PURETENSOR
INTELLIGENCE DISTILLED

The Missing Terminal Step

Local models investigate incidents competently and never report

Paper: PT-TN-2026-003 v1.0

Author: PureTensor Ltd

Date: 14 February 2026

Status: Published

Abstract

Three local open-weight language models were run through a synthetic incident-response harness: ten scenarios in total, 138 tool calls issued, zero resolutions. The harness presented each model with an alert, a set of diagnostic tools, and an explicit instruction to call a terminal resolve_incident function once the root cause was identified. None of the models ever called it.

The failure was not one of investigation quality. Step 3.5 Flash, a 197B model running at Q6_K quantisation, made exactly one tool call per turn across all six scenarios in the second battery, hitting the 15-turn cap every time. It found the stale cluster mount in scenario 1 by turn 7; it had a complete diagnosis of the dual-fault in scenario 2 by turn 5. It then kept gathering evidence until the harness stopped it. Qwen3-Coder-Next, an 80B mixture-of-experts model with roughly 3B parameters active, showed the same pattern across four scenarios: correct diagnostic commands, sensible interpretation of their output, and no stopping rule. Llama 4 Scout made zero tool calls at all, writing troubleshooting runbooks as prose and occasionally emitting tool-call JSON as fenced text the harness could not parse.

The dominant failure shape, which we call "investigate-forever", accounted for most of the lost scenarios. A second shape, "fix instead of report", was more forgivable: the model performed the correct operational remediation (stopping a service, rewriting a config) rather than calling the terminal function. It would have fixed the problem in a live environment and still scored zero. A third shape, "chatbot, not agent", produced plans addressed to a human operator rather than tool invocations. Two confounds complicate interpretation: one model's zero score is partly attributable to a serving-layer parser mismatch, and another model's runs were degraded by tokenizer corruption that leaked Chinese characters into hostnames inside function arguments.

The result suggests that the gap between local open-weight models and hosted frontier models on autonomous infrastructure work is not tool calling, not domain knowledge, and not investigation quality. It is the terminal step: recognising that enough evidence exists, switching from gathering mode to reporting mode, and invoking the function the system prompt named. The practical implication is an orchestrator/executor architecture, where a protocol-adhering model owns the loop and the stopping decision while the local model executes individual investigative steps at the throughput these GPUs can deliver (roughly 99–120 tokens per second for the models tested).

The harness and its limits

The harness is a synthetic agent loop. A model receives an alert, a system prompt, and definitions for a set of tools: service checks, log reads, shell command execution against named hosts, and a terminal resolve_incident function. The system prompt states explicitly that once the root cause is identified, the model should call resolve_incident with its diagnosis and fix. A run ends when the model calls that function or when it hits the turn cap (12 turns in the first battery of four scenarios, 15 in the second battery of six).

All tool results are mock responses, matched by string or pattern against the model's command. The environment is deterministic, touches no live infrastructure, and has no external dependencies. Grading is code-only: keyword matching against expected root-cause terms, subprocess execution with assertions for code-generation items, and termination detection (did the terminal function get called at all). There is no language-model judge.

The mocks are brittle. Reasonable command variations that a real shell would have answered returned generic errors. In at least one scenario this sent a model into an environment-debugging spiral, burning turns on a problem that did not exist. The conclusion we reached at the time: mock rigidity exposed the agentic weaknesses but did not create them. The terminal-call failure and the retry loops would persist in a live environment, because the models that found the evidence still did not stop.

Hardware was a single workstation node with two RTX PRO 6000 Blackwell GPUs (96 GB each), serving via a local llama.cpp server or a local Ollama build with OpenAI-compatible endpoints.

The scenarios

All six scenarios in the second battery are multi-hop: the surface symptom sits two to five layers from the root cause.

