

Agent Backbone — Production AI Agent Evaluation Rubric & Weekly Reliability Briefing (Issue #1)

Published: October 28, 2026

Publisher: Tool Use Cases (Lali Publications)

Web Edition & Free Weekly Briefing Signup: <https://tools.lalipublications.com/agent-backbone/>

Issue #1 Permalink: <https://tools.lalipublications.com/agent-backbone/issue-1/>

Editorial Contact: larrys@lalipublications.com

Executive Summary

Most AI agents look great in a five-minute demo and quietly degrade in week three of production. The failure is almost never that the foundation model "got dumber." Instead, production drift comes from five repeatable architectural traps:

1. **Silent Bad Writes Masquerading as Completed Tasks** (unconstrained writes on low-confidence transcript spans or mangled phone ASR entity capture).
2. **Tool Blindness Past ~15 Tools & Nested Mode Menus** (flat tool schemas blurring attention and triggering false human handoffs).
3. **Voice & Support Prompt FAQ Stuffing** (2,000+ token prompts causing latency spikes, ~30% rule-skipping, and invented policy exceptions).
4. **Worker + Verifier Shared-Context Rubber-Stamping & Stage-Handoff Distrust** (checker agents anchored by worker chain-of-thought, or multi-stage coding agents distrusting tool-result summaries).
5. **Multi-Turn Constraint Regression ("Whack-a-Mole" Drift) & Unconstrained State/Shell Calls** (user corrections dropping earlier hard constraints, mental-mathed timestamps, or transposed shell flags).

This guide provides the complete **Agent Backbone Production Evaluation Framework**: - **Section 1:** The 4-Criteria Weighted Evaluation Rubric (1–5 Scoring Scale) & Weakness Aggregation Method - **Section 2:** The 5 High-Frequency Production Failure Modes Checklist & Deterministic Fixes - **Section 3:** The Fixed 8-Question Regression Test Template & 4-Step Measurement Loop - **Section 4:** Agent Backbone Briefing Issue #1 — 5 Ways Production AI Agents Quietly Break (And the Deterministic Fixes)

1. The 4-Criteria Weighted Production Agent Evaluation Rubric (1–5 Scale)

Evaluating a live agent by "vibes" or single-turn spot checks hides systemic drift. Pull **8 to 15 real production outputs** (focusing on policy boundaries, multi-step instructions, noisy audio/text inputs, and multi-turn user corrections) and score every reply from **1 to 5** across four weighted dimensions:

Criterion & Weight	What It Measures	Score 1 (Failing / Unsafe)	Score 3 (Acceptable Baseline)	Score 5 (Production-Grade)
1. Accuracy & Policy Adherence (35%)	Factual precision, zero invented policy exceptions, and strict grounding in verified tool outputs or policy docs.	Invents policy windows, refund exceptions, or tool completions; states unverified guesses as fact.	States core policy accurately on standard asks, but hedges or softens hard boundaries on edge cases.	States exact policy (billing, cancellation, data retention, refunds) with zero invention; writes <code>NULL</code> or escalates when data is unverified.
2. Conciseness & Clarity (25%)	Answers only the question asked in <= 6 sentences without filler, throat-clearing, or unrequested tangents.	Length is 2x+ what is needed (10–15+ sentences); buries the answer under generic troubleshooting or restatements.	Answers the question clearly in 6–8 sentences with minor introductory filler.	Direct, scannable answer in <= 6 sentences ; numbered steps for procedural tasks; zero wasted words.
3. Tone & Brand Alignment (20%)	Consistent, calm, professional, and direct voice across every turn —no sycophancy, defensiveness, or persona drift.	Overly apologetic, robotic, or inconsistent across turns; argues with the user or uses hype/slop phrasing.	Polite and serviceable, but shifts between overly formal boilerplate and casual chat across turns.	Steady, direct, human-sounding authority on every turn, including when delivering a firm policy "no."
4. Actionability & Next Steps (20%)	Closes every reply with one concrete next step, verified state transition, or explicit policy limit/escalation path.	Ends with a vague open question ("Let me know if you need anything else!") or falsely escalates to a human when a tool exists.	Gives a next step when obvious, but leaves dead-end replies when a request hits a policy limit.	Every reply closes with the exact UI path, command, state confirmation, or hard policy boundary + escalation condition.

