



TRACELOCK BY ODINGARD

TraceLock Technical Product Specification and Architecture Guide

Architecture, data contracts, security model and operational pipeline
for continuous compliance monitoring

VERSION 1.0



© 2026 Odingard. All rights reserved.

TraceLock Technical Product Specification and Architecture Guide

Document title	TraceLock Technical Product Specification and Architecture Guide
Version	Version 1.0
Owner	Odingard — Architecture: James
Classification	Confidential
Product	TraceLock

This document describes TraceLock, a hosted software-as-a-service platform operated by Odingard. TraceLock is not installed into customer cloud projects or on-premises environments; customer systems are connected as read-only scan targets.

The information in this document is provided for the operation and evaluation of TraceLock. Service commitments for a specific subscription are governed by the applicable order form or private offer.

Product support: support@odingard.com

Security and vulnerability reports: security@odingard.com

TraceLock and Odingard are trademarks of Odingard. All other trademarks are the property of their respective owners.

Contents

Product definition and boundaries	1
Value proposition	1
Target personas	1
Product boundaries	2
Service objectives	2
System architecture and component topology	5
Component topology	5
Request path	6
Evaluation path	6
Scheduled workloads	7
Security boundaries	7
Connector engine and ingestion pipeline	9
Credential lifecycle	9
Scheduling	9
Worker execution	9
Ingestion sources	10
Freshness	10
Privacy Wall architecture	12
Design principle	12
Enforcement stages	12
Outbound redaction	13
What the database holds	13
Verification	13
Unified Control Library and crosswalk engine	15
Model	15
Evaluation outcomes	16
AI governance mappings	16

Framework accuracy posture	16
Exceptions	16
Evidence integrity model	19
Artifact anatomy	19
Digest computation	19
Deduplication and append behaviour	20
Retention	20
Provenance and timing	20
Attested file artifacts	20
Independent verification	20
Identity and authorization architecture	23
Authentication paths	23
SAML configuration model	23
SCIM provisioning model	24
Three independent authorization axes	24
Session and audit properties	24
Auditor access control plane	26
Token model	26
Isolation properties	26
Revocation	27
Export	27
Data schemas	29
ControlEvaluationFinding	29
ImmutableEvidenceArtefact	30
Verification contract	31
Threat model and mitigations	33
Trust assumptions	33
Threats and mitigations	33
Residual risks	35

Reference implementations	37
Canonicalization and evidence digest	37
Privacy Wall filter	38
Crosswalk evaluator	40
Deployment, CI/CD and verification	43
Runtime	43
Release pipeline	43
Integrity verification tests	44
Infrastructure as code	44
Observability	45
Glossary	47
Product terms	47
Identity and administration terms	48
AI governance terms	49
Connector terms	50

Product definition and boundaries

TraceLock is a continuous compliance and evidence platform operated by Odingard as a hosted service. It evaluates the security and governance posture of systems a customer already owns, records the result as tamper-evident evidence, and presents that single evidence set against every framework the customer reports on.

Value proposition

TraceLock is built on three product commitments.

Continuous control monitoring. Controls are evaluated on a schedule by read-only connectors rather than assembled by hand before an audit. Every evaluation produces a status, a timestamp, and an evidence record, so control state is a live property of the environment instead of a quarterly snapshot.

Evaluate once, report many. A single internal control code maps to the equivalent requirement in nine frameworks: SOC 2, ISO/IEC 27001, NIST CSF 2.0, NIST AI RMF 1.0, ISO/IEC 42001:2023, the EU AI Act (Regulation (EU) 2024/1689), the HIPAA Security Rule, SEC rule references, and CIS Controls v8. One evaluation and one evidence artifact satisfy every mapped framework requirement simultaneously.

Evidence that survives challenge. Evidence is stored with a SHA-256 digest computed over a canonical form of the recorded posture metadata, on a retention clock, in an append-oriented store. An auditor can recompute the digest independently and confirm that the record they are reading is the record that was collected.

Target personas

Persona	Primary surface	What they need from TraceLock
CISO / Head of Compliance	Dashboard, Estate Map	Framework coverage, failing controls, and audit readiness at a glance
GRC analyst	Control Center, Evidence	Per-control detail, evidence freshness, exceptions, and remediation tracking
Platform / DevOps engineer	Connectors, Remediation	Connector configuration, failure diagnosis, and remediation guidance

Persona	Primary surface	What they need from TraceLock
External auditor or examiner	Auditor Room	Time-bound, read-only access to controls and evidence without a tenant licence
Odingard platform staff	Platform administration	Cross-tenant organization lifecycle, plan, and support operations

Product boundaries

Boundaries are as much a part of the specification as capabilities. The following statements are architectural guarantees, not policy preferences.

- **TraceLock is not installed into customer environments.** No agent, appliance, VPC peering, or on-premises component is deployed into a customer project. Customer systems are connected as read-only scan targets over provider APIs.
- **TraceLock does not ingest customer business data.** The Privacy Wall admits only an allowlisted set of posture keys. Raw provider payloads, object contents, log bodies, and personal data are dropped before persistence.
- **TraceLock does not mutate customer environments.** Connector credentials are read-only by design, and remediation is delivered as guidance, tickets, and infrastructure-as-code snippets that the customer applies.
- **TraceLock is not the customer's system of record for identity.** Enterprise identity remains in the customer's identity provider; TraceLock consumes SAML assertions and SCIM operations from it.
- **Framework mappings are engineering artifacts, not legal advice.** Crosswalk references are curated at clause and article level and are intended to be confirmed by the customer's compliance or legal reviewer before being relied on as an audit deliverable.

Service objectives

Objective	Target
Platform availability	99.9% monthly
Evidence ingestion and digest computation	Under 500 ms at P99 per artifact
Connector scheduling interval	Scans scheduled and drained every 5 minutes
Notification dispatch	Queued deliveries drained every 5 minutes
Auditor Room isolation	Zero cross-tenant read paths; every auditor request is scoped by signed token

