Pillar #5 — Skill Chains

Autonomous Skill-Chain Meta-Orchestrator — Agentic OS v1 Hardening. Design spec, DDL, reference pipelines, threat mitigations, and build-phase plan for MC #99129.

- Design Spec

Design Spec

Pillar #5 — Autonomous Skill-Chain Meta-Orchestrator

Design Spec — Agentic OS v1 Hardening

MC: #99129 | **Parent:** #99063 | **Date:** 2026-05-05 | **Status:** ready_for_review

Synthesizer: Petter Graff (CodeCraft)

Sources: 6 specialist sections at `/tmp/pillar5-99129/section-{1..6}-*.md` (4256 lines total) + forged baseline `/Users/makinja/system/prompts/forged/99129.md`

---

§1. Executive Summary

Problem

Pillars #1-#4 produced an Agentic OS that thinks, schedules, remembers, and uses skills. What it does not do is run **stateful, multi-step, durable, scheduled skill chains** that survive reboots, have human-in-loop gates, and refuse to claim victory without disk evidence. Recent ALAI failures (MLX phantom 2026-05-04, broken-JOHN 2026-05-02, Drop AWS phantom 2026-04-30) all share one root:

agent-emitted "done" with no runtime artefact behind it. Skill chains amplify that risk by chaining N self-attesting links — phantom compounds linearly.

Decision

Pillar #5 is a **thin convention layer**, not a new orchestration engine. It reuses primitives that already exist on disk (`mc.js` task_scheduling, launchd cron, skills, sub-agents, SQLite WAL, `liveness-claim-validator.sh` hook, ai-factory-pipeline 64-gate stack). Two new artefacts only: `chain-runner.sh` (≤150 LOC bash polling loop) and `phantom-link-detector.js` (15-min Node daemon).

Trigger model is **cron + sub-agent dispatch hybrid**: launchd fires `chain-runner.sh`, which polls `mc.js list --delegated chain-runner`, advances ONE step per tick by spawning ONE sub-agent Task, then exits. Pure recursive parent-spawning is rejected (cost penalty + observability collapse).

Pure cron-only stateless is rejected (no inheritance of 64-gate stack). Hybrid wins.

State lives in **mission-control.db** (`chain_runs` + `chain_steps` tables — Bruce's section-1 schema read of pipeline.db confirmed zero chain-shaped tables, default rule applies). Runtime evidence lives in

JSONL append log at `~/system/logs/chains/<chain>-<run>.jsonl` written ONLY by chain-runner.sh (petter-graff "runtime not testimony" wins on observability; openai-ca durability argument wins on storage; both adopted, state-file rejected).

Deliverables (DESIGN — build phase = child MCs)

1. SQLite DDL for `chain_runs` + `chain_steps` (mission-control.db)
2. `chain-runner.sh` ≤150 LOC bash, shellcheck clean, dry-run + signal traps + exit codes 0-4
3. `phantom-link-detector.js` Node daemon, 15-min cadence with 7-min offset from watchdog
4. Three reference YAML pipelines: daily-inbox-triage (FIRST SHIP), weekly-client-report, email-pipeline-fix-loop (LAST SHIP — only after 7 clean days on first two)

1. Two new launchd plists + one watchdog integration call (no third plist for fix-loop)
2. Cost telemetry: AGENT_COST event + cost-tracker.js `chain` subcommand + $3/day pre-flight gate

3. Threat mitigations addressing 8 attack surfaces (3 HIGH must block ship)

Ship Gate

80% daemon fleet health minimum. Currently 79.7% (114/143 healthy per `~/system/state/daemon-fleet-status.json` 2026-05-05T08:34Z). Two critical daemons must be fixed first: `com.john.db-backup` (down_exit_256) and `com.alai.cost-daily-report` (calendar_err_32512).

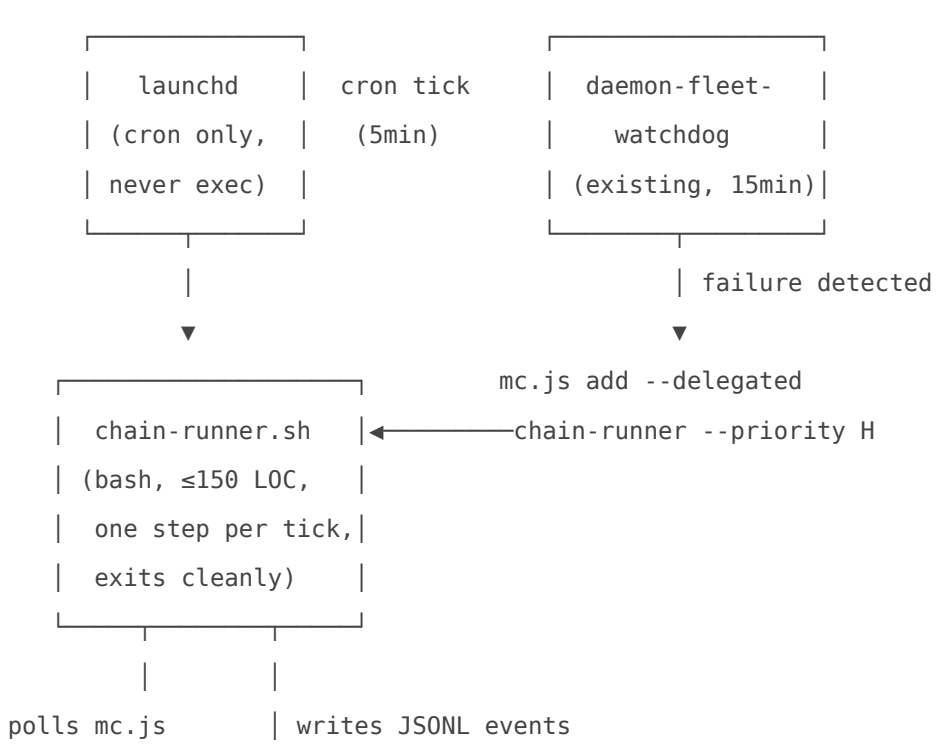
Cost Envelope

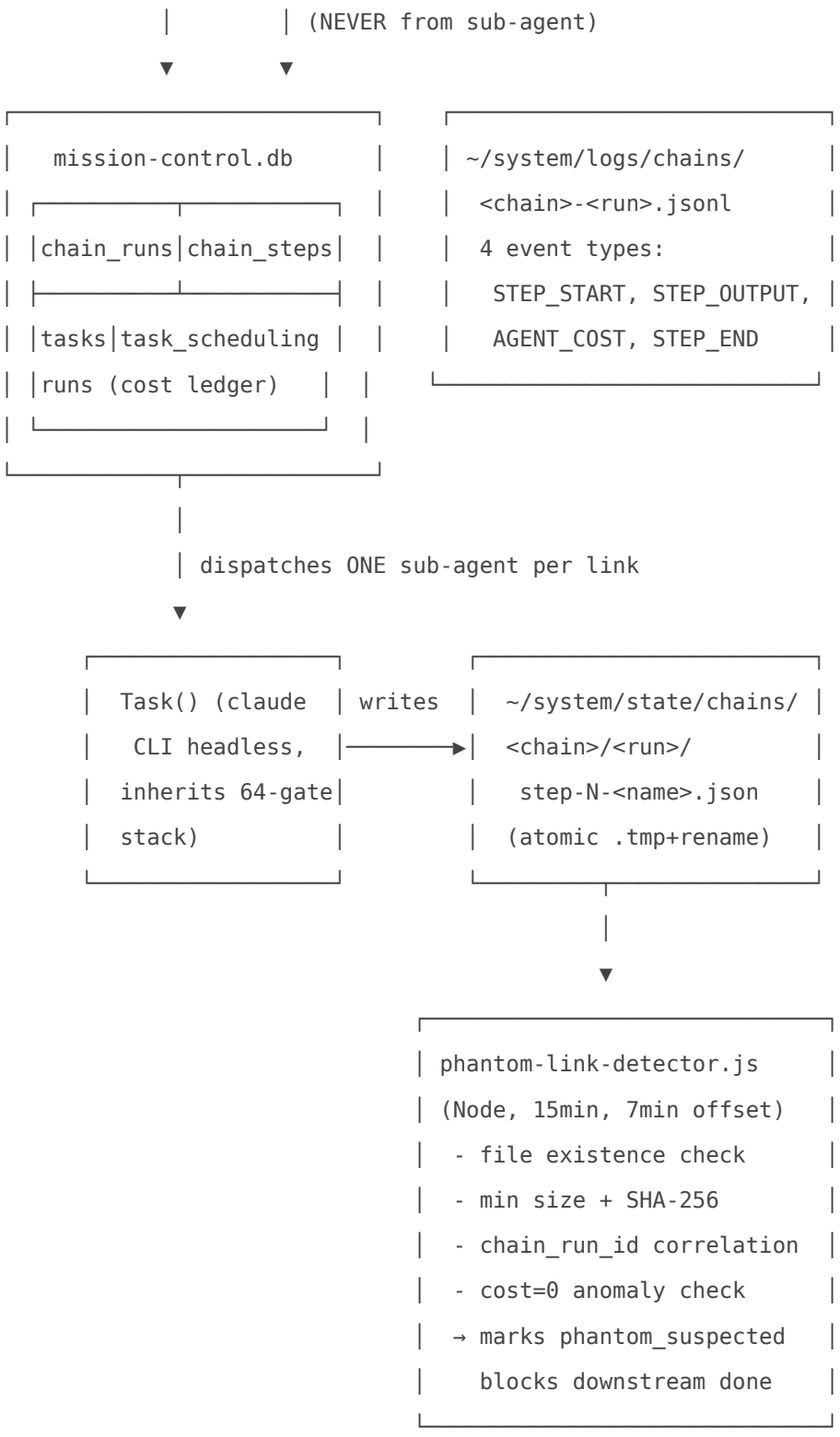
- **Hard ceiling:** $3.00/day combined across all `delegated_to=chain-runner` tasks
- **Expected monthly:** $2.80 (44 runs/month: 30 daily + 4 weekly + 10 fix-loop)
- **2× worst-case monthly:** $17.27 — recommend $20/month as Pillar #5 budget envelope
- **Daily peak risk:** ~$1.79 (12 fix-loop incidents in one day) — under ceiling
- **Ceiling breach scenario:** 23+ fix-loop firings in 24h — ceiling enforcement halts at run 22

