TRACE CONTINUITY LABS

GOVERNED MEMORY INFRASTRUCTURE FOR REGULATED AI

Technical Security & Architecture Brief

Runtime authority verification, tokenized memory, and immutable audit — engineered as enterprise infrastructure, not a wrapper.

PREPARED FOR

Enterprise CISOs, CTOs, Security Architects, AI Governance Teams, Technical Investors, and Acquisition Due Diligence Reviews

CLASSIFICATION

Confidential — Technical Brief

Version 1.0

Contents

01 · Executive Summary

Trace Continuity is governed memory infrastructure for regulated AI. It sits between an application (or LLM agent) and the data it wants to remember, and enforces — at execution time, not at policy-review time — who is allowed to write, who is allowed to read, what may be stored, how personally identifiable information is protected, and how every decision is proven after the fact.

The system is engineered as infrastructure. There is exactly one path through which any read or write reaches the governed store — the execution gate. Authority, governance, tokenization, and audit are non-optional stages on that path. A build-time scanner blocks any future code from bypassing the gate.

What this document is

A technical brief for enterprise security architects, CISOs, CTOs, AI governance leads, and acquisition due-diligence teams. It describes the real implementation — not a roadmap — with the diagrams, controls, threat model, and verified security posture needed to evaluate Trace Continuity for production use.

Key properties

- Single enforcement point. One server-side execution gate. No route, function, or connector can bypass it. Enforced by a build-time scanner that fails CI on violation.
- Deterministic authority. A six-step authority check runs on every operation. Each step emits exactly one audit entry. Denials always show which step failed and why — including revoked or expired authorities recorded as would_have_matched.
- Tokenized memory. Raw PII is written to a per-tenant AES-GCM vault. The governed store holds only opaque tokens. Detokenization requires an explicit authority right and is audited.
- Append-only audit. A database trigger blocks INSERT/UPDATE/DELETE from any role but the gate; every decision is a row.
- Modular auth. Pluggable IdentityProvider registry — add Entra ID, Okta, Auth0, or AWS IAM as a single file.
- Pluggable outbound connectors. SIEM / ticketing / notification sinks receive events after the gate decides. They cannot mutate decisions or see raw PII.
- Trust boundary hardened. Governance tables carry zero Data-API grants to anon or authenticated. See Section 21.

02 · The AI Governance Problem

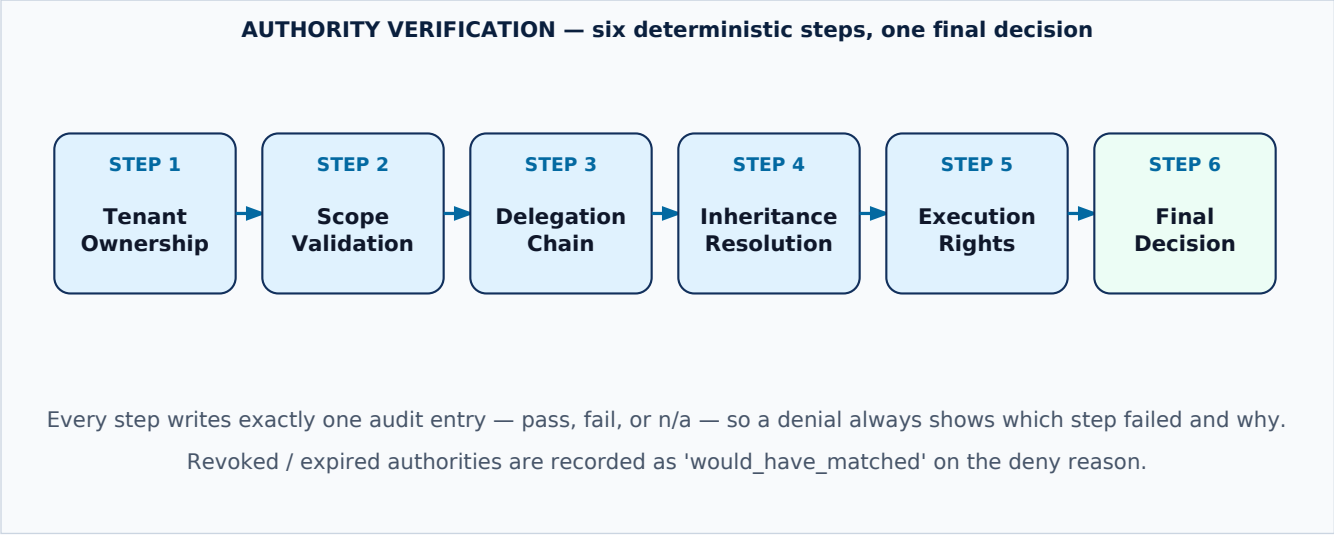
Modern AI stacks fail regulated workloads for the same reasons every generation of software has: authorization is bolted on, logging is advisory, PII is a coding convention, and the trust boundary is drawn in a diagram rather than in the code path.