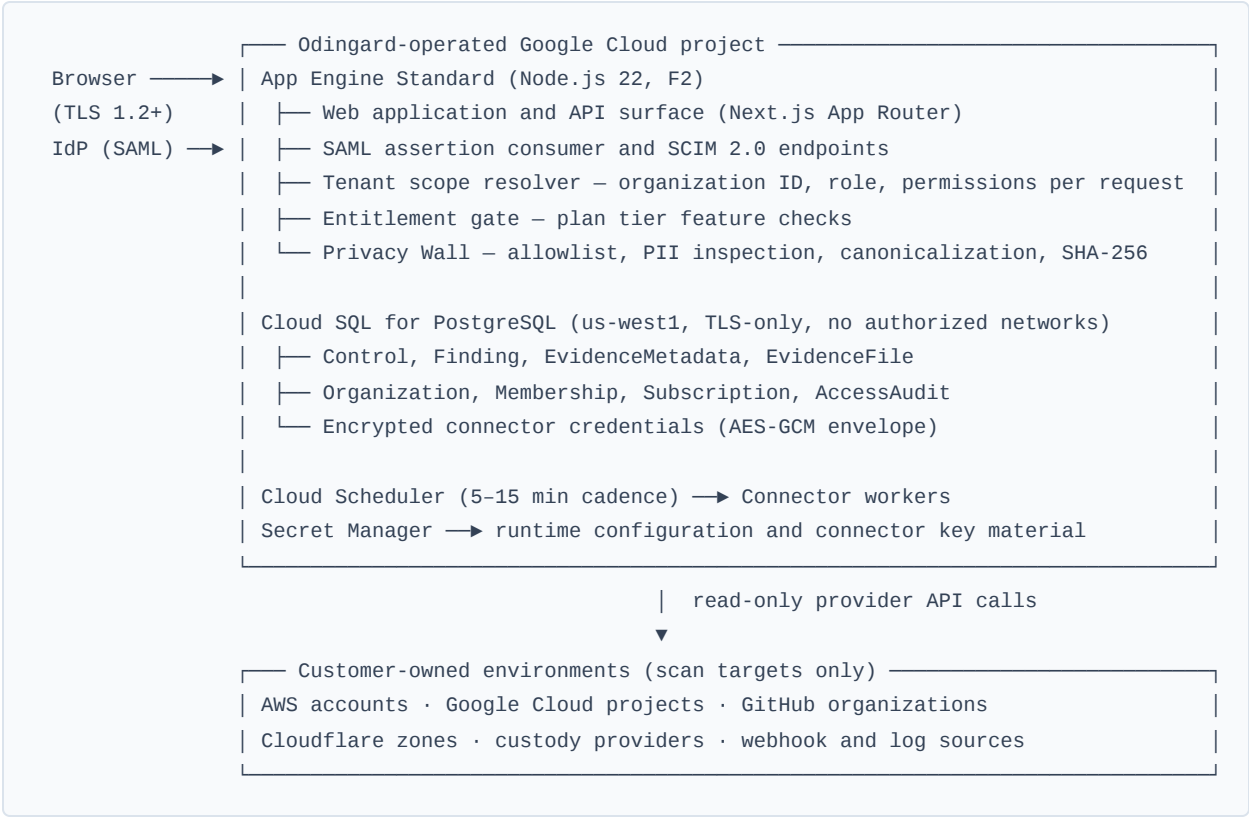
Objective	Target
Evidence retention	Seven years from observation by default, per artifact retention clock

Service commitments for a specific subscription are governed by the applicable order form or private offer.

System architecture and component topology

TraceLock is a single multi-tenant application deployed on Google App Engine Standard with a managed PostgreSQL database, a scheduler, and a set of read-only connector workers. Every tenant is served by the same deployment and separated logically by an organization-scoped data model rather than by per-tenant infrastructure.

Component topology



Three tenancy classes appear in this topology and must not be conflated:

Class	Components	Operated by
Vendor-hosted service	App Engine service, Cloud SQL, Cloud Scheduler, Secret Manager, connector workers	Odingard
Commerce surfaces	Google Cloud Marketplace listing, entitlement and billing messages	Google
Customer-owned scan targets	Cloud accounts, source control, edge and custody providers	Customer

No TraceLock component runs inside the customer-owned class.

Request path

1. **Authentication.** The browser presents a session established by local password authentication with TOTP MFA, or by a SAML assertion validated against the organization's identity provider.
2. **Tenant scope resolution.** Each request resolves a tenant scope containing the user identity, organization ID, and effective permissions. Data access is expressed in terms of that scope; there is no ambient query path that omits the organization ID.
3. **Entitlement evaluation.** Feature access is checked against the organization's plan tier independently of role. Role never implies entitlement, and entitlement never implies role.
4. **Data access.** Reads and writes are executed against Cloud SQL over the Cloud SQL Auth proxy using a Unix socket. The instance has no authorized networks and accepts encrypted connections only.

Evaluation path

1. **Scheduler.** Cloud Scheduler invokes the scan endpoint every five minutes. The scheduler selects due connectors, respecting per-organization cadence and lifecycle state; suspended organizations are read-only and are not scanned.
2. **Connector worker.** The worker decrypts the tenant credential from its AES-GCM envelope in memory, calls the provider's read-only APIs, and produces posture observations. Credentials are never written to logs or returned by any API.
3. **Privacy Wall.** Observations are inspected for personal data, reduced to allowlisted keys, canonicalized, and hashed. Anything outside the allowlist is discarded rather than stored and redacted.
4. **Control evaluation.** The observation is evaluated against the control definition, producing `PASSED`, `FAILED`, `INDETERMINATE`, `NOT_APPLICABLE`, `EXEMPT`, or `NOT_STARTED`, and a finding where the evaluation fails.
5. **Crosswalk projection.** The control's framework mappings project the result onto each mapped framework at clause or article level.
6. **Evidence write.** The sanitized metadata, its digest, the source provider, and the retention clock are written as an evidence record and linked to the control. Identical digests for the same control are deduplicated rather than duplicated, so an unchanged posture does not inflate the evidence set.
7. **Notification and ticketing.** Failure triggers fan out to the configured destinations — Slack, email, webhook — and optionally open a Jira or ServiceNow issue.

Scheduled workloads

Workload	Endpoint	Cadence
Connector scan scheduling and draining	/api/cron/scans	Every 5 minutes
Notification delivery draining	/api/cron/notifications	Every 5 minutes
Evidence expiry notification	/api/cron/evidence-expiry	Every 15 minutes
Training assignment and reminders	/api/cron/training	Every 15 minutes

Security boundaries

Boundary	Enforcement
Tenant boundary	Organization-scoped tenant scope on every query; cross-tenant reads have no code path
Staff boundary	Cross-tenant platform administration requires both the ADMIN role and membership of the staff allowlist
Ingestion boundary	Privacy Wall allowlist and PII inspection reject non-conforming metadata before persistence
Auditor boundary	HMAC-signed, time-bound, read-only scope; GET-only method enforcement
Credential boundary	AES-GCM envelope encryption with key material held in Secret Manager; decryption occurs in process memory only
Machine boundary	Machine access requires a hashed organization key presented per request and is rate limited

Connector engine and ingestion pipeline

Connectors are the only components that touch customer environments. Their design constraints are narrow by intent: read-only credentials, scheduled execution, no write paths into the customer environment, and no path into the database that bypasses the Privacy Wall.

