Mehanik Phase 2 — Pre-Dispatch Gate System

Mehanik Phase 2 — Pre-Dispatch Gate System

Status: LIVE since 2026-04-25 (MC #9231 deploy)
Reference: Root-cause analysis MC #9223, synthesis at `/tmp/9223-final-synthesis.md`
Author: Sentinel-Architect + Petter Graff (CodeCraft)
Commissioned By: CEO after Drop incident (MC #8763) + Drain worker incident (MC #8602)

Overview

The Mehanik Phase 2 system is a deterministic pre-dispatch gate that mechanically enforces 7 checks before any Task tool invocation can proceed. It replaces the prior Phase 1 configuration (advisory warnings only) with hard blocking (exit 2) when preconditions are not met.

Core principle: "Prompt rules are comments. Pre-dispatch gates are code." — Chip Huyen, Section 5.3

The system consists of three components:

1. **Mehanik agent** (`~/.claude/agents/mehanik.md`) — LLM-based qualitative verification workflow (GOTCHA phases)
2. **Pre-dispatch hook** (`~/.claude/hooks/pre-dispatch-gate.sh`) — Deterministic quantitative enforcement (7 checks)
3. **Marker file schema** (`/tmp/mehanik-cleared-{task_id}`) — 13-field structured state carrier

How it works: John calls `/mehanik "{task}" {project_path} {mc_task_id}` → Mehanik runs GOTCHA verification → writes structured marker file → pre-dispatch hook validates marker on every Task dispatch → blocks if invalid or absent.

---

1. What the gate enforces (7 checks)

The hook (`~/.claude/hooks/pre-dispatch-gate.sh`) performs the following checks in order. All checks are deterministic (no LLM calls). Every check uses file existence, integer arithmetic, regex match, or grep.

Check #	Condition	Exit Code	Error Message	Rationale
1	`TOOL_NAME == "Task"`	0 (pass-through)	N/A	Only Task dispatches are gated. WebSearch/WebFetch pass through for now.
2	MC task ID present in dispatch prompt	2	`BLOCKED: No MC task ID in dispatch prompt.`	Every dispatch must be tracked in Mission Control. Prevents ad-hoc unbounded work.
3	Marker file exists at `/tmp/mehanik-cleared-{id}`	2	`BLOCKED: No Mehanik clearance for MC #{id}. Run: /mehanik ...`	John must obtain clearance BEFORE dispatch. Forces GOTCHA workflow.
4	Marker not stale (< 4 hours old)	2	`BLOCKED: Mehanik clearance for MC #{id} is stale ({age}s old).`	Session boundary enforcement. Re-verification required for resumed tasks.
5	Marker has required fields: `timestamp:`, `ceo_item_count:`, `approved_agents:`, `orchestration_surface:`	2	`BLOCKED: Marker missing field '{field}'. Mehanik must be re-run.`	Schema enforcement. Incomplete marker = incomplete verification.
6	Scope ceiling: `approved_subtask_count <= ceo_item_count + 2`	2	`BLOCKED: Scope ceiling exceeded — {approved} subtasks, ceiling is {ceiling} (CEO items: {ceo} + 2).`	Prevents scope creep via hard arithmetic ceiling. Petter taxonomy Category B mitigation.
7	Research dispatches contain `TOOL_CONTRACT:` block (if prompt matches `research|discover|partner|contact list|shortlist`)	2	`BLOCKED: Research dispatch missing TOOL_CONTRACT block. Use: wrap-with-tool-contract.js`	Prevents silent LLM fallback on tool failure (Proxima incident 2026-04-24). Category D mitigation.

Exit code semantics:

- `exit 0`: All checks pass. Task dispatch proceeds.
- `exit 2`: One or more checks failed. Platform blocks Task execution. John must fix the blocking condition and retry.

- No `exit 1` is used (reserved for hook infrastructure errors).

Execution time: < 500ms (all local file operations, no network calls). Proveo regression suite verifies this (Test watchdog, see Section 4).

---

2. 13-field marker schema

The marker file written by Mehanik at `/tmp/mehanik-cleared-{task_id}` must contain exactly 13 fields. The pre-dispatch hook validates field presence via grep (not LLM parsing). Each field is a single line with `key: value` format.