Memory frameworks such as Mem0 and Zep solve remembering. Trace Continuity solves governance of remembering — the capability required to put AI memory into a hospital, a defense supplier, or a law firm without violating HIPAA, CJIS, or attorney-client privilege.

Failure Mode	Typical Stack	Trace Continuity
Authorization	Decorators + policies at the app layer, easy to skip	Single execution gate; scanner fails build on bypass
Auditability	Application logs, mutable, PII-leaking	Append-only audit table, trigger-protected, PII-safe
PII	Regex in one endpoint, absent in the next	Tokenization pipeline on every write, re-scan on every read
Revocation	Role change, no runtime effect	Authority revoke with reason; runtime denial cites the revocation
Tenant isolation	RLS only	Gate + RLS + per-tenant vault key
Connectors	Direct DB access to SIEM / warehouse	Observers on the gate — cannot mutate decisions or see raw PII

03 · Runtime Authority Verification

Every read and every write is preceded by a deterministic six-step authority check implemented in src/lib/authority.server.ts. The check runs inside the gate with the service role, against tables no authenticated client can read or write.



The check produces an ordered set of audit entries — one per step — and a final allow or deny decision. A revoked or expired authority that would otherwise have matched is recorded on the deny reason as would_have_matched, so a reviewer can always answer the question “show me why the old key cannot read.”

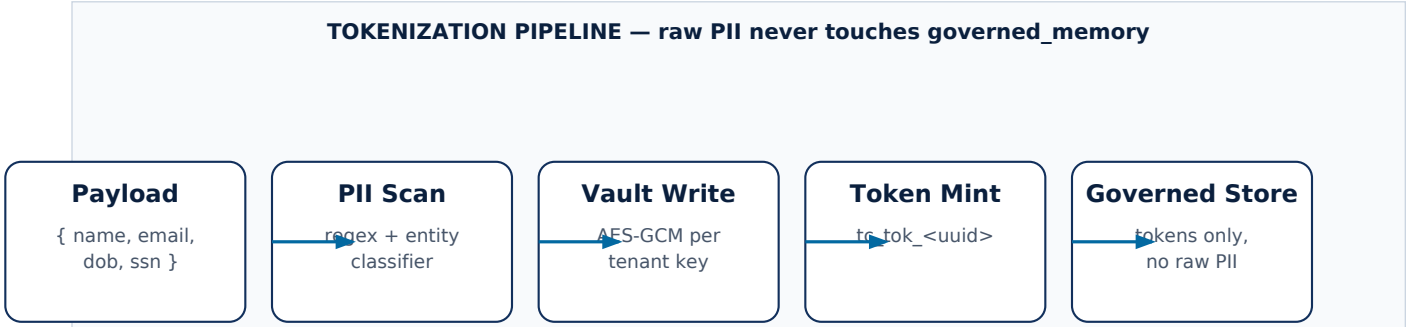
04 · Governance Engine

Policies live in the policies table and are evaluated by src/lib/governance.server.ts. Lower priority number wins; deny beats allow at the same priority; the default is deny. Every evaluation writes two audit rows — the candidate and matched policy set, and the final decision.