Credential lifecycle

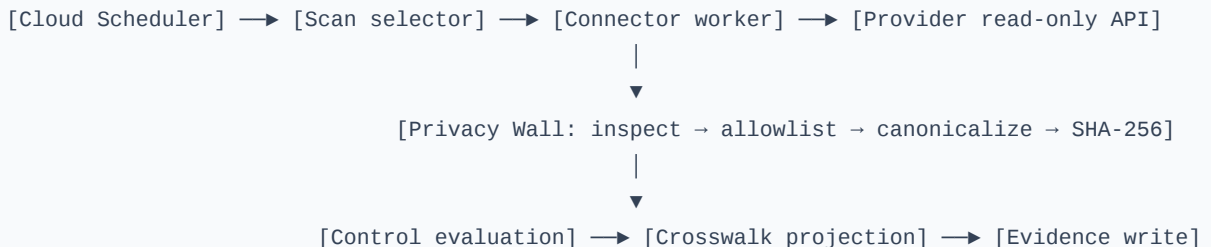
1. **Registration.** An administrator supplies a read-only credential or configures cross-account trust for the provider.
2. **Envelope encryption.** The credential is encrypted with AES-GCM under key material held in Secret Manager and stored as versioned envelope ciphertext. The plaintext is never written to the database, to logs, or to any API response.
3. **Smoke test.** A lightweight validation call confirms that the credential authenticates and holds the expected read scope before the connector is scheduled.
4. **Use.** Workers decrypt the envelope in process memory for the duration of a scan.
5. **Revocation.** Revocation removes the stored envelope and marks the integration disconnected. Because the credential is read-only and provider-side, the customer can also revoke unilaterally at the provider, which the next scan reports as lost permission.

Scheduling

The scan endpoint is invoked every five minutes and performs two functions: selecting connectors that are due, and draining scans that are queued. Selection respects per-organization cadence, connector status, and organization lifecycle state — a suspended organization is read-only and is not scanned. Providers that require tenant-configured cross-account access are not scheduled until that access exists; until then the connector reports the reason rather than failing silently.

Worker execution

A connector run proceeds through a fixed sequence:





Failure semantics are explicit at each stage:

Failure	Result
Credential rejected by provider	Integration marked <code>ERROR</code> ; controls become <code>INDETERMINATE</code> ; no evidence written
Permission narrowed at the provider	Affected controls become <code>INDETERMINATE</code> with the missing permission recorded
Provider rate limiting or transient error	Run is retried on the next scheduled cycle; last known posture and its freshness are retained
Privacy Wall rejection	Write aborts with a Privacy Wall error; no partial record is persisted
Region outside residency allowlist	Observation rejected

Ingestion sources

Three ingestion modes feed the same pipeline and the same Privacy Wall:

- **Scheduled polling** of provider APIs for cloud, source control, edge, and custody providers.
- **Webhook receipt** for providers that push events, with signature verification at the receiving route and the event reduced to allowlisted posture keys.
- **Machine submission** through the organization-scoped machine endpoint, used for AI agent governance events, authenticated by a hashed organization key and rate limited per key.

Freshness

Freshness is a first-class property. Each control records when it was last verified, and evidence records carry an observation timestamp and a retention clock. A control whose last verification has aged past its expected cadence is reported as stale rather than as passing, which prevents a disconnected connector from presenting as continued compliance.

Privacy Wall architecture

The Privacy Wall is the ingestion boundary between customer environments and the TraceLock database. Its purpose is structural: TraceLock cannot leak customer business data it never stored, and it cannot store what the Privacy Wall does not admit.

Design principle

The Privacy Wall is an allowlist, not a redaction pass. A redaction pass assumes the payload is stored and dangerous content is removed; an allowlist assumes nothing is stored and only declared posture keys are admitted. Any key the connector produces that is not on the allowlist is dropped before the record reaches the persistence layer, which means a connector cannot widen the data footprint by returning more fields.

Enforcement stages

Every evidence write passes through four stages in order.

1. Inspection. The metadata object is walked recursively. Strings are tested against patterns for email addresses, national identifiers, payment card numbers, and telephone numbers. A match on an undeclared field aborts the write with a Privacy Wall error rather than sanitizing it silently, so a connector defect surfaces as a failure instead of a partial record. A small set of fields — the declared owner and user email fields required for training and ownership records — are the only paths permitted to carry an address.

2. Allowlist reduction. Object keys are filtered to the declared posture allowlist, which admits only control and posture attributes: provider, mode, control code, finding type, severity, status, freshness, region, resource type and count, configuration drift, observation and receipt timestamps, byte length, framework list, score, AI autonomy and operator role, EU AI Act risk category and Annex III area, exclusion metadata, and similar keys. Provider payload bodies, object contents, log lines, request bodies, and identifiers of individual end users have no admitted key.

3. Canonicalization. Remaining keys are sorted deterministically and nested structures are canonicalized recursively. Canonicalization is what makes the digest reproducible: two observations of the same posture produce byte-identical canonical forms regardless of the order the provider returned them.

4. Digest and residency. A SHA-256 digest is computed over the canonical form, and the declared region is checked against the configured residency allowlist. An observation from a region outside the allowlist is rejected.

Outbound redaction

The Privacy Wall also governs text leaving TraceLock. Text destined for third-party destinations — Slack messages, webhook bodies, Jira and ServiceNow issue descriptions — is passed through an outbound redaction pass that replaces matched personal-data patterns and truncates the payload to a bounded length. Notification content therefore carries control codes, statuses, and remediation references rather than environment detail.

What the database holds

Category	Stored	Not stored
Posture	Control code, status, severity, freshness, resource counts, drift indicators	Provider payload bodies
Provenance	Source provider, region, observation timestamp, digest	Raw API responses
Personal data	Declared owner and learner email addresses required for assignment and ownership	Customer end-user identities, log contents, object contents
Credentials	AES-GCM envelope ciphertext	Plaintext credentials in any store or log

Verification

The Privacy Wall is verifiable from outside the system, which is the property that matters to an auditor. Three checks are available:

1. **Evidence inspection.** Every evidence record exposes its stored metadata and digest. An auditor can confirm that the record contains posture keys only.
2. **Digest recomputation.** Recomputing SHA-256 over the canonical metadata reproduces the stored digest, proving that what was hashed is what is displayed.
3. **Negative testing.** The regression suite asserts that metadata carrying personal-data patterns or undeclared keys is rejected or reduced, and that residency enforcement rejects out-of-region observations.

Unified Control Library and crosswalk engine

The Unified Control Library is the mechanism behind "evaluate once, report many". A control is defined and evaluated once; the crosswalk projects its outcome onto every framework that recognizes an equivalent requirement.

Model

The library has three layers.

Internal control codes. Each control has a stable internal code, a title, an owner, a status, and a verification timestamp. The code — not the framework citation — is the unit of evaluation, ownership, and evidence linkage.

Framework mappings. Each control code carries a mapping record with an optional reference per framework. A missing reference is meaningful: it records that no defensible mapping exists, which is preferable to inventing one.

Framework	Reference granularity
SOC 2	Trust services criterion (for example <code>CC6.1</code>)
ISO/IEC 27001	Annex A control (for example <code>A.9.2.1</code>)
NIST CSF 2.0	Function and category outcome (for example <code>PR.AC-1</code>)
HIPAA Security Rule	Safeguard citation (for example <code>164.312(a)(1)</code>)
SEC	Rule reference (for example <code>206(4)-7</code>)
CIS Controls v8	Control or safeguard family (for example <code>CIS 6.1</code>)
NIST AI RMF 1.0	Function-level reference (for example <code>GOVERN 1.1</code>)
ISO/IEC 42001:2023	Clause reference (for example <code>Clause 5.2</code>)
EU AI Act	Article reference under the adopted 2024 numbering (for example <code>Art. 15</code>)

Projection. An evaluation result is projected across every mapped framework at read time. Coverage, gap, and readiness views are derived from the projection rather than maintained as separate per-framework state, so a single control failure appears consistently in every framework it affects and cannot be green in one report and red in another.