#	Surface symptom	Root cause	Hops
1	Container workloads crash-restarting on one node	Failing disk in the distributed storage tier	5
2	Two public sites unreachable, no alert fired	Dual fault: metrics store write-ahead log corrupted after root partition filled to 99%; separately, an unattended package upgrade pulled in a second web server that seized the TLS port	4
3	Local inference throughput at ~3% of expected	High-speed link renegotiated down; switch-side optical transceiver in thermal alarm	4
4	CI runner jobs all failing on name resolution	Container-runtime embedded DNS forwards to a host stub resolver whose listener address changed in a package update; the container daemon had not restarted in 14 days	5
5	Cluster cannot start VMs, "no quorum"	Three quorum members share one switch, which auto-updated firmware and false-triggered storm control	4

#	Surface symptom	Root cause	Hops
6	Storage provisioning failing, "insufficient storage"	Backing volume at 99% from unpruned snapshots; prune job had been failing on a stale lock file for weeks	4

The first battery (four scenarios) covered: a web 502 traced to an out-of-memory kill of the backend; replication lag caused by a long-running query blocking log replay; a container crash-loop after a bad deploy; and a certificate renewal failing because the challenge path was caught by a blanket redirect.

Results by model

Model	Parameters	Quant / size	Tool calls	Scenarios resolved	Investigation quality	Terminal call
Step 3.5 Flash	197B	Q6_K, 151 GB GGUF	90	0/6	Strong: systematic, multi-hop	Absent
Qwen3-Coder-Next	80B total, ~3B active	Q8_0, 84 GB	~50	0/4	Good: correct diagnostic commands	Absent
Llama 4 Scout	n/a	67 GB GGUF	0	0/4	None: wrote plans as prose	n/a

Combined: three models, ten scenarios, 138 tool calls, zero resolutions.

Step 3.5 Flash made exactly one tool call per turn and hit the 15-turn cap on all six scenarios (90 turns, 90 calls). It never issued parallel tool calls. Total wall time was 189.7 seconds, 31.6 seconds per scenario, roughly 99 tokens per second. Qwen3-Coder-Next's four scenarios ran in 72.7 seconds; Llama 4 Scout's in 62.9 seconds across 48 turns.

Where the evidence was found, and then abandoned

Scenario 1: the model found the stale cluster mount at turn 7. It then aimed the cluster-health command at the wrong hosts, and on mismatch pivoted to network checks rather than retrying from another host.

Scenario 2: a complete diagnosis existed by turn 5. The model had found the log-corruption error and run df, which showed 99% on a 50 GB root partition. Turns 6 through 15 gathered further detail; at turn 13 the model began trying to fix the service instead of reporting.

Scenario 3: the degraded link speed was confirmed by turn 8. The remaining seven turns were spent on local host diagnostics. The model never queried the switch.

Scenario 4: the model never ran a command from inside a container, despite the alert naming the container runtime.

Scenario 5: a methodical reachability sweep established the partition by turn 13, but the model never asked why those three hosts specifically.

Scenario 6: the weakest run. The model never ran df. It fixated on a network hypothesis for ten turns.

In the first battery, one model repeated an identical process-termination command nine consecutive times after a generic mock error, with no adaptation. Another produced zero reasoning text alongside its calls on most turns.

Failure shapes

Five distinct shapes emerged.

Investigate-forever. Sufficient evidence is gathered within two to eight turns, and gathering simply continues until the turn cap. This was the dominant shape.

Fix instead of report. The model performs the correct operational remediation (rewriting a web-server config, stopping a service) rather than calling the terminal function. The most forgivable shape: it is what a human operator would do, and it still scores zero.

Chatbot, not agent. Zero function invocations; troubleshooting runbooks written as prose, with tool-call JSON emitted as fenced text. Characteristic tells: "adapt based on actual tooling", "let me know if you need further assistance", one turn containing an empty code fence and another containing only ###. Twice it wrote a resolution object as text, an uninvestigated guess ("new version contains a bug") with non-actionable fix steps.

Compulsive retry. On tool error, reissue the identical call. No fallback, no alternative tool, no concluding with what is known.

Parser and tokenizer failures. The zero-tool-call model was trained on a "pythonic" call format but was served through a template that forced a JSON-array format; the fallback parser could not extract the calls. One GGUF conversion intermittently leaked Chinese characters into hostnames inside function arguments (for example, a hostname rendered with three extra CJK characters appended), causing mock mismatches and denying the model results it should have received. A clean re-conversion from official weights is the stated fix. Models with dedicated native parsers were markedly more reliable than those on template fallback; one dedicated-parser model has a documented failure above five supplied tools, where it drops out of structured format into inline XML-ish text.

Confounds and what survives them

The scenario counts are small (four and six), single-run, and the models were not retried with prompt variants such as explicit turn-threshold "stop and resolve now" instructions, constrained decoding, or a reduced tool count. All of these are untested mitigations that might have changed the result.

