CONFIDENTIAL — Internal Incident Report
Generated: 2026-05-03 07:17 UTC | Author: Ria (dev-side oversight auditor)
MC Task: #3930 | Classification: DevFM-024

P0 Incident Report: Token-Burn Loop — 2026-05-03

Severity: P0 — active token burn, manually halted by Thomas
Duration: ~21 minutes (06:30–06:51 UTC)
Impact: Max-plan Opus 4.7 tokens burned by 6 concurrent claude processes
Detection: Thomas observed live
Resolution: Manual SIGTERM of 6 claude PIDs + mc-mcp-bridge kill
MC Task: #3930 (assigned to Ria)

Executive Summary

Multiple Mission Control subsystems simultaneously sent prompts to `agent=main` (Jane Main) through the agent_turn sidecar. With no rate limiter, no concurrency cap, and no circuit breaker, each request spawned a separate `claude` SDK process under meridian. At peak, 6 concurrent Opus 4.7 processes were running, each consuming Max-plan tokens. The incident was NOT a single-source loop — it was a multi-source convergence with zero coordination between callers.

Timeline

Phase 1: 06:30–06:42 UTC — Failing calls (old gateway, PID 268705)

Time	Event
06:30:10	First sampling_audit entry: `agent_turn → bridge:main`, TimeoutError
06:30–06:42	6–14 calls/min to `agent=main`, ALL failing
06:34:41	Gateway log: heartbeat error for `agent:main:telegram:direct:8797337631`
06:37:00	Gateway registers `agent:main:main` with heartbeat every 60000ms

Call rate per minute (all failing): - 06:30: 6 | 06:31: 12 | 06:32: 9 | 06:33: 9 | 06:34: 14 - 06:35: 6 | 06:36: 10 | 06:37: 5 | 06:38: 5 | 06:39: 7 - 06:40: 13 | 06:42: 12

No token burn in this phase — all calls error before spawning claude. But the high call rate reveals the underlying problem: MC subsystems were generating 6–14 requests/min with no coordination.

Dominant callers (identified by prompt length fingerprint): - plen=332: 23 occurrences, steady ~2/min from 06:30–06:42 — likely tg-drain loop retrying notes - plen=495: 8 occurrences — secondary caller (planner or neo probe) - plen=479: 5 occurrences — tertiary caller - plen=489, 444, 485: 3–4 each — additional callers

Phase 2: 06:44 UTC — Full Stack Restart

Time	Event
06:44	MC (PID 1148), gateway (PID 1620), meridian (PID 1193), sidecar (PID 1788) all restart
06:44:38	Gateway loads mc-session-register hook
06:44:39	Planner safety sweep fires: 19 duplicate dispatch evaluations (separate bug, all skipped)
06:45:01	Gateway heartbeat starts

Phase 3: 06:45–06:51 UTC — TOKEN BURN (calls now succeed)

Time	Event
06:45:31	First successful sampling call post-restart
06:45–06:51	2–7 calls/min, each spawning a real `claude` process
~06:48	Peak: 6 concurrent claude processes (ages: 12s, 18s, 24s, 30s, 33s, 54s)
~06:48	3 simultaneous ESTAB sockets from mc-mcp-bridge → :18790
06:51	Thomas manually SIGTERMs all 6 claude PIDs
06:51	Thomas SIGTERMs mc-mcp-bridge (PID 1624)
06:51:48	Last sampling_audit entry

Call rate per minute (succeeding, burning tokens): - 06:45: 2 | 06:46: 7 | 06:47: 2 | 06:48: 6 - 06:49: 5 | 06:50: 7 | 06:51: 7

Total: 145 sampling_audit entries in the 25-minute window. Of these, ~36 succeeded post-restart and spawned claude processes that consumed real tokens.

Root Causes

RC-1: No per-agent rate limit on agent_turn sidecar

File: `/home/claw/.openclaw/scripts/agent_turn_endpoint.py`

The sidecar's HTTP handler accepts every POST to `/api/agent/turn` and spawns `openclaw agent --local --json` for each one. No concurrency tracking, no token bucket, no 429 response. It will happily run 10 claude processes simultaneously for the same agent.

Evidence: 6 concurrent claude processes under meridian PID 1193 at peak.

