Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

Context Engineering for AI Agents: A Practical Guide

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-10-29 · context engineering · context engineering for ai agents · context engineering vs prompt engineering · context window management agents · what is context engineering

Original article: https://reliablecontext.com/articles/context-engineering-for-ai-agents-practical-guide



In this article

1. Context engineering, stated plainly
2. Where the term comes from, and who defines it how
3. Prompt engineering versus context engineering
4. The components of an agent's context
5. Why curation is necessary: finite attention, position and context rot
6. Strategies: select, compress, isolate, persist
7. Facts need provenance before they enter context
8. Worked example: Halden Therapeutics AG
9. Failure modes and how to detect them
10. Checklist for a defensible context

Context engineering, stated plainly

Context engineering is the practice of deciding which tokens a language model sees at each step of a task, where those tokens come from, and in what form. For an agent that calls tools, reads documents and keeps notes across many turns, the context is assembled anew at every step from instructions, tool definitions, retrieved facts, memory and history. The engineering problem is to fit the most useful evidence into a finite and unevenly used window.

This guide treats context as a supply chain. Every token has an origin, and for work that must be defended later the question is not only "is this relevant?" but "can this be traced to a source, and should it be here at all?"

Where the term comes from, and who defines it how

Several definitions circulate; they overlap but are not identical.

- **Anthropic (29 September 2025).** "Context refers to the set of tokens included when sampling from a large-language model (LLM)." Context engineering "refers to the set of strategies for curating and maintaining the optimal set of tokens (information) during LLM inference, including all the other information that may land there outside of the prompts." It calls context engineering "the natural progression of prompt engineering" (Anthropic, Effective context engineering for AI agents).
- **Andrej Karpathy, as quoted by LangChain (2 July 2025).** Context engineering is the "delicate art and science of filling the context window with just the right information for the next step" (LangChain, Context Engineering).
- **Mei et al. (July 2025).** The survey describes context engineering as "a formal discipline that transcends simple prompt design to encompass the systematic optimization of information payloads for LLMs". It formalizes the context as an assembly C = A(c1, ..., cn) of components and frames the discipline as an optimization of the context-generating functions under the hard constraint |C| ≤ Lmax (Mei et al., §3.1).

None of these sources claims to have coined the term. The site's glossary entry for context engineering gives a shorter, vendor-neutral definition.

Prompt engineering versus context engineering

Anthropic contrasts the "discrete task of writing a prompt" with context engineering, which "is iterative and the curation phase happens each time we decide what to pass to the model" (Anthropic). Mei et al. tabulate the same contrast in formal terms (Mei et al., Table 1).

Dimension	Prompt engineering	Context engineering
Object	A prompt string, often the system prompt	Every token in the window at each step
When the work happens	Before deployment, then revised	At every inference step, by code and by the model
State	Primarily stateless (Mei et al.)	Stateful: memory and task state are explicit components
Typical failure	Ambiguous or brittle instructions	Missing, stale, buried, conflicting or unsourced material
Evidence question	Is the instruction clear?	Where did each token come from, and is it still true?

The last row is this article's addition. The system prompt remains one component among several.

The components of an agent's context

Anthropic's earlier post on agents distinguishes **workflows**, "systems where LLMs and tools are orchestrated through predefined code paths", from **agents**, "systems where LLMs dynamically direct their own processes and tool usage". Its basic building block is "an LLM enhanced with augmentations such as retrieval, tools, and memory" (Anthropic, Building effective agents). By September 2025 Anthropic had settled on a shorter definition of agents: "LLMs autonomously using tools in a loop" (Anthropic). The loop is what makes context a supply-chain problem: each tool call adds material the next step may not need.

ReAct, which interleaves reasoning traces with actions and feeds each observation back into the prompt, is an early research formulation of such a loop (Yao et al., ReAct). Mei et al. name six context components (§3.1). The table maps them to their usual origin and the provenance question each raises.

Component (Mei et al.)	Typical origin	Provenance question
c_instr: system instructions and rules	Written by the developer	Which version was in force for this run?
c_tools: tool definitions and signatures	Written by the developer or a tool server	Which tools were exposed, and were they all needed?
c_know: external knowledge (retrieval, knowledge graphs)	Search indexes, databases, documents, the web	Which source, which version, observed when, quoted or paraphrased?
c_mem: persistent information from prior interactions	Written earlier, often by the model itself	Was the memory derived from a source, or from a previous answer?

Component (Mei et al.)	Typical origin	Provenance question
c_state: state of the user, world or multi-agent system	Orchestrator, sub-agents, environment	Who produced this state, and is it current?
c_query: the immediate request	The user	When was it asked, and by whom?

Instructions and tools are authored artefacts, versionable like code. Retrieved knowledge and memory carry most claims about the world and are least likely to arrive with their origin attached.

Why curation is necessary: finite attention, position and context rot

Three lines of evidence explain why a larger window does not remove the need to choose.

Attention is a budget. Anthropic argues that context "must be treated as a finite resource with diminishing marginal returns", that LLMs have an "attention budget", and that transformer attention creates "n² pairwise relationships for n tokens" which get "stretched thin" as length grows (Anthropic).