Evaluation outcomes

Status	Meaning
PASSED	The control was evaluated and satisfied
FAILED	The control was evaluated and not satisfied; a finding is recorded
INDETERMINATE	Evaluation could not reach a conclusion, typically due to lost connector permission
NOT_APPLICABLE	The control does not apply to this environment
EXEMPT	An approved exception suppresses the control for a bounded period
NOT_STARTED	No evaluation has been recorded yet

INDETERMINATE is deliberately distinct from FAILED . Losing read access to a provider is an operational problem, and reporting it as a control failure would produce a false compliance signal in both directions.

AI governance mappings

AI-governance references are intentionally coarse — function, clause, and article level rather than sub-control — because finer citation would imply a precision the published texts do not support. EU AI Act citations use the adopted numbering of Regulation (EU) 2024/1689 rather than the 2021 proposal numbering. The AI governance surfaces add determinations that pure control mapping cannot express: operator role (provider or deployer), agent autonomy level, Annex III high-risk area, and the role-shift triggers that move an organization from deployer to provider obligations.

Framework accuracy posture

Crosswalk references for SOC 2, ISO/IEC 27001, NIST CSF 2.0, the HIPAA Security Rule, and SEC rules are curated from published control texts. CIS references use published CIS-to-framework mappings and point at the relevant control family. AI-governance references are a curated first pass. All mappings are engineering artifacts intended for confirmation by the customer's compliance or legal reviewer before they are relied on as an audit deliverable; framework citations are presented so that a reviewer can check them, and gaps are shown rather than filled.

Exceptions

An exception suppresses a specific finding for a bounded period with a recorded rationale, owner, and expiry. Exceptions never delete the finding: the underlying evaluation continues, the control shows as EXEMPT rather than PASSED , and expiry restores the original state automatically.

Exception activity is written to the privileged audit history.

Evidence integrity model

An audit is a dispute about what was true at a point in time. The evidence model is designed so that TraceLock's answer to that dispute does not depend on trusting TraceLock.

Artifact anatomy

Each evidence record carries the fields required to identify it, prove it, and expire it.

Field	Purpose
<code>id</code>	Stable artifact identifier
<code>organizationId</code>	Tenant binding; every read is scoped by it
<code>controlCode</code>	The control the artifact substantiates
<code>type</code>	<code>AUTOMATED</code> for connector output, <code>MANUAL</code> for attested uploads
<code>dataHash</code>	SHA-256 digest over the canonical Privacy Wall metadata
<code>metadata</code>	The canonical posture object that was hashed
<code>sourceProvider</code>	Originating connector or upload path
<code>description</code>	Human-readable summary of what was observed
<code>retentionClockEnd</code>	Expiry boundary, seven years from observation by default
<code>fileUrl</code>	Optional pointer to an attested file artifact
<code>createdAt</code> / <code>updatedAt</code>	Recording timestamps

Digest computation

The digest is computed over the canonical form of the sanitized metadata, not over the raw provider response. This is deliberate: the raw response is never persisted, so hashing it would produce a digest nobody could later reproduce. Hashing the canonical posture object produces a digest that is reproducible by anyone holding the artifact.

```
digest = SHA-256( JSON( canonicalize( allowlist( observation ) ) ) )
```

Because canonicalization sorts keys and drops undeclared ones, the digest is stable across connector runs, library versions, and provider field ordering. Two runs that observe the same posture produce the same digest.

Deduplication and append behaviour

Writes are content-addressed within a control. If an incoming artifact's digest already exists for the same organization and control code, the existing record is reused and the control's verification timestamp is refreshed. The consequences are worth stating explicitly:

- Unchanged posture refreshes freshness without creating a duplicate artifact.
- A change in posture produces a new digest and therefore a new artifact, leaving the prior artifact in place.
- The evidence set is append-oriented: artifacts are added as posture changes, and the history of prior states is not overwritten by later observations.

Retention

Each artifact carries its own retention clock, defaulted to seven years from observation and overridable per artifact where a framework or contract requires a different schedule. Expiry is monitored on a fifteen-minute cadence, and approaching expiry raises an `EVIDENCE_EXPIRED` notification trigger so that a control does not silently lose its substantiation.

Provenance and timing

Three independent facts establish when an artifact was collected: the observation timestamp inside the canonical metadata, the record creation timestamp assigned by the database, and the control verification timestamp updated by the same write. Back-dating an artifact would require all three to be consistent, plus a digest that still recomputes over the altered metadata.

Attested file artifacts

Manual evidence — penetration test reports, signed policies, vendor attestations — is uploaded as a file artifact and referenced from the evidence record. File artifacts are served through an authorization-checked route rather than a public object URL, so access is bound to the tenant scope or to an auditor's signed scope, and every download is attributable.

Independent verification

The verification procedure available to an auditor requires no vendor cooperation beyond read access:

1. Export the evidence set for the control from the Auditor Room.

2. Recompute SHA-256 over the canonical metadata in the export.
3. Compare with the recorded digest.
4. Compare the observation timestamp with the control's verification timestamp and the evaluation history.

A record whose digest recomputes and whose timestamps agree with the control history is a record that has not been edited after collection.

Identity and authorization architecture

TraceLock owns its own identity lifecycle. Local authentication, enterprise single sign-on, and directory provisioning are all served by the platform itself, with no dependency on a separate portal in the sign-in path.

Authentication paths

Path	Mechanism	Typical use
Local sign-in	Email and password with bcrypt hashing, optional or enforced TOTP MFA	Individual and team plans, break-glass access
Self-serve signup	Verified email with single-use, time-bound activation link	New workspaces
Password reset	Single-use, time-bound reset link	Recovery without administrator involvement
Enterprise SSO	SAML 2.0 with per-organization configuration and just-in-time provisioning	Enterprise plans
Directory provisioning	SCIM 2.0 Users and Groups with per-organization bearer tokens	Enterprise plans
Machine access	Hashed per-organization key presented on each request, rate limited	Agent governance submission
Auditor access	HMAC-signed, 15-minute, read-only capability token	External audit engagements

SAML configuration model

Each organization holds its own SAML configuration: issuer, sign-in URL, signing certificate, and domain association. The service exposes three per-organization endpoints — metadata, login initiation, and assertion consumer — so an identity provider is configured directly against TraceLock rather than against an intermediary.

Assertions are validated for signature and audience against the configured certificate. Successful assertion for an unrecognized user provisions a member just-in-time with the organization's default role. Domain-based discovery routes a user entering a corporate address to their organization's identity provider. Certificate rotation is an explicit administrative action, so an expiring certificate is a scheduled operation rather than an outage.