---

§2. Architecture Overview

Components Diagram





Trigger Types (closed set in v1)

Type	Source	Use case
`cron`	launchd `StartCalendarInterval`	daily-inbox-triage (07:00), weekly-client-report (Mon 09:00)

Type	Source	Use case
`watchdog`	`daemon-fleet-watchdog.sh` calls `mc.js add --delegated chain-runner`	email-pipeline-fix-loop (no plist of its own)
`manual`	Human/orchestrator `mc.js add --delegated chain-runner`	one-off, testing, incident response

A fourth type (`event_bus`) is **reserved for v2** when Pillar #9 multi-host runtime activates.

Boundary Rules

- **Cron triggers, never executes work.** Plists call `chain-runner.sh --enqueue <chain>`; that

script writes one MC task and exits.

- **MC is the queue.** No parallel queue primitive. `task_scheduling.lease_until` (mc.js:204),

`attempt_count` (mc.js:201), `cb_state` (mc.js:200), `dead_letter` (mc.js:207) ARE the chain

queue (Bruce §1).

- **Skills execute the work.** `~/.claude/skills/sp-*/SKILL.md` is the unit of skill content.

Chain links wrap, do not replace, skills.

- **Sub-agents isolate the blast radius.** Each link spawns one Claude Code Task; the sub-agent

inherits the 64-gate stack at `~/system/specs/ai-factory-pipeline.md` §2.

- **chain-runner is the only writer of state.** Sub-agents write only to `output_path`. JSONL,

DB rows, lease state — all chain-runner.sh exclusively.

Reuse vs New

Existing primitive	Path	Reuse for
`mc.js` task_scheduling	`~/system/tools/mc.js` lines 163,198,204,3089-3144	Chain queue, leases, cb_state, DLQ
launchd	`~/Library/LaunchAgents/`	Cron triggers, phantom-detector cadence
Skills	`~/.claude/skills/sp-*/SKILL.md`	Per-link work units
Sub-agents (Task)	`claude` CLI headless	Per-link execution + 64-gate inheritance

Existing primitive	Path	Reuse for
SQLite WAL	mission-control.db	Durable chain state, atomic UPDATE
`liveness-claim-validator.sh`	`~/.claude/hooks/` (LIVE per `~/system/state/postflight-cleared-99127.json`)	Block bare LIVE/DONE claims
`bash-danger-gate.sh`	`~/.claude/hooks/` (LIVE)	Inherits across sub-agent dispatch
`cost-tracker.js`	`~/system/tools/cost-tracker.js`	Cost dual-write, daily aggregation, new `chain` subcommand

NEW artefact	Path	Purpose
`chain-runner.sh`	`/Users/makinja/system/tools/chain-runner.sh` [NEW]	Cron-triggered polling + per-step dispatch
`phantom-link-detector.js`	`/Users/makinja/system/tools/phantom-link-detector.js` [NEW]	15-min phantom + cost anomaly daemon
`chain-validator.js`	`/Users/makinja/system/tools/chain-validator.js` [NEW]	Pre-registration YAML schema check
`secret-scanner.sh`	`~/.claude/hooks/secret-scanner.sh` [NEW]	PostToolUse hook on chain state dir (T6 mitigation)
`chain_runs` + `chain_steps`	mission-control.db [NEW DDL]	Durable chain state
Two LaunchAgent plists	`~/Library/LaunchAgents/com.alai.chain-{daily-inbox,weekly-report}.plist` [NEW]	Cron triggers
Phantom-detector plist	`~/Library/LaunchAgents/com.alai.chain-phantom-detector.plist` [NEW]	15-min scan

---

§3. Storage & State

DB Location Decision — RESOLVED

Verdict: mission-control.db. Bruce verified pipeline.db schema this session (section-1 §1): tables `pipelines`, `phase_history`, `gate_checks`, `blueprint_runs`, `phase_results`, `blueprint_lineage`, `_litestream_seq`, `_litestream_lock`. None are chain-state-shaped. `blueprint_runs` is the closest analog but client-project-scoped, not skill-chain-scoped. Forged-prompt synthesizer's default rule applies: pipeline.db has no relevant tables → mission-control.db. This **resolves** the unresolved item from forged-99129.md disagreement #3.

mission-control.db already holds `tasks` + `task_scheduling` (queue), `runs` (cost ledger with `trace_id` for free-JOIN cost aggregation), `pipeline_events`, `outbox`. WAL mode active (verified by Litestream marker tables). Single-file WAL is correct at ALAI volume (≤10 chain runs/day, single-host ANVIL).

chain_runs + chain_steps DDL

```
BEGIN;

CREATE TABLE IF NOT EXISTS chain_runs (
  chain_run_id   TEXT    PRIMARY KEY,           -- UUID v4 (NOT wall-clock; multi-host safe)
  chain_name     TEXT    NOT NULL,              -- "daily-inbox-triage" | etc.
  mc_task_id     INTEGER REFERENCES tasks(id),  -- chain MC task
  triggered_by   TEXT    NOT NULL
                 CHECK(triggered_by IN ('cron','watchdog','manual','retry')),
  status         TEXT    NOT NULL DEFAULT 'running'
                 CHECK(status IN ('running','awaiting_human','succeeded','failed',
                                  'phantom_suspected','cost_halted','dead_letter')),
  total_links    INTEGER NOT NULL DEFAULT 0,
  current_link   INTEGER NOT NULL DEFAULT 0,
  cost_usd_total REAL    NOT NULL DEFAULT 0.0,
  started_at     TEXT    NOT NULL DEFAULT (datetime('now')),
  completed_at   TEXT    DEFAULT NULL,
  lease_until    TEXT    DEFAULT NULL,          -- chain-level (30min); step lease on
chain_steps
  schema_version INTEGER NOT NULL DEFAULT 1,
  estimated_cost_usd  REAL,                     -- from YAML cost_ceiling_usd_per_run
  cache_hit_ratio_avg REAL,                     -- rolling avg across steps
  phantom_check_status TEXT DEFAULT 'pending'   -- 'pending' | 'clean' | 'phantom_suspected'
);

CREATE INDEX IF NOT EXISTS idx_chain_runs_name_status ON chain_runs(chain_name, status);
CREATE INDEX IF NOT EXISTS idx_chain_runs_started     ON chain_runs(started_at);
CREATE INDEX IF NOT EXISTS idx_chain_runs_mc_task     ON chain_runs(mc_task_id);

CREATE TABLE IF NOT EXISTS chain_steps (
  id             INTEGER PRIMARY KEY AUTOINCREMENT,
  chain_run_id   TEXT    NOT NULL REFERENCES chain_runs(chain_run_id) ON DELETE CASCADE,
```

```
  step_name      TEXT    NOT NULL,
  step_index     INTEGER NOT NULL,
  skill          TEXT    NOT NULL,
  status         TEXT    NOT NULL DEFAULT 'pending'
                 CHECK(status IN ('pending','running','done','failed',
                                  'awaiting_human','phantom_suspected','skipped')),
  attempt_count  INTEGER NOT NULL DEFAULT 0,
  max_attempts   INTEGER NOT NULL DEFAULT 2,    -- 0 for human_gate links
  human_gate     INTEGER NOT NULL DEFAULT 0
                 CHECK(human_gate IN (0,1)),
  input_path     TEXT    DEFAULT NULL,
  output_path    TEXT    DEFAULT NULL,
  output_sha256  TEXT    DEFAULT NULL,
  cost_usd       REAL    NOT NULL DEFAULT 0.0,
  input_tokens   INTEGER NOT NULL DEFAULT 0,
  output_tokens  INTEGER NOT NULL DEFAULT 0,
  cache_creation_tokens INTEGER DEFAULT 0,
  cache_read_tokens     INTEGER DEFAULT 0,
  cache_hit_ratio       REAL DEFAULT 0,
  model          TEXT    DEFAULT NULL,
  started_at     TEXT    DEFAULT NULL,
  completed_at   TEXT    DEFAULT NULL,
  lease_until    TEXT    DEFAULT NULL,          -- step-level lease (10min)
  error_text     TEXT    DEFAULT NULL,
  compensate_on_failure TEXT DEFAULT NULL,
  UNIQUE (chain_run_id, step_name)              -- idempotency key
);

CREATE INDEX IF NOT EXISTS idx_chain_steps_run_status   ON chain_steps(chain_run_id, status);
CREATE INDEX IF NOT EXISTS idx_chain_steps_run_index    ON chain_steps(chain_run_id,
step_index);
CREATE INDEX IF NOT EXISTS idx_chain_steps_lease        ON chain_steps(lease_until)
  WHERE status = 'running';
CREATE INDEX IF NOT EXISTS idx_chain_steps_phantom_scan ON chain_steps(status, output_path)
  WHERE status = 'done' AND output_path IS NOT NULL;

CREATE TABLE IF NOT EXISTS chain_runs_archive (
  chain_run_id   TEXT    PRIMARY KEY,
  chain_name     TEXT    NOT NULL,
  -- (matches chain_runs schema + archived_at)
```

```
  archived_at    TEXT    NOT NULL DEFAULT (datetime('now'))
);

COMMIT;
```