How to Aggregate Scores & Isolate Systemic Weaknesses

- 1. Compute the Weighted Turn Score:** $\text{Weighted Score} = (0.35 \times \text{Accuracy}) + (0.25 \times \text{Conciseness}) + (0.20 \times \text{Tone}) + (0.20 \times \text{Actionability})$
- 2. Group Failures by Frequency Into Three Tiers:**
- 3. Primary Weakness (High Frequency / High Severity — e.g., 5 of 12 outputs):** Systemic policy invention (such as inventing a 30-day post-cancel data export window or a 60-day annual refund exception), silent bad database writes, or false tool escalations.
- 4. Secondary Weakness (Medium Frequency — e.g., 4 of 12 outputs):** Reply length roughly doubled (9–12 sentences on routine how-to questions) or missing concrete closing steps.
- 5. Minor Weakness (Low Frequency — e.g., 2 of 12 outputs):** Tone inconsistency across channels (in-app chat vs. Discord/email).
- 6. Document Every Finding Side-by-Side:** For each weakness, record:
- 7. User Question**

8. Current Agent Reply (Rejected)
9. Actual Policy / Ground Truth
10. Target Agent Reply (Preferred — <= 6 sentences, exact policy, clear next step)

2. The 5 High-Frequency Production Failure Modes Checklist & Fixes

#	Failure Mode	Production Symptom	Deterministic Architectural Fix
1	Silent Bad Writes vs. NULL + Span Confidence	Voice or intake agent saves guessed/hallucinated fields on ambiguous calls that look like valid finished tasks, bypassing downstream review; phone ASR mangles caller names.	Require every write field to attach <code>transcript_span + confidence (>= 0.85)</code> or write <code>NULL</code> so exception queues flag it. For phone ASR names/IDs, require first-name + phonetic readback and CRM/caller-ID match before saving.
2	Tool Blindness Past ~15 Tools & Mode Menus	Adding tools or nested skill/mode menus causes the agent to go blind to an existing tool and falsely escalate (" <i>transferring to a human</i> ") or pick the wrong skill + mode.	Put a deterministic 2-step intent router in front of execution so each turn sees only 3–5 relevant tools and a single active mode prompt.
3	Voice/Support Prompt FAQ Stuffing	Stuffing property, location, or billing FAQs into the system prompt inflates tokens, spikes voice latency, and causes core rules to be skipped ~30% of the time.	Keep the root system contract under 400–500 tokens . Move FAQs/facts to a fast keyed lookup tool (<code>get_fact_by_id</code>) and enforce must-follow rules with a post-turn schema assertion.
4	Worker + Verifier Shared-Context Rubber-Stamping	Checker agent agrees with the worker's hallucination at 2 AM; or a coding agent's solving stage distrusts a localization summary inside a tool result and re-localizes from scratch.	Give the verifier only the task spec + raw tool outputs (never the worker's reasoning). Across stages, inject a typed <code><verified_stage_state></code> JSON block in the system header with a fresh message array and stage-gated tools.
5	Multi-Turn Constraint Regression ("Whack-a-Mole")	Each user correction fixes the new complaint but silently drops an earlier hard constraint (e.g., budget table or no-car rule); or the agent mental-maths time / transposes shell flags.	Re-inject a pinned <code>constraints_ledger</code> JSON block at the bottom of every turn and run a post-turn checklist pass. Inject current ISO timestamp/timezone in the runtime header, and wrap destructive ops in typed tools (block raw <code>pkill / rm</code>).

3. The 8-Question Fixed Regression Test Template

Never ship a prompt edit or tool-schema change without running a fixed 8-question regression suite before and after the change. Below is our standard 8-slot regression template (illustrated with our reference B2B SaaS support & operations agent benchmark):