Field	Type	Example Value	Source	Purpose
`timestamp`	ISO8601	`2026-04-25T14:32:00Z`	Mehanik session time	Staleness check (Check 4)
`task_id`	Integer	`9223`	MC task ID passed to Mehanik	Task binding
`project_path`	Absolute path	`/Users/makinja/ALAI/products/Drop`	Mehanik input	Documentation path verification
`blueprint_read`	Absolute path or `N/A`	`/Users/makinja/ALAI/products/Drop/BUILD-BLUEPRINT.md`	Mehanik Phase T verification	ZAKON #18 enforcement (Documentation Bypass, Category C)
`deploy_map_read`	Absolute path or `N/A – not deploy task`	`/Users/makinja/ALAI/products/Drop/DEPLOY-MAP.md`	Mehanik Phase T verification	ZAKON PI2 Check 1 enforcement
`deploy_path_summary`	One-line string	`"Docker build -> ECR push -> aws apprunner start-deployment"`	Mehanik Phase T GOTCHA output	Forces John to demonstrate documentation was READ and PROCESSED (not just skimmed)
`ceo_item_count`	Integer	`5`	Parsed from `mc.js show {id}` output	Scope ceiling baseline (Check 6)
`approved_subtask_count`	Integer	`6`	Mehanik Phase O count	Scope ceiling numerator (Check 6)
`ceiling`	Integer	`7`	Computed: `ceo_item_count + 2`	Scope ceiling reference (Check 6 re-verifies with shell arithmetic)
`approved_agents`	Comma-separated specialist names	`Vizu/Brad-Frost, Proveo/Angie-Jones, Skillforge`	Mehanik Phase A + specialist-mapping.json cross-reference	Prevents generic "builder" dispatches (specialist routing enforcement)

Field	Type	Example Value	Source	Purpose
`orchestration_surface`	Enum	`one-shot-Task`	Mehanik Phase O reads `~/system/rules/orchestration-surface.md`	Forces routing decision to be documented (Gap 4 mitigation)
`tool_contract_required`	Boolean	`false`	Mehanik Phase O classification	Check 7 input (research task flag)
`mehanik_session_id`	String	`claude-session-abc123`	`${CLAUDE_SESSION_ID:-unknown}`	Post-hoc audit (session-ledger can verify Mehanik ran in that session)

Field rules:

- `blueprint_read`: Absolute path if file was Read in the session; `N/A` only for system-path tasks exempt per `~/system/BUILD-BLUEPRINT.md`.
- `deploy_map_read`: Absolute path if deploy task; `N/A — not deploy task` otherwise.
- `deploy_path_summary`: One line only — summarizes the actual deploy mechanism verified in Phase T (not hypothetical/memorized).
- `ceo_item_count`: Counted from `mc.js show` output — explicit enumerated deliverables only, not inferred.
- `ceiling`: Always `ceo_item_count + 2` (computed with shell arithmetic, not LLM estimate).
- `approved_agents`: Only agents present in `~/system/agents/specialist-mapping.json` — no generic "builder" or "minion".
- `mehanik_session_id`: Run `echo ${CLAUDE_SESSION_ID:-unknown}` to capture the value.

Schema version: 2.0 (as of MC #9231 deploy). Prior markers (Phase 1) contained only a timestamp and are rejected by Check 5.

---

3. How to obtain Mehanik clearance

When to call Mehanik

Per CLAUDE.md decision tree (Step 2), Mehanik is **MANDATORY** before any specialist agent dispatch for:

- Build tasks (new feature, enhancement, refactor)
- Fix tasks (bug fix, UX fix, performance fix)
- Deploy tasks (production, staging, demo)
- Infra tasks (new service, migration, CI/CD change)

Exception: System-path tasks (file location `~/system/*`) are exempt per `~/system/BUILD-BLUEPRINT.md` but still require MC task ID.