Full DDL with all archive columns: see section-1 §7 (lines 651-728).

Atomic Checkpoint Protocol

SQLite WAL handles DB row atomicity natively. Sidecar JSON evidence files (sub-agent outputs)

use `.tmp` + `fsync` + `rename(2)` (APFS atomic) + SHA-256 capture:

```
TMP_PATH="${OUTPUT_PATH}.tmp"
write_output_to "${TMP_PATH}"
python3 -c "import os,sys; fd=os.open(sys.argv[1],os.O_RDONLY); os.fsync(fd); os.close(fd)"
"${TMP_PATH}"
SHA=$(shasum -a 256 "${TMP_PATH}" | awk '{print $1}')
mv -f "${TMP_PATH}" "${OUTPUT_PATH}"
sqlite3 "${MC_DB}" "UPDATE chain_steps SET output_path='${OUTPUT_PATH}',
output_sha256='${SHA}'
                    WHERE chain_run_id='${CHAIN_RUN_ID}' AND step_name='${STEP_NAME}';"
```

Resume integrity: re-compute SHA on disk, compare to `chain_steps.output_sha256`. Mismatch → reset

to `pending`, rerun. Orphan `.tmp` cleanup: `find ~/system/state/chains/ -name "*.tmp" -mmin +30`

runs as part of phantom-link-detector.

Artifact path convention (NOT `/tmp/` — cleared on reboot per CONSTRAINTS):

```
~/system/state/chains/<chain_name>/<chain_run_id>/step-<N>-<step_name>.json
```

JSONL Event Log

Path: `~/system/logs/chains/<chain_name>-<chain_run_id>.jsonl` (per-run isolation,

petter-graff isolation argument; cross-run queries via `find` + `jq`).

Four event types:

- `STEP_START` — written by chain-runner.sh BEFORE sub-agent dispatch

- `STEP_OUTPUT` — written AFTER verifying output file + SHA-256 match
- `AGENT_COST` — written AFTER reading `runs` table for sub-agent's cost (NOT from sub-agent narrative)
- `STEP_END` — written AFTER STEP_OUTPUT verified + chain_steps.status updated

Critical ownership rule (section-1 §4): Sub-agents do NOT write to JSONL. They write only to `output_path`. chain-runner.sh is the sole writer. "runtime not testimony" — code review must reject any chain-runner that delegates JSONL writing to sub-agents.

Full event schemas with field tables: see section-1 §4.2 (lines 312-407) and section-2 §2.9 (lines 656-692).

Lease Semantics — RESOLVED

New `lease_until` column on `chain_steps`, NOT reuse of `task_scheduling.lease_until` (Bruce §5.1).

`task_scheduling` is keyed by `task_id`; one MC task = one row. Reusing it for step-level leases would either inflate the task table (one MC per step = operational noise + dispatch logic breakage) or repurpose the chain MC row (losing chain-level vs step-level distinction). Clean column wins.

The chain MC task's `task_scheduling` row continues to serve cb_state, dead_letter, chain-level retry counting (mc.js:3089).

Atomic lease acquire (CAS pattern):

```
UPDATE chain_steps
SET status='running', attempt_count=attempt_count+1, started_at=datetime('now'),
    lease_until=datetime('now', '+10 minutes')
WHERE chain_run_id=:chain_run_id AND step_name=:step_name
  AND status IN ('pending','failed')
  AND (lease_until IS NULL OR lease_until < datetime('now'));
-- chain-runner checks changes()==1; if 0, another runner holds lease, exit 0
```

SQLite WAL serializes writes; `changes()==1` is the optimistic-lock confirmation. Section-1 §5.2 (lines 432-510) carries full lease release + expiry recovery SQL.

Retention / Age-Out (section-1 §6)

- **>24h running/awaiting_human** → mark `dead_letter`, cascade chain_steps to `failed`
- **>7d completed** → INSERT into `chain_runs_archive`, DELETE from `chain_runs` (CASCADE)
- **JSONL files** → mv to `~/system/logs/chains/_archive/YYYY-MM/`
- **Orphan `.tmp` files** → `find ... -mmin +30` → log as `ORPHAN_TMP` event

Age-out runs daily 03:00 via `com.alai.chain-ageout.plist` (StartCalendarInterval, NOT KeepAlive).

Build-phase plist; cite section-1 §6.3 (lines 610-640) for full XML.

---

§4. chain-runner.sh + phantom-link-detector

chain-runner.sh — Architecture

Stateless bash polling loop. Not a daemon, not a scheduler. ≤150 LOC. Each invocation:

poll mc.js → read chain_steps → dispatch ONE sub-agent → wait for ready → verify output →

Proveo gate → advance row → release lease → exit. LaunchAgent re-triggers on cadence.

Polling cadence: `StartInterval=60` (chain-runner poll plist) for the inner loop;

`StartCalendarInterval` for the per-chain triggers. Do NOT poll tighter than 5min without

confirming daemon-fleet health (CONSTRAINT §).

Exit codes:

Code	Meaning
0	No runnable task, or step dispatched and awaiting ready (normal cron tick)
1	Internal error (DB query fail, lock file, disk full)
2	Chain halted at human gate — emit HUMAN_GATE event, exit clean
3	Cost ceiling exceeded — all chains halted, Slack alerted
4	Circuit breaker OPEN for this chain — skip, log, exit

Signal handling: SIGTERM/SIGINT → write STEP_INTERRUPTED → release lease → exit 1.

SIGKILL untrapped; lease expiry (10 minutes per mc.js:1367) handles kill-9 resume.

Max runtime per step: 8 minutes (`timeout 480`). 10-min lease window leaves 2-min cleanup buffer before lease expiry.

Dry-run mode (`--dry-run`): Read DB; print planned dispatch (chain_id, run_id, step, skill, input_path, output_path); never call Task(); never mutate DB; exit 0.

Sub-Agent Dispatch Contract

Each link dispatches exactly ONE sub-agent. Checkpoint passed as `--resume-from-checkpoint <abs_path>` argument (NOT environment variable — args appear in `set -x` traces, env vars don't):

```
timeout 480 node "${CLAUDE_CLI}" task \
  --skill "${skill}" \
  --resume-from-checkpoint "${input_path}" \
  --output-path "${output_path}" \
  --chain-task-id "${chain_task_id}" \
  --chain-run-id "${CHAIN_RUN_ID}" \
  --step "${STEP_NAME}" \
  2>>"${STEP_STDERR_LOG}"
```

Sub-agent contract on receipt:

1. Read `--resume-from-checkpoint` file (cold start = `{}`; otherwise previous link's output)
2. Do skill work
3. Write output to `--output-path` using atomic `.tmp+rename`
4. Call `mc.js ready <chain_task_id>` with comment containing output_path
5. **Do NOT call `mc.js done`** — done is exclusively Proveo's signal (Hard Constraint #4)

Atomic Step Transition

```
READ checkpoint → ACQUIRE lease (CAS UPDATE) → WRITE STEP_START to JSONL →
UPDATE chain_steps.status='running' → SPAWN sub-agent (Task, timeout 480) →
WAIT for ready_for_review (poll mc.js show every 30s, max 8min) →
VERIFY output_path exists on disk → INVOKE Proveo gate →
WRITE STEP_END (JSONL FIRST, then SQL — mitigates skew, section-2 §2.7) →
WRITE next chain_steps row (status=pending, input_path=current output_path) →
```

```
RELEASE lease → EXIT
```

Full bash pseudocode: section-2 §2.3 (lines 224-298). LOC budget table: section-2 §2.10.