SCIM provisioning model

The SCIM 2.0 surface implements the resource endpoints an enterprise directory expects: service provider configuration, schemas, resource types, Users, and Groups. Tokens are issued per organization, stored as hashes, and scoped to that organization only, so a leaked token cannot address another tenant. Deprovisioning through SCIM deactivates the member and terminates their access; deactivation rather than deletion preserves the audit history that the member's actions belong to.

Three independent authorization axes

TraceLock separates three concepts that are frequently and incorrectly collapsed into one.

Axis 1 — Tenant role. `ADMIN`, `CLIENT_OWNER`, `CLIENT_USER`, and `AUDITOR` govern permissions inside an organization. The platform-administration permission is the single permission that distinguishes `ADMIN` from `CLIENT_OWNER`.

Axis 2 — Staff access. Cross-tenant platform administration requires the `ADMIN` role *and* membership of the staff allowlist held in platform configuration. Because the allowlist is configuration rather than tenant data, no tenant administrator can grant themselves cross-tenant reach, and platform staff hold no path to tenant evidence or user impersonation.

Axis 3 — Entitlement. Feature availability is decided by the organization's plan tier, evaluated independently of role. No role bypasses entitlement. Internal and comped access is granted as a real subscription record, which keeps internal accounts on the same code path a customer takes and preserves the audit story.

The rule that follows from these axes: privileged access is granted by adding a subject to the correct axis, never by exempting them from a check.

Session and audit properties

Sessions carry the effective role at issue time, so a role change takes effect at the next sign-in. Privileged and identity-affecting events — role changes, invitations, SSO configuration, SCIM token issuance, lifecycle transitions, exception decisions — are written to an audit history that records the actor kind, distinguishing a human user, a SAML-provisioned identity, an operator, and the system itself.

Auditor access control plane

External auditors need to read a tenant's controls and evidence without holding a licensed seat, without joining the tenant's identity provider, and without any possibility of reaching another tenant. The Auditor Room is a separate access plane built for exactly that.

Token model

Auditor access is granted as a signed capability token rather than as a user account.

Property	Value
Construction	Base64url payload with an HMAC-SHA-256 signature over the payload
Signing key	Platform secret held in Secret Manager; never exposed to clients
Payload	Organization ID, auditor email, scope, expiry
Scope	<code>READ_ONLY_AUDITOR</code> — the only scope the verifier accepts
Lifetime	15 minutes from issue
Comparison	Constant-time signature comparison
Transport	Delivered as a link and held in a dedicated auditor cookie, separate from the application session

Verification rejects a token whose signature does not match, whose scope is not the auditor scope, or whose expiry has passed. There is no refresh path: expiry is absolute, and continued access requires a newly issued link.

Isolation properties

- **Tenant binding.** The organization ID is inside the signed payload. Substituting another organization's identifier invalidates the signature, so the token cannot be widened.
- **Method restriction.** The auditor plane admits GET requests only. There is no write path — no comment, no status change, no export mutation — reachable with an auditor scope.
- **Surface restriction.** Auditor routes expose controls, evidence, comments, export, and issue endpoints. Administration, billing, identity configuration, connector credentials, and platform routes are not part of the plane.

- **Separate credential.** The auditor cookie is distinct from the application session cookie, so an auditor scope cannot be escalated by an existing browser session and a signed-in user does not inherit auditor scope.

Revocation

Because tokens are short-lived and stateless, revocation is achieved by expiry and by not re-issuing. Where immediate revocation is required, rotating the signing secret invalidates every outstanding auditor token at once. Issuance is recorded, so who was granted access, to which organization, and when, is answerable after the fact.

Export

The export endpoint produces the defensible package an auditor takes away: the control set with framework mappings, evaluation history, evidence records with their digests and observation timestamps, and exceptions with their rationale and expiry. The package is intended to be verifiable offline — digests recompute without contacting TraceLock — so the auditor's conclusion does not rest on vendor cooperation after the engagement ends.

Data schemas

The two schemas below define the contracts that matter to an integrator: the finding produced by a control evaluation, and the evidence artifact that substantiates it. Both are expressed as JSON Schema 2020-12.

ControlEvaluationFinding

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://tracelock.odingard.com/schemas/control-evaluation-finding.json",
  "title": "ControlEvaluationFinding",
  "type": "object",
  "additionalProperties": false,
  "required": [
    "finding_id",
    "organization_id",
    "control_id",
    "evaluation_timestamp",
    "status",
    "raw_posture_metadata_hash",
    "framework_mappings"
  ],
  "properties": {
    "finding_id": { "type": "string", "description": "Stable identifier for this evaluation result." },
    "organization_id": { "type": "string", "description": "Tenant that owns the finding. Every read is scoped by this value." },
    "control_id": { "type": "string", "description": "Internal control code, for example CC6.1." },
    "connector_source_id": {
      "type": ["string", "null"],
      "description": "Integration that produced the observation. Null for manual attestation."
    },
    "evaluation_timestamp": { "type": "string", "format": "date-time" },
    "status": {
      "type": "string",
      "enum": ["PASSED", "FAILED", "INDETERMINATE", "NOT_APPLICABLE", "EXEMPT", "NOT_STARTED"]
    },
    "severity": { "type": ["string", "null"], "enum": ["LOW", "MEDIUM", "HIGH", "CRITICAL", null] },
    "raw_posture_metadata_hash": {
      "type": "string",
      "pattern": "^[a-f0-9]{64}$",
      "description": "SHA-256 over the canonical Privacy Wall metadata for this evaluation."
    },
    "posture_metadata": {
      "type": "object",
      "description": "Allowlisted posture keys admitted by the Privacy Wall. No provider payload bodies.",
      "additionalProperties": true
    },
    "framework_mappings": {
      "type": "array",
      "minItems": 0,
      "items": {
        "type": "object",
        "additionalProperties": false,
        "required": ["framework", "reference"],
        "properties": {
          "framework": {
            "type": "string",
```

```

        "enum": [
            "SOC2",
            "ISO_27001",
            "NIST_CSF_2_0",
            "NIST_AI_RMF",
            "ISO_42001",
            "EU_AI_ACT",
            "HIPAA",
            "SEC",
            "CIS_v8"
        ]
    },
    "reference": {
        "type": "string",
        "description": "Clause, article, criterion or safeguard reference, for example 'Art. 15' or 'A.9.2.1'."
    }
}
},
"exclusion": {
    "type": ["object", "null"],
    "additionalProperties": false,
    "required": ["exclusion_id", "reason", "expires_at"],
    "properties": {
        "exclusion_id": { "type": "string" },
        "reason": { "type": "string" },
        "expires_at": { "type": "string", "format": "date-time" }
    }
}
}
}

```

ImmutableEvidenceArtefact

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://tracelock.odingard.com/schemas/immutable-evidence-artefact.json",
  "title": "ImmutableEvidenceArtefact",
  "type": "object",
  "additionalProperties": false,
  "required": [
    "artefact_id",
    "organization_id",
    "control_id",
    "type",
    "sha256_digest",
    "recorded_custody_metadata",
    "observed_at",
    "recorded_at",
    "retention_clock_end"
  ],
  "properties": {
    "artefact_id": { "type": "string" },
    "organization_id": { "type": "string" },
    "control_id": { "type": "string", "description": "Control the artefact substantiates." },
    "type": { "type": "string", "enum": ["AUTOMATED", "MANUAL"] },
    "sha256_digest": {
      "type": "string",
      "pattern": "^[a-f0-9]{64}$",
    }
  }
}