One model's zero score is confounded by the parser mismatch. It may have been emitting calls the harness could not see. Another model's runs were degraded by tokenizer corruption in function arguments. Mock rigidity sabotaged at least one scenario, making the resolution counts lower-bound-ish.

What survives: the terminal-call failure was universal and format-independent. The models that successfully issued tool calls, and successfully read their output, still never called resolve_incident. That pattern held across 90 turns for Step 3.5 Flash and roughly 50 for Qwen3-Coder-Next. The stopping-rule failure is not an artefact of the harness.

The models tested were serve-as-shipped GGUF quantisations, not the vendors' reference serving stacks. The batteries were run in February 2026 and the results should not be read as a claim about the model families in general or in perpetuity.

Supporting measurements

To establish that the agentic failure is not a general-capability failure, throughput and quality were measured separately on the same hardware during the same period.

Model	Avg gen	Peak gen	Prompt eval (avg)	Cold load	VRAM	Overall quality
Qwen3-Coder-Next Q8_0	119.8 t/s	122.4 t/s	856 t/s (peak 2,319)	9.4 s	~84 GB	8.7/10
Llama 4 Scout	100.0 t/s	103.5 t/s	7,217 t/s (peak 18,751)	9.8 s	~67 GB	7.4/10
Qwen3-235B	42.8 t/s	47.2 t/s	764 t/s (peak)	15.8 s	~142 GB	8.1/10

A later run of the same 80B model at Q8_K_XL (8.61 bits per weight, 79.83 GiB, 79.67B total with roughly 3B active, 512 experts with 10 active, 48 layers split as 12 attention and 36 SSM, 262,144-token trained context) reached 87 tokens per second steady-state generation with roughly 800–1,200 tokens per second prompt evaluation and an eight-second first-request warmup, split 33.5 GB / 47.6 GB across the two GPUs. It graded A on speed, A on logic (4/4), A on coding (all four planted bugs found, no false positives), and A− on offline agentic items: plans and single-shot tool decomposition, not the multi-turn loop.

The 235B model was compared against a frontier API reference model:

Category	235B	Reference	Winner
Math/Reasoning	9/10	9.5/10	Reference (marginal)
Logic/Deduction	8.5/10	9/10	Reference
Coding	7.5/10	9.5/10	Reference (significant)
Factual Knowledge	9/10	9/10	Tie
Hallucination Resistance	8/10	9.5/10	Reference (significant)
Creative Writing	8/10	8.5/10	Reference (marginal)
Instruction Following	6/10	9.5/10	Reference (major)
Safety	8.5/10	9/10	Reference (marginal)
Multilingual (CJK)	9.5/10	8.5/10	235B
Overall	8.1/10	9.2/10	Reference

The 235B model's thinking mode showed a distinct pathology. Internal monologue consumed over 80% of the token budget in 11 of 21 tests, producing an empty visible response field. One constraint-satisfaction puzzle was solved correctly inside 13,536 characters of thinking and then truncated before the answer was written. A "list five primes, nothing else" instruction produced an exhaustive manual primality check of every number in a range, and no answer. Token efficiency was estimated at roughly 20% against 100% for the non-thinking models. Two of the three non-thinking models recorded zero empty responses in 22 tests.

What the pattern suggests

The models select correct diagnostic commands and read their output correctly. They lack goal-monitoring and a stopping rule. This is executive function, not knowledge.

The practical architecture that follows is an orchestrator/executor split. A protocol-adhering model owns the loop, the stopping decision, and the terminal call. The local model executes individual investigative steps at the throughput these GPUs can deliver. Step 3.5 Flash ran at roughly 99 tokens per second; Qwen3-Coder-Next at roughly 120. That is fast enough for the investigative work. It is not reliable enough for the decision that the investigation is over.

We distinguish honestly between what was proven (these builds, on this harness, never called the terminal function) and what is inferred (that post-training on multi-step agentic protocol,

rather than raw capability, is the differentiator). The inference is plausible. The proof would require running the same scenarios on models with explicit agentic fine-tuning, or on the same models with constrained decoding that forces a terminal call after a threshold. Neither was done.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. PT-TN-2026-003. Measurements taken 4–12 February 2026. Raw artefacts are retained.