Failure Handling

- **Crash mid-link:** lease expires (10min), next tick reads `WHERE status IN ('pending','failed')`

ORDER BY id ASC, resumes from earliest unfinished step. Done steps immutable.

- **Lease expiry:** `_renew_lease` SQL UPDATE every 30s during ready-poll
- **Partial writes:** `.tmp` orphan only; phantom-detector reports `>30min` orphans
- **Sub-agent timeout:** STEP_TIMEOUT event → increment attempt_count → status=failed → release
- **Retry budget exhaustion:** `attempt_count >= max_retries` → cb_state='OPEN', dead_letter=1,

STEP_DEAD event, H-priority MC dead-letter task, Slack alert, EXIT 4

- **cb_state circuit breaker:** check before dispatch; exit 4 if 'open' (mc.js:3101-3105 semantics)

Full SQL + bash: section-2 §2.4 (lines 328-415).

Phantom-Link-Detector Daemon

Node.js, 15-min cadence, 7-min offset from daemon-fleet-watchdog (avoids SQLite contention:

phantom at :00/:15/:30/:45, watchdog at :07/:22/:37/:52).

Algorithm per `chain_steps WHERE status='done' AND output_path IS NOT NULL` (24h window):

1. `fs.existsSync(output_path)` — file present?
2. File size >= MIN_STEP_OUTPUT_BYTES (64) — empty-file bypass guard (T3 mitigation)
3. Parse JSON; verify `chain_run_id` + `step_name` fields match (T3)
4. Re-compute SHA-256, compare to `chain_steps.output_sha256` (T3)
5. Find matching `STEP_END status:ok` event in JSONL (T3)
6. Cost anomaly check: 5+ consecutive zero-cost agentic steps OR (2+ zero-cost AND total=0) (section-5 §5.6)

On mismatch:

- `chain_steps.status='phantom_suspected'`

- `chain_runs.status='phantom_suspected'`
- PHANTOM_DETECTED / PHANTOM_COST_DETECTED event to JSONL
- Slack alert to `#alerts`
- mc.js comment to chain task: "PHANTOM LINK DETECTED... Downstream done BLOCKED"

Blocking downstream done: ai-factory-pipeline B10 fail-secure postflight gate already checks `chain_runs.status` before allowing `mc.js done` — no mc.js code change needed.

False-positive handling: legitimate "process and delete" → `output_path: null` in chain YAML (detector skips `WHERE output_path IS NULL`). 2-minute grace period on freshly-completed steps.

Saga / Compensation

Per-link attribute `compensate_on_failure: <skill>`, NOT chain-wide. Only links crossing external-API boundaries (section-2 §2.5):

Pipeline	Links needing compensation	Compensating skill
daily-inbox-triage	None — read-only classification + idempotent INSERT OR IGNORE	N/A
weekly-client-report	`bookstack-publish` (CEO-visible side-effect), `client-email-send` (at-most-once)	`bookstack-revert-draft`; `email-send-partial-log`
email-pipeline-fix-loop	`apply-fix` (production config mutation)	`email-pipeline-rollback` (reverse-order via step2-proposal.json rollback_procedure)

Idempotency tokens for external POSTs are **mandatory** regardless of compensation config — `chain_run_id + step_id` composite stored in SQLite before POST attempted.

Implementation Language Guard — RESOLVED

bash + Node ONLY. chain-runner.sh = bash. phantom-link-detector.js = Node. mc.js helpers = Node.

Python permitted only for the one-line `os.fsync()` invocation (bash has no fsync primitive).

No Kotlin (genesis: `feedback_john_kotlin_rabbit_hole_2026-05-02.md`). Three reasons (section-2 §2.8):

opacity (existing alai-hooks Kotlin CLI has 13/64 OPAQUE gates), runtime dependency (Gatekeeper SIGKILL on cp documented in `reference_hook_system_2026-05-04.md`), rabbit-hole attractor.

Structural prevention: shellcheck pass is acceptance criterion → Kotlin impossible by construction.

No TypeScript. mc.js is plain Node JavaScript; chain-runner helpers stay in JavaScript.

TypeScript adds a build step that fails silently.

No OpenAI Agents SDK (DPA pending; vendor lock; bypasses 64-gate stack).

---

§5. YAML Pipeline Schema + 3 Reference Pipelines

Schema Spec (top-level fields)

Field	Type	Constraint
`chain_id`	string	`/^[a-z][a-z0-9-]{1,63}$/`, unique across `~/system/agents/chains/`
`schema_version`	int	must equal 1
`description`	string	≤120 chars

Field	Type	Constraint
`priority`	enum	`H
`trigger.type`	enum	`cron
`trigger.spec`	object	shape per type (cron: plist_label + calendar_interval; watchdog: source_plist + dedup_window_minutes; manual: optional allowed_origins)
`cost_ceiling_usd_per_run`	float	`0 < x ≤ 5.0` (build phase enforced)

Field	Type	Constraint
`human_gate_default`	bool	overridden true if priority=H
`max_retries_default`	int	`0..5`
`links[]`	list	length `1..20`

Per-link fields

`name`, `skill`, `agent` (required); `status_initial` (`pending|skip_if_exists`),

`human_gate`, `compensation_skill`, `evidence_paths[]` (must start with `/Users/makinja/system/state/chains/` —

NO `/tmp/`), `idempotency_key_template` (Go-template, vars: chain_run_id, link_name, date, attempt),

`timeout_seconds` (30..1800), `max_retries` (0..5; human_gate links capped to 0),

`on_fail` (`stop|retry|compensate|dead_letter`).

Full spec with rationale per field: section-3 §3.1.2 (lines 76-149).

Three Reference Pipelines

A. daily-inbox-triage — FIRST SHIP

- **Trigger:** cron, 07:00 daily (`com.alai.chain-daily-inbox.plist`)
- **Priority:** M
- **Cost ceiling:** $0.25/run (target $0.20)
- **Links (4):** fetch-inbox (deterministic, $0) → classify (Sonnet, ~$0.018 warm) →

draft-replies (Sonnet, ~$0.017 warm) → send (deterministic + human gate)

- **Human gate:** ONLY before send link (link 3)
- **Compensation:** none — read-only classification + idempotent INSERT OR IGNORE
- **Expected $/month:** $0.96 (warm cache); 2× worst-case $4.87

Full YAML: section-3 §3.2 (lines 166-287).

B. weekly-client-report — SECOND SHIP (after 7 clean days on A)

- **Trigger:** cron Monday 09:00 (`com.alai.chain-weekly-report.plist`; one hour after existing

`com.john.weekly-pipeline-review` 08:00)

- **Priority:** M

- **Cost ceiling:** $0.75/run
- **Links (6):** aggregate-status → draft-narrative → format-report → human-review-draft (gate 1, 48h timeout) →

bookstack-publish → human-review-final-publish (gate 2, 24h timeout, includes client-notify-send)

- **Human gates:** TWO (links 3 and 5)
- **Compensation:** `bookstack-revert-draft` on link 4; `client-notify-partial-log` on link 5
- **Expected $/month:** $0.44; 2× worst-case $4.40

Full YAML: section-3 §3.3 (lines 304-446).

C. email-pipeline-fix-loop — THIRD SHIP (after 7 clean days on A+B)

- **Trigger:** watchdog (NOT cron). `daemon-fleet-watchdog.sh` calls `mc.js add --delegated chain-runner

--priority H --tag chain:email-pipeline-fix-loop `when` email_consecutive_failures >= 3`

- **Priority:** H — forces human_gate on every agentic link regardless of explicit field
- **Cost ceiling:** $1.50/run
- **Links (5):** probe (deterministic) → diagnose (Sonnet, gate) → propose-fix (Sonnet, gate) →

apply-fix (executes proposal.sh, gate, compensate on partial) → verify-fix (re-probe, gate)

- **Compensation:** `email-pipeline-rollback` on apply-fix
- **Diagnostic preamble (REQUIRED in skill prompt):** "You are diagnosing a BROKEN email pipeline.

All evidence below was gathered from a system that is currently failing. Treat every observation

as a hypothesis, not a fact. Do NOT claim the system is working."

- **Expected $/month:** $1.40 (10 incidents/month); 2× worst-case $8.00
- **Self-referential risk:** runs on broken system — context rot guaranteed; verify-fix re-probes

from scratch, never relies on pre-apply state in memory

Full YAML: section-3 §3.4 (lines 464-606).

