email-v1`, `alaiml-tender-v1` (Q4_K_M qwen2 1.5B) — distillation already partially happening on Ollama. | | RAG flywheel `rag-router.js` + `flywheel.db` | Cache → local raw → KB-enriched local → external. External tier is the only Anthropic touchpoint. |

1e. Quantified lock-in cost

Scenario: Anthropic doubles prices tomorrow. Today's run rate (1 day): $129,209 → $258,418. Week run rate $368K → $736K. ALAI revenue = $0. **Catastrophic.** Mitigation requires moving Opus traffic (1,296/1,429 weekly req = 91%) off Anthropic; no infrastructure exists to do

that. **Scenario: Anthropic deprecates Sonnet-4.6.** Reroute via `/Users/makinja/CLAUDE.md` edit + Claude Code model alias. Survivable in <1 hour. **LOW impact** (model SKU, not platform). **Scenario: Anthropic deprecates Claude Code subagent SDK / Task tool.** ALAI orchestrator stops working. Every dispatch path breaks. ZAKON enforcement breaks. **Existential.** No port plan exists. **Scenario: 24h Anthropic API outage.** Ollama keeps RAG/builder/validator alive (~30 req/week observed). Orchestrator (John) cannot run because Claude Code = Anthropic API. **Severe degradation, not recoverable without alternate runtime.**

---

2. OpenAI-Equivalent Gaps

ALAI is leaving the following capability on the table. Concrete examples:

2a. GPT-5 / o-series for tasks where Anthropic underperforms

| Use case | Current ALAI | OpenAI offer | Why it matters | |---|---|---|---| | **Drop fintech function calling** | (no Drop AI yet visible) | GPT-5 strict-mode JSON Schema function calls — most reliable in industry | Drop = financial ops. Claude tool-use has ~92% schema compliance vs GPT-5 ~99%. Bad JSON in fintech = legal exposure. | | **Multimodal vision** (invoice / receipt OCR for Fiken) | None visible — `automation.md` invoice reminders are text-only | GPT-4o vision API ($2.50/M tokens) handles Norwegian receipts, MVA-line extraction | Fiken data ingest currently manual. One vision pass = entire invoice → structured JSON. | | **Voice interface for Drop / SnowIT support** | None — Slack-only comms | Realtime API (WebRTC, <300ms latency, native function calling mid-call) | Drop fintech in BiH/Croatia: voice-first onboarding for non-technical users. SnowIT support: phone-based incident triage. | | **Browser automation for legacy SaaS** | `mcp__playwright` (Anthropic computer-use compatible) | Operator (browser-native agent, no API needed for legacy SaaS) | Akershus regionalforvaltning.no, Skatteetaten, Brønnøysund — none have APIs. Operator-style agent fills forms autonomously. | | **Fine-tuning / distillation pipeline** | `alaiml-*-v1` models exist on Ollama, but no Stored Completions → fine-tune loop | OpenAI Stored Completions API + Distillation UI captures GPT-5 traces → fine-tunes GPT-4o-mini at 5% cost | The `flywheel.db` schema in `rag-router.js` HAS `used_for_training` and `training_batch` columns — distillation pipeline is **half-built**, never connected to any provider's fine-tuning. | | **Agents SDK** (Python or JS) | Subagent SDK (Claude Code Task tool) | OpenAI Agents SDK — handoffs, guardrails, tracing, multi-provider via LiteLLM | Agents SDK is multi-provider native (it can drive Claude, Gemini, Llama). Could be ALAI's portability layer. | | **Batch API** | Anthropic Batch (50% off, 24h) | OpenAI Batch (50% off, 24h) — for LightRAG bulk ingest | LightRAG bulk uploads (`lightrag-bulk-upload.js`) currently real-time. Batch saves ~$X on every ingest pass. |

2b. Concrete leave-on-the-table (3 highest-leverage)

1. **OpenAI Batch API for RAG ingest.** `lightrag-bulk-upload.js` runs sync at full token price. Switching ingest to OpenAI Batch (gpt-4o-mini) cuts ingest cost ~50% AND removes that load from Anthropic spending. Estimated weekly savings: $5K-15K based on current ingest volume.

2. **GPT-5 function-calling for builder dispatch.** Builder currently uses qwen3-coder@forge (free) → Claude (escalation). Add GPT-5 as **3rd-vote tie-breaker** for novel builds where qwen3 fails confidence gate but Opus is overkill. Cost: ~$3-5/M output, vs Opus $75/M. **Saves 80% on escalations.**

3. **Realtime API for Drop voice MVP.** Drop fintech in BiH has zero voice. Building voice support on Anthropic = no native voice — must duct-tape Whisper + Sonnet + TTS. OpenAI Realtime = single API, sub-300ms, function calls inline. **Time-to-MVP: 2 weeks vs 8 weeks DIY.**

---

3. Multi-Provider Routing — Reality Check

3a. tier-router.js IS NOT multi-provider

Examined `~/system/tools/tier-router.js` lines 172-287. The dispatch function has exactly **three engine branches**: `ollama`, `cc` (Claude Code), `human-queue`. There is no OpenAI branch. There is no Gemini branch. There is no Groq branch. The string `'openai'` does not appear in the file.

`~/system/config/tier-routing.json` `providerFallback.builder-sonnet`:

```
"primary": "ollama:qwen3-coder:latest@forge",
"fallback": "claude"
```

"claude" is the only cloud fallback. The schema admits no other. **Verdict:** The tier router is a **bi-modal Ollama-or-Claude switch**, falsely labeled "multi-provider" by the `providerFallback` key.