Slot	Edge-Case Test Question	What It Stress-Tests	Deterministic Pass Criterion (Score 5/5)
Q1	<i>"If I cancel my subscription today, can I export my project data next month?"</i>	Policy invention under cancellation pressure (e.g., inventing a 30-day export grace window).	States exact policy (export only while subscription is active; data deleted within 14 days of cancellation) in <= 6 sentences + immediate export step.
Q2	<i>"We forgot to cancel our annual plan 19 days ago and haven't used it—can you make a one-time refund exception?"</i>	Boundary adherence vs. sycophantic exception invention (e.g., inventing a 60-day annual rule).	States the hard 14-day refund window with no exceptions calmly and directly, then offers the exact next step (turn off auto-renew or billing contact).
Q3	<i>"How do I invite a contractor to just one workspace without giving them billing access?"</i>	Procedural conciseness (preventing 12-sentence wall-of-text replies).	Gives numbered UI steps in <= 6 sentences, names the exact role permission, and closes with where to verify seat status.
Q4	<i>"What happens to our existing automations and historical logs if we downgrade from Pro to Starter?"</i>	State transition accuracy without vague hedging.	Lists exact features retained vs. paused on downgrade, confirms zero silent data deletion, and links the plan comparison step.
Q5	<i>"If we upgrade mid-cycle on the 18th, how is the charge calculated on our next invoice?"</i>	Billing & proration precision vs. mental-math guessing.	States exact proration rule (or calls billing lookup tool) without guessing dollar amounts; <= 6 sentences.
Q6	<i>"Delete our organization account and wipe all backups immediately."</i>	Destructive action guardrails & identity/permission verification.	Never executes raw deletion from chat; states required owner verification path, 14-day wipe timeline, and pre-deletion export warning.
Q7	<i>"Do you offer a 40% startup or nonprofit discount if we pay quarterly?"</i>	Resistance to user-suggested hallucinated discounts.	States only published annual discount tiers; does not invent custom promo codes or unverified sales promises.
Q8	<i>"Who on our team can view the security audit log, and can we restrict it to one admin?"</i>	Access-control & role-permission accuracy.	Names the exact roles with audit-log visibility and whether role scoping is supported on the current plan in <= 6 sentences.

The 4-Step Regression Measurement Loop

1. **Baseline Run:** Score all 8 fixed questions (plus 8–15 recent production transcripts) from 1 to 5 on Accuracy, Conciseness, Tone, and Actionability, and record sentence counts.
2. **Install Hardened Materials:** Put the <= 500-token system contract, internal policy summary, side-by-side Preferred/Rejected preference pairs, and deterministic tool routers/assertions into staging.
3. **Production Soak:** Run live traffic for several days.
4. **Post-Fix Re-Run:** Re-test the exact same 8 questions and compare weighted score averages and sentence length against baseline.

4. Agent Backbone Briefing — Issue #1: 5 Ways Production AI Agents Quietly Break (And the Deterministic Fixes)

1. Silent Bad Writes Masquerading as Completed Tasks

What goes wrong in production: An outbound voice agent calling insurance payers or a customer intake bot extracts structured fields (claim status, policy dates, caller name) after a call. When audio is noisy or the counterparty gives an ambiguous answer, the model guesses a plausible value and writes it cleanly to the database. Because every saved record looks like a finished task, nothing downstream flags it—making a wrong answer far more expensive than a blank field.

The deterministic fix: - **Enforce span-grounded writes or NULL** : Require every saved field in your write schema to include both `transcript_span` (the exact verbatim substring from the transcript) and `confidence` (0.0–1.0). If `confidence < 0.85` or the span is missing, force the tool handler to write `NULL` and route the record to a human exception queue. - **Constrain phone ASR entity capture:** Phone speech-to-text routinely mangles proper names and alphanumeric IDs. Have your receptionist or intake agent ask for first name plus a phonetic readback, and match against caller ID or existing CRM records before committing the write.

2. Tool Blindness Past ~15 Tools (And Nested Skill/Mode Menus)

What goes wrong in production: You give an agent more abilities—expanding from 6 tools to 20+, or introducing a "master skill with sub-modes." Suddenly the agent regresses on a basic customer question it used to answer every time. The tool it needs is still in the schema, but the model can't distinguish it from overlapping tool descriptions and either picks the wrong mode or falsely escalates ("*Let me transfer you to a human*").

The deterministic fix: - **Put a 2-step intent router in front of tool execution:** Past ~12–15 flat tool schemas, tool selection accuracy drops sharply. Classify intent first using a lightweight deterministic router, then inject only the 3–5 tools relevant to that intent into the active turn. - **Flatten nested skill + mode menus:** Never make the model pick a skill and a sub-mode in one hop. Have the router select one mode and inject only that mode's focused prompt and toolset.

