STRIX

Execution Control for AI Systems · strixgov.com

Signed Evidence v1

Ed25519-signed governance evidence with public-JWKS verification.

Version	1.0.0 (approved)
Owner	Strix Evidence
Status	Public technical whitepaper
Related	Execution Binding Guarantee v1, SE-1, SE-5, SE-7, SE-12, SE-14, SE-18, EU AI Act Articles 12/14/28
Generated	2026-05-10
Source	github.com/Tarshann/strix-platform

Verifiable claim. Every cryptographic claim in this whitepaper is independently verifiable using standard primitives (Ed25519, RFC 7517 JWKS, SHA-256). The reference implementation lives at *github.com/Tarshann/strix-platform*; the public verifier package is *@strixgov/verifier* on npm. The public JWKS endpoint is *https://www.strixgov.com/.well-known/strix-jwks.json*.

Part I — Gate Report (approval record)

A. Feature Summary

Feature: Signed Evidence v1 (with EU AI Act compliance extension)

What changed:

- Evidence is now cryptographically signed with a deterministic payload
- A `regulatoryContext` block is embedded inside the signed payload
- Verification surface now produces compliance-derived flags (Articles 12, 14, 28)
- New verification states introduced: `VERIFIED`, `UNSIGNED`, `LEGACY_UNSIGNED`, `COMPLIANCE_VIOLATION`
- Public key discovery + independent verification becomes first-class

Why it matters: This upgrades Strix from "execution control with auditability" to "execution control with externally verifiable, cryptographically bound, regulator-aligned proof." That is a trust-claim expansion, triggering GOG-1 and GOG-2.

B. Affected Surfaces

Surface	Status	Notes
Capability Registry	Not touched	Still upstream of evidence
Policy Engine	Not touched	Compliance is proof-layer, not decision-layer
Governed Procedure	Touched	Now must produce signed evidence
Token Layer	Not touched	No semantic change
Evidence Layer	Touched (core)	Signature, schema, regulatoryContext
Proof Layer	Touched (core)	Verification + compliance flags
Reliability Layer	Touched	Signing failure modes introduced
Multi-tenant Layer	Touched	Key separation + verification semantics

---

C. Required Gates

Gate	Required	Why
Gate D	■	Changes trust claim + payload schema
Gate E	■	Signature integrity + binding
Gate F	■	End-to-end propagation
Gate G	■	Signing failures + rotation
Gate H	■	External verification + compliance flags

Gate	Required	Why
Gate J	■ (strong)	Cryptographic primitive introduced

D. Gate Report

Gate D — Design / Architectural Coherence

Status: PASS

Checked:

- Signed payload schema versioned
- Regulatory context cryptographically bound
- No change to execution ordering
- Trust claim explicitly expanded

Findings:

- Clean separation preserved (intent → evaluation → execution → evidence → proof)
- Compliance layer correctly sits in proof/evidence, not policy

Risks:

- Over-marketing compliance without verification discipline (mitigated via CI-5)

Remediation: None

Gate E — Enforcement / Integrity

Status: PASS

Checked:

- Signature generated from canonical payload
- Payload deterministic
- All required fields bound (evidenceId, hash, chain, capabilityId, etc.)
- Tamper scenarios addressed

Findings:

- Strong binding between record and signature
- No path where signed record can be mutated without detection

Risks:

- If canonicalization drifts → signature mismatch across environments

Remediation:

- Lock canonical serializer version + test vector

Gate F — Integration / Embodiment

Status: PARTIAL

Checked:

- Schema defined
- Verification response defined
- Compliance flags defined

Missing:

- DB migration implementation
- SDK propagation
- Outbox/sync preservation guarantees
- Trust center UI updates
- JWKS endpoint implementation

Risks:

- Partial rollout → inconsistent proof surface

Remediation — implement full pipeline before marking complete:

· DB migration

· Payload builder

· Signing hook in evidence creation

· Verification endpoint

· UI + Trust Center updates

· JWKS endpoint

Gate G — Reliability / Failure Modes

Status: PASS (with explicit behavior)

Checked:

- Missing key
- Signing failure
- Mixed-version deployments
- Rotation

Findings:

- Clear separation: Standard mode → degraded (**`UNSIGNED`**), Compliance mode → **`COMPLIANCE_VIOLATION`**

- Optional strict mode (block on failure)

Risks:

- Operators misunderstanding default non-blocking behavior

Remediation:

- Document operational modes clearly

Gate H — Observability / Proof

Status: PASS

Checked:

- Verification response includes: hashValid, chainValid, signaturePresent, signatureValid, compliance flags

Findings:

- CI-5 satisfied: compliance flags derived, not asserted
- External verification reproducible

Risks:

- UI could accidentally collapse statuses

Remediation:

- Force UI to display all verification layers separately

Gate J — Specialized Crypto Gate

Status: PARTIAL

Checked:

- Ed25519 assumed
- Key ID included
- Payload versioned

Missing:

- Golden test vectors
- Rotation test suite
- Algorithm mismatch tests

Risks:

- Silent crypto regressions

Remediation — add regression pack:

- Golden payload + signature

- Wrong key test
- Wrong algorithm test
- Mutation matrix

E. Invariant Check

Category	Status	Notes
K-1 through K-6 (Kernel)	PASS	No change to execution ordering or boundary
EVI-1 through EVI-7 (Evidence)	PASS	Improved via signature + regulatory binding
P-1 through P-4 (Proof)	PASS	Proof now stronger + externally reproducible
R-1 through R-5 (Reliability)	PASS	Failure modes explicit and visible
GOG-1 through GOG-3	PASS	GOG-1 triggered correctly; GOG-2 satisfied; GOG-3 enforced via derived flags
SE-1 through SE-13, SE-15 through SE-18	PASS	—
SE-14 (rotation)	PARTIAL	Needs implementation tests
CI-1 Traceability	PASS	—
CI-2 Tamper Resistance	PASS	—
CI-3 Provenance	PASS	—
CI-4 Independent Verifiability	PASS	—
CI-5 Truthful Representation	PASS	Flags derived, not asserted
CI-6 Oversight Support	PASS	—

F. Test Coverage

Conceptually defined (not yet implemented):

- Valid signature verification
- Tamper detection (hash, chain, fields)
- Wrong key
- Missing key
- Legacy behavior
- Compliance flag derivation

Missing — critical, must be built before any public claims:

- Deterministic payload serialization test

- Golden signature vector
- Mixed-version environment test
- Key rotation historical verification
- Sync/outbox byte-preservation test

G. Trust Claim Impact

Before:

> "Strix provides execution governance with verifiable audit trails"

After:

> "Strix provides execution governance with independently verifiable, cryptographically signed, regulator-aligned proof of control and oversight"

This claim holds only because:

- Signatures are verifiable externally
- Compliance flags are derived from verification
- Regulatory context is cryptographically bound

If any of those break, the claim collapses.

H. Final Decision

APPROVED WITH REQUIRED FOLLOW-UPS

This is architecturally sound. This is category-defining. This is not fully implemented yet.

I. Next Actions (In Order)

· **Core Implementation**

- DB migration (signature, algorithm, keyId, regulatoryContext columns)
- Canonical payload builder (strict deterministic)
- Signing module (Ed25519)

· **Enforcement Wiring**

- Integrate signing into evidence creation path
- Enforce SE-6 behavior by mode

· **Verification Layer**

- Build verification engine (hash + chain + signature)

- Implement compliance flag derivation

· **Public Trust Surface**

- `/verify/:evidenceId` endpoint
- `/.well-known/jwks.json` endpoint
- Trust Center UI updates

· **Reliability + Ops**

- Key rotation system
- Env config handling
- Compliance mode toggles

· **Test Pack (MANDATORY BEFORE CLAIMS)**

- Golden payload + signature
- Tamper matrix
- Wrong key / wrong algorithm
- Mixed-version test
- Rotation test

· **Documentation**

- Public verification guide
- Compliance mapping doc (Articles 12, 14, 28)
- Architecture note (Signed Evidence v1)

Closing Constraint

> This feature is correct. It is aligned. It is powerful.
>
> But until the full proof surface, verification, and tests exist,
>
> you do not yet have the right to claim it publicly.
>
> That is the line Strix cannot cross.