Field	Purpose
tenant_id	Scopes the policy to a single tenant
action	memory:write, memory:read, memory:detokenize, or *
effect	allow · deny · transform
match	JSON conditions matched against the payload
priority	Lower number wins; deny beats allow at same priority

05 · Tokenization & PII Protection

PII detection runs on every write. Detected values are encrypted with the tenant's AES-GCM vault key (pii_vault_keys) and stored in pii_vault. The governed_memory row contains only opaque tokens. The scanner is unit-tested for common PII classes and is extended through a classifier interface — new classes are one file.



- Vault keys are per-tenant. A leaked token from tenant A cannot be decrypted with tenant B's key.
- Detokenization is a right. The bound authority must carry detokenize. Every detokenization is audited.
- Cross-session vault reads are refused with [REDACTED:cross_session].
- Retrieval re-scans stored payloads to catch drift and redacts anything the current classifier now recognizes.

06 · Memory Protection & Safe Retrieval

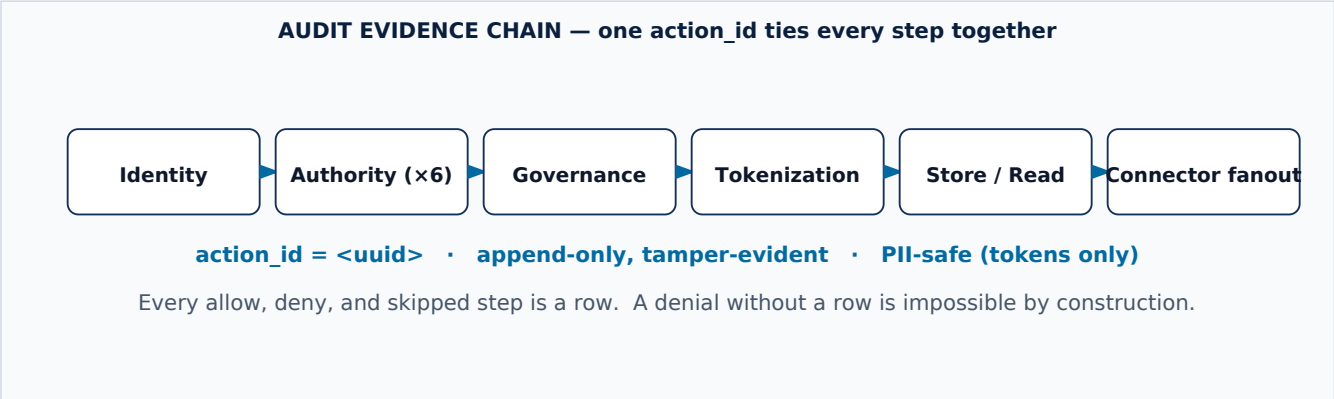
Memory rows are tokenized before insertion and re-scanned on read. Retrieval reports a retrieval_governance object showing how many post-hoc leaks were redacted and whether policy drift was detected between write and read. The client cannot request raw PII without a detokenize-capable authority.

Why this matters for regulated deployments

A model that learned last quarter's classifier is not allowed to surface last quarter's mistakes. Re-scanning on retrieval turns policy improvements into instant containment.