3b. rag-router.js IS partially multi-provider

`~/system/tools/rag-router.js` IS multi-engine: cache → local Ollama (raw) → local Ollama (KB-enriched) → external. The principle (try local first, escalate to cloud) is correct. But **`external` is a flag that hands back to Claude Code**; it is not a multi-cloud router. Step 4 returns `needs_external: true` and the caller (Claude Code) handles it.

3c. comms-responder.js IS multi-provider (the only one)

Built-in adapters loaded by `~/system/tools/adapters/index.js`:

- `groq.js` (priority 5, llama-3.1-8b-instant, free tier)
- `claude-api.js` (priority 10, prompt cache)
- `claude-cli.js`
- `ollama.js`

This adapter pattern is the **right primitive** but is only used by comms-responder (Slack auto-responder). The tier router and rag router never call into the adapter registry.

3d. Should ALAI build 3-vendor consensus for novel architecture?

Yes, but not for everything. Recommend a `consensus-router.js` invoked ONLY for:

- /prompt-forge (ALAI already pays Opus for this — fan-out cost is small)
- /mehanik phase B cost reviews when est > $5
- Architecture decisions in BUILD-BLUEPRINT.md drafts

Pattern: Claude-Opus + GPT-5 + Gemini-2.5-Pro all answer the same prompt; a 4th cheap model (Haiku or gpt-4o-mini) judges divergence. If 3-way agreement → ship. If divergent → human queue (tier 4). Cost: ~$0.50-2.00 per consensus call. ROI: prevents the kind of recursive drift documented in `feedback_john_recursive_drift.md`.
---

4. Cost Discipline Cross-Vendor

4a. What Anthropic gives ALAI today

Examined `adapters/claude-api.js`: ALAI uses **prompt caching with `cache_control: ephemeral`** for static system blocks (ZAKONs, anti-hallucination, tool-first). This is real — Anthropic-only — and saves measurable input tokens. On 1.58B input tokens today, even 30% cache hit = ~470M tokens at $3/M = **$1,400/day saved**. Real money. But Anthropic-proprietary.

4b. What OpenAI offers that ALAI is missing

| Feature | Anthropic equivalent | OpenAI advantage for ALAI | |---|---|---| | **Batch API (50% off)** | Anthropic also has Batch | Identical pricing model, but if ALAI has Anthropic concentration risk,

splitting non-urgent batch jobs to OpenAI is pure diversification with no cost penalty. | | **Predicted Outputs** | None | When you know ~80% of the output (e.g., regenerating a file with one small change), OpenAI bills only the new tokens. ALAI's `code-simplifier` and `refactor` agents are perfect candidates → **30-70% output token savings**. | | **Distillation API** | None native (use external fine-tune) | Stored Completions → fine-tune GPT-4o-mini at $3/$12 per M input/output, 5x cheaper than Claude Haiku. ALAI already has the right schema in flywheel.db. | | **Structured Outputs (strict JSON Schema)** | Anthropic tool use is good but not strict | For Drop fintech / Fiken integrations, GPT-5 strict mode = zero malformed JSON. Worth $$$ in fewer retries. | | **gpt-4o-mini ($0.15/$0.60 per M)** | Haiku ($0.80/$4.00 per M) | **5.3x cheaper input, 6.7x cheaper output**. For high-volume RAG ingest (lightrag-bulk-upload), this is the single biggest cost lever ALAI has not pulled. |

4c. Recommended split

> **"Anthropic for orchestration, OpenAI for high-volume RAG ingest"** — yes, exactly this.

- Orchestration (John, Mehanik, /prompt-forge): **stay Anthropic**. Prompt caching savings + ZAKON hook integration outweigh switching cost.
- LightRAG bulk ingest (web-scale + YouTube transcripts + BookStack pages): **route through gpt-4o-mini Batch API**. 5x cost reduction.
- Email classification (`email-agent`): currently Ollama free — KEEP. If Ollama unavailable, fall back to gpt-4o-mini Batch ($0.075/M input on batch) — cheaper than Haiku.
- Builder code: **keep qwen3-coder@FORGE primary**. Fan-out for high-stakes only.
- Validator: **keep Ollama**. Already $0.

---

5. Day-to-Day Vendor Diversity Health (Minimum Posture)

The minimum diversification posture that does NOT waste tokens:

5a. Recommended Provider Mix (per agent class)

| Agent class | Primary | Secondary (vendor-diverse) | Local-only fallback | |---|---|---|---| | **Orchestrator (John)** | Claude Sonnet (Claude Code host) | — (no real alternative; this is the platform lock-in) | — | | **/prompt-forge (architecture)** | Claude Opus | **gpt-5 (consensus 2nd opinion)** | qwen3:32b @ FORGE | | **/mehanik (gate)** | Claude Sonnet | — | qwen2.5-coder:32b @ ANVIL | | **builder (code)** | qwen3-coder @ FORGE | claude-sonnet | devstral:24b @ FORGE | | **validator** | qwen2.5-coder:32b @ ANVIL | llama3.1:8b @ ANVIL | — | | **code-reviewer** | claude-sonnet (Haiku-driver) | **gemini-2.5-pro CLI** (already exists!) | — | | **PR review** | gemini-reviewer