RC-2: No circuit breaker for claude process count

Nothing in the system monitors how many `claude` SDK processes are running. No hard cap prevents accumulation. The only protection was manual intervention (Thomas checking `ps -ef`).

Evidence: Ages staircased at 12s/18s/24s/30s/33s/54s — each new spawn happened while previous ones were still running.

RC-3: Multiple independent callers, no coordination

At least 4 MC subsystems call `agent_turn` for `agent=main` independently:

Caller	Interval	Path	Code Location
tg-drain loop	300s (5 min)	`_send_via_jane()` → `/api/mcp/sample` → sidecar	`main.py:1317–1576`
Planner safety sweep	1800s (30 min)	`_phase3_call_agent_turn()` → direct POST	`planner_routes.py:1222`
Gateway heartbeat	60s	Session heartbeat poll → agent_turn	OpenClaw gateway
MCP agent_request	On-demand	Any MCP client → `sampling.py:agent_request()` → sidecar	`mcp_server/sampling.py:358`
/tom auto-decide	On-demand	`_agent_call()` → direct POST	`main.py:4698`

None of these callers know about each other. No semaphore, no coordination, no mutual exclusion.

Evidence: Multiple distinct prompt lengths (332, 495, 479, 489, 444, 485) appearing at overlapping times.

RC-4: Dedup at wrong layer

The tg-drain's `_drain_notes` has content-hash dedup within a 6-hour window — it catches duplicate *notes*. But the dedup happens AFTER the agent_turn invocation, not before. Each call to `_send_via_jane` fires a separate agent_turn even for content that will be dedup'd.

Similarly, `_drain_james_notifications` has `delivered=1` tracking but no retry limit — it retries indefinitely on failure.

Evidence: Note hash `3b55913453a1553d` was dedup'd at insert time but each invocation still spawned a separate claude process.

What Was NOT the Root Cause

James q#82 — partially incorrect initial hypothesis

Task #3930 was originally filed as "James q#82 reflect-error → main agent re-spawn." Investigation disproved this:

Check	Finding
`james_notifications WHERE queue_id=82`	1 row (#43), created 2026-05-02 09:15, `delivered=1` — already delivered yesterday
`notes WHERE category='telegram-queue'` in incident window	0 rows
`intake_queue WHERE id=82`	`status='failed'` — properly parked, not looping

James q#82 was **one of many signals** various callers were trying to relay, but it was NOT an active loop source.

OpenClaw cron jobs — not the source

Only 2 crons were enabled (MC DB backup every 2h, Memory Dreaming daily). All others had been disabled for review. The high-frequency calls came from MC's internal `@scheduler` decorators and `asyncio.create_task` loops, not from OpenClaw crons.

Secondary Findings

Planner duplicate-dispatch bug

At 06:44:39 (immediately after restart), the planner's safety sweep evaluated the same 3 tasks 5–7 times each in a single second:

Task	Evaluations	Result
#3136	5x	All skipped: `risk_flags:CREDENTIALS`
#2962	7x	All skipped: `risk_flags:SCOPE-CREEP`
#2948	7x	All skipped: `risk_flags:PRODUCTION,SCOPE-CREEP`

Impact: No token burn (all skipped before agent_turn), but 19 duplicate `planner_dispatches` rows + 19 duplicate `planner-dispatch-skipped` notes. Indicates a dedup bug in `planner_pass()`.

jobs-state.json temp file accumulation

524 orphaned `.tmp` files in `/home/claw/.openclaw/cron/`. Not directly related to the incident but indicates a write-loop or cleanup failure in the OpenClaw cron state manager.

Impact Assessment

Token burn estimate

- **Burn window:** 06:45–06:51 UTC (6 minutes of successful spawns)
- **Concurrent claude processes:** Peak 6, average ~3–4
- **Model:** Opus 4.7 (Max plan)
- **Estimated spawns:** ~36 successful agent_turn calls
- **Per-spawn cost:** Each process loads full agent context (~100K+ tokens input) + generates response (~500–2000 tokens output)
- **Rough estimate:** 36 spawns × ~100K input tokens × $15/M = ~$54 input + output costs
- **Actual cost:** Requires Meridian/provider billing reconciliation (TODO)