```

```

    "description": "SHA-256 over the canonical form of recorded_custody_metadata."
  },
  "storage_uri": {
    "type": ["string", "null"],
    "description": "Authorization-checked reference to an attested file artefact. Not a public object URL."
  },
  "source_provider": { "type": ["string", "null"] },
  "description": { "type": "string" },
  "recorded_custody_metadata": {
    "type": "object",
    "description": "Canonical, allowlisted posture object. Key order is normalised so the digest is reproducible.",
    "additionalProperties": true
  },
  "observed_at": { "type": "string", "format": "date-time" },
  "recorded_at": { "type": "string", "format": "date-time" },
  "retention_clock_end": {
    "type": "string",
    "format": "date-time",
    "description": "Default seven years from observation; overridable per artefact."
  }
}
}

```

Verification contract

Given an artifact, verification is a pure function of the artifact itself:

```

canonical = sort_keys_recursive( recorded_custody_metadata )
assert sha256_digest == SHA256( JSON( canonical ) )
assert observed_at <= recorded_at
assert recorded_at <= retention_clock_end

```

An implementation that satisfies these assertions has verified the artifact without contacting TraceLock.

Threat model and mitigations

The threat model addresses the adversaries that matter for a compliance platform: an insider who wants a better audit result, a tenant that wants another tenant's data, and a compromised integration that wants a foothold.

Trust assumptions

- The customer's identity provider is authoritative for enterprise user identity.
- Provider credentials issued to TraceLock are read-only; TraceLock cannot mutate customer environments even if compromised.
- Odingard platform staff can perform tenant lifecycle operations but cannot read tenant evidence or impersonate tenant users.
- Google Cloud is a trusted operator of the underlying compute, database, secret, and scheduling services.

Threats and mitigations

T1 — Back-dated or fabricated evidence

An administrator wants an audit to show that a control passed at a date when it did not.

Evidence digests are computed over canonical metadata, so altering metadata invalidates the digest. Three independent timestamps — observation, record creation, and control verification — must agree, and prior artifacts remain in place when posture changes because writes are content-addressed and append-oriented. Evidence-affecting administrative actions are written to the privileged audit history. An auditor recomputing digests and comparing timestamps against evaluation history detects an inconsistency.

T2 — Rogue or over-collecting connector

A connector, or a compromised provider response, attempts to inject data outside its posture scope.

The Privacy Wall admits allowlisted keys only, so a connector cannot widen its data footprint by returning additional fields. Personal-data patterns on undeclared fields abort the write rather than being sanitized silently, converting a collection defect into a visible failure. Residency enforcement rejects observations from regions outside the configured allowlist.

T3 — Cross-tenant read

A user or auditor attempts to read another organization's controls or evidence.

Every query is expressed against a resolved tenant scope containing the organization ID; there is no ambient data path that omits it. Auditor tokens carry the organization ID inside the HMAC-signed payload, so substituting an identifier invalidates the signature. Machine access resolves the organization from a hashed key and rejects a request whose declared organization does not match the key's organization.

T4 — Auditor scope escalation

An auditor link is used to write, to reach administrative surfaces, or after the engagement ends.

The auditor plane admits GET requests only and exposes a fixed set of read routes. Tokens expire fifteen minutes after issue with no refresh path, are held in a dedicated cookie separate from the application session, and are invalidated en masse by rotating the signing secret.

T5 — Credential theft

An attacker with database access attempts to use stored connector credentials.

Credentials are stored only as AES-GCM envelope ciphertext, with key material in Secret Manager rather than in the database, so database access alone does not yield usable credentials. Plaintext exists in process memory for the duration of a scan and is never logged or returned by any API. Credentials are read-only at the provider, bounding the impact of compromise to disclosure of posture data the customer already shared.

T6 — Privilege confusion between tenant role, staff access, and entitlement

A tenant administrator gains cross-tenant powers, or a role grants paid features.

The three axes are enforced independently. Tenant role governs in-tenant permissions. Cross-tenant platform administration requires the `ADMIN` role and membership of the staff allowlist, which is configuration rather than tenant data. Feature access is decided by plan entitlement, which no role bypasses — so an internal comped subscription, not a code exception, is how internal accounts obtain full features.

T7 — Authentication bypass

An attacker forges a SAML assertion, replays a reset link, or brute-forces sign-in.

SAML assertions are validated against the organization's configured certificate with signature and audience checks; certificate rotation is an explicit administrative operation. MFA is enforceable per organization and required on privileged accounts. Password reset and signup links are single-use and time-bound. Sensitive endpoints, including the machine endpoint, are rate limited per key or address.

T8 — Notification and ticketing exfiltration

Configured outbound destinations are used to extract environment detail.

Outbound text passes through the redaction pass and length bound, so notifications and tickets carry control codes, statuses, and remediation references rather than provider payloads. Destination configuration is an administrative action recorded in audit history.

Residual risks

Risk	Current position
Single-region database	Primary data plane runs in one region; recovery relies on managed backups
Compliance certification	Odingard operates TraceLock against these controls; TraceLock is not itself represented as holding a SOC 2 or ISO certification
Crosswalk correctness	Mappings are curated engineering artifacts requiring compliance review before audit reliance
Customer-side revocation	A customer that revokes provider access without notice sees controls become <code>INDETERMINATE</code> until access is restored

Reference implementations

The listings below are working reference implementations of the three mechanisms an integrator or auditor most often needs to reproduce: the evidence digest, the Privacy Wall filter, and the crosswalk evaluator. They are written in TypeScript against the Node.js standard library and depend on nothing outside it.

Canonicalization and evidence digest

```
import { createHash } from "node:crypto";

type Json = string | number | boolean | null | Json[] | { [key: string]: Json };

/** Deterministic canonical form: object keys sorted, arrays order-preserving. */
export function canonicalize(value: Json): Json {
  if (Array.isArray(value)) return value.map(canonicalize);
  if (value && typeof value === "object") {
    return Object.keys(value)
      .sort()
      .reduce<{ [key: string]: Json }>((acc, key) => {
        acc[key] = canonicalize((value as { [key: string]: Json })[key]);
        return acc;
      }, {});
  }
  return value;
}

export function sha256(value: Json): string {
  return createHash("sha256").update(JSON.stringify(canonicalize(value))).digest("hex");
}

export type EvidenceArtefact = {
  readonly organizationId: string;
  readonly controlId: string;
  readonly recordedCustodyMetadata: Json;
  readonly sha256Digest: string;
  readonly observedAt: string;
  readonly recordedAt: string;
  readonly retentionClockEnd: string;
};

export function buildArtefact(input: {
  organizationId: string;
```

```

    controlId: string;
    metadata: Json;
    observedAt?: Date;
    retentionYears?: number;
}): EvidenceArtefact {
    const observedAt = input.observedAt ?? new Date();
    const retention = new Date(observedAt);
    retention.setFullYear(retention.getFullYear() + (input.retentionYears ?? 7));
    const metadata = canonicalize(input.metadata);
    return {
        organizationId: input.organizationId,
        controlId: input.controlId,
        recordedCustodyMetadata: metadata,
        sha256Digest: sha256(metadata),
        observedAt: observedAt.toISOString(),
        recordedAt: new Date().toISOString(),
        retentionClockEnd: retention.toISOString(),
    };
}

/** Independent verification. Requires only the artefact. */
export function verifyArtefact(artefact: EvidenceArtefact): boolean {
    return (
        sha256(artefact.recordedCustodyMetadata) === artefact.sha256Digest &&
        Date.parse(artefact.observedAt) <= Date.parse(artefact.recordedAt) &&
        Date.parse(artefact.recordedAt) <= Date.parse(artefact.retentionClockEnd)
    );
}