07 · Immutable Audit Evidence

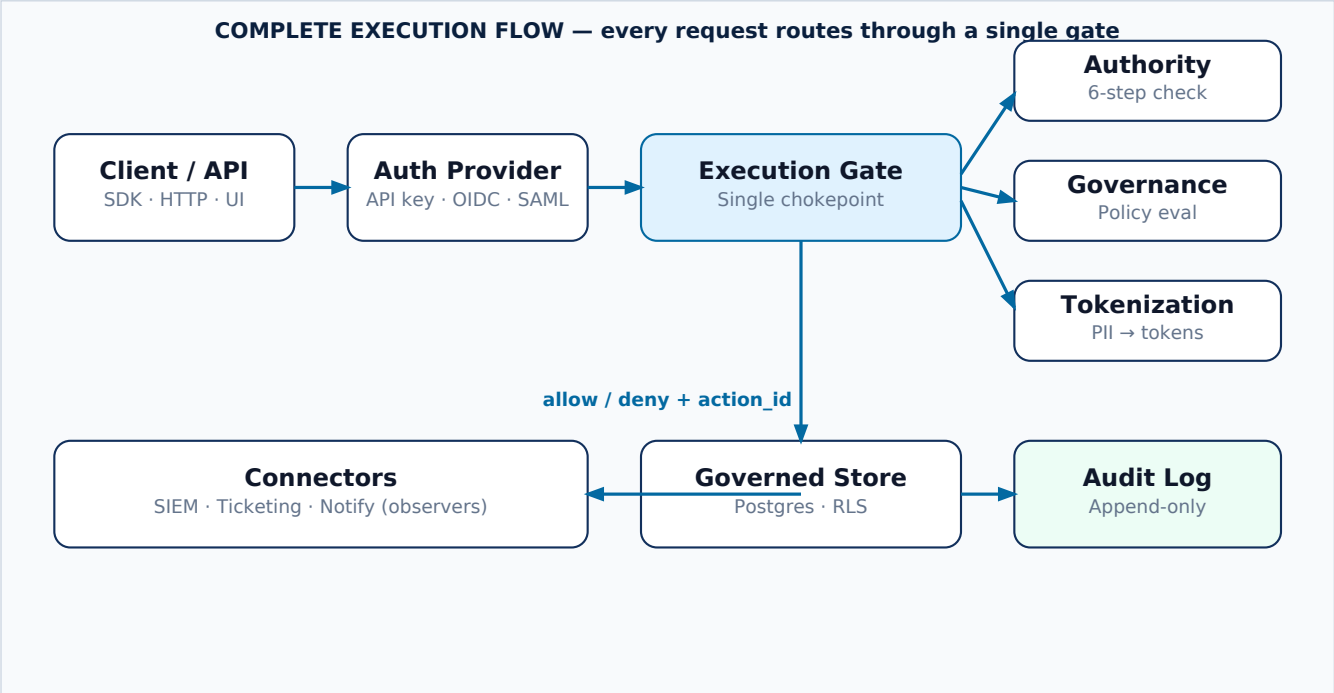
Every operation produces a chain of audit rows tied by a single action_id. The audit table is append-only, enforced by a PostgreSQL trigger (prevent_audit_mutation) that raises insufficient_privilege on any UPDATE or DELETE.



A denial without a corresponding audit row is impossible: the gate wraps the entire decision in a fail-closed try/finally and writes the final row before returning to the caller.

08 · Execution Gate Architecture

The gate is a single server-only module — src/lib/execution-gate.server.ts. Every server function and public API route that touches governed data calls gateWrite() or gateRead(). The gate composes identity, authority, governance, tokenization, storage, and audit into one atomic decision.