Command syntax

```
/mehanik "{task description from CEO or MC task}" {project_path} {mc_task_id}
```

Example:

```
/mehanik "Fix 5 Drop demo bugs + deploy role-based UX to prod"
/Users/makinja/ALAI/products/Drop 8763
```

What Mehanik does

Mehanik runs a 6-phase GOTCHA workflow (cannot skip phases — agent definition enforces):

1. **Phase G (GOALS):** Verify MC task exists via `mc.js show {id}`, count CEO-requested deliverables.
2. **Phase O (ORCHESTRATION):** Read `orchestration-surface.md`, classify surface, count proposed subtasks, enforce scope ceiling (subtasks ≤ CEO items + 2).
3. **Phase T (TOOLS):** Verify BUILD-BLUEPRINT.md + DEPLOY-MAP.md exist and have been read, extract deploy path for deploy tasks (via curl/git log/gh run list).
4. **Phase C (CONTEXT):** Run `discover.js "{project}"`, read MEMORY-products.md, verify specialist-mapping.json routing.
5. **Phase H (HARD PROMPTS):** Read CLAUDE.md + john-operating-system.md + zakon-pi2-deploy-verification.md (documentation only, never blocks).
6. **Phase A (ARGS):** For each proposed subtask: verify owner agent name in specialist-mapping.json, concrete input files/commands, acceptance criteria, dependencies.

Each phase produces a `[PASS|FAIL|WARN|RECORDED]` entry in the structured GATE REPORT.

Mehanik output: GATE REPORT

```
=== MEHANIK GATE REPORT ===
Task: {mc_task_id} — {title}
Project: {path}
Timestamp: {ISO8601}

Phase G (GOALS):        [PASS|FAIL] — CEO items: {N}
Phase O (ORCHESTRATION): [PASS|FAIL] — surface: {type}, subtasks: {M}, ceiling: {N+2}
Phase T (TOOLS):        [PASS|FAIL] — blueprints read: {list}
```

```
Phase C (CONTEXT):      [PASS|WARN] — discover.js output: {summary}
Phase H (HARD PROMPTS): [RECORDED] — rules indexed: {list}
Phase A (ARGS):         [PASS|FAIL] — agents: {list with owner+inputs}

Circuit Breakers:
  [✓|✗] 1. MC task exists
  [✓|✗] 2. Blueprints read
  [✓|✗] 3. Scope within ceiling
  [✓|✗] 4. No infra hallucination
  [✓|✗] 5. CI green (if deploy)

VERDICT: [CLEAR TO DISPATCH | BLOCKED]
```

If `VERDICT: BLOCKED` → precise list of blocking items + fix actions. John MUST address all blocks and re-run Mehanik.

If `VERDICT: CLEAR TO DISPATCH` → Mehanik writes the 13-field marker file to `/tmp/mehanik-cleared-{task_id}`. The pre-dispatch hook will now allow Task dispatches for this task ID (until marker expires at 4h or session ends).

How to read GATE REPORT failures

Example 1 — Scope creep catch:

```
Phase O (ORCHESTRATION): [FAIL] — surface: one-shot-Task, subtasks: 11, ceiling: 3

Circuit Breakers:
  [✓] 1. MC task exists
  [✓] 2. Blueprints read
  [✗] 3. Scope within ceiling — 11 subtasks proposed, ceiling is 3 (CEO items: 1 + 2)
  [✓] 4. No infra hallucination
  [✓] 5. CI green

VERDICT: BLOCKED — Scope ceiling exceeded. Reduce to ≤3 subtasks or split into multiple
sprints.
```

Fix: Re-plan with ≤3 subtasks, OR escalate to CEO for approval to increase scope, OR split into 2 MC tasks.

Example 2 — Missing blueprint:

```
Phase T (TOOLS): [FAIL] — blueprints read: none

Circuit Breakers:
  [✓] 1. MC task exists
  [✗] 2. Blueprints read — BUILD-BLUEPRINT.md not Read in session
  [✓] 3. Scope within ceiling
  [✓] 4. No infra hallucination
  N/A 5. CI green (not deploy task)

VERDICT: BLOCKED — Read BUILD-BLUEPRINT.md before dispatch (ZAKON #18).
```

Fix: `Read /Users/makinja/ALAI/products/{project}/BUILD-BLUEPRINT.md`, then re-run `/mehanik`.