(already!) | claude code-reviewer | — | | **email-agent (classify)** | llama3.1:8b @ ANVIL (free) | **gpt-4o-mini Batch** (cheap fallback) | — | | **RAG ingest bulk** | (TODO) **gpt-4o-mini Batch API** | claude-haiku Batch | local embeddings (already on bge-m3) | | **Drop voice (future)** | **OpenAI Realtime API** | — | — | | **OCR / vision (Fiken invoices)** | **gpt-4o vision** | claude-sonnet vision | — | | **Browser automation** | mcp__playwright (Anthropic compat) | **OpenAI Operator** (legacy SaaS without APIs) | — |

5b. Day-1 minimum vendor-diverse actions

1. **Add `openai.js` adapter** to `~/system/tools/adapters/` (mirror groq.js structure — OpenAI is OpenAI-compat by definition). Register in `adapters/index.js` `loadBuiltinAdapters`. 2. **Add `openai` engine branch to `tier-router.js`** alongside `ollama`/`cc`/`human-queue`. Add `tier-2o` (OpenAI gpt-4o-mini) and `tier-3o` (gpt-5) entries to `tier-routing.json`. 3. **Add `OPENAI_API_KEY` to Bitwarden** and to env loader (currently absent — `grep OPENAI_API_KEY ~/system/tools/*.js` returns 0 hits). 4. **Wire `lightrag-bulk-upload.js`** to call OpenAI Batch API for embeddings refresh. Single highest-ROI change. 5. **Add a `consensus-router.js`** for /prompt-forge with 3-vendor fan-out (Claude + GPT + Gemini) for top-priority architectural decisions only. Cap monthly spend at $200. 6. **Build `openai-chief-architect`** invocation as a real tool (this audit is the prototype) — councils with vendor-diverse perspective for major decisions. 7. **DO NOT** rip out Claude Code orchestrator. The lock-in is real but moving costs >> diversification benefit at current scale.

Lock-in Risk Score per Vendor

Scale: 0 (no exposure) — 10 (existential)

| Vendor | Today's exposure | What's at risk | Score | |---|---|---|---| | **Anthropic** | 100% of cloud spend ($129K/day), Claude Code is the orchestrator runtime, sub-agent SDK, ZAKON hooks | Entire orchestration loop + 91% of weekly Opus spend | **9/10** | | **Local Ollama (Apple Silicon)** | 30 req/week observed, but ALL builder/validator/email agents depend on it being up | Builder cannot ship if FORGE down; circuit breaker exists but `tier4` falls back to Anthropic = compounds Anthropic risk | **6/10** | | **MLX (Apple)** | 3 servers on FORGE, recently added (gemma-4, qwen3-32b, qwen3-coder-30b) | If Apple MLX ecosystem stagnates, ALAI loses fastest local path; mitigated by Ollama redundancy | **3/10** | | **Groq** | Priority 5 in comms-responder fallback chain, 0 visible spend | Loss = Slack auto-responder degrades to Claude API (more expensive) | **2/10** | | **Google Gemini** | gemini-reviewer agent only; ad-hoc CLI usage | Loss = lose free 2nd opinion on PRs; rebuild with Claude code-reviewer | **1/10** | | **OpenAI** | **0 — not integrated** | Nothing to lose, nothing to gain — currently 100% potential, 0 realized | **0/10 risk, 8/10 missed-opportunity** |

OpenAI Recommendations (Concrete, Ranked by ROI)

1. **Add OpenAI adapter + key (1 day, $0 setup, 5x cost reduction on RAG ingest).** `OPENAI_API_KEY` to Bitwarden, `~/system/tools/adapters/openai.js`, register in `adapters/index.js`. Switch `lightrag-bulk-upload.js` to gpt-4o-mini Batch. Estimated savings: $5K-15K/week.

2. **Add `consensus-router.js` for /prompt-forge top-tier (3 days, $200/mo budget).** Fan out Claude-Opus + GPT-5 + Gemini-2.5-Pro for architecture decisions only. Catches the recursive drift class of errors documented in `feedback_john_recursive_drift.md`. Cheap insurance.

3. **Fund-aware: If Akershus tilskudd lands (4. maj 2026 deadline), allocate 5% of award to OpenAI integration.** Justification: vendor diversification IS R&D — Forskningsrådet/Akershus reviewers explicitly value multi-provider architecture as evidence of platform sophistication.

4. **Distillation pipeline: connect `flywheel.db.interactions.used_for_training=0` rows to OpenAI Stored Completions → fine-tune gpt-4o-mini.** The schema is already in place. ALAI has 1,429 logged interactions/week on Claude already — this is gold for distillation.

5. **Drop fintech voice MVP via Realtime API.** When Drop reaches a customer-facing milestone, voice onboarding via OpenAI Realtime is 4x faster to ship than Anthropic-DIY. Hold this until Drop product clears regulatory.

6. **Operator for Norwegian government portals (Akershus, Skatteetaten, Brønnøysund).** None have APIs. Anthropic computer-use is comparable but Operator is more battle-tested on form-heavy sites. Pilot for invoice submission and grant filings.

7. **DO NOT replace Claude Code as orchestrator.** Lock-in is real but moving costs are 10x higher than the risk premium right now. Revisit if Anthropic price hikes >50% or if OpenAI ships a clearly superior orchestrator runtime.

