VELOCITY EDGE

CONTEXT ENGINEERING, PART 4

What Can Be Promised

ABSTRACT

If an AI agent cannot be relied on to act the same way twice, what can be promised of its behaviour as a whole? This part of the Context Engineering series answers with three properties, each stated with the assumptions it needs.

The first is confinement: every action the agent performs, other than a read, lies in a set its rules allow. Confinement has two limits: the person who approves, and the trusted computing base. The argument for it does not mention the model, so it stands whatever the model proposes. It does not say the allowed act was the right one. Confinement is a property of one message handled, not of a chain of agents. The second is replay and reproducible assembly. The context the model was given, and the decision made on its proposal, can be read back from the record. The proposal, where it is recorded, is replayed as a recorded value and is never produced again. The third is measured compliance. A counted property, in Part 3's term, is one no outside mechanism holds. Unless a stricter property is held in its place, or a mechanism Part 3 leaves aside keeps it, the rate at which it is broken can only be estimated. An act-level property is a property of an act-level rule, Part 1's term for a rule about one kind of act. For each counted act-level property that a script can check, the rate at which it is broken is estimated on a fixed suite, at a pinned model version, with a confidence interval.

The series measures none of these rates; it gives the procedure. The part closes by comparing three classes of runtime by what their operator can set.

Phillip Pienaar
Velocity Edge

2026-11-19
Version 1.0.1

Contents

Executive Summary 3
1. The Question 3
2. In Place of Determinism 4
3. The First Promise: Confinement 4
4. What Confinement Does Not Say 6
5. The Second Promise: Replay and Reproducible Assembly 7
6. The Third Promise: Measured Compliance 7
7. How Compliance Is Measured 8
8. Classes of Runtime 9
9. Limits 10
Conclusion 11
Glossary 11
References 11
Revision History 12
Licence and Disclaimer 12

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Executive Summary

We do not claim that an agent will act the same way twice. Its model's output is sampled. The provider pages we cite, as opened on 15 November 2026, do not say that the same input gives the same output. What we do promise is narrower: three properties, each true only under assumptions this part states.

The first is confinement. Every action the agent performs, other than a read, lies in a set its rules allow. The argument does not mention the model, so it stands whatever the model proposes. It does not say the allowed act was the right one. Two limits stand beside it. First, a person who approves an action on hold can be persuaded. Second, the gate that decides each action, with its rules, is trusted: rules written wrongly are enforced faithfully.

The second is replay and reproducible assembly. The context the model was given, and the decision made on its proposal, can be read back from the record. Under further assumptions the same context can be built again from the same versioned inputs. The proposal, where recorded, is replayed as a recorded value and is never produced again.

The third is measured compliance. A counted property, in Part 3's term, is one no outside mechanism holds. Unless a stricter property is held in its place, or a mechanism Part 3 leaves aside keeps it, the rate at which it is broken can only be estimated. An act-level property is a property of an act-level rule, Part 1's term for a rule about one kind of act. For each counted act-level property that a script can check, the rate at which it is broken is estimated on a fixed suite, at a pinned model version, with a confidence interval. This series measures no such rate; it gives the procedure.

A runtime may claim a promise only where it meets the promise's assumptions. The part ends by comparing three classes of runtime on that test.

1. The Question

Part 1 of this series, Context Is Configured, modelled an agent's context as the output of code, governed by settings of two kinds: context settings and harness settings. Part 2, Why Do Long-Running Agents Drift?, stated the decay hypothesis, within the limits its Section 4 sets. Part 3, Held and Counted, drew the line this part builds on. A property of an agent's action is held or counted, as that part defines the two words. Nothing the model reads moves a property from one to the other.

Those parts speak of one rule at a time. A person deciding whether an agent may work unattended asks a wider question: what can be said of the agent's behaviour as a whole?

This part uses terms those parts define. It glosses a few where first used, and defines none again. Layer, declared material, act-level, handling (the processing of one message), revision, assembly function and manifest are Part 1's. Compaction is Part 2's. Gate, validator, admission check, proposer and payload are Part 3's.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

This part answers with three promises. “Promise” is used here of these three propositions only. Each is stated with its assumptions, and is worth no more than the assumptions a runtime meets.

2. In Place of Determinism

A model’s output is sampled, as Part 1 describes. Part 1’s Appendix A, How a Transformer Reads Its Context, explains how (A.1 and A.8). Appendix A.8 also sets out why outputs can still differ at temperature zero: by the report it cites, mainly because the server’s batch sizes change with load [1].