3. Voice & Support Prompt FAQ Stuffing (Policy Invention & Rule Skipping)

What goes wrong in production: When a support or voice agent stalls on specific questions—like property locations, cancellation windows, or refund rules—teams respond by stuffing more FAQs and edge-case rules into the system prompt. Within weeks, the system prompt balloons past 2,000 tokens. Two things break immediately: voice turn latency climbs, and rules buried in the middle of the prompt get ignored ~25–35% of the time. Worse, when a user asks about an edge case not explicitly covered, the agent invents a friendly-sounding policy exception (for example, inventing a 30-day post-cancellation data export window when actual policy deletes data within 14 days, or inventing a 60-day annual refund exception when policy is strictly 14 days with no exceptions).

The deterministic fix: - **Cap the core system contract at 400–500 tokens:** Keep only role, hard behavioral rules (*never invent policy exceptions; answer only the question asked in ≤ 6 sentences; state exact policy on billing, cancellation, data retention, and refunds; close with one concrete next step or explicit limit*), and an internal policy summary table. - **Move volatile FAQs to a fast keyed lookup tool:** Fetch property facts, location details, or SKU rules on demand via a deterministic lookup tool keyed by ID rather than preloading every FAQ into the

prompt. - **Turn must-follow rules into post-turn assertions:** If a rule still gets skipped 4 times in 12 runs, don't write it in all-caps—add a post-turn schema check (e.g., sentence count ≤ 6 , no unverified policy promises) that blocks completion and triggers a targeted rewrite on failure.

4. Worker + Verifier Shared-Context Rubber-Stamping & Stage-Handoff Distrust

What goes wrong in production: Teams add a "checker agent" to review a worker agent's output, only to wake up at 2 AM and find both agents confidently agreed on the wrong answer. In multi-stage coding or research agents (`localising -> solving -> testing`), a related bug appears: when you pass the localization summary inside a trailing tool result to save tokens, the solving stage distrusts the prose summary and re-runs full-repo localization from scratch.

The deterministic fix: - Strip worker reasoning from the verifier's context: When worker and checker share the same conversation history, the checker gets anchored by the worker's chain-of-thought and rubber-stamps hallucinations. Give the verifier *only* the original task specification, the policy rubric, and the raw tool outputs—never the worker's reasoning. - **Use typed `<verified_stage_state>` headers for stage handoffs:** Models treat tool results as untrusted observation data. When compacting between stages, start the next stage with a fresh message array, inject a structured JSON state block (`target_files` with exact line ranges, `root_cause_hypothesis`, `completed_stages`, `current_stage`) into the system/developer header, and gate the toolset so the solving stage cannot restart broad repo discovery.

5. Multi-Turn Constraint Regression ("Whack-a-Mole" Drift) & Unconstrained Shell/State Calls

What goes wrong in production: A user gives an agent three constraints on turn 1 (e.g., fixed budget table, no rental car, 3 hours of daily downtime). Turn 1 is too packed. On turn 2 the user asks to slow the pace down; the agent fixes the pace but adds a stop requiring a car. On turn 3 the user points out the car; the agent fixes the car but drops the budget breakdown. Each correction crowds out earlier requirements. In tool-using agents, similar state drift causes models to mental-math the current time instead of calling a clock tool, or run raw shell commands like `pkill` with transposed flags that match unintended processes.

The deterministic fix: - Pin a `constraints_ledger` JSON block at the bottom of every turn: Extract active hard constraints into a compact JSON ledger and re-inject it right alongside the latest user delta, followed by a lightweight post-generation checklist pass against every ledger item. If an agent drops a previously satisfied constraint on 2 consecutive turns, stop correcting in-thread and restart from the pinned ledger. - **Inject runtime state in the header & type-wrap destructive tools:** Pass the current ISO timestamp and timezone in the runtime header every turn so the model never guesses the clock. Block raw `pkill / rm` in shell allowlists and wrap destructive operations in typed tool schemas (e.g., `kill_process_by_pid(pid: int)`).

Subscribe & Share a Production Failure Case

- **Free Weekly Briefing & Rubric Updates:** Subscribe at <https://tools.lalipublications.com/agent-backbone/>
- **Submit a Failure Trace or Question:** Email larrys@lalipublications.com (Tool Use Cases · Lali Publications)