System impact

System	Effect
mc-mcp-bridge	Killed, currently DOWN — no MCP sampling for any agent
Agent-turn sidecar	Running but idle (no callers with bridge down)

System	Effect
MC schedulers	Running normally (tg-drain, neo, ria, planner)
James worker	Running, q#82 parked as failed
OpenClaw gateway	Running, heartbeat active but no bridge to relay through

---

Remediation Plan

Must-land before mc-mcp-bridge restart

1. Sidecar rate limiter (RC-1)

File: `agent_turn_endpoint.py`
Spec: - Per-agent concurrency cap: max 2 concurrent for same `agent` arg - Per-agent rate limit: max 10 requests/min per agent - Return HTTP 429 + body `{"ok": false, "error": "cost_breaker_tripped", "concurrent": N, "limit": 2}` - Log `cost_breaker_tripped` to sampling_audit on every rejection

2. Global circuit breaker (RC-2)

File: `agent_turn_endpoint.py`
Spec: - Before spawning: count `claude` processes under meridian PID - If count >= 3: reject with 429 + `global_circuit_breaker` - Meridian PID resolved at startup from `pgrep -f meridian` or env var

3. Invocation-layer dedup (RC-4)

File: `main.py` (tg-drain section)
Spec: - Before calling `_send_via_jane`, compute prompt fingerprint (first 200 chars normalized) - Check a TTL cache (60s expiry) keyed on fingerprint - If hit: skip the call, log "dedup'd at invocation layer" - Same for `_drain_james_notifications`

Should-land (separate tasks)

4. Planner dedup fix

File: `planner_routes.py`
Spec: `planner_pass()` should track evaluated task IDs within a single pass and skip duplicates.

5. DevFM-024 registration

Ria assigns "sampling/createMessage runaway" as DevFM-024. Neo registers detector (task #3934).

6. james_notifications retry limit

File: `main.py:1489` (`_drain_james_notifications`)
Spec: Add retry_count column or max-age cutoff (e.g., don't retry notifications older than 24h).

Verification Protocol (post-patch)

1. Deploy sidecar rate limiter
2. Restart mc-mcp-bridge
3. Monitor for 5 minutes: `tail -f /home/claw/.openclaw/logs/agent-turn.log` — expect 0 new POSTs
4. Verify sidecar responds to test POST with 200 (not 429) when under limit
5. Verify sidecar responds with 429 when rate exceeded (send 3 rapid-fire POSTs for same agent)
6. Check `ps -ef | awk '$3==<meridian_pid> && /claude/'` — should show ≤2 at any time
7. 30-minute quiet-window check: no sampling_audit entries with status=error for bridge:main

---

Open Questions

1. **Exact token cost** — needs Meridian billing API or provider dashboard reconciliation
2. **What was the plen=332 prompt?** — the most frequent caller (23 occurrences). Sampling_audit stores length but not content. Need to check if MC logs preserve the prompt text.
3. **Why did the full stack restart at 06:44?** — was it manual, a crash, or a systemd watchdog? This is the event that converted 14 minutes of harmless failures into an active token burn.
4. **Are there other latent convergence patterns?** — any other scenario where 4+ subsystems target the same agent simultaneously deserves the same rate-limit protection.

Lessons Learned

1. **Rate limiting is not optional for LLM-spawning endpoints.** Every path that can spawn a claude process MUST have a concurrency cap. This is an architectural invariant, not a nice-to-have.
2. **Dedup must happen at the invocation layer, not the content layer.** Catching duplicate notes after the LLM has already been called is closing the barn door after the horse has bolted.
3. **Multi-source convergence is harder to detect than single-source loops.** The original hypothesis (James q#82 loop) was clean and obvious but wrong. The real cause — independent callers converging — required tracing 145 audit entries, cross-referencing 4 code paths, and checking the gateway journal.
4. **Failing calls hide the problem until a restart makes them succeed.** For 14 minutes, the same convergence was happening but causing zero damage because all calls failed. The restart at 06:44 was the triggering event, not the root cause.
5. **Kill switches work.** Thomas's manual intervention (kill PIDs + bridge) was fast and effective. But it shouldn't be the only protection.

Report written by Ria, 2026-05-03 07:15 UTC. MC task #3930. DevFM classification: DevFM-024 (pending Neo registration, task #3934).