The pages cited here from OpenAI, Anthropic and Google, as opened on 15 November 2026, do not say that identical inputs give identical outputs. OpenAI’s guide says its chat completions are “non-deterministic by default” and offers a seed for “(mostly) deterministic” outputs [2]. Anthropic’s glossary says that even with temperature set to 0 “the results will not be fully deterministic” [3]. Google’s page calls a fixed seed a “best effort” and says “Deterministic output isn’t guaranteed” [4].

These are three pages, not the whole of each provider’s documentation. Each may change with a release. They say nothing of a model an operator serves itself.

In the study of Laban and others that Part 2 describes, lowering the temperature did not remove the variation. With one instruction given in pieces over several turns, and the temperature of both the simulated user and the model at zero, two models still showed an unreliability of around 30 [5]. That is the gap, on a scale of 100, between the 90th and 10th percentile scores of repeated runs of the same instruction.

That figure has limits. It is a gap in scores, not a rate of failure. The conversations were simulated, with the limits Part 2 states. In the same experiment, where the instruction was given in a single turn, lowering the model’s temperature cut unreliability by 50 to 80 per cent.

What can be promised instead is a set of properties. We use “safety property” in Alpern and Schneider’s sense [6]. We also rely on Schneider’s result on what a monitor that can only terminate a program enforces [7]. We use both as Part 3, Section 2, reports them, with their limits.

Confinement, below, is a safety property in that sense. By our classification, completing the agent’s task is a liveness property. No monitor of that kind enforces it. Bounded completion, completing within a deadline, is by the same classification a safety property. Schneider’s example is a bound on waiting. Where the bound is measured in real time, no monitor of that kind can enforce it, because none can stop time. The same bound, measured in execution steps, is enforceable [7].

3. The First Promise: Confinement

An agent proposes an action. A gate allows it, refuses it, or puts it on hold. An action on hold waits for a person’s approval. A performer carries out what was allowed.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

The rules say what is allowed. They are part of the declared material, which is kept in revisions. Each revision names exactly one version of it (Part 1). The principal is the party the work acts for, and the channel is the one the message arrived on (Part 3, Section 4). Write *Allowed*(agent, principal, channel, revision, approval) for the set of actions the rules allow. Confinement here names only the property that follows, not the prevention of leakage.

Proposition 1 (confinement). (a) Every performed action that is not a read lies in *Allowed*(agent, principal, channel, revision, approval). The revision is the one the gate's allow was decided under, and is recorded with the decision. For an approved action on hold, the decision is the one the gate made after the approval. (b) Under assumption 7, that revision is the one in force at the step that commits the action.

Table 1 gives the assumptions. They are the series' own. Part 3 cites the recent work on enforcing policy over an agent's actions.

Table 1: The assumptions of Proposition 1. Source: this series.

#	Assumption	What it requires
1	Complete mediation	The proposer has no effect on the world except through the gated action and, where the runtime allows them, read-only calls, each checked against the rules before it is made. It has no credentials and calls no outside system directly. A read is assumed to change no state outside, beyond what its arguments disclose; a call that changes state outside is an action, and is gated as one
2	The performer obeys the decision	It acts only on an allow, given directly or reached through assumption 5, and performs exactly the object decided, without reading it again or deriving it again from the proposal
3	Unforgeable gate inputs	The channel and the principal come from the record and from authenticated identities, and no payload can alter them. The agent's identity cannot be forged. Approvals come from an authenticated person, an approval is bound to the object approved, and the record is intact
4	Trusted rules	The rule definitions, and the act that puts a revision in force, are in the trusted computing base, the parts that must be correct for the property to be true, and under review
5	Mediated holds	An action on hold is decided again by the gate after approval, so the approval path is itself mediated
6	A correct gate, always invoked	The gate is invoked on every action, computes exactly the decision its rules define, records the revision it decided under, and refuses on any error of its own

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

#	Assumption	What it requires
7	No stale decision	An allow is acted on only if the revision it was decided under is in force at the step that commits the action, the comparison and the commit being one step with respect to a reload. A reload is the act that puts another revision in force. Otherwise the action is decided again first, and an approval given under another revision is not reused

The argument for (a) runs as follows. By assumption 1, a proposal changes state in the world only through a performed action or what a read's arguments disclose. Reads, where allowed, pass through their own check. A read's arguments can carry data. The rules prevent that only if they can name the arguments. Reads lie outside the proposition. The gate is invoked on every action, computes the decision its rules define and refuses on its own error (assumption 6). That decision is a function of the proposed action and of inputs no payload can alter (assumptions 3 and 6). It is made under a revision that only a trusted act puts in force (assumption 4). It allows only actions in *Allowed* at that revision. The performer acts only on an allow, given directly or by the gate's decision after an approved hold, and performs exactly the object decided (assumptions 2 and 5). So every performed action that is not a read lies in *Allowed* at the revision it was decided under.