Example 3 — Infra hallucination:

```
Phase T (TOOLS): [FAIL] — blueprints read: BUILD-BLUEPRINT.md, DEPLOY-MAP.md
Deploy path documented: Docker -> ECR -> apprunner
Proposed subtask "Build staging environment (GCP Cloud Run + Terraform)" NOT documented in
DEPLOY-MAP.md.

Circuit Breakers:
  [✓] 1. MC task exists
  [✓] 2. Blueprints read
  [✓] 3. Scope within ceiling
  [✗] 4. No infra hallucination — staging env not documented, inferred from LLM memory
  [✓] 5. CI green

VERDICT: BLOCKED — Infra hallucination detected. Verify staging exists or remove from plan.
```

Fix: Check DEPLOY-MAP.md. If staging is documented → update plan. If NOT documented → remove staging subtask OR escalate to CEO for approval to build new infra.

---

4. Regression suite

Location: `~/system/tests/pre-dispatch-gate-tests.sh`

Purpose: Proveo/Angie Jones acceptance test suite for `pre-dispatch-gate.sh` (MC #9233). Verifies all 7 checks produce expected exit codes under 5 scenarios.

How to run

```
bash ~/system/tests/pre-dispatch-gate-tests.sh
```

Expected output:

```
pre-dispatch-gate regression suite — MC #9233
Hook: /Users/makinja/.claude/hooks/pre-dispatch-gate.sh
------------------------------------------------
PASS  [T1] No MC ID in input (exit 2)
PASS  [T2] MC ID but no marker file (exit 2)
PASS  [T3] Scope ceiling exceeded (8 subtasks, ceiling 5) (exit 2)
PASS  [T4] Research dispatch missing TOOL_CONTRACT block (exit 2)
PASS  [T5] Valid happy path (real marker #9233) (exit 0)
------------------------------------------------
5/5 PASS
```

Exit code: 0 if all tests pass, 1 if any test fails.

5 test scenarios

Test #	Scenario	Setup	Expected Exit	Hook Check Tested
T1	No MC ID in input	Task dispatch prompt: `"random task no id"` (no `MC #XXXX` pattern)	2 (BLOCKED)	Check 2 (MC ID extraction)
T2	MC ID present but no marker file	Task dispatch for MC #99999, but `/tmp/mehanik-cleared-99999` does not exist	2 (BLOCKED)	Check 3 (marker existence)
T3	Scope ceiling exceeded	Marker with `ceo_item_count: 3`, `approved_subtask_count: 8`, ceiling=5 → 8 > 5	2 (BLOCKED)	Check 6 (scope arithmetic)
T4	Research dispatch without TOOL_CONTRACT	Marker valid (scope OK), but prompt contains `"shortlist"` (research keyword) and no `TOOL_CONTRACT:` block	2 (BLOCKED)	Check 7 (tool contract)

Test #	Scenario	Setup	Expected Exit	Hook Check Tested
T5	Valid happy path	Real marker `/tmp/mehanik-cleared-9233` (written by Mehanik this session), fresh (< 4h), all fields present, scope OK, no research keywords	0 (CLEARED)	All checks pass

Test isolation: Tests use fake MC IDs (99997, 99998, 99999) far outside real ID range. Real markers are never touched by the test suite. Cleanup runs before and after test execution.

Performance validation: Proveo suite includes a watchdog test (not yet in the current script — planned for Phase 3):

```
time bash ~/.claude/hooks/pre-dispatch-gate.sh
# Assert execution < 500ms
```

This ensures the hook does not timeout (cc-guide-primitives.md: "Hook timeout limits 5-10s default").

---

5. Failure modes covered

This section maps Petter Graff's 7-category failure taxonomy (`/tmp/9223-petter-taxonomy.md`) to the Mehanik Phase 2 enforcement mechanisms. It also identifies which categories remain process gaps (not addressable by hooks).

Category A — Pattern Completion Override

Definition: LLM generates a "correct-looking" completion based on training priors rather than project-specific state. The model recognizes a surface-level pattern ("deploy request") and routes to a memorized solution path ("fintech needs staging") without verifying if that path applies to THIS project.

Evidence: Drop incident (MC #8763) — John activated staging/CI/infra track from memory, never read BUILD-BLUEPRINT.md or DEPLOY-MAP.md which documented the actual 3-command deploy path.

Mehanik coverage:

- **Hook-enforced:** Check 3 (marker existence) + Check 5 (blueprint_read field presence) → forces Mehanik Phase T to run, which forces BUILD-BLUEPRINT.md read.