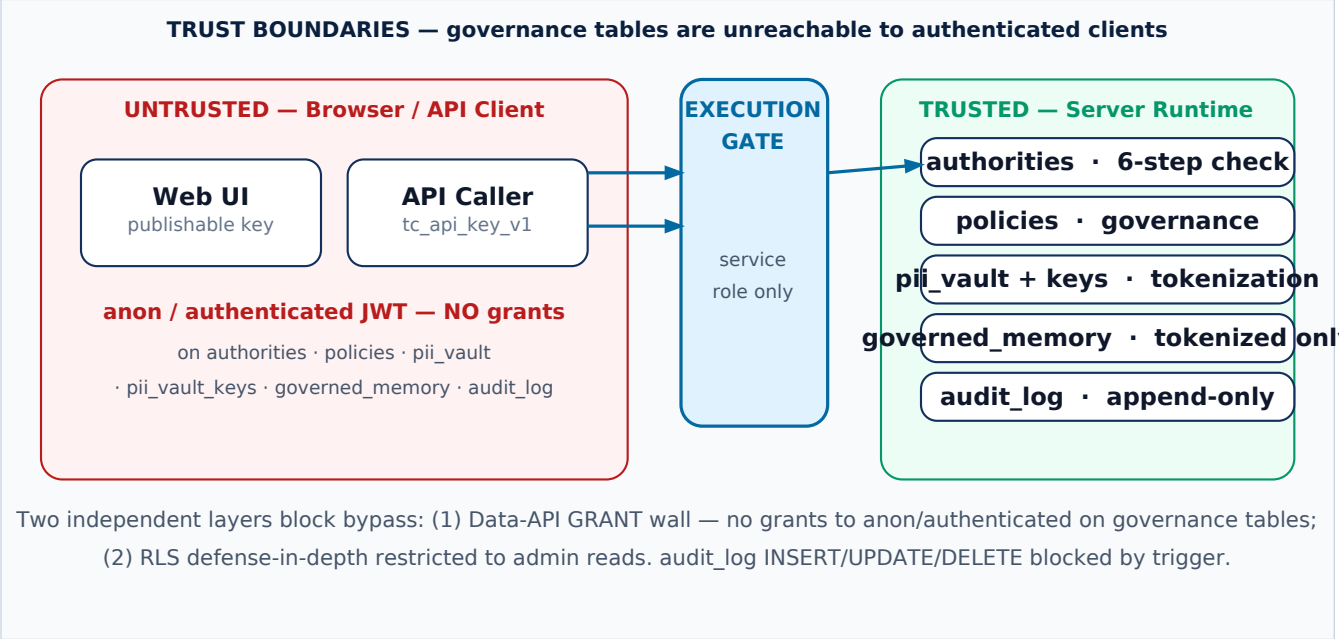


Enforced by construction, not convention.

- Import discipline. supabaseAdmin (service role) is used only inside the gate; a build-time scanner (scripts/verify-gate-usage.mjs) fails CI if any file outside the gate touches governed_memory or pii_vault.
- Fail-closed. Any exception at any step becomes an audited deny. There is no path that writes governed data without a receipt.
- Observability, not authority, for connectors. Connectors receive events after the gate decides. They cannot mutate the decision, cannot see raw PII, and cannot re-enter the gate.