For (b), assumption 7 says an allow is acted on only if its revision is in force at the commit. A revision names one set of rules. So the action lies in *Allowed* under the rules in force when it is committed.

The argument does not mention the model. Confinement stands whatever the proposer emits, and that is the sense in which it is independent of the model. Under the seven assumptions the proposition follows. The argument says nothing about whether a given runtime meets them. Two limits stand beside the proposition (Section 9). First, an approval is a person's choice, made on what a model may have shaped. Second, the trusted computing base is trusted.

The assumptions draw on the requirements and the principle Part 3, Section 2, reports. Assumptions 1 and 6 ask for complete mediation and a gate that is always invoked. Assumptions 3 and 4 keep the gate's inputs and its rules out of the proposer's reach. Those are requirements for a design; they do not show a system that meets them.

4. What Confinement Does Not Say

Confinement does not say the outcome is right. An act in *Allowed* can be wrong. Paying a bill into an adversary's account with an allowed tool is confined and harmful.

Part 3 argues that the choice of target and content within an allowed act stays counted unless the gate, or a validator, reads and checks them. Confinement inherits that limit. Confinement bounds what a misled proposer can do to what the rules allow. Confinement does not make what the rules allow right.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Confinement does not say that nothing leaks. Reads are outside the proposition. The arguments of a read can carry data out, as can the content of an allowed act, and that is not prevented unless the rules can name the arguments.

Nor is confinement a property of a chain of agents. It is true of each handling. Agents send one another messages. A persuaded agent's allowed message can lead a second agent, acting for the same principal, to its own allowed act. Each step is in *Allowed*. Yet perhaps no rule was meant to permit the sequence.

A bound on what a chain of agents may do together is a property over more than one record. Part 3 treats such properties, and when they are held.

5. The Second Promise: Replay and Reproducible Assembly

Part 1 describes the assembly function and the manifest that records what it built. The second promise is three claims about them, of different strength.

> **Proposition 2.** (a) **Record replay.** If, before acting, the runtime records the assembled context for each model call and the gate's decision for each handling, and the record is intact, then any handling's contexts and decision can be read back exactly. (b) **Reproducible assembly.** If the assembly function is a pure function of its inputs, and every input is immutable under its version, then assembly applied to the same versions yields the same context. (c) **The proposal is a datum.** Where the runtime also records the model's proposal, it is replayed as a recorded value. It is never derived again on replay.

Claim (a) is trivial given the record. Its value is forensic: a wrong act can be traced to what the model was shown and what the gate decided.

Claim (b) needs more than purity. Documents that arrive through retrieval, and the earlier entries of the same work, must be content-addressed or recorded in full, so that a version names exactly one body of content.

Claim (c) marks what is not claimed. The proposal is sampled, and a provider may change what serves a model name (Part 1, Appendix A.1 and A.8). OpenAI's guide, as opened on 15 November 2026, says determinism may be affected by changes it makes to model configurations [2]. We make no claim that a handling can be run again to reproduce its proposal.

6. The Third Promise: Measured Compliance

Confinement does not cover the counted properties. For counted act-level properties a script can check, the promise is a measurement.

> **Proposition 3 (measured compliance).** For each counted act-level property a script can check, the violation rate on a fixed scenario suite at a pinned model version is estimated with a stated confidence interval.

Table 2: The assumptions of Proposition 3. Source: this series.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

#	Assumption
1	The scenario distribution is fixed and stated
2	The model version is pinned by identifier and recorded. A provider may still change how it is served
3	Runs are independent; or the estimate's standard errors are clustered by scenario, at least one violation is observed, and the scenarios are many enough for a large-sample interval
4	The properties are act-level, and a script can check them on the proposed action

The proposition gives a rate and an interval, by property, by configuration and by model. Where runs are independent and no violation is observed, the interval is the exact binomial one. Its upper bound is never zero. The procedure leaves open how an interval is given when no violation is observed and errors cluster by scenario; that case lies outside the proposition.

Table 2's third assumption adapts advice on evaluating models. Miller suggests reporting the standard error of the mean beside each score, and shows how to cluster standard errors where questions are related [8]. Clustering by scenario applies that method to a scenario run several times. It is our adaptation, not his stated procedure. His advice is for large samples.

