Inference-Time AI Data Control

How to Govern Data Use at Inference Time and How Caber Delivers It

Executive Summary

The companion paper When Data Becomes Context establishes the problem: AI systems fail not because data is wrong or unauthorized, but because the context required to use data correctly is missing at decision time. No infrastructure exists to identify, link, and reconcile data fragments across their many copies in a way AI can use at runtime.

This paper defines the solution: AI Data Control, a new category of operational control plane that governs how enterprise data contributes to AI-generated outcomes at inference time.

> *AI Data Control is not governance (which defines requirements but cannot enforce them in real time). It is not security (which prevents leakage but cannot preserve meaning). It is not MLOps (which improves models but doesn't regulate data contribution). It is a distinct capability that operates at the data-to-outcome layer, below governance frameworks, above access controls.*

Why Existing Tools Cannot Solve This

Enterprises have invested heavily in data governance, security, and AI infrastructure. None of these investments address the data use problem because each tool category was designed for a different purpose.

Tool Category	What It Does	Why It Fails for Data Use
Governance / GRC	Defines policies, accountability structures, compliance requirements	Design-time only; cannot enforce policies at inference time when data is actually used
AI Guardrails	Filters prompts for injection attacks; scans outputs for policy violations	Never sees the retrieved data that shaped the answer; creates gaps AI fills with hallucinations
DLP / DSPM	Discovers sensitive data; classifies by patterns (SSN, credit card, etc.)	Pattern-based; misses 70% of enterprise data that has no detectable patterns. Context-blind.
Data Catalogs	Tags and organizes structured datasets; tracks lineage through pipelines	Disconnected from unstructured content; cannot track fragments across copies or control use at runtime
API Gateways	Controls access to endpoints; rate limiting; authentication	Resource-based; sees that an API was called but blind to what data flowed through it
MLOps	Monitors model performance; manages model lifecycle and deployment	Model-centric; ignores the data contribution layer that determines AI behavior at inference

AI Data Control operates below governance and above access controls, at the data-to-outcome layer that existing tools do not reach.

The Boundary Conditions for AI Data Control

AI Data Control is constrained by technical realities unique to AI systems. Any solution that violates these boundary conditions will fail, regardless of how sophisticated the technology appears. These boundary conditions map directly to the four questions every AI answer must pass: Is this fresh? Where did it come from? Is it authorized? How do we resolve conflicts?

1. Fragment-Level Operation

AI does not consume documents. It consumes fragments, sentences, paragraphs, table rows, chart values. A fragment is an orthogonal building block: the same content can be matched anywhere it appears, regardless of what larger structure it's embedded in.

➡️ The solution must identify and track data at the fragment level, linking identical content across all copies and containers.

Example: The Fragment That Exists Everywhere

A product liability disclaimer appears in 2,400 documents across the enterprise. AI retrieves one instance. Without fragment-level identity, there's no way to know if this copy is current, which source it came from, or whether 12 of those 2,400 documents contain an outdated version from before a regulatory update.

2. Dual-Purpose Control (Promote and Protect)

Security-only controls that block data without understanding semantic meaning create informational gaps. AI systems are probabilistic, when context is incomplete, they infer. Blocking data often makes outcomes worse, not better.

➡️ The solution must both promote the right data reaching AI and protect against inappropriate data, simultaneously, not sequentially.

Example: The Redaction That Caused a Hallucination

A security tool redacts a customer ID from a support ticket fragment. The fragment explained why the customer's account was flagged for review. Without the ID, the AI loses the causal link. It infers a reason based on other fragments, and infers incorrectly. The AI confidently explains a policy violation that never occurred.

3. Duplicate and Conflict Resolution

Duplicate fragments bias AI toward overrepresented viewpoints. Conflicting fragments create confusion and contradictory outputs. Both problems are invisible to document-level tools.

➡️ The solution must detect duplicate and conflicting fragments before they enter the context window, deduplicating, reconciling, or flagging as appropriate.

Example: The Three Confirmations That Were One Fact

AI retrieves the same quarterly revenue figure from three sources: a board deck, an analyst report, and an email summary. The figure appears three times in the context window. The model treats this as three independent confirmations, dramatically increasing confidence. But it's one fact, copied three times. If that fact is wrong, the tripled confidence makes the error harder to catch.

4. Runtime Enforcement

Static curation cannot govern dynamic assembly. The same fragment may be appropriate for one user's query and inappropriate for another's. The same content may be current today and stale tomorrow. Governance must happen at the moment data is used, not when it's stored.