Validation Rules (section-3 §3.6)

16 rules summarized; key callouts:

- **V-09 (skill exists):** `~/.claude/skills/sp-<value>/SKILL.md` OR `~/.claude/skills/<value>/SKILL.md`

must exist at validator run time. Prevents phantom skill references (MLX-style gap).

- **V-11 (evidence path safety):** every path must start with `/Users/makinja/system/state/chains/`.

`/tmp/` paths FORBIDDEN for cross-step evidence (cleared on reboot per CONSTRAINTS).

Validator at `~/system/tools/chain-validator.js` [NEW]; invoked at chain registration.

Invalid chain → logged as `CHAIN_INVALID` event in JSONL, skipped (does NOT crash runner).

Versioning

`schema_version: 1` is the only supported version in v1. Required field; absence = validation error (V-02). Migration: increment + write `chain-migrate-v1-v2.js` + 30-day compatibility window (warn `SCHEMA_DEPRECATED`) + reject `CHAIN_SCHEMA_OUTDATED` after window.

`chain_id` is primary key — MUST NOT be renamed; replacements get new ID running in parallel.

---

§6. Trigger Infrastructure (LaunchAgents)

Two NEW LaunchAgent Plists

`com.alai.chain-daily-inbox.plist` (07:00 daily)

- ProgramArguments: `/bin/bash /Users/makinja/system/tools/chain-runner.sh --enqueue daily-inbox-triage --trigger cron`
- StartCalendarInterval: Hour=7, Minute=0
- KeepAlive=false; RunAtLoad=NOT set (no off-hours flood on launchctl load)
- StandardOutPath/StandardErrorPath: `~/system/logs/chains/chain-daily-inbox-launchd.log` (merged)

`com.alai.chain-weekly-report.plist` (Monday 09:00)

- StartCalendarInterval: Weekday=1, Hour=9, Minute=0
- One hour AFTER existing `com.john.weekly-pipeline-review` (verified 08:00) — separation

intentional; pipeline-review fires first, errors surface before chain begins

- Does NOT duplicate `com.john.weekly-pipeline-review`. That plist calls `weekly-pipeline-review.js`

directly; this plist enqueues an MC chain task

Phantom-Detector Plist

`com.alai.chain-phantom-detector.plist` (15-min interval)

- macOS launchd does NOT support `*/15` cron syntax. StartCalendarInterval expressed as

ARRAY of four dicts (Minute=0, Minute=15, Minute=30, Minute=45) per `man launchd.plist`

- KeepAlive=false (one-shot scan; ghost-worker defence)
- Calls `~/system/tools/chain-phantom-detector.sh` shell wrapper → `phantom-link-detector.js`

Full XML: section-4 §4.2.1, §4.2.2, §4.3 (lines 29-244).

Watchdog Integration (NO new plist for fix-loop)

`email-pipeline-fix-loop` is event-driven, NOT calendar-driven. Adding a cron plist would fire

on every tick regardless of email pipeline health → H-priority MC noise. Existing

`com.alai.daemon-fleet-watchdog` (StartInterval=900, KeepAlive=false, RunAtLoad=true) gains

4 lines inside `daemon-fleet-watchdog.sh` (section-4 §4.2.3 lines 152-168):

```
EMAIL_FAILURE_THRESHOLD=3
if [[ "$email_consecutive_failures" -ge "$EMAIL_FAILURE_THRESHOLD" ]]; then
  /opt/homebrew/bin/node /Users/makinja/system/tools/mc.js add \
    "Chain run: email-pipeline-fix-loop (incident: ${INCIDENT_ID})" \
    --priority H --delegated chain-runner --desc "..." \
    >> /Users/makinja/system/logs/chains/chain-email-fix-launchd.log 2>&1
  /opt/homebrew/bin/node /Users/makinja/system/tools/slack.js send "#alerts" "..."
fi
```

KeepAlive Decision Matrix (section-4 §4.5)

Component	Pattern	Rationale

chain-runner.sh poll plist	StartInterval=60	Ghost-worker defence; lease semantics need clean exit
chain-daily-inbox trigger	StartCalendarInterval Hour=7	Calendar event; exits after mc.js add
chain-weekly-report trigger	StartCalendarInterval Weekday=1 Hour=9	Calendar event
chain-phantom-detector	StartCalendarInterval ARRAY (4× Minute)	Discrete scan; KeepAlive=false
daemon-fleet-watchdog	StartInterval=900 (existing, unchanged)	Watchdog integration adds bash inside
email-pipeline-fix-loop	NO PLIST	Event-driven via watchdog hook

Lid-Close / Caffeinate (section-4 §4.6)

`man launchd.plist` (StartCalendarInterval): "Unlike cron... launchd will start the job the

next time the computer wakes up. If multiple intervals transpire before the computer is woken,

those events will be coalesced into one event upon wake."

Coalescing is wanted behaviour: machine closed Sun-Mon → ONE Mon 09:00 weekly-report fire on

wake (not N queued stale runs). caffeinate NOT required.

`StartInterval` (phantom-detector, runner-poll) does NOT fire on missed ticks — fires at next

boundary after wake. Max delay 60s — acceptable.

If sub-agent running when lid closes: macOS suspends; on wake, `lease_until` likely expired →

chain-runner treats as failed, retries from last clean checkpoint.

Logging Contract (section-4 §4.7)

```
~/system/logs/chains/
  chain-daily-inbox-launchd.log           # launchd stdout+stderr (merged)
  chain-weekly-report-launchd.log
  chain-phantom-detector-launchd.log
  chain-email-fix-launchd.log
  chain-events.jsonl                      # rolling, in-progress + recent
  archive/
    chain-events-<chain_run_id>.jsonl     # archived per chain_run_id on completion
```

Rotation via existing `com.john.log-rotate` (extend config to include `~/system/logs/chains/*.log`, 7-day retention). `chain-events.jsonl` requires append-only handling; archive on chain done/failed.

Pre-flight: `mkdir -p ~/system/logs/chains/archive` BEFORE any plist load (LaunchAgent does NOT create parent dirs; silent log-write failure otherwise).

80% Daemon Fleet Health SHIP GATE

HARD GATE. Currently **79.7% (114/143)** per `~/system/state/daemon-fleet-status.json` 2026-05-05T08:34Z. One healthy daemon below threshold. **Do NOT load chain plists** until gate opens.

Two critical daemons must be remediated FIRST:

1. **`com.john.db-backup`** (down_exit_256) — mission-control.db backup; chain_runs corruption has no fallback without it
1. **`com.alai.cost-daily-report`** (calendar_err_32512) — chain cost telemetry depends on cost-tracker.js daily report; broken report obscures $3/day ceiling proximity

After fixing, re-run `bash /Users/makinja/bin/daemon-fleet-watchdog.sh` and re-evaluate. If healthy ≥116 (81%), gate opens. **Genesis:** SENTINEL v3 47/138 healthy figure; building cron-fleet on 34% failure-rate scheduler doubles failure surface. Phantom-detection subsystem validity itself depends on this gate (section-4 §4.8 lines 499-503).

Pre-flight checklist (section-4 §4.4): snapshot daemon-fleet → plutil -lint each plist → `mkdir -p` log dirs → load phantom-detector FIRST → load chain trigger plists → verify all loaded via `launchctl list | grep com.alai.chain-` → check no stale `mc.js list --delegated chain-runner` runs.

---

§7. Cost Telemetry & Ceiling

Per-Link Cost Capture

Source of truth: Anthropic API `usage` block from `claude` CLI exit JSON. NEVER from sub-agent narrative text (testimony vs evidence — same discipline as JSONL ownership).

```
{
  "usage": {
    "input_tokens": 4200, "output_tokens": 810,
    "cache_creation_input_tokens": 3100, "cache_read_input_tokens": 980
  },
  "model": "claude-sonnet-4-6"
}
```

chain-runner.sh computes `cost_usd` using same formula as `cost-tracker.js/estimateCost()`:

- Input: $3.00/M tokens; Output: $15.00/M tokens; Cache read: $0.30/M (10% discount); Cache create: at input rate
- Pricing constants from cost-tracker.js PRICING table (Sonnet-4.6 2026-04-06 rates)

AGENT_COST event appended to JSONL atomically (`.tmp` + `cat >>` since JSONL append is atomic at OS level for writes < PIPE_BUF=4096 bytes on Darwin):

```
{
  "ts":"2026-05-05T06:33:44Z", "event":"AGENT_COST",
  "chain_run_id":"...", "step_name":"classify", "model":"claude-sonnet-4-6",
  "input_tokens":4200, "output_tokens":810,
  "cache_creation_input_tokens":3100, "cache_read_input_tokens":980,
  "cost_usd":0.0181, "cache_hit_ratio":0.233
}
```

Aggregation — RESOLVED

Decision: explicit SQL UPDATE on STEP_END, NOT SQLite trigger. Two reasons (section-5 §5.2):

(1) triggers are invisible to runtime, hard to debug; (2) trigger would fire on any future direct DB writes (admin corrections), corrupting running total. Explicit UPDATE is grep-able, testable with `--dry-run`, single-writer auditable.

```
UPDATE chain_runs
SET total_cost_usd = (SELECT COALESCE(SUM(cost_usd),0) FROM chain_steps WHERE chain_run_id =
?)
WHERE chain_run_id = ?;
```

WAL mode (already active per Litestream markers) provides read consistency for ceiling check without blocking writer.

