active-thread-lock — 4th Anti-Drift Structural Layer

1. TL;DR

`active-thread-lock.sh` is a PreToolUse hook that fires on every `Task`, `Agent`, `WebSearch`, and `WebFetch` tool call. It reads the `## ACTIVE_THREAD:` block from `~/.claude/session-state.md`, extracts the approved MC IDs (5-digit `#NNNNN` references), and blocks any dispatch whose prompt references an MC ID outside that approved set with exit code 2. To perform a legitimate CEO-authorized thread switch, include the token `[CEO_APPROVED_THREAD_SWITCH]` anywhere in the dispatch prompt. On any parse failure or missing configuration the hook exits 0 (fail-open), so it never blocks legitimate non-MC work.

2. Why This Exists (Genesis)

ZAKON #27 (one product per session) has existed as a written rule since the ALAI operating system was established, but had no machine enforcement. The consequence was documented in `feedback_drift_after_step1_completion.md` (2026-05-02): John completed Step 1 of a CEO multi-step sequence, then drifted to a self-ranked priority (Akershus) instead of proceeding to Step 2, requiring the CEO to manually correct course. The CEO observation was: "*ja vise ne mogu da te stalno vracam*" ("I cannot keep pulling you back").

The fix was approved as part of the **system-uvezivanje master spec §4** (`~/system/specs/system-uvezivanje-master-2026-05-02.md`), which defines four anti-drift structural layers. This hook is layer 4. The specific CEO directive is recorded in `~/system/specs/ai-factory-pipeline.md` §6 Q3 answer: "*Da*" (2026-05-03).

The Four Anti-Drift Layers (system-uvezivanje §4)

Layer	Hook / Mechanism	What it enforces
1	`john-max-depth-gate.sh` (ZAKON #28)	Emergent-spawn depth ≤ 3 beyond Mehanik clearance
2	`pre-mc-add-gate.sh`	1 CEO turn = max N MC dispatches

Layer	Hook / Mechanism	What it enforces
3	`memo-citation-gate.sh`	Drift-stop protocol on feedback memo citations
4	`active-thread-lock.sh` (this runbook)	ACTIVE_THREAD sequence enforcement — blocks off-thread MC dispatches

3. How It Works

Execution Flow (Step-by-Step)

1. **Read JSON from stdin.** Claude Code passes a JSON object with `tool_name` and `tool_input` fields. The hook extracts `tool_input.prompt` via Python.
2. **Extract dispatched MC IDs from prompt.** Regex patterns matched (4-6 digit numbers):
   - `MC #NNNNN` or `#NNNNN`
   - `mc_task_id NNNNN` or `task-id NNNNN`

   If no MC IDs are found in the prompt, exit 0 (fail-open — no IDs to check).
3. **Check bypass token.** If `[CEO_APPROVED_THREAD_SWITCH]` is present anywhere in the prompt, exit 0 (authorized override). This check fires before any file I/O.
4. **Read session-state.md.** File: `~/.claude/session-state.md`. If the file is missing, exit 0 (fail-open).
5. **Extract approved IDs from ACTIVE_THREAD block.** Python regex finds the block starting at `## ACTIVE_THREAD:`, continuing until the next `---` separator or next `## [A-Z]` heading. All `#NNNNN` patterns within that block form the approved set. If the block is absent or yields no IDs, exit 0 (fail-open).
6. **Compare.** For each dispatched MC ID, check against the approved set. First non-member triggers exit 2 with a BLOCKED message to stderr naming the offending MC ID, the full approved set, and the override token. All members pass with exit 0.

Pseudocode

```
INPUT = read_stdin_json()
PROMPT = INPUT.tool_input.prompt

DISPATCHED = extract_mc_ids(PROMPT)
# Patterns: #NNNNN, MC #NNNNN, mc_task_id NNNNN, task-id NNNNN (4-6 digits)
if DISPATCHED is empty:
    exit 0  # no IDs -- fail-open
```

```
if "[CEO_APPROVED_THREAD_SWITCH]" in PROMPT:
    exit 0  # bypass token

if not exists("~/.claude/session-state.md"):
    exit 0  # missing file -- fail-open

APPROVED = extract_mc_ids_from_active_thread_block("~/.claude/session-state.md")
if APPROVED is empty:
    exit 0  # no block or malformed -- fail-open

for id in DISPATCHED:
    if id not in APPROVED:
        stderr("BLOCKED [active-thread-lock]: MC #" + id
               + " not in ACTIVE_THREAD sequence (approved set: "
               + join(APPROVED) + "). Override: include [CEO_APPROVED_THREAD_SWITCH] in
prompt.")
        exit 2

exit 0
```

4. Override

Include the literal string `[CEO_APPROVED_THREAD_SWITCH]` anywhere in the dispatch prompt. The hook checks for this token before reading session-state.md, so it incurs no file I/O on bypass.