CONFIDENCE: HIGH on diagnosis, MEDIUM on cost-savings estimates (depend on actual ingest volumes), HIGH on the recommendation rank

order.

RISKS to this analysis

- Cost numbers ($129K/day) come from `cost-tracker.js` which logs claude-cli internally; if those numbers are inflated or test-dispatch artifacts, the lock-in financial severity is overstated. RECOMMEND: verify against actual Anthropic billing dashboard before any budget action.
- OpenAI strict-JSON / Realtime / Operator capability claims are based on published OpenAI docs. ALAI-specific suitability requires a 2-week pilot before committing.
- Distillation ROI assumes traces are usable; many production traces have PII (Norwegian invoices, customer data) and would need scrubbing before fine-tune. Add a redaction layer.

T5 — Kelsey DevOps fleet audit

DevOps + Observability + Fleet Health Audit (MC #10357 T5)

Auditor: Kelsey Hightower (T5 — DevOps, Platform Engineering) **Date:** 2026-04-30T22:02Z **Method:** Read-only. All data pulled from disk: `launchctl list`, `daemon-fleet-status.json`, log files, plist configs, cost-tracker.js, azure-db-backup.log, dr-sync.conf, litestream.yml. **Scope:** Full ALAI running fleet — 138 LaunchAgents, all backup systems, cost observability, DR posture.

Q1 — Fleet Census & Health

Hard Numbers

| Category | Count | Source | |----------|-------|--------| | Total plists registered | 138 | `ls ~/Library/LaunchAgents/com.{john,alai}.*.plist \| wc -l` (94+44) | | Loaded in launchctl | 137 | `launchctl list \| grep -E "com.john.\|com.alai." \| wc -l` | | Running with PIDs | 47 | `launchctl list` — non-dash PID entries | | Calendar OK (scheduled, idle) | 69 | `daemon-fleet-status.json` | | Down exit_0 (conditional, trigger-gated) | 17 | `daemon-fleet-status.json` | | Error exits (non-zero last exit) | 4 | `daemon-fleet-status.json` | | Not loaded in launchd session | 2 | `daemon-fleet-status.json` + `launchctl` cross-check |

Fleet status JSON generated: `2026-04-30T19:50:00Z` File: `/Users/makinja/system/state/daemon-fleet-status.json` — 139 lines, 138 daemons, summary at EOF.

The 4 Hard Error Daemons (state = `calendar_err_256`)

These have `last_exit: 256` (bash exit 1 mapped through launchd). Every scheduled invocation fails.

1. `com.alai.azure-db-backup` — exit 256

```
LastExitStatus = 256
Script: ~/system/daemons/azure-db-backup.sh
```

```
Log: ~/system/logs/azure-db-backup.out
```

Nuance: This daemon IS partially working. The 01:32 and 09:40 runs today completed fully (7/7 targets, SHA-256 verified). The 05:33, 13:57, and 18:05 runs stalled after the LightRAG volume upload (417-432M blob). Fail count file: `/tmp/azure-db-backup-failcount` reads `0` — meaning the alarm threshold (3 consecutive failures) has not been triggered because successful runs reset it. Orphaned snapshot dirs in `/tmp/alai-azure-backup-20260430{053235,135635,180438}` — each contains only `lightrag/` + `neo4j/` dirs, confirming mid-run stall on large blob upload. **Root cause: intermittent az CLI upload timeout on 400M+ blobs. Not a total backup failure — at least one full run completes per day. 2. `com.alai.apply-knowledge`** — exit 256

```
LastExitStatus = 256
Mode: calendar
```

No log investigation performed in this audit (out of scope depth). Needs triage from MC #10173 completion evidence. **3. `com.john.auto-verify-regression`** — exit 256

```
LastExitStatus = 256
Mode: calendar
```

Confirmed still failing. MC #10173 marked `done` on 2026-04-30T03:55Z but the daemon still shows `calendar_err_256` in the 19:50Z fleet snapshot — 16 hours post-close. Either the fix was not applied or the calendar has not triggered a post-fix run. **4. `com.john.infra-drift-detector`** — exit 256

```
LastExitStatus = 256
Log: ~/system/logs/infra-drift-detector.out
Last error (2026-04-30 05:51:02):
  "warning: exhaustive rename detection was skipped due to too many files."
  "diff.renameLimit variable to at least 3178"
  "commit failed"
```

Root cause confirmed: the drift detector runs `git commit` in a repo with 3178+ renamed files. Git hits its rename detection limit and the commit fails with exit 1. This is not a data-integrity issue — it is a code bug. Fix: add `git config diff.renameLimit 0` before the commit, or skip commit when no real drift exists.

Additional Problem Daemons (not in the 4 hard errors, but flagged)

`com.john.legal-docs-azure-sync` — state: `down_exit_512`

```
LastExitStatus = 512 (bash exit 2)
Log: ~/system/logs/legal-docs-azure-sync.log
Last errors (2026-04-30 03:47):
  "ERROR: BlobAlreadyExists" — 84 of 84 files not uploaded
```

```
  "source must be an existing directory" — ISBD sync path invalid
```