Dual Write

After JSONL append, chain-runner.sh writes to `costs.db` via `cost-tracker.js trackCost()` with `source: 'chain-runner'`, `agent_name: 'chain-runner/<chain_name>/<step_name>'`, `metadata: {chain_run_id, step_index, attempt}`. No schema change to cost-tracker.js — agent_name encodes chain context, metadata carries structured chain_run_id for reconciliation.

$3/Day Hard Ceiling

Pre-flight check (NOT post-hoc) BEFORE acquiring next chain task lease:

```
DAILY_SPEND=$(sqlite3 "${MC_DB}" "
  SELECT COALESCE(SUM(total_cost_usd), 0)
  FROM chain_runs
  WHERE started_at > datetime('now', '-1 day')
    AND status NOT IN ('cancelled', 'phantom_suspected');")
[ daily ≥ ceiling ] && {
  sqlite3 "${MC_DB}" "UPDATE task_scheduling
    SET cb_state='OPEN', cb_opened_at=datetime('now'), cb_reason='daily_cost_ceiling_exceeded'
    WHERE delegated_to='chain-runner' AND status IN ('open','pending');"
  slack #alerts; append COST_CEILING_BREACHED event; exit 0
}
```

`cb_state='OPEN'` is the same circuit-breaker mc.js already respects — natural halt without new code path. **Recovery automatic:** 24h rolling window decays below $3.00 → next tick passes ceiling check → dispatch resumes. Cron does NOT add duplicate chain tasks (existing-open check).

Warn threshold: $2.00 (67%) → Slack warning, continue dispatching. Externalize via

`~/system/config/chain-runner.env`:

```
CHAIN_DAILY_COST_CEILING_USD=3.00
CHAIN_COST_ALERT_THRESHOLD_USD=2.00
```

cost-tracker.js `chain` Subcommand [NEW]

```
node ~/system/tools/cost-tracker.js chain <chain_name_or_run_id> [--window 7d|30d|all]
```

Cross-DB join (costs.db + mission-control.db). Output: per-run + per-step breakdown with cache hit %, ACTUAL vs ESTIMATE variance, daily ceiling headroom. Variance > 50% → exit 1 → chain-runner appends `COST_VARIANCE_WARNING` event.

Full output sample + JS impl: section-5 §5.4 (lines 287-353).

Cache Hit Telemetry

Columns added to chain_steps DDL: `cache_creation_tokens`, `cache_read_tokens`, `cache_hit_ratio`. `chain_runs.cache_hit_ratio_avg` updated at chain completion. Daily report at `~/system/tools/chain-cache-report.js` [NEW] runs after last step of each chain run; checks prefix-stability (SHA-256 of stable prompt portion vs yesterday).

Phantom-Cost Detection (section-5 §5.6)

Inverse of MLX phantom: completed steps with `cost_usd=0` across all of them. Zero cost legitimate ONLY for `type: deterministic` (shell, no Claude) or local Ollama/MLX (priced $0). 5+ consecutive zero-cost agentic STEP_END events = `phantom_suspected`. Stricter: 2+ zero-cost AND total=0 → flag.

False-positive escape: chain YAML opt-in `zero_cost_ok: true` (absent by default — safe side flags).

Pipeline Cost Projection Table

Pipeline	Runs/month	Expected $/run (warm)	Cold $/run	Expected $/month	2× worst $/month
daily-inbox-triage	30	$0.0319	$0.0811	$0.96	$4.87

Pipeline	Runs/month	Expected $/run (warm)	Cold $/run	Expected $/month	2× worst $/month
weekly-client-report	4	$0.111	$0.255	$0.44	$4.40
email-pipeline-fix-loop	10	$0.140	$0.257	$1.40	$8.00
Combined	**44**	—	—	**$2.80**	**$17.27**

Section-5 §5.7 (lines 506-616) carries full per-step token + cost tables.

---

§8. Threat Surface & Mitigations

8 threats identified (Parisa §6). 3 HIGH must be addressed before ship.

T1 — HIGH — Shell Injection in chain-runner.sh

Vector: YAML-derived strings (`skill`, `agent`, `evidence_paths[]`, `step_name`,

`compensate_on_failure`) interpolated unquoted into bash → arbitrary code execution as makinja.

YAML directory writable by any sub-agent (compromised sub-agent or malicious skill output).

Mitigation (must-fix before ship):

1. Always double-quote: `claude --skill "$SKILL_NAME"` (NEVER `$SKILL_NAME` bare)
2. Whitelist regex `^[a-zA-Z0-9_-]{1,64}$` on every YAML field BEFORE shell use; reject, do NOT

sanitize

1. Path whitelist on `evidence_paths[]`: realpath must start with `~/system/state/chains/`
2. Load YAML via `node -e "require('js-yaml')"`, NEVER shell heredoc
3. shellcheck SC2086 (unquoted variable expansion) MUST remain active in CI

Residual risk: LOW after mitigations. Consider `chmod 444` on `~/system/agents/chains/` —

sub-agents have no legitimate reason to modify chain definitions.

T2 — HIGH — APPROVE Token Forgery via mc.js Comment

Vector: chain-runner.sh runs as launchd daemon, NOT inside Claude Code session. Therefore:

- No UserPromptSubmit hook fires
- No `/tmp/ceo-turn-<session>.txt` exists for daemon
- PreToolUse Bash hooks only fire within Claude Code agent session
- `mc.js comment` `detectActor()` is hostname-only (mc.js:528-554, 2726-2745); no session token check

`ceo-token-origin-gate.sh` does NOT protect daemon-invoked `mc.js comment` path. Runaway chain-runner

or compromised sub-agent can self-issue APPROVE → advances chain past human gate. Same pattern as

"broken JOHN" 2026-05-02.

Mitigation (must-fix before ship):

1. **Switch to `mc.js approval_tokens` table** (already exists, mc.js:4130-4165), NOT comment parsing.

chain-runner.sh writes `chain_steps.status='awaiting_human'` + new approval_token record →

exits. Next poll checks `approval_tokens WHERE status='approved'`. **chain-runner NEVER writes

APPROVE — only reads one.**

1. CEO approval via `mc.js approval grant <token_id>` invoked in CEO's Claude Code session

(where ceo-token-origin-gate.sh IS active)

1. mc.js `comment` handler extension: if text matches `\bAPPROVE\b/i` AND no

`CLAUDE_SESSION_ID` env + valid CEO turn log → exit 2

Residual risk: MEDIUM. approval_tokens table writable by any process running as makinja.

Cryptographic signing out of scope for v1 bash+Node. **System-wide implication:** mc.js comment

APPROVE forgery gap is NOT Pillar #5-specific — propose system-wide hardening MC (see §12).

T3 — HIGH — Phantom-Link False-Negative (Empty File Bypass)

Vector: Sub-agent writes empty file at `output_path` before exiting → file existence check

passes → chain advances → downstream reads empty → silent failure or null propagation. Direct analog of MLX HTTP 200 / no actual work pattern.

Mitigation:

1. `MIN_STEP_OUTPUT_BYTES=64` in phantom-link-detector.js
2. JSON parse + verify `chain_run_id` + `step_name` fields match
3. Re-compute SHA-256 vs `chain_steps.output_sha256` (catches replace-with-empty after checksum)
4. Find matching `STEP_END status:ok` in chain-events.jsonl
5. `required_output_fields[]` per-step YAML schema contract; non-null + non-empty validation

Residual risk: LOW. Phantom-link-detector is structural + provenance verifier, NOT semantic. Semantic correctness = Proveo gate's domain.

T4 — MED — JSONL Log Tampering

Vector: Same-UID sub-agents can append fake STEP_END events; insert mid-file via temp+concat; truncation attack.

Mitigation:

1. SHA-256 hash chain — each event includes `prev_hash` + `seq`; phantom-detector re-walks chain
2. `chain_runs.event_seq_watermark` — detects file shrinkage
3. `fs.open(path, 'a')` O_APPEND only; never `writeFileSync` (truncates)
4. Sub-agent prompt explicit: "MUST NOT write to `~/system/logs/chains/`. Evidence files go to `~/system/state/chains/<chain_id>/<run_id>/` only."
1. File mode 0644 owned makinja (necessary, not sufficient — same-UID still writes)

Residual risk: MEDIUM. Hash chain prevents undetected injection but not full-log rewrite. True append-only requires `chattr +a` (Linux) — macOS lacks direct equivalent. For ALAI v1 threat model (accidental hallucination, not adversarial external attacker), hash chain + watermark sufficient.