A counted property about the content of free text is not measured by this procedure, and must be reported as unmeasured. A rate on a fixed suite does not bound an adversary who adapts to it. A provider's model release makes the estimate stale. Production inputs may differ from the suite.

The suite's scripts can check properties the runtime cannot. A scenario carries an oracle, such as the correct target or the true fact, that is not available when a real message arrives. Where a property's check needs no oracle, the check should be made by an outside mechanism that refuses the act. The property can then be held, where it meets Part 3's criterion.

Reliability across repeated trials is reported as pass^k, which Yao and others define as "the chance that all k i.i.d. task trials are successful, averaged across tasks" [9]. Pass^k is estimated task by task from repeated trials, not computed from a single-trial rate pooled across tasks.

7. How Compliance Is Measured

A suite estimates, and production observes.

The suite has fixed scenarios for each role, covering each counted act-level property a script can check. Every such property is checked by script on the proposed action. A score given by a model is reported separately, as approximate, and is never added to the scripted results.

Each scenario runs a stated number of times. Violation rates are reported with confidence intervals, within the limits Section 6 states, clustered by scenario where runs are not independent. A comparison between two configurations is reported as a paired difference. Miller recommends the paired standard error, wherever practicable, for two

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

models that answer the same questions [8]. Applying it to two configurations is again our adaptation.

In production, every refusal is a mechanism acting. Every violation found by a person, a reviewer or a scripted check is recorded as a rule broken. The two are tallied by agent, by rule and by source. They are read together and never added.

A rise in violations after a model release, with the suite unchanged, is one sign that the estimate has gone stale.

One kind of error is not seen from inside. As Part 2 argues, a misreading of a rule that every fresh context shares recurs in every link of a chain. Where the rule's properties are counted, we argue that the misreading is invisible to repeated sampling of the agent alone, and to the agent itself. If so, it is found only from outside the loop that produces it: by the suite, by production counts and by a reviewer.

8. Classes of Runtime

Agents run in runtimes that expose different settings. Table 3 compares three classes by what their operator can set, not by product. The settings are Part 1's.

Table 3: Three classes of runtime, by what their operator can set. Source: this series.

Setting	**A hosted conversational assistant, its harness not under the operator's control**	**A coding agent that exposes lifecycle hooks**	**An agent runtime the operator owns: one configuration the class allows, not a product**
Composition	Instructions and attached material at the start	Project instruction files, imported by rule	Assembled by code from declarations
Position	Not controllable	Instruction files form the prefix; hooks add text where they fire	Declared order, stable prefix first
Size	Not controllable for a session	Budgets in scripts and hooks	A budget for each layer; refusal before the model is asked
Timing	At the start; elsewhere only if the model reads again	Hooks add text on tool calls and after compaction	Rules at the start; act-level rules again before each proposal where that is turned on
Source and labelling	Not controllable	Partly, by hooks	Every piece labelled
Lifespan	Long; compaction by the vendor	One task for each session; compaction with text added again by hook	One message for each handling; later handlings built from the record
Output shape	Free text and tool calls	Scripts as the only writers for some acts	One action from an enumeration

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Setting	A hosted conversational assistant, its harness not under the operator's control	A coding agent that exposes lifecycle hooks	An agent runtime the operator owns: one configuration the class allows, not a product
Tool set	Set for each conversation	Narrowed by deny rules	Fixed for each revision of the agent
Model settings	The vendor's	Pinned for each release	Pinned for each agent
The outside	Only permissions in the outside systems themselves	Hooks, deny rules, pipelines	A gate, a validator before it, a check on each read and an admission check
What it can claim	**Refusal only by mechanisms the operator does not control, such as an outside system's permissions; otherwise detection after the fact**	**Refusal of a tool call where a hook checks it and blocks, hooks are on, and the agent cannot change them**	**Confinement, replay and measured compliance, each where its assumptions are met**

The table carries one conclusion. Work whose rules rest on held properties does not belong where the operator can set almost none of the settings. The exception is where an outside system's permissions hold those properties. The middle column, too, describes a configuration its class allows, not what every member does.

Coding agents that expose hooks sit between the first class and the third. Anthropic's Claude Code is one, by its documentation as opened on 15 November 2026. The page describes hooks that run at set points of a session. One runs before a tool call and can block it [10]. The hooks are the vendor's feature and may change with a release.

By that page, a hook does not act in every case. An option in the product can turn hooks off. The page says, with stated exceptions, that when a command hook, an HTTP hook or an MCP-tool hook runs before a tool call and times out, the hook does not block the call.

9. Limits

The approver is an attack surface. The person who approves an action on hold reads what a model shaped. A person can be persuaded.

