TraceScript Retrieval Integrity

Governing Retrieved Context Before It Becomes Trusted AI State

Canonical Public White Paper v1.0

Subtitle:
Retrieval integrity envelopes, influence control, RAG poisoning detection, canonical conflict checks, filter receipts, and replayable proof for agentic retrieval systems

Primary contribution: AI Retrieval Security
Secondary contribution: Context Integrity
Tertiary contribution: A runtime architecture for governing what agents retrieve before retrieved content becomes trusted context, memory, or action basis

---

Abstract

Retrieval is not neutral.

When an agent retrieves a document chunk, policy excerpt, ticket summary, or prior conversation, that content becomes **candidate substrate** — it may be shown to a user, summarized, written into memory, cited as evidence, or used to justify external action.

Most systems treat retrieval as a search problem. TraceScript treats retrieval as a **security boundary**.

The question is not only:

Did we find relevant content?

The deeper question is:

May this retrieved trace influence generation, memory, policy, or action — given its source trust, authority, canonical status, poisoning risk, contradiction state, scope, sensitivity, and replay burden?

TraceScript Retrieval Integrity is a runtime module that wraps retrieved content with **integrity envelopes**, filters candidate traces by **allowed influence**, detects **RAG poisoning** and **canonical conflict**, and emits **filter receipts** that bind to the integrity ledger.

Its winning sentence is:

Retrieved content cannot become trusted context without checks.

Its core doctrine is:

Every retrieved trace that may influence generation or action should carry an integrity envelope — and fail closed when poisoning, revocation, or scope violation is detected.

Keywords

TraceScript
Retrieval Integrity
RAG security
retrieval integrity envelope
influence control
canonical conflict
RAG poisoning
retrieval filter receipt
cross-tenant retrieval
action-basis eligibility
safe-to-trust check
agent memory contamination
retrieval authority score

---

1. Problem Statement

Enterprise AI systems retrieve constantly:

- Vector stores and RAG corpora
- Policy libraries and knowledge bases
- CRM records and ticket histories
- Prior agent turns and summarized threads
- Vendor PDFs and internal wikis

Failures appear as:

- **Retrieval poisoning** — adversarial or drifted chunks override policy
- **Silent authority** — retrieved summaries treated as official truth
- **Revoked influence** — superseded traces still retrieved and cited
- **Cross-scope leakage** — customer A context retrieved for customer B
- **Stale canonicalization** — old policy retrieved to support new action

These failures occur **before** the agent acts — at the moment retrieval becomes context.

---

2. Security Boundary

```
Candidate traces → influence evaluation → integrity envelope → allowed retrieval set
```

Stage	Question
Source classification	What produced this trace?
Influence request	What use is requested — working context, canonical memory, action basis?
Integrity envelope	What trust metadata wraps this trace?
Filter decision	allow · allow_with_warning · exclude · quarantine
Receipt	Immutable record of the filter decision

---

3. Retrieval Integrity Envelope

An envelope attaches trust metadata to each retrieved trace:

- source id, type, trust, and authority
- canonical status and allowed influence level
- freshness, scope, and sensitivity class
- poisoning and canonical conflict scores

- contradiction and revocation state
- action-basis eligibility and warnings
- filter receipt reference and replay status

The envelope answers:

- May this be shown?
- May it be summarized?
- May it update memory?
- May it support external action?
- May it become canonical?

Rule: No envelope, no trusted retrieval influence.

---

4. Influence Control

Retrieved traces carry **allowed influence** levels:

Level	Meaning
`none`	Display-only or audit — no downstream influence
`working_context`	Ephemeral session context
`canonical_memory`	May persist as trusted memory
`action_basis`	May support external action release

The retrieval filter compares **requested influence** against **trace allowed influence**, revocation state, and integrity scores.

Revoked or superseded traces are downgraded or excluded — preventing old policy from supporting new action.

---

5. RAG Poisoning Detection

Product-depth runtime evaluates retrieved chunks for:

- poison pattern scores (override language, bypass phrases)
- retrieval authority score (source trust × canonical conflict)
- canonical conflict against approved substrate
- decision: allow · allow_with_warning · require_review · quarantine · block

Use `evaluationProfile: "product_deep"` on integrity evaluate for RAG signals, or call product-depth RAG governance directly.

---

6. Runtime API Surface

Endpoint	Purpose
`POST /substrate/retrieval/filter`	Filter candidate trace IDs by influence + integrity
`POST /substrate/integrity/evaluate`	Full integrity path for retrieval signals
`GET /tracescript/observability/retrieval-integrity`	Tenant retrieval integrity dashboard view

Filter responses include **retrieval filter receipts** bound to the receipt ledger.

---

7. Proof Model

Retrieval Integrity is proven when:

1. Poisoned RAG chunks are flagged and blocked or quarantined
2. Revoked traces are excluded from canonical retrieval

3. Cross-tenant receipt reads fail at API and SQL layers
4. Filter decisions emit receipts with replay verification
5. Product-depth RAG events appear in observability views

See `/proof.html` module ladder and Developers retrieval demo preset.

---

8. Relationship to Platform Modules

Module	Relationship
Substrate Integrity Monitor	Parent substrate security — trusted-state formation
Policy Corpus Integrity	Policy retrieval and canonical policy authority
Memory Firewall	Memory writes after retrieval influence
Agent Action Firewall	Action basis must not rely on unfiltered retrieval
Receipt Ledger	Filter receipts + replay

Retrieval Integrity is the **believe-zone gate** between search results and trusted AI context.

---

9. Buyer Checklist Mapping

Retrieval Integrity helps answer:

- Can it block cross-customer retrieval?
- Can it prevent old policy from supporting action?
- Can it detect when weak information is made to look official through summaries?
- Can it repair, revoke, or block unsafe information from future use?

Full checklist: `/evaluation-checklist.html`

---

10. Deployment Path

1. Ingest retrieval signals with `targetSubstrate: "rag_store"`
2. Evaluate with `evaluationProfile: "product_deep"` for RAG depth
3. Filter candidate traces via `/substrate/retrieval/filter` before prompt assembly
4. Bind filter receipts; replay on audit
5. Monitor `v_tracescript_retrieval_integrity` per tenant

TraceScript Retrieval Integrity — safe-to-trust check for everything agents retrieve.