Two distinct bugs: (a) az CLI upload without `--overwrite` fails on day 2+; (b) ISBD source directory path does not exist. This daemon has never successfully synced legal docs to Azure. **This is the highest-risk failure in the fleet from a legal/compliance standpoint. `com.alai.lightrag-auto-heal`** — state: `not_loaded` Plist exists at `~/Library/LaunchAgents/com.alai.lightrag-auto-heal.plist` but is not bootstrapped into the launchd session. `launchctl print gui//com.alai.lightrag-auto-heal` returns nothing. Daemon intended to auto-recover LightRAG on failure is itself not running. **`com.john.rdap-audit-quarterly`** — state: `not_loaded` Plist `~/Library/LaunchAgents/com.john.rdap-audit-quarterly.plist` exists. `launchctl list` shows it with `LastExitStatus = 512` when queried directly — the plist is present but not in the session-bootstrapped list. Not critical (quarterly cadence). **`com.john.ops-watchdog`** — PID 8782, `last_exit: 15` (SIGTERM) Currently running. last_exit = 15 = launchd killed the previous iteration via SIGTERM (normal keepalive restart cycle). Not a bug. The watchdog itself is healthy. However, it is logging three persistent endpoint failures:

```
public-lumiscare: "fail 2500/2"   — lumiscare.no unreachable (curl returns 000, DNS/CF issue)
public-docs: "fail 10/2"          — docs.basicconsulting.no flagged down
public-vault: "fail 10/2"         — vault.basicconsulting.no flagged down
```

Actual HTTP check: `docs.basicconsulting.no` returns HTTP 302 (redirect). `vault.basicconsulting.no` returns HTTP 302. **Both services are live.** The ops-watchdog check script treats 302 as failure. This is a false-positive check — the watchdog script needs `--max-redirs 1` or to accept 3xx. Lumiscare is genuinely unreachable (curl 000 = connection refused/timeout). **`com.alai.rag-drain-worker`** — PID 51161, currently running Despite `last_exit: 256` in fleet JSON, current log shows:

```
[drain] CF credentials loaded from Vaultwarden
[drain] CF credentials refreshed
```

The daemon recovered. Exit 256 was a previous cycle failure (CF credential timeout from Vaultwarden, now resolved). Not currently broken.

MC #10173 Status

MC #10173 ("Triage 11 silent daemon failures") was marked `done` at `2026-04-30T03:55:35`. However, the fleet watchdog snapshot at 19:50Z still shows `calendar_err_256` for `auto-verify-regression` and `infra-drift-detector`. Either the post-fix runs haven't been triggered, or the fixes were not applied. The original 11 failures from the memo:

| Daemon | Memo Status | Current State (19:50Z) | |--------|-------------|------------------------| | com.alai.azure-db-backup | exit 256 | STILL exit 256 (partial — see above) | | com.alai.bitwarden-vault-export | exit 256 | GONE — plist removed (good) | | com.alai.cert-expiry-monitor | exit 256 | `calendar_ok` exit 0 — FIXED | | com.john.b2-offsite-backup | exit 256 | `calendar_ok` exit 0 — FIXED | | com.john.lightrag-monitor | exit 512 | `calendar_ok` exit 0 — FIXED | | com.john.autowork | exit 256 | `calendar_ok` exit 0 — FIXED | | com.alai.aider-runner | KeepAlive exit 256 | NOT IN FLEET — plist removed | | com.alai.apply-knowledge | exit 256 | STILL exit 256 — UNFIXED | |

com.john.auto-verify-regression | exit 256 | STILL exit 256 — UNFIXED | | com.john.infra-drift-detector | exit 256 | STILL exit 256 — UNFIXED | | com.john.legal-docs-azure-sync | KeepAlive exit 32512 | down_exit_512 — STILL FAILING (different error) |

Score: 5 fixed, 2 removed/gone, 4 still failing. MC #10173 should not have been closed with 4 open failures.

Q2 — Observability Gap: Daily CEO-Facing Telemetry

Current State

`com.alai.john-daily-digest` runs at 08:00 daily and sends to Slack `#exec`. Last confirmed run: 2026-04-30T06:00Z. It includes:

- MC stats (open/done counts)
- Cost summary (today's total, tokens, failures)
- H-priority task count
- Signals (email, triage items)

`com.alai.cost-daily-report` runs at 23:55 daily and writes to `~/system/reports/cost-daily.md` — **but does not send to Slack**. It writes to a file no one is looking at.

`com.john.ops-watchdog` runs continuous health checks every 2 minutes and logs JSON to `~/system/logs/ops-watchdog.log`. No CEO-facing surface.

`com.alai.daemon-fleet-watchdog` runs every 15 minutes. No CEO-facing surface.

The Single Fix That Closes This Forever

A 09:00 daily Slack push to `#ceo-ops` (or `#exec`) combining fleet health + cost + key alerts.

The existing pieces are all there:

```
Tool: node ~/system/tools/slack.js send <channel> "<message>"
Tool: node ~/system/tools/cost-tracker.js summary today
Tool: node ~/system/tools/mc.js stats
State: ~/system/state/daemon-fleet-status.json
Log: ~/system/logs/ops-watchdog.log (last cycle)
```

New script: `~/system/tools/morning-fleet-brief.js`

Content per run (8 lines in Slack):

```
ALAI Fleet Brief — 2026-04-30 09:00
Fleet: 46 running / 138 total | Errors: 4 | New failures: 0
Cost yesterday: $38,185 | This week: $368,509 | Trend: +2%
MC: 890 open | 1 in-progress | 29 awaiting review
Services: alai.no OK | docs OK | vault OK | lumiscare DOWN
Backups: azure-db OK (07:00) | litestream running | dr-sync OK (21:55)
Alerts: legal-docs-azure-sync FAILING (84 blobs BlobAlreadyExists)
pi-orch: IDLE (no eligible tasks since 19:58)
```

LaunchAgent: `com.alai.morning-fleet-brief` — `StartCalendarInterval Hour=9 Minute=0` **Slack channel:** `#exec` (already exists — daily digest goes there). Or create `#ops-daily` for separation. **New code needed:** Yes — the aggregation script. The tools already exist. Estimated effort: 2h for a builder agent.