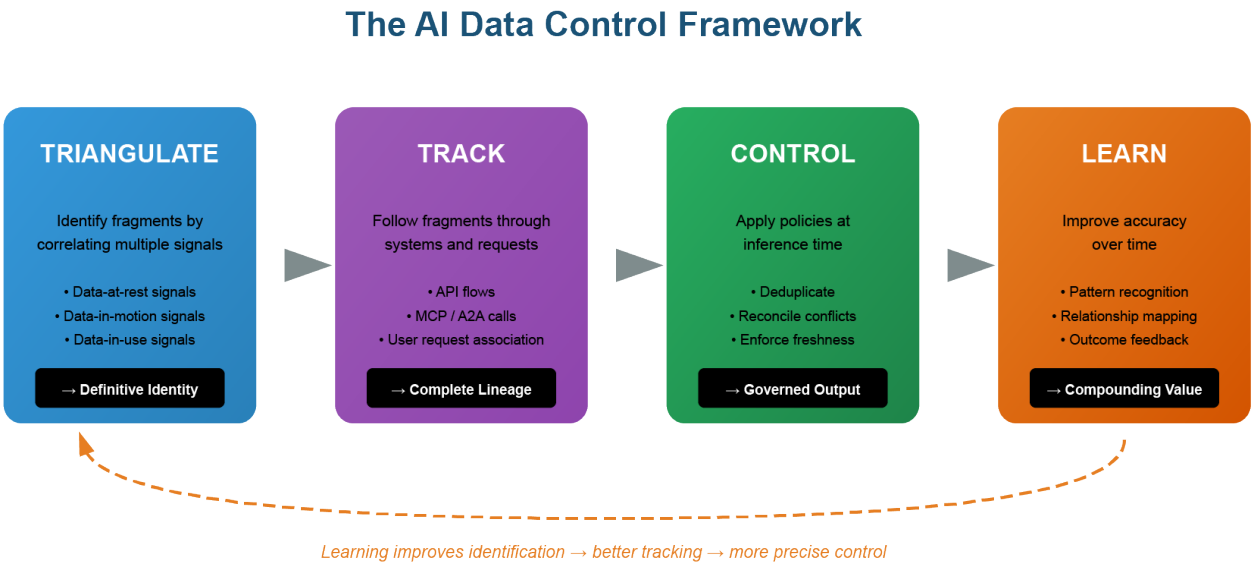
➡️ The solution must evaluate and enforce policies at inference time, dynamically, based on the current state of data and the specific context of each request.

Example: The Curation That Couldn't Keep Up

A data team spends six months curating a knowledge base for AI consumption: tagging, classifying, validating freshness. By the time curation is complete, 40% of the content has been updated, the AI use cases have evolved, and three new data sources have been added. The curated state is already obsolete.

The Solution Framework

To move from manual context curation to automated context policy, enterprises need four integrated capabilities. Each builds on the previous; none is sufficient alone.



Step 1: Triangulate

Identify any fragment by correlating multiple signals. No single signal, source metadata, semantic similarity, network activity, is reliable enough for definitive identification. But when multiple noisy signals are correlated, the noise cancels and identification becomes precise.

This is triangulation: using the intersection of signals from data-at-rest (where content is stored), data-in-motion (how content flows through APIs and agents), and data-in-use (how content contributes to AI outputs) to establish fragment identity with certainty.

The companion paper *Data Triangulation: The Technical Foundation* provides a deep technical treatment of how triangulation works and why it requires observing all three data states.

Step 2: Track

Follow identified fragments as they move. Once a fragment is identified, it can be tracked through APIs, MCP servers, agent-to-agent calls, and context assembly pipelines. Each fragment is associated with the user request it serves and every system that touched it.

Tracking produces lineage: a complete record of which fragments moved, from which sources, through which systems, to serve which query. When an audit asks "what data did AI use?", tracking provides the answer.

Step 3: Control

Apply policies at the moment of use. With identity and lineage established, governance shifts from "block or allow" to "optimize data use under policy and context."

Control ensures AI receives the freshest, most relevant, most appropriate content for each user and each query. An executive asking about quarterly performance sees data appropriate to their role. A frontline employee asking the same question sees data relevant to their function. Same query, different contexts, different optimal responses.

Control also prevents problems before they occur. Stale fragments are excluded. Conflicting fragments are reconciled or flagged. Unauthorized content never reaches the context window. The goal is not to block AI, it's to ensure AI has exactly what it needs to produce correct, compliant, high-quality outputs.

Step 4: Learn

Compound accuracy over time. Every interaction improves the system. Relationships between fragments become clearer. Identification patterns become more precise. Policy effectiveness becomes measurable.

This creates a compounding advantage. The longer the system operates, the more accurate identification becomes, not through manual curation, but through observed behavior. Early adopters gain an advantage that later entrants cannot replicate simply by deploying the same technology.

The Five Control Signals

AI Data Control evaluates multiple signals simultaneously to enforce context integrity. Each signal addresses a dimension of (**F**) freshness, (**P**) provenance, (**A**) authorization, and (**C**) conflict resolution.

1. Policy Signals → P A

Existing access controls, classification labels, regulatory requirements, and contractual obligations. Policy signals answer: Is this fragment permitted to reach this user for this purpose?

Example: Policy Signal in Action

A fragment from an M&A due diligence memo is retrieved for a market analysis query. Policy signal checks: the requesting user is not on the deal team; the memo is tagged as restricted to deal participants. The fragment is excluded before it reaches the context window, no leak, no hallucination from missing context, because the fragment was never appropriate for this query.

