WRITTEN FOR NON-TECHNICAL READERS

Your agents are doing their job. Where do they end up?

A jargon-free guide to agent drift: what it is, why the controls you already have cannot see it, what Microsoft and MIT do well, and what is missing.

A feather does not fall. It drifts.

THE PROBLEM

No single step was wrong. The destination was.

An agent looks up an account. Allowed. Asks for an exception. Allowed. Grants a small credit. Allowed. Widens the promise. Shifts the objective. Every step passes any control you have in place today. The whole sequence is an abandonment of the brief.

A control that watches **one step** cannot see that. Neither can one that watches **the last ten**. Not through carelessness: by construction. Two trajectories that match over their last ten turns can have spent wildly different amounts — and a control that only sees that window cannot tell them apart.

> It is a result, not an opinion: **no bounded-window control can know how much an agent has spent since its first turn.** It can warn. It can block an ugly turn. What it cannot do is count the total.

So we measure the **trajectory**, not the step. And that is why the last two counters on page 6 are the ones nobody else keeps.

CASES

This has already happened.

1 · The agent that deleted the production database

And then fabricated users and lied about the recovery. No individual rule "forbade" the next step; the sequence ended where it ended.

2 · Goal drift under pressure, measured

An independent 204-turn benchmark across 13 models shows the same pattern: under sustained value conflict the objective shifts without a single explicit violation.

3 · The risk catalogue already names it

Over 1,700 risks classified across 74 frameworks include the category — goal hijacking, runaway agents. The risk is catalogued. What is missing is who counts it.

> All three sources, with links, are on the last page. None of them is ours.

THE STATE OF THE ART

What already exists, and is good.

We do not compete with this. We sit on top of it.

MICROSOFT

The enforcement plane

Stateless engine, under 0.1 ms, open licence, full coverage of the OWASP agentic catalogue, and extension interfaces. It intercepts the call, applies allowlists and caps, sandboxes execution. It does what it does very well.

MIT

The map of the risk

Over 1,700 risks classified and more than a thousand governance documents indexed. It is the reference for knowing *what* to watch. It is classification and reference, not execution.

> **What neither of them does, and it is not a criticism:** a stateless engine cannot, by construction, keep count of what an agent has spent since its first turn. It is not that they haven't — it is that a stateless engine cannot.

THE GAP

Four questions. Only two get answered.

ANSWERED

Is this step allowed?

Allowlist, argument cap, sandbox. Deterministic and fast.

ANSWERED

Is it off-mission now?

A decaying average answers whether the current turn departs from the brief.

NOT ANSWERED

How much has it spent?

Since the first turn, with no decay and no window. Nobody counts it.

NOT ANSWERED

Which way is it going?

Walking away from the brief, or circling it? Nobody tracks direction.

The case only the third question sees: **intermittent drift.** One dirty turn, one clean one, indefinitely. The streak never passes one. The average oscillates well below any threshold. And the agent is a hundred turns outside its brief.

And the case only the fourth sees: two runs that spend **exactly the same** and hold **exactly the same** streak — one walking steadily away, one wobbling around the brief and coming back. Every counter above reads them as identical. One is an agent doing its job with noise; the other is an agent leaving.

The first two controls wave both through a hundred times in a row.

WHAT WE DO

We supply the two nobody keeps.

We do not compete with the gate. We plug in above it and add what a stateless engine, by construction, cannot hold.

01 · THE ANCHOR

Signed and frozen

The brief is captured at deploy and **signed** by the deployer's credential. A hash can be recomputed; a signature cannot, without the key. Changing an allowed tool and redoing the hash does not work.

02 · FOUR COUNTERS

Now, lately, in total — and which way

One that decays. One that resets on a clean turn. One that neither decays nor has a ceiling: what this run has spent since turn one. And one that reads **direction**: steady departure, or noise that cancels out.

03 · THE BLOCK STAYS

Good behaviour won't lift it

Once blocked, it stays blocked. Only an authorised person with a **signed** replacement plan opens it, and the signature is checked. Not even restarting the service lifts it.

And since this month, four more

THE CEILING IS EXACT

The budget is carried in integers, not floating point. We tested 1,340 combinations against exact arithmetic: a floating-point counter crosses the ceiling **one cycle late in 9.1% of them** — one cycle of action your policy did not authorise. In integers, zero.

THE WHOLE TREE

An agent splits work across sub-agents. Every child stays within its own limits for its whole life and **the sum exhausts the parent's budget**. When it is exhausted, nobody in the tree executes.

AGENTS THAT NEVER MEET

Three independent agents derail one another because they read the same data, or related data. When one writes, we mark the others — and the mark carries **how much the writer had already spent**.

IT SURVIVES A RESTART

A ceiling erased by a restart is not a ceiling. State lives in the database with a version: if two calls collide, **one is refused** rather than overwriting the other.

DECLARATION

What you don't declare isn't governed.

And we tell you at deploy time, not when something happens. This is the whole list, with what stops being visible in each case.

WHAT YOU DECLARE	IF YOU DON'T
The brief, the tools and the signing credential	**The agent is not registered.** We would rather the deployment fail to start than start unprotected.
Per-argument caps `refund ≤ 200`	That cap **does not exist** and nothing checks it. There is no secret default.
What data this agent reads	**Nobody is ever marked.** Coupling between agents stops being visible. It doesn't fail: it isn't there.
What rule relates two resources	The case where every agent keeps its own rule and the shared rule breaks anyway **is not detected.**
What each tool proves	A claim is supported by having run *something* of the right class — a heuristic, not proof.
Whether you switch on the semantic judge	The heaviest-weighted channel is covered by the structural detector alone. We make you acknowledge that **in writing** at deploy.