T5 — MED — Lease Takeover / Split-Brain

Vector: Two chain-runner instances acquire same chain_run_id lease → both dispatch same

sub-agent → both write same output_path → both append STEP_END → external POSTs execute twice.

Mitigation:

1. Atomic SQL UPDATE with WHERE clause on lease_until — `changes()==0` → exit 0 (CAS pattern,

section-1 §5.2 + section-6 mitigation 1). SQLite WAL serializes; impossible for two writers

both see `changes==1`.

1. `node mc.js active | grep chain-runner` startup check (soft guard; SQL UPDATE is hard guard)
2. StartCalendarInterval (NOT KeepAlive) on chain trigger plists; ProcessType=Background
3. Log `chain_runs.runner_pid` on lease acquisition; second instance sees PID mismatch → exit clean

Residual risk: LOW after WAL atomic UPDATE. Standard SQLite job-queue pattern.

T6 — HIGH — Secrets Exfiltration Through Skill Output

Vector: daily-inbox-triage reads inbox containing API keys, Bitwarden exports, GCP tokens,

contract terms. Prompt injection in malicious email → sub-agent writes secrets to chain-events.jsonl

or `~/system/state/chains/`. Persistent on disk (no TTL in spec); readable by future chains.

No existing secret-scanner hook in `~/.claude/hooks/` for Write/Edit on chain state dirs.

Mitigation (must-fix before ship; NEW HOOK REQUIRED):

1. **NEW hook `~/.claude/hooks/secret-scanner.sh`** [NEW] — PostToolUse Write/Edit targeting

`~/system/state/chains/` and `~/system/logs/chains/`. Pattern:

```
(sk-[a-zA-Z0-9]{20,}|AIza[0-9A-Za-z\-_]{35}|AKIA[0-9A-Z]{16}|ghp_[a-zA-Z0-9]{36}|
 -----BEGIN (RSA|EC|OPENSSH) PRIVATE KEY|["\s]password["\s]*[:=]["\s]*[^\s]{8,})
```

Match → block Write, log `/tmp/secret-scanner.log`, Slack alert

1. Sub-agent context isolation: email-fetch writes ONLY structured metadata (no raw bodies) to

`step1.json`. If classification needs body content, ephemeral via Claude context window, NOT disk

1. `~/system/state/chains/` mode 0700 (owner-only)
2. Evidence file TTL: 72h auto-purge via phantom-link-detector.js or separate cleanup
3. Email-triage skill prompt explicit: "Process metadata only. Do not reproduce raw email body

text. Write only {sender, subject, classification, priority, mc_id}."

Residual risk: MEDIUM. Prompt injection in emails persistent. Recommend human-gate at

email-fetch→classify boundary for first 30 days production.

T7 — MED — Cost Exhaustion DoS

Vector: Pre-flight gap (Lee §3 post-link timing) — single Opus call >$1; three Opus calls

exhaust $3/day before ceiling fires post-step. Aggregator timing (per-run-completion not real-time)

allows runaway concurrent dispatch. Retry cascade: 2N invocations at $0.50/call.

Mitigation:

1. **Pre-STEP_START budget check** — sum today's chain spend + step's `cost_estimate_usd` >

ceiling → halt + cb_halted before dispatch

1. **REQUIRED** `cost_estimate_usd` **per link** in YAML; chain-runner rejects YAML at load time if absent
2. **Per-step** `max_cost_usd` **hard ceiling** — actual STEP_END cost > max → cb_state=OPEN, halt chain
3. Model enforcement: chain YAML header `ceo_approved_opus_links: []` (empty = no Opus). Validator

rejects `agent: opus` or `model: claude-opus*` without [CEO_APPROVED] marker

Residual risk: LOW for intentional overspend after pre-flight. Pathological wrong estimate

(model pricing change) not covered → monthly pricing-update reminder.

T8 — LOW — Liveness-Claim-Validator Bypass via Synonym Avoidance

Vector: `liveness-claim-validator.sh` CLAIM_PATTERN at hook line 122 (regex citation, not a claim):

```
\bLIVE\b|\bverified\b|\boperational\b|cost=\$0 verified|\bDONE\b
```

Case-sensitive match.

Synonyms bypass: "finished", "completed", "succeeded", "all links passed", "pipeline up and running",

"confirmed", "active". MLX-style failure mode.

Mitigation:

1. Extend CLAIM_PATTERN with chain-completion synonyms (false-positive-prone in legitimate context;

prefer targeted approach)

1. **Dedicated** `chain-completion-validator` **hook** (PreToolUse Write, same three paths). Triggered

on `chain_run_id` AND past-tense completion language. Required evidence: `chain-events\.jsonl` +

`STEP_END` + `chain_run_id.*ok` + `postflight-cleared` within ±15 lines

1. **mc.js ready cross-validation:** before any chain-associated task ready, mc.js checks

`chain_runs.status='succeeded'` AND all `chain_steps.status='done'` AND no `phantom_suspected`.

Fail → exit 2

Residual risk: MEDIUM. Pattern-matching natural language inherently incomplete. Strongest

mitigation is structural: architecturally impossible for chain to declare done without runtime

STEP_END + phantom-detector clearance.

Threat Summary Table

#	Threat	Sev	Existing Defense	Gap	Ship-Blocking?
T1	Shell injection (YAML→bash)	HIGH	bash-danger-gate (pipe-to-shell)	No pre-quoted variable expansion guard	YES

#	Threat	Sev	Existing Defense	Gap	Ship-Blocking?
T2	APPROVE forgery (daemon path)	HIGH	ceo-token-origin-gate (Claude session only)	Daemon has no session; mc.js comment hostname-only	YES
T3	Phantom empty-file bypass	HIGH	phantom-link-detector existence check	No content/schema validation	NO (Phase 1b)
T4	JSONL tampering	MED	runtime-only convention	Same-UID sub-agents append; no integrity	NO (Phase 1b)
T5	Lease takeover	MED	task_scheduling.lease_until column	chain-runner.sh CAS UPDATE not yet written	NO (initial build)
T6	Secrets exfil through email	HIGH	bash-danger-gate (pipe-to-shell)	No credential scanner on chain state dir	YES (NEW HOOK)
T7	Cost exhaustion DoS	MED	$3/day post-link ceiling	Pre-flight gap; single Opus spike	NO (initial build)
T8	Validator synonym bypass	LOW	liveness-claim-validator (LIVE/DONE)	"finished/completed/succeeded" bypass	NO (post-v1)

Three SHIP-BLOCKING child MCs identified (T1, T2, T6).

Full exploitation paths + code snippets: section-6 §Threat 1-8 (lines 32-644).

---

§9. Build-Phase Plan (Child MCs)

Child MC	Owner	Estimate	Deliverables
#99129-A	Petter Graff (CodeCraft)	8h	`chain-runner.sh` ≤150 LOC + `phantom-link-detector.js` + Node helpers; shellcheck clean; dry-run; T1 + T5 mitigations baked in
#99129-B	Bruce Momjian (CodeCraft)	1h	chain_runs + chain_steps DDL applied via migration to mission-control.db; verify `.schema` returns expected; chain_runs_archive ready

Child MC	Owner	Estimate	Deliverables
#99129-C	Parisa Tabriz (Securion)	4h	NEW `~/.claude/hooks/secret-scanner.sh` (T6); mc.js comment APPROVE forgery hardening — switch to approval_tokens primitive (T2); register hook in `~/.claude/settings.json`
#99129-D	Hadi Hariri (CodeCraft)	1h	Commit 3 reference YAMLs to `~/system/agents/chains/`; chain-validator.js with 16 rules; V-09 + V-11 enforced
#99129-E	Kelsey Hightower (FlowForge)	2h	Fix `com.john.db-backup` + `com.alai.cost-daily-report` to clear 80% gate; load 3 plists + watchdog integration; pre-flight checklist run
#99129-F	Lee Robinson (CodeCraft)	2h	cost-tracker.js `chain` subcommand; AGENT_COST event spec; pre-STEP_START ceiling check (T7); chain-cache-report.js
#99129-G	Angie Jones (Proveo)	4h	E2E validation: cold start, resume-after-kill mid-link 2, phantom trip, human-gate APPROVE advance + ceo-origin block, cost ≤$0.20/run. Real evidence in `/tmp/proveo-99129/`
#99129-H	Skillforge	2h	BookStack page on Pillars shelf at https://docs.basicconsulting.no/books/agentic-os/page/pillar-5-skill-chains

Ship sequence:

1. -A through -F build artefacts (parallel where independent)
2. -B + -E gate: 80% daemon health PASSED + DDL applied → unblock plist load
3. **daily-inbox-triage v1 ships first**
4. **7 clean days** (no phantom alerts, cost within projection, no broken-JOHN events) → ship

weekly-client-report v2

1. **7 more clean days** → ship email-pipeline-fix-loop v3
2. -G validation runs end-to-end against daily-inbox-triage v1 BEFORE marking Pillar #5 done
3. -H BookStack publish

§10. Acceptance Criteria for #99129 (DESIGN PHASE)

AC	Criterion	Verification
AC1	Spec file exists at `/Users/makinja/system/specs/agentic-os-pillar5-skillchains-2026-05-05.md`	`ls -la <path>` returns file
AC2	All 6 specialist sections referenced with file paths	grep `/tmp/pillar5-99129/section-` in spec returns 6+
AC3	3 reference pipeline YAMLs documented (NOT committed — that's build phase)	§5 contains daily-inbox-triage, weekly-client-report, email-pipeline-fix-loop section headers + cite section-3 line ranges
AC4	Threat mitigations enumerated with severity	§8 contains 8-row threat table with HIGH/MED/LOW + ship-blocking column
AC5	Cost envelope + ship gate documented	§1 + §6 + §7 contain $3/day ceiling + 80% fleet gate + projected $2.80/mo
AC6	ZAKON PLAN compliance (validation + docs in §11)	§11 lists Proveo MC + Skillforge MC explicitly
AC7	4 forged-prompt unresolved items resolved or escalated	DB location resolved §3; lease-column resolved §3; language guard resolved §4; saga-scope resolved §4

§11. ZAKON PLAN — Required Follow-Up Tasks

Both REQUIRED before Pillar #5 can be marked done.