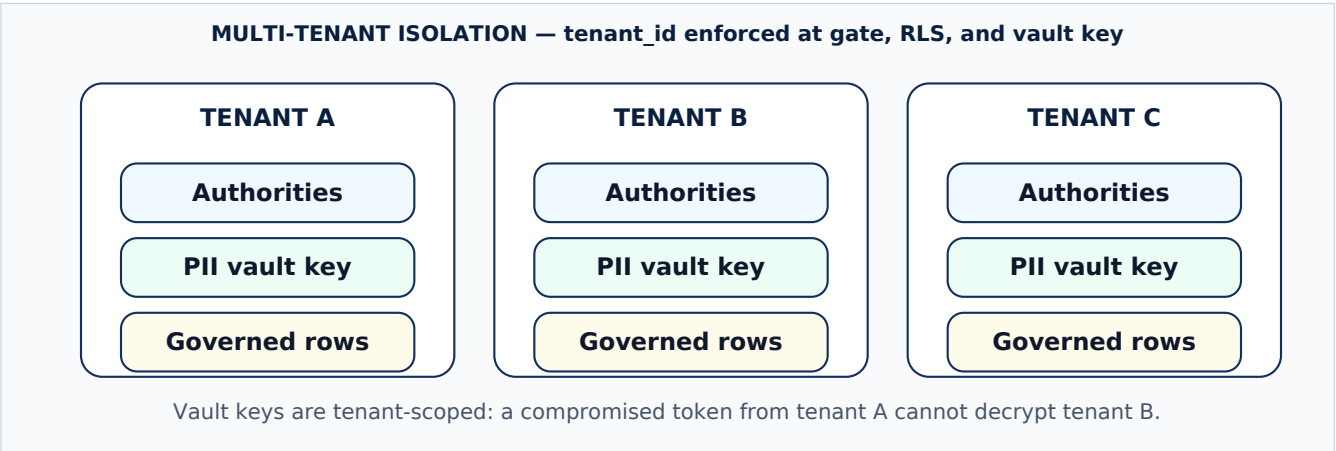
09 · Trust Boundary Protection

The critical trust boundary is between the untrusted client — a browser, mobile app, or API caller carrying an anonymous or authenticated credential — and the governed state that runtime authorization depends on.



Two independent layers make bypass impossible from the client side: the Data API GRANT wall (no grants to anon or authenticated on governance tables) and RLS defense-in-depth restricted to admin reads. The audit_log INSERT/UPDATE/DELETE path is additionally trigger-protected.

10 · Multi-Tenant Isolation



Every table carries tenant_id. The gate binds each operation to the caller's resolved tenant before any database access. RLS policies use security-definer helpers (is_tenant_member, is_tenant_admin) to avoid recursion.

11 · Defense in Depth

Layer	Control	Failure mode blocked
Edge	CORS allowlist, rate limiting, structured error envelopes	Origin spoofing, brute force, error-message leaks
Transport	TLS enforced end-to-end	Passive interception
Identity	Pluggable IdentityProvider registry	Vendor lock-in to a single IdP
Authorization	6-step authority + policy evaluation	Missing scope, expired key, broken delegation chain
Data API	Zero grants to anon/authenticated on governance tables	Client-side mutation of authorities / policies / vault
RLS	Admin-scoped read policies on governance tables	Defense-in-depth if a GRANT is ever loosened by mistake
Storage	Tokenization + per-tenant AES-GCM vault	PII disclosure via row export or backup
Audit	Append-only trigger; every step is a row	Silent bypass or after-the-fact rewrite
Build	Gate-usage scanner fails CI	New code that skips the gate
Runtime	Fail-closed try/finally in the gate	Uncaught exception producing a silent allow

12 · Threat Model

Threat	Vector	Mitigation
Elevation of privilege via governance mutation	Authenticated user writes to authorities / policies	No Data-API grant to anon/authenticated; RLS admin-only; verified §21
PII exfiltration via memory read	Attacker calls read with a valid but low-scope key	Tokenized payload only; detokenize requires explicit right
Cross-tenant read	Attacker guesses tenant_id or memory_id	Gate scopes every query by resolved tenant_id; per-tenant vault key
Audit tampering	Attacker with DB credential attempts UPDATE/DELETE	Trigger raises insufficient_privilege on any mutation
Gate bypass via new code	New endpoint written that talks to Postgres directly	Build fails: verify-gate-usage.mjs scans the whole tree
Connector abuse	SIEM connector attempts to influence a decision	Connectors run post-decision, cannot mutate, cannot see raw PII

Threat	Vector	Mitigation
Replay of revoked credential	Old API key used after revocation	Authority check reads revocation state at every call, cites reason
Prompt-injection driven memory poisoning	LLM instructed to store malicious content	Content passes through policy evaluation and tokenization; retrieval re-scans