```

Privacy Wall filter

```

export class PrivacyWallError extends Error {}

/** Posture keys admitted into storage. Anything else is dropped. */
const ALLOWED_KEYS = new Set([
    "provider",
    "mode",
    "controlCode",
    "findingType",
    "severity",
    "status",
    "freshness",
    "region",
    "resourceType",
    "resourceCount",

```

```

    "configurationDrift",
    "observedAt",
    "receivedAt",
    "byteLength",
    "frameworks",
    "score",
  ]);

  /** Fields explicitly permitted to carry an address, for ownership and assignment records. */
  const ADDRESS_FIELDS = new Set(["ownerEmail", "userEmail"]);

  const PATTERNS: ReadonlyArray<readonly [string, RegExp]> = [
    ["email", /\w.+@\w-+\.\w-+\/],
    ["national-id", /\b\d{3}-\d{2}-\d{4}\b/],
    ["payment-card", /\b(?:\d[ -]){13,19}\b/],
    ["phone", /\b\+?\d[\d ()-]{8,}\d\b/],
  ];

  function inspect(value: unknown, path: string, key?: string): void {
    if (typeof value === "string") {
      if (key && ADDRESS_FIELDS.has(key)) return;
      for (const [label, pattern] of PATTERNS) {
        if (pattern.test(value)) {
          throw new PrivacyWallError(`Privacy Wall rejected ${label} content at ${path}`);
        }
      }
      return;
    }
    if (Array.isArray(value)) {
      value.forEach((item, index) => inspect(item, `${path}[${index}]`));
      return;
    }
    if (value && typeof value === "object") {
      for (const [childKey, childValue] of Object.entries(value)) {
        inspect(childValue, `${path}.${childKey}`, childKey);
      }
    }
  }

  function reduce(input: Record<string, unknown>): Record<string, unknown> {
    return Object.keys(input)
      .filter((key) => ALLOWED_KEYS.has(key) || ADDRESS_FIELDS.has(key))
      .sort()
      .reduce<Record<string, unknown>>((acc, key) => {
        acc[key] = input[key];
        return acc;
      }, {});
  }

```

```

/** Inspect, reduce to the allowlist, canonicalize, then hash. */
export function protectEvidence(input: {
  provider: string;
  controlCode: string;
  metadata: Record<string, unknown>;
  region?: string;
  allowedRegions?: ReadonlySet<string>;
}) {
  const candidate = {
    ...input.metadata,
    provider: input.provider,
    controlCode: input.controlCode,
    region: input.region ?? "US",
  };
  inspect(candidate, "$");
  const metadata = reduce(candidate);
  const allowed = input.allowedRegions ?? new Set(["US"]);
  if (!allowed.has(String(metadata.region))) {
    throw new PrivacyWallError(`Residency rejected region ${String(metadata.region)}`);
  }
  return { metadata, dataHash: sha256(metadata as Json), observedAt: new Date() };
}

```

Crosswalk evaluator

```

export type Framework =
  | "SOC2"
  | "ISO_27001"
  | "NIST_CSF_2_0"
  | "NIST_AI_RMF"
  | "ISO_42001"
  | "EU_AI_ACT"
  | "HIPAA"
  | "SEC"
  | "CIS_V8";

export type FrameworkMapping = Partial<Record<Framework, string>>;

export type ControlEvaluation = {
  controlId: string;
  status: "PASSED" | "FAILED" | "INDETERMINATE" | "NOT_APPLICABLE" | "EXEMPT" | "NOT_STARTED";
  mapping: FrameworkMapping;
};

export type FrameworkCoverage = {
  framework: Framework;
  satisfied: string[];
}

```

```

    failing: string[];
    indeterminate: string[];
    unmapped: string[];
  };

  /** Project one evaluation set onto every mapped framework. Evaluate once, report many. */
  export function projectCoverage(
    evaluations: readonly ControlEvaluation[],
    frameworks: readonly Framework[],
  ): FrameworkCoverage[] {
    return frameworks.map((framework) => {
      const coverage: FrameworkCoverage = {
        framework,
        satisfied: [],
        failing: [],
        indeterminate: [],
        unmapped: [],
      };
      for (const evaluation of evaluations) {
        const reference = evaluation.mapping[framework];
        if (!reference) {
          coverage.unmapped.push(evaluation.controlId);
          continue;
        }
        if (evaluation.status === "PASSED" || evaluation.status === "NOT_APPLICABLE") {
          coverage.satisfied.push(reference);
        } else if (evaluation.status === "FAILED") {
          coverage.failing.push(reference);
        } else {
          coverage.indeterminate.push(reference);
        }
      }
      return coverage;
    });
  }
}

```

A control in **EXEMPT** state is reported as indeterminate rather than satisfied. An approved exception documents a decision to accept a gap; it does not assert that the requirement is met, and reporting it as satisfied would misstate the control environment to an auditor.

Deployment, CI/CD and verification

TraceLock is delivered as a single hosted service. There is no customer-side installation step, so the deployment pipeline described here is the vendor's release path, and its guarantees are what a customer inherits.

Runtime

Element	Configuration
Compute	Google App Engine Standard, Node.js 22 runtime, F2 instance class
Scaling	Automatic, scaling to zero when idle, bounded maximum instances
Region	us-west1
Database	Cloud SQL for PostgreSQL, TLS-only, no authorized networks, reached over the Cloud SQL Auth proxy on a Unix socket
Secrets	Secret Manager, injected at deploy time; no secret material in source or images
Scheduling	App Engine cron invoking authenticated scan, notification, evidence-expiry and training endpoints
Transport	TLS 1.2 or later, HSTS, managed certificates for the service domain
Object artifacts	Attested evidence files served through an authorization-checked route rather than public object URLs

Release pipeline

Every change to the default branch passes the same gate before it can be promoted.

1. **Static verification.** Type checking and linting must pass with no suppressions introduced.
2. **Automated tests.** The unit and integration suite must pass in full. Skipped tests are treated as failures of the change, not of the suite.
3. **Security pipeline.** Secret scanning across the full history, static analysis, and a dependency audit gated at high severity. A finding at or above the gate blocks the merge.