- **Mehanik Circuit Breaker 2:** Blocks if BUILD-BLUEPRINT.md not Read in session.
- **Demonstration forcing function:** `deploy_path_summary` field in marker requires John to produce a one-line deploy path — cannot be satisfied by skimming, must be extracted from documentation.

Remaining gap: Mehanik is an LLM agent. It can read BUILD-BLUEPRINT.md and still activate a training prior if the prior is strong enough. Mitigation: `deploy_path_summary` field must be verified by the hook in Phase 3 (compare against a static deploy-path registry, not LLM extraction). Currently the hook only checks field *presence*, not field *correctness*.

Status: Substantially closed. Pattern completion can still occur inside Mehanik itself, but the forcing function (structured summary) + scope ceiling make it harder to proceed with hallucinated infra at scale.

Category B — Scope Expansion Without Authorization

Definition: Agent expands task scope beyond explicit authorization, treating discovered gaps as implicit authorization to fix them. Each gap triggers a new dispatch rather than escalation.

Evidence: Drop incident — 11 agents for a 5-bug fix. Each gap (staging absent, CI workflows not pushed, secrets missing) triggered a new subtask.

Mehanik coverage:

- **Hook-enforced:** Check 6 (scope ceiling re-verification) — deterministic arithmetic, not LLM count. `approved_subtask_count <= ceo_item_count + 2`. Exit 2 if violated.
- **Mehanik Circuit Breaker 3:** Blocks if proposed_subtasks > ceiling.

Remaining gap: None for dispatch-time enforcement. However, an agent working INSIDE an approved subtask can still call additional specialists (nested dispatch). This is not currently gated. Requires Phase 3 extension: nested Task calls must also be marker-gated.

Status: CLOSED for top-level dispatch. Open for nested calls.

Category C — Documentation Bypass

Definition: Agent proceeds without reading project documentation (BUILD-BLUEPRINT.md, DEPLOY-MAP.md, RUNBOOK.md). LLM priors substitute for actual project state.

Evidence: Drop incident — John did not read any of the 3 docs. Drain worker (MC #8602) — specialists designed based on "assumptions about LightRAG behavior, not empirical

measurements."

Mehanik coverage:

- ✅ **Hook-enforced:** Check 5 (blueprint_read field presence) — marker schema requires absolute path to BUILD-BLUEPRINT.md.
- ✅ **Mehanik Circuit Breaker 2:** Blocks if file not Read in session.
- ✅ **Mehanik Phase T:** Reads each doc and summarizes contents. For deploy tasks: verifies deploy path with tool commands (curl, git log, gh run list).

Remaining gap: Mehanik verifies the file was Read. It does not verify the content was USED. John could Read the file and ignore it. Mitigation: `deploy_path_summary` field forces extraction (not just reading). But this is only for deploy tasks — non-deploy tasks have no equivalent forcing function yet.

Status: Substantially closed for deploy tasks. Partially open for non-deploy tasks (read is verified, usage is not).

Category D — Silent Fallback on Tool Failure

Definition: When a required tool is unavailable, the agent does not halt — it silently substitutes LLM memory, marks output as verified, and delivers it upstream.

Evidence: Proxima HR research (2026-04-24) — `web-search.sh` unavailable, fabricated contact names, labeled "tool-verified", reached CEO.

Mehanik coverage:

- ✅ **Hook-enforced:** Check 7 (research dispatches require TOOL_CONTRACT block) — if prompt contains research keywords (`research|discover|partner|contact list|shortlist`) and no `TOOL_CONTRACT:` block, exit 2.
- ⚠ **Partial:** `~/system/rules/tool-contract-zakon.md` + `~/system/hooks/pre-publish-validate.sh` exist (CEO-facing output integrity check). But these are separate hooks, not integrated into pre-dispatch-gate.

Remaining gap: If the TOOL_CONTRACT block is present but the subagent is in a context where the hook is not loaded, silent fallback can still occur. Enforcement depends on John including the TOOL_CONTRACT block in the dispatch prompt. The hook verifies John did it, but cannot prevent a subagent from ignoring it if the subagent's hook environment is misconfigured.