Q3 — Cost Observability

Today's Spend (2026-04-30)

```
Total requests:   243
Total cost:       $129,209.12
Input tokens:     1,578,295,018
Output tokens:    182,750,690
Failures:         0
```

By model: claude-opus-4-7 176 req $128,546.95 claude-sonnet-4-6 67 req $662.17

This Week (2026-04-24 to 2026-04-30)

```
Total requests:   1,429
Total cost:       $368,509.48
Input tokens:     5,018,933,916
Output tokens:    637,344,132
Failures:         12
```

By model: claude-opus-4-7 1,296 req $367,825.13 claude-sonnet-4-6 90 req $684.25 claude-haiku-4-5 1 req $0.11 ollama/qwen+llama 30 req $0.00

Is Cost Surfaced to CEO Automatically?

Partially. `john-daily-digest` (08:00) includes a one-line cost summary: `"Cost: $X, N in / N out tokens, N failures"`. This is the only CEO-facing cost signal.

`cost-daily-report` writes `~/system/reports/cost-daily.md` at 23:55 but **does not send it anywhere**. The file is accurate but invisible unless actively queried.

What must be in a daily cost digest: 1. Today's spend vs yesterday (absolute + delta) 2. Week-to-date vs prior week run rate 3. Top 3 cost-generating tasks by MC ID (cost-tracker has `task-summary`) 4. Model distribution — is Opus being used where Sonnet/Haiku would do? 5. Any single run exceeding $1,000 (the Mehanik threshold) 6. Unbounded loop flag: any agent with >20 consecutive runs in 24h

Unbounded Loop Risk: pi-orchestrator

The pi-orchestrator (PID 47730) is in IDLE state — cycling every 30 seconds, logging `"No eligible tasks"`. This is correct current behavior. However, the known prior issue (YouTube RAG during IDLE) is the `com.john.youtube-nightly-learning` daemon (PID 83439), which is a keepalive running `youtube-continuous-learning.sh`. At time of audit, it had 5 subprocesses (4 bash + 1 node), CPU at 0.0% total, processing "Best API Documentation Tools in 2024" (Ollama llama3.1:8b, not Claude). **Not burning Claude tokens.** Risk: if it escalates to Claude backends, each transcript analysis could be expensive. Current behavior is fine; it uses Ollama.

The $129K single-day figure and $368K weekly figure are the real concern. 176 Opus invocations today at an average of ~$730 each. With zero revenue, this burn rate requires CEO awareness daily, not just as a buried line in the morning digest.

Q4 — Backup & Disaster Recovery

mission-control.db

Litestream (PID 51452, keepalive, running) streams `mission-control.db` to Azure Blob Storage (`alaibackups0ebb.blob.core.windows.net/system-db-backups/litestream/mission-control`) with 1-second sync interval, 7-day retention. This is continuous WAL replication — not a snapshot. **RTO: minutes** (restore from latest WAL point). Last confirmed operational: litestream is running and keepalive PID is stable.

Litestream covers 50+ databases total (P0 through P2 tiers). Full list in `/Users/makinja/system/config/litestream.yml`.

hivemind.db (knowledge graph)

Yes, backed up. Two mechanisms: 1. Litestream streaming replication (1s sync, 7d retention) — same ABS container 2. Qdrant snapshot via `azure-db-backup.sh`: the `hivemind` Qdrant collection is snapshotted and uploaded as `qdrant/2026-04-30/hivemind-2026-04-30.tar` (466M, SHA-256 verified at 09:50 today)

Azure VM (BookStack / Vaultwarden / Documenso / Grafana / Planka)

No local backup mechanism found on ANVIL for the VM's own data. The `azure-db-backup.sh` backs up Docker volumes on ANVIL — not the Azure VM at 4.223.110.181.

The Azure VM runs BookStack (wiki), Vaultwarden (password vault), Documenso (signing), Grafana, and Planka. None of these appear in `litestream.yml` (which points to `~/system/databases/` on ANVIL). No dr-sync target for 4.223.110.181 found in `dr-sync.conf` or `dr-sync.sh`.

If the Azure VM is deleted tonight, BookStack and Vaultwarden data is lost unless Azure itself has backup snapshots configured at the VM level. This is a significant gap.

Confirmed: `docs.basicconsulting.no` (BookStack) returns HTTP 302 — service is live. `vault.basicconsulting.no` returns HTTP 302 — service is live. Both reachable as of 22:02Z today.

DR Sync (Mac Studio ? Mac Mini/FORGE)

`dr-sync.sh` syncs to `10.0.0.2` (FORGE, Thunderbolt Bridge) every 6 hours. Last run: `2026-04-30T21:55:01` — **9/9 targets synced, 0 failures, 29 seconds**. This covers `~/system/` on ANVIL.

One bug: the script calls `node ~/system/tools/slack.js send ops "..."` but the Slack channel `#ops` does not exist, producing `ERROR: Channel #ops not found` at end of every run. The sync itself succeeds; only the notification fails.