Validation (Proveo / Angie Jones) — Child MC #99129-G

End-to-end test of `daily-inbox-triage` v1 with REAL evidence (NOT dry-run only):

1. Cold start — fresh chain_runs row, fresh JSONL file
2. Resume-after-kill mid-link 2 — `kill -9 chain-runner` → next tick reads

`WHERE status IN ('pending','failed')` ORDER BY id ASC → resumes from link 2 only,

links 0-1 untouched

1. Phantom-link-detector trip — `rm <output_path>` mid-run → next 15-min tick → status=phantom_suspected,

downstream `mc.js done` BLOCKED by postflight gate

1. Human-gate halt + `mc.js comment <chain_task_id> "APPROVE <run_id>"` advances chain;

ceo-token-origin-gate blocks self-issued APPROVE (T2 mitigation verified)

1. Cost ≤$0.20/run (warm) verified via `cost-tracker.js chain daily-inbox-triage --window 1d`

Real evidence committed to `/tmp/proveo-99129/`. PASS verdict at

`/tmp/proveo-99129-postflight.json`.

Documentation (Skillforge) — Child MC #99129-H

BookStack publish on Pillars shelf:

- URL: https://docs.basicconsulting.no/books/agentic-os/page/pillar-5-skill-chains
- Cover: architecture, chain_runs/chain_steps DDL, runbook for adding 4th chain,

phantom-link-detector incident response, T1-T8 threat playbook

- URL recorded in `/Users/makinja/system/state/postflight-cleared-99129.json`

---

§12. Open Questions / Risks

1. **pipeline.db unused for chains.** Bruce's section-1 schema read found zero chain-shaped

tables. Should it be archived or repurposed in Pillar #11 cleanup MC? **Defer to Pillar #11

cleanup; not Pillar #5 scope.**

1. **Daemon-fleet 79.7% structural risk.** Chains depend on a partially-broken scheduler. Even

after fixing db-backup + cost-daily-report (target 81%+), 19% of fleet remains unhealthy.

The phantom-detection subsystem itself depends on a healthy phantom-detector plist. Recommend

quarterly daemon-fleet audit MC as ongoing operational hygiene.

1. **mc.js comment APPROVE forgery gap (T2) is system-wide.** Affects ALL daemon-invoked mc.js

comment paths (not just chain-runner). Propose **system-wide hardening MC** —

"Audit all daemon → mc.js comment paths for APPROVE forgery vector" — separate from #99129-C

which only addresses Pillar #5 surface.

1. **First-ship epistemics tension (Parisa flag).** daily-inbox-triage is LOWEST complexity but

HIGHEST data sensitivity (CEO inbox contains secrets). Recommend human-gate at email-fetch→

classify boundary for first 30 days production, in addition to send-link gate. Trade slight

automation cost for substantially reduced exfiltration blast radius.

1. **Cache prefix invalidation risk.** Cost projections assume ≥75% cache hit on warm runs. If

chain system prompt changes (new skill schema, system prompt edit), cache_creation_tokens

spike + cache_read_tokens drop to zero → costs spike 2-4×. `chain-cache-report.js` prefix-

stability check (SHA-256 of stable prompt portion) flags this within 1 day.

1. **24h vs 7d vs 30d retention windows** are estimates from Bruce §6 + petter-graff §3. May

need tuning after first 30 days production telemetry — increment via build-phase MC, not

spec change.

Appendix A — Source Section Files

File	Lines	Author	Domain

`/tmp/pillar5-99129/section-1-storage-schema.md`	763	Bruce Momjian	DDL + DB location decision + retention
`/tmp/pillar5-99129/section-2-chain-runner-architecture.md`	722	Petter Graff	bash runner + signals + phantom detector + saga + language guard
`/tmp/pillar5-99129/section-3-yaml-pipelines.md`	882	Hadi Hariri	YAML schema + 3 pipelines + validation V-01..V-16 + versioning
`/tmp/pillar5-99129/section-4-launchd-daemon.md`	522	Kelsey Hightower	plist XML + daemon-fleet pre-flight + KeepAlive matrix + 80% gate
`/tmp/pillar5-99129/section-5-cost-telemetry.md`	644	Lee Robinson	AGENT_COST event + ceiling + cache hit + phantom-cost + cost-tracker.js extension
`/tmp/pillar5-99129/section-6-threat-surface.md`	723	Parisa Tabriz	8 threats T1-T8 + exploitation paths + mitigations + ship-blocking matrix
Total source lines	**4256**	—	—
`/Users/makinja/system/prompts/forged/99129.md`	121	Synthesizer (petter-graff)	Forged baseline with 10 disagreements + decisions

Appendix B — Disagreements Resolved

Per `/Users/makinja/system/prompts/forged/99129.md` block:

#	Tension	Resolution	Citation
1	Recursive sub-agent vs cron+stateless	**HYBRID**: chain-runner.sh = cron+stateless trigger and link advancer; each link's work dispatched as sub-agent Task call (inherits 64-gate stack)	§2 Architecture Overview; section-2 §2.0
2	Checkpoint storage shape (JSONL vs JSON file vs SQL)	**BOTH SQL + JSONL, NO state-file**. SQL = canonical state (durability, queryable, joins); JSONL = runtime-written event log; state-file rejected	§3 Storage; section-1 §4

#	Tension	Resolution	Citation
3	DB location (mission-control.db vs pipeline.db)	**RESOLVED — mission-control.db**. Bruce's section-1 schema read confirmed pipeline.db has zero chain-shaped tables; default rule applies	§3 Storage; section-1 §1
4	Human-in-loop primitive (mc.js comment vs file-flag)	**mc.js comment canonical, Slack mirror-only.** File-flag bypasses ceo-token-origin-gate (broken-JOHN pattern) → REJECTED. T2 in §8 hardens this further (use approval_tokens, not comment parsing)	§5 + §8 T2; section-6 Threat 2
5	Saga compensation pattern	**Per-link attribute, not chain-wide.** External-API boundary only (email send, BookStack publish). Read-only links + idempotent DB writes do NOT need compensation	§4 Saga; section-2 §2.5
6	OpenAI Agents SDK	**Both panelists rule out.** Hard constraint regardless	§4 Implementation Language Guard; section-2 §2.8
7	Implementation language	**bash + Node ONLY.** No Kotlin (rabbit-hole genesis), no TypeScript (silent build failure), no Python beyond fsync one-liner	§4 Implementation Language Guard; section-2 §2.8
8	First-ship pipeline ranking	**AGREED: daily-inbox-triage first; email-pipeline-fix-loop last.** Build order: triage → weekly → fix-loop. Fix-loop ships ONLY after 7 clean days on first two	§5 + §9 Build-Phase Plan
9	liveness-claim-validator integration	**Hook is LIVE per `/Users/makinja/system/state/postflight-cleared-99127.json` 2026-05-05T06:54Z. Pillar #5 reuses, does not reimplement.** Operationalized via phantom-link-detector	§2 Components Diagram; §8 T8

#	Tension	Resolution	Citation
10	ANVIL/FORGE distributed nature	**v1 single-host (ANVIL), distributed concerns documented for v2.** UUID `chain_run_id` adopted NOW to avoid future schema migration. Clock skew, lid-close, mc.js↔JSONL skew already mitigated in v1	§2 Boundary Rules; section-2 §2.7

All 10 forged-prompt disagreements resolved. AC7 satisfied.

End of spec — Pillar #5 Skill-Chain Meta-Orchestrator design.

Next: Mehanik clearance → child MC dispatch (#99129-A through -H) → 80% daemon health gate → daily-inbox-triage v1 first ship.

Bismillah.