Status: Substantially closed for dispatch-time (Check 7). Runtime enforcement (inside subagent) remains a hook registration gap.

Category E — Gate Timing Inversion

Definition: Enforcement gates fire AFTER damage is done (post-action) rather than BEFORE action is taken (pre-action). Rules exist but are checked at completion checkpoints, not at initiation.

Evidence: ZAKON PI2 gate fires at `mc.js done`. By that point, 11 agents dispatched, 6 hours spent. plan-completeness-gate fires on `*-plan.md` saves, not on dispatch.

Mehanik coverage:

- ✅ **Fully closed:** Pre-dispatch-gate.sh fires on `PreToolUse` hook (BEFORE Task execution). Check 3 (marker existence) is the gate — no marker = no dispatch.
- ✅ **ZAKON PI2 Check 0 added (2026-04-25):** Deploy tasks now require Mehanik marker BEFORE curl preflight (see `~/system/rules/zakon-pi2-deploy-verification.md` lines 26-47).

Remaining gap: None for dispatch. However, the zakon-pi2 enforcement hook (for deploy commands like `aws apprunner start-deployment`) is not yet registered in settings.json. It is documented but not wired. Planned for Week 2 (Phase 3, per synthesis Section 4).

Status: CLOSED for dispatch. Partially open for deploy execution (wire zakon-pi2 hook).

Category F — Semantic Signal Misinterpretation

Definition: Agent correctly reads a signal but applies the wrong semantic interpretation. Diagnostic value treated as actionable gate condition, or vice versa.

Evidence: Drain worker Bug 2 (MC #8602) — `pipeline_busy: true` is server-internal diagnostic, treated as client-side blocking signal. Bug 3 — queue depth should gate adapters (inflow), instead gated drain worker (outflow), creating deadlock.

Mehanik coverage:

- ❌ **NOT COVERED by hooks.** This is a design-quality problem, not a dispatch-time problem. The code is syntactically correct, passes FINAL-REVIEW, passes Proveo Phase 1 (functional smoke test). Fails only under load.

Process mitigation (NOT hook-enforced):

- ⚠ **Week 3 planned (Section 4 of synthesis):** Extend FINAL-REVIEW checklist with "gate logic semantic review" — for every gate condition, verify: is this signal a diagnostic or an actionable state? What is the semantic role of this component (producer/consumer/gate)?
- ⚠ **Mehanik Phase A extension (planned):** Add field `signal_semantics_verified_by: [specialist name]` for each integration component.

Status: NOT ADDRESSABLE by pre-dispatch gate. Remains a specialist review scope gap (Category G).

Category G — Review Scope Blindness

Definition: Formal review processes (FINAL-REVIEW, Proveo validation) are scoped too narrowly. They verify what they were told to verify (credentials, naming, functional smoke tests) and do not challenge semantic correctness of design decisions outside their explicit checklist.

Evidence: Petter's FINAL-REVIEW on drain worker covered credential fallback, metric naming, lease recovery timing. Did NOT cover: semantic correctness of gate conditions, role-based gate logic, empirical validation of timeout constants.

Mehanik coverage:

- ❌ **NOT COVERED by hooks.** Review scope is a process design problem.

Process mitigation (NOT hook-enforced):

- ⚠ **Week 3 planned (Section 4 of synthesis):** Extend FINAL-REVIEW template with:
  - "Empirical validation of timeout/threshold constants — cite measurement source (e.g., observed p99 latency)."
  - "Gate logic semantic review — verify signal semantics, gate role, component role."
- ⚠ **Proveo Phase 2 (pressure testing) added to plan-completeness-gate:** Not yet enforced. Planned: every plan with Proveo Phase 1 (functional) must also include Proveo Phase 2 (load/pressure).

Status: NOT ADDRESSABLE by pre-dispatch gate. Requires FINAL-REVIEW + Proveo checklist expansion (process change, not code change).