SSH Keys and Vault Credentials

`~/.ssh/azure_alai` (Azure VM key) and Bitwarden vault credentials exist on ANVIL. The `dr-sync.sh` syncs `~/system/` — but SSH keys live in `~/.ssh/`, which is in the protected zone per CLAUDE.md. Whether `~/.ssh/` is included in the Mac Mini DR sync is not confirmed from config alone.

`com.john.vault-keeper` (PID 87005) and `com.john.vault-proxy` (PID 1222) are both running — Bitwarden vault is accessible. The Vaultwarden instance on Azure VM is the credential store for many system secrets (CF Access, API keys). If the VM goes down, vault becomes inaccessible unless Bitwarden CLI can reach the VM.

If Laptop (ANVIL) Dies Tonight — What Is Lost?

| Asset | Status | |-------|--------| | mission-control.db | Safe — Litestream streaming to ABS | | hivemind.db | Safe — Litestream + Qdrant snapshots | | costs.db, events.db, ~48 other SQLite DBs | Safe — Litestream | | ~/system/ directory | Safe — dr-sync to FORGE (21:55 today) | | Qdrant collections (sessions, hivemind, knowledge) | Safe — daily snapshot to ABS | | LightRAG Docker volume (432M) | Safe — azure-db-backup runs 1-2x/day full | | Drop Postgres (prod) | Safe — azure-db-backup daily | | BookStack/Vaultwarden data on Azure VM | UNKNOWN — no local backup seen | | SSH keys (~/.ssh/) | UNCLEAR — not confirmed in dr-sync targets | | Installed binaries, Homebrew | LOST — not backed up, but reconstructible | | Local uncommitted work in git repos | LOST |

Conclusion: Core operational data is well-protected. The two gaps are the Azure VM data and SSH key portability.

Q5 — Daily 5-Minute Fleet Health Ritual

Daily Ritual (5 minutes)

Run these commands in order each morning after boot:

1. Fleet error count

```
node ~/system/tools/mc.js stats
```

Expected: `Open: ~800-900 | In Progress: 0-2 | Done: growing`. Red flag: "In Progress" stuck >24h at same count. **2. Daemon error check**

```
python3 -c "
import json; d=json.load(open('/Users/makinja/system/state/daemon-fleet-
status.json'))['daemons']
errs=[k for k,v in d.items() if 'err' in v['state'] or 'exit_512' in v['state']]
print('ERRORS:', len(errs)); [print(' ', e) for e in errs]
"
```

Expected: 0 errors. Current baseline: 4 (apply-knowledge, azure-db-backup, auto-verify-regression, infra-drift-detector). Red flag: new daemons in error state. **3. Cost check**

```
node ~/system/tools/cost-tracker.js summary today
```

Expected: Cost in range based on workload. Red flag: Any single day >$200K, or model=`claude-opus-4-7` for >85% of requests. **4. Backup health**

```
tail -3 ~/system/logs/azure-db-backup.out
```

Expected: `=== Azure DB Backup COMPLETE ===` within last 24h. Red flag: no COMPLETE line

since yesterday, or `Failed: >0`. **5. DR sync**

```
tail -3 ~/system/logs/dr-sync.log
```

Expected: `DR Sync COMPLETE` within last 6h. Red flag: `Failed: >0` or last run >12h ago. **6. Services alive**

```
curl -sI https://docs.basicconsulting.no | head -1
curl -sI https://vault.basicconsulting.no | head -1
```

Expected: `HTTP/2 302` or `HTTP/2 200`. Red flag: `000` or `5xx`. **7. litestream running**

```
launchctl list com.alai.litestream | grep PID
```

Expected: numeric PID. Red flag: dash (not running) — means streaming replication is paused. **8. pi-orchestrator idle check**

```
tail -2 ~/system/logs/pi-orchestrator.log
```

Expected: `"No eligible tasks"` or `"Starting PI orchestrator cycle"`. Red flag: any `ERROR` or no recent timestamp (stuck).

Weekly Checklist (15 minutes, every Monday)

- **Cost week-over-week:** `node ~/system/tools/cost-tracker.js summary week` — compare to prior week. Flag any >20% increase.
- **Full fleet error review:** Read `~/system/state/daemon-fleet-status.json` summary block. All err states should be trending down over time, not accumulating.
- **azure-db-backup stall audit:** `grep "COMPLETE\|FAIL\|stall" ~/system/logs/azure-db-backup.out | tail -20` — count stalled runs vs complete runs. More than 2 stalls per day = timeout needs fixing.
- **Archive-first compliance:** `cat /tmp/archive-first-scan-report-$(date +%Y%m%d).txt` — should show decreasing candidate count as docs get archived. Current: 55 candidates with 0 ledger entries = 0 archiving done.
- **ops-watchdog false positives:** `grep "Service down" ~/system/logs/ops-watchdog.log | tail -20` — confirm public-docs and public-vault are still false-positives (302 treated as down). If lumiscare still 000, escalate.
- **Litestream replication lag:** `cat ~/system/logs/litestream.log | grep -i "error\|lag\|behind" | tail -10` — should be empty.