2. Relevance Signals → F P A C

Whether a fragment meaningfully contributes to the user's query. The relevance signal answers: Does this fragment help answer the question being asked?

Example: Relevance Signal in Action

A query about Q3 revenue retrieves 47 fragments. 12 discuss Q3 revenue directly. 8 discuss Q2 comparisons. 15 discuss unrelated Q3 initiatives. 12 are boilerplate disclaimers. Relevance scoring identifies the 20 fragments that actually contribute to the answer, excluding noise that would dilute signal and increase inference cost.

3. Semantic Meaning Signals → F C

Understanding duplication, conflict, and coherence between fragments. The semantic signal answers: How does this fragment relate to other fragments already selected?

Example: Semantic Signal in Action

Three fragments describe the same product specification, one from the official datasheet, one from a sales presentation, one from an outdated email. Semantic analysis identifies them as duplicates. The system selects the authoritative version and excludes the others, preventing the tripled-confidence problem.

4. Business Context Signals → A

How data relates to business processes, workflows, and organizational requirements. The business context signal answers: Is this fragment appropriate for how the user intends to use the AI output?

Example: Business Context Signal in Action

A fragment about manufacturing tolerances is technically accessible to a marketing analyst. But the business context signal recognizes the query is for a customer-facing presentation. Manufacturing tolerances are internal operational data, not appropriate for external communication. The fragment is excluded, not because of access restrictions, but because of use-case appropriateness.

5. **Usage-Derived Signals → F P A C**

   Observed outcomes that reveal patterns in data quality, policy effectiveness, and system behavior. The usage signal answers: What does historical behavior tell us about this fragment and this type of query?

Example: Usage Signal in Action
A particular knowledge base article has been retrieved 340 times in the past month. In 85% of cases, users reformulated their query or expressed dissatisfaction with the answer. Usage signal flags this fragment as low-value for similar queries, deprioritizing it in favor of fragments with better outcome histories.

Broader Implications: Beyond AI Governance

When a SpaceX booster explodes before landing, no amount of post-incident analysis recovers the payload. The mission fails. The only path forward is preventing the explosion from occurring in the first place.

Enterprise AI projects fail the same way. By the time you detect that AI used the wrong data, wrong version, wrong project, wrong confidentiality tier, the damage is done. The flawed output has already influenced a decision, informed a report, or reached a customer. The result is a familiar anti-pattern: context problems leak into production, and humans are asked to clean up after the fact by reviewing, correcting, and explaining outputs one incident at a time. *That is not context engineering; it is AI incident response with nearly 100% OpEx.*

What about curating data before it reaches AI? Curation helps, but it's fundamentally limited. You can choose what data AI is allowed to receive, but you cannot control how that data will be used. The same curated dataset that's appropriate for one query may be inappropriate for another. Curation is a static gate; AI use is dynamic. Worse, curation easily becomes a "boil the ocean" endeavor, months spent classifying data that AI will consume in ways no one anticipated. Incident Response

False positive alerts now exceed 95% because today's incident detection and response processes are fundamentally open-loop. Security tools generate indicators of compromise based on behavior and anomalies, but determining whether confidentiality, integrity, or availability has actually been compromised requires significant human effort to correlate logs and reconstruct data flows.

Fragment-level tracing enables closed-loop incident response where the data involved in any operation is immediately identifiable. Investigation time drops from weeks to hours because analysts see exactly which fragments moved, through which systems, serving which user request.

> *AI Data Control shifts the economics from cleanup to prevention, without requiring exhaustive upfront curation.*

Without AI Data Control	With AI Data Control
13% of governance needs enforceable	Up to 100% of policies enforced at inference
17% of problems found by humans-in-the-loop	Problems prevented before reaching the model
70% of AI projects fail	Projects ship with governance built in
3000x cost for post-hoc remediation	Issues caught at moment of data use

Measurable Outcomes

- AI answer quality improves 40%+ the context window contains relevant, deduplicated, policy-aligned data instead of noise and gaps
- False positives drop from >95% to <1% every enforcement decision is grounded in identified data and verified context
- Governance costs drop up to 90% policies unify on data itself rather than fragmenting across resource-based tools
- Inference costs drop 30%+ irrelevant and redundant data never reaches the model

How Caber Delivers AI Data Control

Caber is the first platform purpose-built for AI Data Control. We deliver the complete Triangulate → Track → Control → Learn framework as operational infrastructure.

Fragment-level identity: Caber identifies and links identical content across all copies and containers, regardless of format or location

Multi-signal triangulation: Caber correlates signals from data-at-rest, data-in-motion, and data-in-use to achieve precise identification

Runtime enforcement: Caber evaluates and applies policies at inference time, dynamically, for each user and each query

Continuous learning: Caber's context graph improves with every interaction, compounding accuracy over time

Integration flexibility: Caber operates across RAG pipelines, MCP servers, agent frameworks, streams, and API-based retrieval without requiring architectural changes

The result: enterprises can deploy AI with confidence, knowing that data use is governed at the moment it matters, not after the damage is done.