4. **Build.** A production build must succeed with the same configuration used in the deployed environment.
5. **Migration.** Schema migrations are applied forward; destructive migrations are not run as part of automatic promotion.
6. **Deploy and promote.** The new version is deployed and promoted, and the previous version remains available as an immediate rollback target.
7. **Post-deploy verification.** Liveness and readiness endpoints must return healthy before the release is considered complete.

Integrity verification tests

Three classes of test protect the properties this specification claims.

Digest reproducibility. Canonicalization is asserted to produce byte-identical output for semantically identical observations with differing key order, and digests are asserted to be stable across runs. A change that alters canonicalization changes every digest, so this test is the tripwire that prevents silent breakage of historical verifiability.

Privacy Wall enforcement. Metadata carrying personal-data patterns on undeclared fields is asserted to be rejected; undeclared keys are asserted to be dropped rather than stored; out-of-region observations are asserted to be refused. These are negative tests — they fail when the Privacy Wall becomes more permissive.

Isolation and scope. Tests assert that tenant-scoped reads cannot return another organization's rows, that auditor tokens fail verification when the organization identifier or scope is altered, that expired tokens are refused, that non-GET methods are refused on the auditor plane, and that platform routes require both the role and the staff allowlist.

Infrastructure as code

Runtime configuration lives in version-controlled deployment descriptors rather than in console state: the App Engine service definition, the scheduled job definitions, and the CI workflow definitions. A representative declaration of the equivalent topology, for readers reproducing the pattern:

```
resource "google_sql_database_instance" "tracelock" {
  name          = "tracelock-db"
  database_version = "POSTGRES_15"
  region        = "us-west1"

  settings {
    tier = "db-custom-2-7680"
  }
}
```

```

ip_configuration {
  ipv4_enabled    = true # required by the App Engine Standard connection path
  ssl_mode        = "ENCRYPTED_ONLY"
  authorized_networks = [] # no direct internet reachability
}

backup_configuration {
  enabled                = true
  point_in_time_recovery_enabled = true
}

deletion_protection = true

resource "google_secret_manager_secret" "evidence_key" {
  secret_id = "tracelock-evidence-key"
  replication { auto {} }
}

```

Observability

Application errors and performance are reported to an error-tracking service with scrubbed payloads. Liveness and readiness endpoints are exposed for platform probes, and a privileged diagnostics endpoint reports database connectivity, migration state, and scheduled-job health for staff use. Delivery attempts for notifications and webhooks are recorded so a missed alert is diagnosable after the fact.

Glossary

Product terms

Term	Definition
Audit	A workspace record for an audit engagement. The application displays framework, progress, dates, and readiness context.
Control	A tenant-scoped compliance requirement represented by a code, title, description, category, framework references, status, and verification information.
Control Center	The compliance overview for readiness, control health, failing controls, and posture by category.
Control execution log	A record of a connector check attempt, including its connector type, status, raw mode/reason data, and evidence hash.
Control test	A connector or other verification operation that evaluates a control and can produce a finding and evidence.
Crosswalk	The mapping between a control and one or more framework references. TraceLock uses the mapping to calculate framework coverage and display mapped controls.
Evidence record	A persisted record associated with a control. It includes a type, description, file reference when present, content hash, retention end, metadata, and source provider.
Estate Map	The live view of control coverage across the framework perspectives exposed by the application, with status filtering and framework drill-down.
Exception	A time-bounded request against a specific control. A request includes a reason and expiry and can be approved, rejected, or marked expired by an authorized reviewer.
Finding	A connector result describing a passed, failed, not-applicable, indeterminate, warning, or not-assessed outcome for a control check.

Term	Definition
Framework	A compliance or regulatory perspective used by controls and crosswalks, such as SOC 2, ISO 27001, NIST CSF 2.0, or EU AI Act.
Integration	A configured external provider connection used by a workspace for posture or evidence collection.
Privacy Wall	The metadata protection boundary that permits declared metadata fields, rejects sensitive text patterns, canonicalizes accepted metadata, hashes it, and enforces allowed regions.
Remediation	Guidance associated with a failed check. TraceLock can also dispatch a configured remediation ticket for a failure.
Tenant	A TraceLock organization boundary. Tenant-scoped queries include the organization identifier and do not use another organization's records.
Verification Room	The read-only auditor surface for reviewing mapped controls, evidence metadata, content hashes, and audit timeline information.
WORM	Write once, read many. TraceLock's universal webhook evidence view describes a seven-year SEC 204-2 WORM retention clock for signed events.

Identity and administration terms

Term	Definition
ACS URL	The SAML assertion consumer service URL that receives an identity provider response. TraceLock derives it from the public origin and organization identifier.
Claimed domain	An email domain associated with an enabled SAML connection. When enforcement is enabled, ordinary users in claimed domains must use SAML.
JIT provisioning	Just-in-time user creation during a successful SAML flow. TraceLock creates a local user when the SAML login does not find one.

Term	Definition
Lifecycle state	An organization's operational state. The implementation recognizes <code>ACTIVE</code> , <code>SUSPENDED</code> , <code>OFFBOARDING</code> , and <code>OFFBOARDED</code> .
Platform administration	The cross-tenant organization and action surface reserved for approved Odingard staff. It requires the platform permission and the staff allowlist.
SCIM	System for Cross-domain Identity Management 2.0, used here for bearer-token-based user provisioning and deprovisioning.
SSO	Single sign-on. In TraceLock's identity configuration, SAML SSO uses an enterprise identity provider.
Tenant scope	The server-side object containing the authenticated user, organization, role, lifecycle state, and permissions used for authorization and tenant scoping.

AI governance terms

Term	Definition
Accepted role	The recorded organization role for an AI system after reviewing the proposed operator role. A written justification is required when it differs from the proposal.
Agentic system	An AI inventory record with an agentic posture and autonomy level. TraceLock records tool scope, human oversight, action logging, and shutdown procedure fields.
Article 25 role shift	A TraceLock AI-governance review event produced when recorded facts can change the proposed operator role under the EU AI Act logic implemented by the application.
AI inventory	The registry of AI systems, their purpose, provider, risk, approval, operator role, and governance posture.
AI role assessment	The saved determination input and result used to record an AI system's operator role, risk category, Annex III fit, and related rationale.

Term	Definition
Governance ledger	The AI Governance view of locally hashed agent activity. Prompt and response hashes are displayed instead of prompt or response content.

Connector terms

Term	Definition
Connector	A provider-specific implementation that declares a category, control codes, credential environment variables, and a check operation.
Connector mode	The execution mode returned by a connector: <code>live</code> , <code>demo</code> , or <code>not_executed</code> .
External ID	The tenant-specific AWS value used in the customer role trust policy to reduce confused-deputy risk.
Least privilege	Granting only the provider actions required by the connector checks. The AWS template lists the exact read actions used by the AWS connector.
Smoke test	An explicit connection test from the Integrations page that exercises the provider configuration and reports checks.