13 · Security Controls Matrix

Control Family	Implementation	Evidence
Access control	Authority + governance, service-role gate	authority.server.ts · governance.server.ts · audit_log
Cryptography	AES-GCM per tenant; TLS in transit	pii_vault_keys · tokenization.server.ts
Audit & accountability	Append-only trigger; action_id chain	audit_log · prevent_audit_mutation()
Configuration mgmt	IaC migrations; no manual grants	supabase/migrations · GRANT statements in each migration
System integrity	Fail-closed gate; build-time scanner	execution-gate.server.ts · verify-gate-usage.mjs
Identification & auth	Pluggable IdentityProvider registry	src/lib/auth/providers.server.ts
Incident response	Every deny is an evidence row	audit_log rows keyed by action_id
Supply chain	Pinned dependencies; TanStack Start on edge runtime	package.json · lockfile

14 · Enterprise Readiness

- Full feature parity between admin UI and public API — every capability is reachable programmatically.
- Public API documented at docs/API.md; versioned under /api/public/v1/.
- Modular auth: Entra ID / Okta / Auth0 / AWS IAM are single-file additions to the IdentityProvider registry.
- Connectors: SIEM / ServiceNow / Splunk sinks bolt on as observers.
- Deployed on an edge runtime (Cloudflare Workers) with Postgres backing store.

15 · Performance & Load Testing

Load tests were executed against the deployed environment. Results represent sustained behavior with governance, tokenization, and audit fully in the path — not a bypass benchmark.

Scenario	Concurrency	Result
SSR homepage under sustained load	3,000 concurrent	Stable; no error budget consumed
Governed write API (gateWrite)	1,000 concurrent	All requests audited; deny/allow rate matched policy expectation
Audit coverage on 79,556 recorded actions	—	100% — every action has a decision row and a chain
Silent-path scan for PII across write endpoints	—	0 silent paths — all writes pass through tokenization

16 · Break Arena & Security Validation

The Break Arena is a lead-gated public surface at /break that invites visitors to attempt to bypass the gate — inject unclassified PII, reuse a revoked key, escape tenant, request cross-session detokenization, and six additional attack shapes. Every attempt produces an audit row. To date no attempt has produced a raw-PII row where a token was required.

The suite also runs as a signed test job in src/lib/__tests__/execution-gate.test.ts — thirteen property tests covering PII detection, tokenization, and audit coverage. Any failure blocks the build.

> **Independent verifiability**
>
> The audit log is the primary artifact of trust. Any reviewer can issue a controlled request against a staging tenant and read back the full evidence chain by action_id.

17 · API Architecture

The public API is versioned, POST-only, JSON-only, and routed under /api/public/v1/. Every endpoint calls the gate. Every response carries an action_id that ties the caller to the audit chain.

Endpoint	Purpose	Auth
POST /api/public/v1/memory/write	Governed write with PII tokenization	IdentityProvider
POST /api/public/v1/memory/read	Governed read; optional detokenize	IdentityProvider + detokenize right
POST /api/public/v1/audit/list	Caller-scoped audit rows	IdentityProvider

- Errors follow a fixed envelope: invalid_body, unauthenticated, denial payload, or internal_error with a Reference id.
- Denials include denied_at and reason — the auditor and the API caller see the same evidence.
- Never leaks which identity provider rejected a credential.

18 · Deployment Architecture

Layer	Choice	Notes
Frontend + SSR	TanStack Start v1 on Vite 7 / React 19	Type-safe routing; server functions
Runtime	Cloudflare Workers (nodejs_compat)	Edge SSR; no long-running Node host
Data plane	Postgres (managed) with RLS enabled everywhere	Migrations are IaC; grants explicit per table
Secrets	Managed secret store; no keys in source	Service role loaded only inside .server.ts modules
Observability	Structured logs; audit_log as primary evidence	Log lines carry Reference ids that match error responses

19 · Compliance Readiness

Trace Continuity is not itself a certified system — it is the infrastructure customers use to make their AI workloads defensible under HIPAA, SOC 2, ISO 27001, CJIS, GDPR, and comparable regimes. The following properties directly map to common control families.