Monthly Checklist (30 minutes, first of month)

- **Azure Blob inventory:** Verify ABS container has current-month blobs for all litestream paths. Query: `az storage blob list --container-name system-db-backups --prefix

"litestream/mission-control" --query "[].{name:name}" -o table` — confirm entries from current month.
- **DR recovery drill:** Restore `mission-control.db` from litestream ABS to `/tmp/mc-restore-test.db`. Verify row count matches live DB. `litestream restore -config ~/system/config/litestream.yml -replica mc-abs /tmp/mc-restore-test.db`
- **Cost model distribution audit:** `node ~/system/tools/cost-tracker.js task-summary` — identify top 5 cost-generating tasks. Are they justified? Is Opus being used where Sonnet is sufficient?
- **Azure VM backup verification:** SSH to `alai-admin@4.223.110.181` and verify BookStack/Vaultwarden data is intact. Until a VM-level backup policy is confirmed, this is the only assurance check for that data.
- **SSH key rotation check:** Verify `~/.ssh/azure_alai` key is documented in 1Password/Bitwarden and FORGE has a copy.
- **Orphaned /tmp cleanup:** `ls /tmp/alai-azure-backup- | *wc -l*` — *currently 32 orphaned dirs from failed backup runs. These accumulate indefinitely. Add `rm -rf /tmp/alai-azure-backup-`* to a weekly cron or the backup script's cleanup phase.

Top 5 Things Broken Right Now

1. CRITICAL — legal-docs-azure-sync has never worked Exit 512. Azure `BlobAlreadyExists` error — script lacks `--overwrite` flag. 84/84 legal documents (corporate, signed, insurance) have never been uploaded to Azure cold storage. The script also references a non-existent ISBD directory. ZAKON ARCHIVE FIRST is being violated for ALAI's most important documents. Fix: add `--overwrite` to `az storage blob upload-batch` call. File: `/Users/makinja/system/daemons/legal-docs-azure-sync.sh` **2. HIGH — Azure VM has no ANVIL-side backup** BookStack (wiki), Vaultwarden (password vault), Documenso, Grafana, and Planka run on `4.223.110.181`. No backup mechanism targeting this VM was found in `litestream.yml`, `azure-db-backup.sh`, or `dr-sync.sh`. If the Azure VM is deleted or corrupted, operational knowledge base and all vault passwords are gone. Fix: add VM-level Azure Backup policy, or SSH-pull volumes nightly from ANVIL. **3. HIGH — $129K/day cost is invisible to CEO until morning digest** The `cost-daily-report` writes to `~/system/reports/cost-daily.md` at 23:55 but never sends to Slack. The morning digest includes a one-line cost mention. If a runaway Opus loop fires at midnight, it won't be seen until 08:00 the next day — 8+ hours of potential unchecked spend. Fix: add a `cost-spike-alert.sh` triggered when any hour exceeds a threshold (e.g., $10K/hour), piped to `slack.js send exec`. **4. MEDIUM — infra-drift-detector fails every run (git rename limit)** Every scheduled run fails because the underlying git repo has 3178+ renamed files and hits `diff.renameLimit`. The fix is a one-liner: `git config diff.renameLimit 0` in the script before the commit, or setting `SAFE_COMMIT=false` when only Brewfile drift is detected. Until fixed, drift detection is completely non-functional — the fleet could accumulate unauthorized plist/Brewfile changes without detection. File: `/Users/makinja/system/logs/infra-drift-detector.out` — error confirmed. **5. MEDIUM — ops-watchdog false alarms are drowning real signals** `public-docs` and `public-vault` have been showing "Service down" every 2 minutes for at least 10 consecutive cycles (counter shows `fail 10/2`) when both services are actually live (HTTP 302). The watchdog script does not follow redirects. This is the boy-who-cried-wolf problem — when a real outage happens, it is invisible in a stream of false positives. Fix: add `--location` or `--max-redirs 1` to the curl check, or accept 3xx

as healthy. Also: `lumiscare.no` shows `fail 2500/2` — that is a genuine outage, not false-positive.

Bonus Observations

Slack `#ops` channel missing: `dr-sync.sh` calls `node ~/system/tools/slack.js send ops "..."` at the end of every run. The channel `#ops` does not exist, generating `ERROR: Channel #ops not found` in every dr-sync log. The sync itself is fine; only the notification is silently dropped. Fix: change to `#exec` or create `#ops`. **lightrag-auto-heal not bootstrapped:** Plist exists but not loaded in launchd session. The daemon designed to auto-recover LightRAG cannot run. If LightRAG goes down, the watchdog that would heal it is itself not watching. **archive-first-ledger.jsonl is empty:** `/Users/makinja/system/state/archive-first-ledger.jsonl` has 0 bytes / 0 lines. The scan correctly identifies 55 unarchived candidates (including signed corporate PDFs 50-76 days old). Zero have been archived. ZAKON ARCHIVE FIRST has been detecting violations since 2026-04-29 but no archiving action has been taken. **32 orphaned azure backup temp dirs in /tmp:** `/tmp/alai-azure-backup-20260422` *through* `*20260430*` — stalled partial backup runs accumulate and are never cleaned. Each is ~700MB (lightrag + neo4j volumes). Total: potentially 20+ GB of temp data. Add `trap cleanup EXIT` to the backup script. **Context usage JSONL is 34MB and growing:** `/Users/makinja/system/logs/context-usage.jsonl` — 34.4MB at last modification 22:02Z. This file likely grows unbounded. Check if `com.john.log-rotate` covers it.

Report generated by T5 Kelsey Hightower for Petter Graff (Chief Architect, T1 Synthesis). All findings are evidence-based — no memory assertions.