Position matters. Liu et al. varied where the answer-bearing document sat among retrieved documents and found that performance "is often highest when relevant information occurs at the beginning or end of the input context, and significantly degrades when models must access relevant information in the middle of long contexts, even for explicitly long-context models." In one multi-document setting, GPT-3.5-Turbo's accuracy with the relevant document in the middle fell below its closed-book accuracy of 56.1%, that is, below answering with no documents at all. In an open-domain case study, using 50 retrieved documents instead of 20 improved performance by only about 1.5% for GPT-3.5-Turbo and about 1% for Claude-1.3 (Liu et al., §1 and §5). These were 2023 models; the site's evidence position robustness test describes how to measure the effect on the model you actually deploy.

Length alone degrades performance. Chroma's technical report evaluated 18 models while holding task difficulty constant and varying only input length, and found that "model performance consistently degrades with increasing input length". On a LongMemEval subset, models did markedly better on focused prompts of about 300 tokens than on the full prompts of about 113,000 tokens that contained the same answer plus irrelevant material. Topically related distractors had non-uniform effects that grew with length (Chroma, Context Rot). Anthropic links the term "context rot" to that report.

Strategies: select, compress, isolate, persist

LangChain groups agent context strategies into four buckets: **write** ("saving it outside the context window"), **select** ("pulling it into the context window"), **compress** ("retaining only the tokens required to perform a task") and **isolate** ("splitting it up") (LangChain). This guide calls the first "persist".

Strategy	What it does	Examples from the sources	Provenance risk
Select	Load only what the step needs	Anthropic's "just in time" retrieval: keep lightweight identifiers (file paths, stored queries, links) and load data with tools at runtime; curate "a minimal viable set of tools"	Low if the item carries its source

Strategy	What it does	Examples from the sources	Provenance risk
Compress	Shrink what is already in context	Compaction: summarize a conversation near the limit and restart with the summary; tool result clearing as "one of the safest lightest touch forms" (Anthropic)	High: a summary is a model-written derivative that may lose source spans
Isolate	Give sub-tasks their own windows	Sub-agents that explore with many tokens but return "a condensed, distilled summary of its work (often 1,000-2,000 tokens)" (Anthropic)	Medium: the lead agent sees conclusions, not evidence
Persist	Store outside the window, recall later	Structured note-taking (Anthropic); MemGPT's paging between main context and external storage	High: model-written notes can re-enter as if they were facts

MemGPT makes the hierarchy explicit: "main context" (the prompt tokens: system instructions, a "working context" and a FIFO message queue) and "external context" in recall and archival storage. When prompt tokens exceed a warning threshold (the paper's example is 70% of the window), the system inserts a "memory pressure" warning so the model can save important information; at the flush threshold (for example 100%), it evicts messages (for example 50% of the window) and writes "a new recursive summary using the existing recursive summary and evicted messages" (Packer et al., MemGPT, §2.1-2.2). The summary that stays in context is a model-generated derivative of a derivative.

Anthropic's caution about compaction applies to every compressing step: "overly aggressive compaction can result in the loss of subtle but critical context whose importance only becomes apparent later" (Anthropic). For work that must be defended, the "subtle but critical context" is often the date, the qualifier or the source locator. The site's compression evidence-loss acceptance test scores exactly that.

Facts need provenance before they enter context

An agent's answer can be no better sourced than its context. ReAct gives an early measurement of why grounding matters. On a sample of HotpotQA trajectories, hallucination accounted for 56% of chain-of-thought failures and 0% of ReAct failures, because ReAct read from an external knowledge base. The same analysis found the cost: for ReAct, "non-informative search, which counts for 23% of the error cases, derails the model reasoning" (Yao et al., Table 2). Retrieval helps only when what it returns is relevant and sound.

The reverse also holds. LangChain cites Drew Breunig's list of long-context failure modes, which begins with **context poisoning**, "when a hallucination makes it into the context", and includes **context clash**, "when parts of the context disagree" (LangChain). A memory note written from an earlier, unsupported answer looks identical in the window to one copied from a registry record.

A defensible context therefore applies an admission rule before the relevance rule. Each item that asserts something about the world should carry, at minimum: the source identifier, the source version or retrieval time, whether the text is verbatim or derived, and who or what produced it. Items without those fields may be true, but they can only prompt a lookup; they should not be cited.

Worked example: Halden Therapeutics AG

Halden Therapeutics AG is a fictional Basel pharmaceutical company. The product code, registry, documents and token counts below are invented for illustration. The token counts are illustrative, not measurements.

A compliance analyst asks an internal research agent: *"What is the current development phase of HLD-204, and when is the next data readout expected?"* Before the answering step, the orchestrator holds about 44,650 tokens of candidate material.