Use case: CEO explicitly authorizes work on an MC outside the current thread, e.g., an urgent hotfix on a separate product. The CEO must include or authorize this token in their directive — it cannot be inserted by John autonomously.

Important: Inserting `[CEO_APPROVED_THREAD_SWITCH]` without explicit CEO authorization is itself a drift violation tracked by the memo-citation gate (layer 3).

5. Fail-Open Conditions

The hook exits 0 (allow) under every condition below. It never produces false positives against legitimate non-MC dispatches.

Condition	Exit	Stderr signal

No 5-digit MC ID extractable from prompt	0	(silent)
`[CEO_APPROVED_THREAD_SWITCH]` token present in prompt	0	(silent)
`~/.claude/session-state.md` does not exist	0	`[active-thread-lock] session-state.md not found — fail-open.`
session-state.md exists, no `## ACTIVE_THREAD:` block found	0	`[active-thread-lock] No ACTIVE_THREAD block or no MC IDs found in session-state.md — fail-open.`
ACTIVE_THREAD block present but contains no parseable `#NNNNN` IDs	0	`[active-thread-lock] No ACTIVE_THREAD block or no MC IDs found in session-state.md — fail-open.`
Python internal exception during parse	0	`[active-thread-lock] ACTIVE_THREAD block parse error — fail-open.`

6. Smoke Test Procedure

Independent Proveo replay completed 2026-05-03. Evidence: `/tmp/evidence-99014-proveo/replay-log.txt` and `verdict.txt`. Overall verdict: **7/7 PASS**.

To replay a single TC manually:

```
echo '{"tool_name":"Task","tool_input":{"prompt":"Dispatch codecraft agent to build MC #10612."}}' \
  | bash ~/.claude/hooks/active-thread-lock.sh
echo "Exit code: $?"
```

TC	Description	Fixture prompt	Expected exit	Expected stderr signal
TC1	MC is in ACTIVE_THREAD approved set	`Dispatch codecraft agent to build MC #10612 system-uvezivanje hook.`	0	(silent)
TC2	MC is NOT in approved set	`Dispatch flowforge agent to work on MC #99999 some unrelated task.`	2	`BLOCKED [active-thread-lock]: MC #99999 not in ACTIVE_THREAD sequence (approved set: 10424,10429,10536,10611,10612,99012,99013,99014,99015,99016). Override: include [CEO_APPROVED_THREAD_SWITCH] in prompt.`

TC	Description	Fixture prompt	Expected exit	Expected stderr signal
TC3	session-state.md removed entirely	`Dispatch agent to work on MC #99999.`	0	`[active-thread-lock] session-state.md not found — fail-open.`
TC4	session-state.md present, no ACTIVE_THREAD block	`Dispatch agent to work on MC #99999.`	0	`[active-thread-lock] No ACTIVE_THREAD block or no MC IDs found in session-state.md — fail-open.`
TC5	[CEO_APPROVED_THREAD_SWITCH] token present + unapproved MC	`[CEO_APPROVED_THREAD_SWITCH] Dispatch agent to work on MC #99999 special task.`	0	(silent)
TC6	Prompt has no 5-digit MC ID at all	`Dispatch agent to review the documentation and run tests.`	0	(silent)
TC7	ACTIVE_THREAD block present but contains no parseable #NNNNN IDs	`Dispatch agent to work on MC #99999.`	0	`[active-thread-lock] No ACTIVE_THREAD block or no MC IDs found in session-state.md — fail-open.`

7. How to Update ACTIVE_THREAD When Starting a New Master Thread

The hook reads `~/.claude/session-state.md` fresh on every dispatch. No restart or cache clear is needed — edits take effect on the very next dispatch call.