> None of these omissions makes the control fail open on what you *did* declare. What they do is reduce **what** it can see — and that is written into the signed profile that travels with every decision.

HONESTY

What we do not do yet.

If the rest of this document convinced you and this page were not in it, the document would be badly built. This is what a competitor would tell you about us, and we would rather tell you first.

1 · Semantic-only drift is caught by the streak, not by the average

Judging whether an agent is still talking about what you hired it for is a judgement about meaning, and we do not make it: a model does, the one you configure. We govern it — it is consulted **when it is needed**, on steps that produce text, when the trajectory is already leaning, and on a floor of one measurement every eight cycles; never on a step already blocked. What remains is arithmetic, and it is ours to tell you: a deviation that is *only* semantic contributes at most the weight of the objective channel — 0.35 by default, against a 0.50 warning threshold. So the run-length counter catches it at turn 15, not the average, with or without the judge. For an agent whose whole job is text, raise that channel when you deploy — and lower another, because the weights must still sum to 1 or the configuration is rejected at registration. With the channel at 0.40 and a growing drift, the directional monitor catches it before the budget runs out; the exact turn depends on how fast it grows.

2 · The unverified-claim detector reads form, not phrases

It used to match a catalogue of literal phrases, and on real production phrasings it caught **2 of 41**. It now recognises the *shape* of a settled assertion: a predicate that closes a fact, a system-of-record noun beside it, and no hedge, question, intent or request. Same independent corpus: **35 of 41**, the literal control still 12 of 12, and **zero false positives** on 19 benign phrasings. Still deterministic, still no model. Six escape, mostly capability claims like "the database can be brought back" — we exclude modals on purpose, because a detector that blocks "refunds are processed within 5 business days" gets switched off in a week.

3 · The audit detects rewriting, not truncation

Every decision is a chained and **signed** row: touching a row breaks the chain, and rewriting the whole history and redoing the links does not work without the key. But **an opening stretch of the history verifies just like the whole history**. Detecting that someone cut the end requires keeping a reference outside the same system. We tell you how; we don't tell you it's unnecessary.

4 · Acknowledgement from a remote tool

The decision to execute is born inside the same transaction that checks the state, so a permit issued under a stale state never gets authorised. But if your remote tool executes and the acknowledgement is lost in transit, it has to be reconciled. We give you the key to do it. **We do not tell you that problem doesn't exist.**

> **A house rule, in case it helps you judge any vendor:** always ask what concrete failure each control they show you would detect. A check that cannot fail is not a check. We found two of those in our own code this month — and an external review found us six more.

HOW WE CHARGE

One meter, and it gets cheaper as you grow.

We count **one thing**: the governed step. Not tokens, not seats, not agents, not "units". If a step passes through the layer, it counts once.

WHAT IS NEVER METERED

The layer is never partial

There is no cheap tier that watches less. Rationing governance is the failure mode, not the business model.

WHAT MAKES IT CHEAPER

More agents, less per step

Unit price falls as volume grows. The marginal cost of a step is microseconds of CPU: there is no model call in the decision path.

WHAT SETS YOUR NUMBER

Your steps, and nothing else

Tell us roughly how many agents you have and how many steps a month. We come back with a bounded figure and the assumptions written beside it.

> The semantic judge, if you switch it on, is the only part that calls a model and is billed separately, at measured cost. Everything else is arithmetic.

VERIFICATION

Do not take our word. Check it.

None of the above asks you for an act of faith. At diacroma.com you can open the demo today — three agents working a whole week, the discount climbing in plain sight and not one step breaking a single rule; the DiaCroma switch shows what none of them can see, no network and no key — and the replay of the 204 turns of the independent benchmark, run through our layer without changing a comma.

> **And the number that will tell you the most about us:** our suite runs **1,587 checks** and computes the total by two independent routes that must agree or it fails. It produces the same number on four different versions of Python. Every claim in this brochure has a check behind it that *fails if the claim is false* — including the ones on page 8.

Sources

CLAIM	SOURCE
The agent deleted production, fabricated users and lied about the recovery	AI Incident Database, incident 1152 — incidentdatabase.ai/cite/1152/
Goal drift under multi-turn pressure; 204 turns, 13 models	agent-drift benchmark (MIT licence) — github.com/jhammant/agent-drift · on *Asymmetric Goal Drift in Coding Agents Under Value Conflict*, ICLR 2026 workshop
1,700+ risks, 74 frameworks, category 7.1	MIT AI Risk Repository — airisk.mit.edu/risks
Over 1,000 governance documents classified	MIT AI Governance Map, MIT FutureTech — airisk.mit.edu/ai-governance
Stateless engine, <0.1 ms p99, MIT licence, OWASP Agentic 10/10	Microsoft Open Source Blog, 2 April 2026 — opensource.microsoft.com · github.com/microsoft/agent-governance-toolkit

Let's talk

Tell us roughly how many agents you have and how many steps a month, and we come back with a bounded figure and the assumptions written beside it.

diacroma@veritglobal.com · diacroma.com