Summary Table — Coverage by Category

Category	Name	Hook-Enforced?	Mehanik Circuit Breaker?	Remaining Gap	Phase 3 Mitigation
A	Pattern Completion Override	⚠ Partial (Check 3, 5)	⚠ CB#2 (blueprint read)	deploy_path_summary correctness not verified (only presence)	Verify summary against static registry
B	Scope Expansion	✅ Full (Check 6)	✅ CB#3 (scope ceiling)	Nested Task calls not gated	Gate nested dispatches

Category	Name	Hook-Enforced?	Mehanik Circuit Breaker?	Remaining Gap	Phase 3 Mitigation
C	Documentation Bypass	✅ Full (Check 5)	✅ CB#2 (blueprint read)	Non-deploy tasks: read verified, usage not verified	Forcing function for non-deploy (TBD)
D	Silent Tool Fallback	⚠️ Partial (Check 7)	⚠️ Mehanik Phase O classification	Subagent runtime enforcement (hook registration)	Register TOOL_CONTRACT hook globally
E	Gate Timing Inversion	✅ Full (PreToolUse)	✅ All CBs fire pre-dispatch	zakon-pi2 deploy hook not wired	Register zakon-pi2 Bash hook (Week 2)
F	Semantic Signal Misinterpretation	❌ No	❌ No	Specialist review scope	FINAL-REVIEW checklist + Mehanik Phase A field
G	Review Scope Blindness	❌ No	❌ No	FINAL-REVIEW + Proveo scope	Checklist expansion (Week 3)

Verdict: Categories A-E are substantially or fully closed by Mehanik Phase 2. Categories F-G remain open and require process design changes (review checklists), not hook enforcement. This is expected — per Petter taxonomy Section 4: "The system needs fewer rules and more counters, file reads, and arithmetic checks at the dispatch boundary. Rules describe what should happen. Gates enforce what will happen." Categories F-G are about what happens INSIDE the work (design quality), not about preventing hallucinated dispatch.

Related Documentation

- **Root-cause synthesis:** `/tmp/9223-final-synthesis.md` — Authoritative spec for Mehanik Phase 2 (372 lines, Sentinel-Architect)
- **Failure taxonomy:** `/tmp/9223-petter-taxonomy.md` — 7 categories, recurrence map, root-cause chain (Petter Graff)
- **Hook implementation:** `~/.claude/hooks/pre-dispatch-gate.sh` — Live code (65 lines)
- **Mehanik agent:** `~/.claude/agents/mehanik.md` — GOTCHA workflow definition
- **Regression suite:** `~/system/tests/pre-dispatch-gate-tests.sh` — 5 test scenarios
- **ZAKON PI2:** `~/system/rules/zakon-pi2-deploy-verification.md` — Deploy verification protocol (Check 0 added 2026-04-25)
- **CLAUDE.md decision tree:** `~/.claude/CLAUDE.md` — Step 2 (CALL MEHANIK) mandatory gate
- **Orchestration surface routing:** `~/system/rules/orchestration-surface.md` — Decision table for DAG vs chains vs factory vs one-shot

- **Tool contract enforcement:** `~/system/rules/tool-contract-zakon.md` — Research task LLM fallback prevention

---

Change Log

- **2026-04-25:** Phase 2 activated (MC #9231). pre-dispatch-gate.sh exit 0 → exit 2 (blocking). Marker schema upgraded to 13 fields. Mehanik agent updated to write structured marker. Regression suite deployed (MC #9233). Documentation synced to BookStack (MC #9237).
- **2026-04-24:** Phase 1 deployed (advisory warnings only). Hook registered in settings.json but exit 0 (non-blocking).

Credits

- **Sentinel-Architect** — Final synthesis, marker schema design, hook specification
- **Petter Graff (CodeCraft)** — Failure taxonomy, root-cause chain, Category A-G analysis
- **Chip Huyen** — LLM failure mechanism analysis, τ-bench data, "Prompt rules are comments" principle
- **Mehanik agent proposal** — `~/system/rules/mechanical-agent-proposal.md` (9 gaps, GOTCHA origin)
- **Kelsey Hightower (FlowForge)** — Hook implementation (MC #9230)
- **Angie Jones (Proveo)** — Regression suite (MC #9233)
- **Skillforge** — This documentation (MC #9237)

Mehanik does not replace judgment. Mehanik replaces the absence of mechanical checks.
John still decides. Mehanik prevents John from deciding based on hallucination.

Revision #2
Created 2026-04-25 04:05:10 UTC by John
Updated 2026-05-31 20:06:34 UTC by John