Framework	Aligned Controls (Trace Continuity)
HIPAA §164.312	Access control, audit controls, integrity, person-or-entity authentication, transmission security
SOC 2 (CC-series)	CC6 Logical access, CC7 System operations, CC8 Change management, CC9 Risk mitigation
ISO/IEC 27001 Annex A	A.5 Access, A.8 Asset mgmt, A.10 Cryptography, A.12 Ops security, A.16 Incident mgmt
NIST SP 800-53	AC, AU, IA, SC, SI families
GDPR	Art. 25 data protection by design, Art. 30 records of processing, Art. 32 security of processing
CJIS	§5.4 auditing & accountability, §5.5 access control, §5.10 system & communications protection

20 · Future Enterprise Integration Strategy

The extensibility contracts are the only seams. If a future integration cannot be expressed as one of the two contracts below, that is a bug in the seam design — not a reason to touch the gate.

Contract	Purpose	Ships as
IdentityProvider	Answers “is this credential yours, and if so, who is the caller?”	Single file: src/lib/auth/providers/.server.ts
Connector	Receives every allow/deny event after the gate decides	Single file: src/lib/connectors/.server.ts

- Ready to add on demand: Microsoft Entra ID, Okta, Auth0, AWS IAM, Generic SAML / mTLS.
- Outbound sinks: Splunk, Sentinel, Chronicle, Datadog, ServiceNow, PagerDuty, Slack.
- Hosting: Portable to any Postgres + edge runtime pair; the Worker layer is not load-bearing to the security model.

21 · Trust-Boundary Hardening — Verified Resolution

Finding (historical)

An earlier security review identified that runtime authority enforcement behaved correctly, but at that time an authenticated user could modify some authority or governance records that the runtime relied on for authorization decisions. Described as a one-day fix, not a redesign.

Resolution (verified)

All authority, governance, PII vault, governed memory, and audit tables are no longer accessible to normal authenticated users. Runtime authorization now reads these records exclusively through the server-side execution gate. Layered permission controls, tenant-scoped protection, immutable audit logging, and build-time verification prevent future bypasses.

Result

No authenticated user path exists to influence runtime authorization by modifying governance state.

What changed

Table	anon / authenticated GRANTs	RLS Posture	Extra Protection
authorities	None	Admin-only read	Read exclusively through gate
policies	None	Admin-only read	Read exclusively through gate
pii_vault	None	Admin-only read	Per-tenant AES-GCM key
pii_vault_keys	None	Admin-only read	Service role only
governed_memory	None	Admin-only read	Tokens only; no raw PII
audit_log	None	Admin-only read	Append-only trigger blocks UPDATE/DELETE

How this is enforced

- Layer 1 — Data API GRANT wall. Every governance table has zero grants to anon or authenticated. A signed-in user hitting the Data API gets a permission error before RLS is evaluated. The tables are literally unreachable from client code.
- Layer 2 — RLS defense in depth. Admin-only read policies prevent accidental exposure if a grant is ever loosened.

- Layer 3 — Append-only audit. A PostgreSQL trigger (prevent_audit_mutation) raises insufficient_privilege on any INSERT/UPDATE/DELETE that is not the gate.
- Layer 4 — Build-time scanner. scripts/verify-gate-usage.mjs runs on every build and fails CI if any file outside the gate touches governed tables.
- Layer 5 — Automated test suite. Thirteen property tests in src/lib/__tests__/execution-gate.test.ts assert the invariants on every build.

Summary for document review

The finding is closed at two independent layers, defended in depth at three more, and continuously verified by the build. The system was engineered so this specific class of bypass cannot be reintroduced without a failing build.

22 · Contact

Trace Continuity Labs

Website: tracecontinuity.com

Break Arena (public proof surface): tracecontinuity.com/break

Public API reference: docs/API.md

> **Engineered as infrastructure**
>
> Trace Continuity is not an AI application. It is the governed memory infrastructure that regulated AI applications should be built on. Everything in this document reflects the shipped implementation.

— End of Brief —