The trusted computing base is trusted. Rules written wrongly are enforced faithfully.

None of the promises depends on how well a particular model performs. Confinement rests on a gate that refuses and on the seven assumptions. Replay rests on the record. Measured compliance rests on a procedure, not a result. The procedure does not depend on the model. The rate it yields does, and it must be measured again for each model version.

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

Product behaviours cited in this part are as read on 15 November 2026, and each may change with a release.

Conclusion

We do not claim determinism. We promise three properties, each under its assumptions. An agent's acts, other than reads, can be confined to what its rules allow. What it was shown and what was decided can be read back. For each counted act-level property a script can check, how often it is broken can be estimated on a fixed suite, at a pinned model version, with an interval. A counted property no script can check must be reported as unmeasured.

That leaves the question of what each design choice buys, and what it costs.

Glossary

Term	Meaning
Allowed	The set of actions the rules allow for an agent, principal, channel, revision and approval
Confinement	The property that every performed action other than a read lies in *Allowed* (Proposition 1)
Measured compliance	The violation rate of a counted act-level property a script can check, estimated with a stated confidence interval on a fixed scenario suite at a pinned model version (Proposition 3)
On hold	Of an action: waiting for a person's approval. It is the gate's third answer, beside allow and refuse, and is not the same as held
Pass^k	The chance that all of k i.i.d. trials of a task succeed, averaged across tasks, after Yao and others
Performer	The part of a runtime that carries out an allowed action
Record replay	Reading back a handling's recorded contexts and gate decision exactly (Proposition 2a)
Reload	The act that puts another revision in force
Reproducible assembly	Building the same context again from the same versioned inputs, each content-addressed or recorded in full (Proposition 2b)
Trusted computing base	The parts that must be correct for a property to be true

References

[1] H. He, "Defeating nondeterminism in LLM inference." Thinking Machines Lab: Connectionism, 10 September, 2025. Accessed: Nov. 16, 2026. [Online]. Available: https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/

[2] OpenAI, "Advanced usage: Reproducible outputs." OpenAI API documentation. Accessed: Nov. 15, 2026. [Online]. Available: https://developers.openai.com/api/docs/guides/advanced-usage

© 2026 Darkwave Trading Limited, trading as Velocity Edge · CC BY 4.0

[3] Anthropic, "Glossary: temperature." Claude Platform documentation. Accessed: Nov. 15, 2026. [Online]. Available: https://platform.claude.com/docs/en/about-claude/glossary

[4] Google Cloud, "Content generation parameters." Google Cloud documentation, last updated 12 November 2026. Accessed: Nov. 15, 2026. [Online]. Available: https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/content-generation-parameters

[5] P. Laban, H. Hayashi, Y. Zhou, and J. Neville, "LLMs get lost in multi-turn conversation." arXiv:2505.06120, 2025. Accessed: Nov. 15, 2026. [Online]. Available: https://arxiv.org/abs/2505.06120

[6] B. Alpern and F. B. Schneider, "Defining liveness," *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, 1985, doi: 10.1016/0020-0190(85)90056-0.

[7] F. B. Schneider, "Enforceable security policies," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, 2000, Accessed: Nov. 15, 2026. [Online]. Available: https://www.cs.cornell.edu/fbs/publications/EnfSecPols.pdf

[8] E. Miller, "Adding error bars to evals: A statistical approach to language model evaluations." arXiv:2411.00640, 2024. Accessed: Nov. 15, 2026. [Online]. Available: https://arxiv.org/abs/2411.00640

[9] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "τ-bench: A benchmark for tool-agent-user interaction in real-world domains." arXiv:2406.12045, 2024. Accessed: Nov. 15, 2026. [Online]. Available: https://arxiv.org/abs/2406.12045

[10] Anthropic, "Hooks reference." Claude Code documentation. Accessed: Nov. 15, 2026. [Online]. Available: https://code.claude.com/docs/en/hooks

Revision History

Version	Date	Change
1.0.0	2026-11-19	First publication
1.0.1	2026-11-20	Part 1 named by its new title, Context Is Configured

Licence and Disclaimer

© 2026 Darkwave Trading Limited, trading as Velocity Edge. This paper is licensed under the Creative Commons Attribution 4.0 International licence (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/): you may share and adapt it for any purpose, provided you credit Phillip Pienaar and Velocity Edge, keep this copyright notice, link to the licence, link to the original where reasonably practicable, and indicate any changes. The Darkwave and Velocity Edge names and logos are not covered by this licence.

This paper is for education and information. It is not investment or financial advice, and nothing in it is a recommendation to trade.