Context budget worksheet

#	Component	Origin	Provenance status	Raw tokens	Decision	Tokens in final context
1	System instructions	Developer, version 14	Versioned, authored	1,800	Keep	1,800
2	Tool definitions (6)	Developer	Versioned, authored	2,400	Select 3 relevant tools	1,100
3	User question	Analyst, timestamped	Attributed	60	Keep, repeat near the end	60
4	Record: HLD-204 phase = Phase 2	Canonical record store	Cited, verbatim quote, registry version 9 read 11 March 2026	350	Keep, place next to the question	350
5	Registry entry, version 9, full text	Trial registry snapshot	Primary source, versioned, hashed	9,200	Compress to the quoted span plus locator	400
6	Web search results (8 pages)	Search tool	Mixed: 1 dated issuer press release; 1 aggregator stating "Phase 1" from 2024; 6 undated or unattributed	14,500	Keep the dated press release excerpt; drop the rest; log the superseded aggregator value	900
7	Memory note: "readout expected Q3 2026"	Written by the agent in an earlier session	No source recorded	40	Drop from context; trigger a lookup	0
8	Prior conversation (12 turns)	Analyst and agent	Mixed	6,800	Compress; retain identifiers and decisions	700
9	Earlier raw tool results (5 calls)	Tools	Already consumed	7,900	Clear	0
10	Sub-agent report on competitors	Sub-agent summary	Derived; sources listed	1,600	Drop: off-question	0

#	Component	Origin	Provenance status	Raw tokens	Decision	Tokens in final context
11	Readout date: declared unknown	Lookup log from step 7	No published source found; sources searched listed with dates	0	Add	120
	Total			**44,650**		**5,430**

Three decisions carry the method.

Row 7 is the trap. The memory note is short, relevant and confident, so a relevance-only rule would keep it. It has no source; admitted, it would let the agent state a readout date nobody published, the context-poisoning pattern above. The worksheet converts it into a lookup, which finds nothing, so row 11 records the date as unknown with the sources searched.

Row 5 compresses without losing evidence. Instead of summarizing the registry text, the orchestrator extracts the exact span and keeps its locator, so the compressed form is a quotation, not a paraphrase:

```
<evidence id="E1" kind="verbatim"
          source="https://registry.example.org/HLD-204" version="9"
          observed="2026-03-11T08:14:03Z">
Phase 2 study of HLD-204 in KRAS-mutant tumours
</evidence>
```

Rows 3 and 4 use position deliberately. Given Liu et al.'s results, the decisive evidence and the question sit at the end, not between conversation summaries.

The answer states "Phase 2, per registry entry version 9 observed 11 March 2026", cites E1, and reports that no published source for the readout date was found, rather than filling it from memory.

Failure modes and how to detect them

Failure mode	What it looks like	How to detect it
Context poisoning	A model-written note or earlier answer re-enters as a fact	Count factual context items without a source identifier; fail the run if one is cited
Lost in the middle	Correct evidence present but ignored	Paired runs moving the same evidence to start, middle and end
Context rot and distraction	Accuracy falls as unrelated or near-miss material accumulates	Run a focused variant of each case beside the full one, as Chroma did
Compression evidence loss	The answer survives a summary but its date, qualifier or locator does not	After each compaction, check every cited span still resolves to its source
Stale memory	An older value outranks a newer one	Temporal fixtures with known updates; measure the stale-answer rate (see the agent memory update test)

Failure mode	What it looks like	How to detect it
Context clash	Two sources disagree and the agent picks one silently	Detect conflicting values before generation; require disclosure
Tool overload	Wrong tool chosen, or tool definitions crowd out evidence	Log tool choice per step; Anthropic: if a human engineer cannot say which tool to use, "an AI agent can't be expected to do better"
Non-informative retrieval	Empty or irrelevant search results derail the loop	Record retrieval quality per step (23% of ReAct's errors)

Each detector depends on logging what entered the context at every step, with origins; otherwise a failure is visible but cannot be attributed.

Checklist for a defensible context

1. Version instructions and tool definitions; record them per run.
2. Expose only the tools the step needs.
3. Admit factual items only with source, version or observation time, and a verbatim-or-derived flag.
4. Treat model-written memory as a lead, not as evidence, until it resolves to a source.
5. Compress evidence by extracting spans with locators; summarize only conversation.
6. Clear raw tool results once they have been used.
7. Place decisive evidence where the deployed model uses it best, and test that.
8. Detect conflicts before generation and require disclosure.
9. Record unanswerable fields as declared unknowns, with the sources searched.
10. Log the assembled context, with origins, at every step.

Anthropic's overall advice is to "find the smallest set of high-signal tokens that maximize the likelihood of your desired outcome" and, for agent design, to find "the simplest solution possible" (Anthropic, Effective context engineering; Anthropic, Building effective agents). Provenance adds one condition: high signal also means traceable.

This is the problem Reliable Context's published guarantees address: a cited, verbatim, dated source behind every value, and unknowns declared, never blank.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
2. https://blog.langchain.com/context-engineering-for-agents
3. https://arxiv.org/abs/2507.13334
4. https://www.anthropic.com/engineering/building-effective-agents
5. https://arxiv.org/abs/2210.03629
6. https://arxiv.org/abs/2307.03172

7. https://research.trychroma.com/context-rot
8. https://arxiv.org/abs/2310.08560

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.