Operational Procedure

1. Open `~/.claude/session-state.md`.
2. Find or create the `## ACTIVE_THREAD:` block at the top of the file, before any archived thread sections or `---` separators.
3. Write the block in the format below, listing every approved child MC ID using the `#NNNNN` pattern anywhere in the block (the hook scans the full block for all such patterns).
4. Save. The hook picks up the new state automatically on the next dispatch.

Example Block Format (Actual from Current Session)

```
## ACTIVE_THREAD: system-uvezivanje-master (CEO approved 2026-05-02 23:55)

**Spec:** ~/system/specs/system-uvezivanje-master-2026-05-02.md
**Master MC:** #10612
**SEQUENCE:** B -> C -> A
**CURRENT_STEP:** B
**LAST_COMPLETED:** (none)

**Children (CEO answers 2026-05-03 to ai-factory-pipeline.md §6):**
1. #99012 [H] Blueprint-check Phase 3 build
2. #99013 [H] alai-hooks Kotlin source check-in
3. #99014 [H] active-thread-lock hook
4. #99015 [H] one-ceo-turn-mc-cap.sh counter fix
5. #99016 [H] Migrate duplicate bash gates to Kotlin

**DRIFT-STOP:** Any task outside ACTIVE_THREAD = STOP, write memo, ask CEO.
Override = explicit CEO [CEO_APPROVED_THREAD_SWITCH] token in CEO message.
```

Adding a new approved MC mid-session: Append a line with `#NNNNN` to the block. The hook includes it on the next dispatch.

Closing a thread: Archive the block by moving it below a `---` separator or rename the heading to `## ARCHIVED:`. With no active `## ACTIVE_THREAD:` block, the hook is fully fail-open and imposes no constraint.

8. Wiring

Position in settings.json

File: `~/.claude/settings.json`. Event: `PreToolUse`. Matcher: `Task|Agent|WebSearch|WebFetch`. The hook is at index position **4** (0-indexed) within the matcher block, sitting after `pre-dispatch-gate.sh` (index 3) and before `john-max-depth-gate.sh` (index 5).

```
PreToolUse — matcher: Task|Agent|WebSearch|WebFetch
  [0] bash ~/.claude/hooks/lock-john-dispatch-cap.sh
  [1] ~/.claude/hooks/claude-hooks pre         (Kotlin alai-hooks binary)
  [2] bash ~/.claude/hooks/pre-action-da-gate.sh
  [3] bash ~/.claude/hooks/pre-dispatch-gate.sh
  [4] bash ~/.claude/hooks/active-thread-lock.sh   <-- THIS HOOK
```

```
[5] bash ~/.claude/hooks/john-max-depth-gate.sh
[6] bash ~/.claude/hooks/one-ceo-turn-dispatch-cap.sh
```

Hook Artifact Details

Field	Value
Path	`~/.claude/hooks/active-thread-lock.sh`
Size	3984 bytes (104 lines)
sha256	`e3c7ce8b8b1cb45968e368a4e7872df923f1af97a37303296ecb5cf28bf6fb79`
Language	Bash + inline Python 3 (consistent with all other ALAI hooks)
Activated	2026-05-03

9. Genesis MC + Commits

- **Genesis MC:** #99014 [H] active-thread-lock hook — 4th anti-drift layer. Estimated effort: 2h. CEO authorized via ai-factory-pipeline.md §6 Q3 "Da" (2026-05-03).
- **Master MC:** #10612 system-uvezivanje-master umbrella. Approved children: #99012, #99013, #99014, #99015, #99016.
- **Spec authority:**
  - `~/system/specs/system-uvezivanje-master-2026-05-02.md` §4 — anti-drift mechanism (four-layer architecture)
  - `~/system/specs/ai-factory-pipeline.md` §6 — CEO Q&A gate matrix with Q3 directive
- **Proveo evidence:** `/tmp/evidence-99014-proveo/verdict.txt`, `replay-log.txt`, `comparison.txt`. 7/7 PASS. sha256 confirmed: `e3c7ce8b`.
- **Related ZAKON:** ZAKON #27 (one product per session) — this hook is the machine enforcement of that written rule. ZAKON #28 (max-depth boundary) — sibling hook `john-max-depth-gate.sh` at position 5 in the same matcher block.
- **Root-cause feedback memo:** `feedback_drift_after_step1_completion.md` (2026-05-02) — the documented CEO correction event that precipitated this hook.

Revision #2
Created 2026-05-03 09:56:36 UTC by John
Updated 2026-06-07 20:01:02 UTC by John