Part II — Implementation Specification

Paste this document in full at the start of your agent session.

It is an operational control document, not a brainstorm.

The agent must produce a Gate Report (Part VII) at task completion.

Part I — System Context

You are working inside the Strix Governance Kernel.

Strix is an execution control kernel that sits between intent and side effect. It determines whether a requested action may execute, under what conditions, with what proof, and with what recoverability constraints.

The core architectural truth you must never violate:

Nothing executes until it is evaluated.

Nothing is trusted unless it is provable.

Nothing that affects trust may bypass governance.

A demo claim may not exceed what the gate-approved system can actually verify.

The feature you are implementing is **Signed Evidence v1**.

Strix's external trust claim after this feature:

"Every governance decision is cryptographically signed by the Strix kernel,

independently verifiable by any third party with the public key,

and mapped to EU AI Act Articles 12, 14, and 28 in compliance mode."

Part II — Feature Definition

What You Are Building

Add Ed25519 cryptographic signatures to Strix governance evidence records, with optional EU AI Act compliance mode that elevates signing to a mandatory regulatory substrate.

Today: Evidence records are SHA-256 hashed and chain-linked. This proves record integrity relative to what is stored. It does not prove authorship, and it cannot be verified without internal DB access.

After this feature: Evidence records also carry a digital signature produced by the Strix signing key, with a compliance context block that is cryptographically bound to the record.

This upgrades Strix's trust claim in three steps:

· "This record has not been modified since it was stored." (hash — today)

· "This record was produced by the holder of the Strix signing key." (signature — this feature)

· "This record satisfies EU AI Act Articles 12, 14, and 28." (compliance mode — this feature)

What This Is Not

- Not a replacement for hashing or chain verification
- Not a new execution path
- Not a token system change
- Not a redesign — additive to the evidence layer only
- Not a compliance assertion layer — compliance flags are DERIVED from verification outcomes

---

Part III — Closed Design Decisions

These decisions are closed. Do not reopen them without a formal Gate D revision.

Decision 1 — Key management: Option A. Single Ed25519 keypair per environment. Private key as env var **`STRIX_SIGNING_PRIVATE_KEY`** (base64-encoded). Public key at **`/.well-known/strix-jwks.json`** (JWKS array). Key ID format: **`strix-{env}-{YYYY-MM}`**. Historical keys retained in JWKS indefinitely. Never remove a key ID that signed active records.

Decision 2a — Failure mode (standard): Transitional. Signing failure produces a degraded proof state. Evidence record is always created. **`signatureAlgorithm = "failed"`**. **`verificationStatus = "UNSIGNED"`**. Mutation proceeds.

Decision 2b — Failure mode (compliance mode): Compliance violation. When **`STRIX_COMPLIANCE_MODE=eu_ai_act`**, signing failure produces **`verificationStatus = "COMPLIANCE_VIOLATION"`** and an explicit compliance event record. Mutation still proceeds by default. Operators may set **`STRIX_COMPLIANCE_BLOCK_ON_SIGN_FAILURE=true`** to hard-block in this case.

Part IV — Required Implementation

Complete every item. Do not mark work done until gate criteria in Part VI pass.

A. Canonical Signed Payload Schema

The exact object that will be serialized and signed. This schema is locked.

```
interface SignedEvidencePayload {
  schemaVersion: "signed_evidence_v1";
  evidenceId: number;
  evidenceHash: string;           // SHA-256 hex of decision fields
  proofChainHash: string;         // SHA-256 chain link from DB
  capabilityId: string;
  action: "allow" | "deny";
  actorId: string;
  actorRole: string;
  createdAt: string;              // ISO 8601, from DB record — not from signing time
  signingKeyId: string;           // e.g. "strix-prod-2026-04"
  regulatoryContext: {
    framework: "EU_AI_ACT" | null;
```

```
    articles: string[];           // ["12", "14", "28"] or []
    systemMode: "standard" | "compliance_enforced";
  };
}
```

Serialization: **`canonicalSerialize()`** from **`shared/normalizePayload.ts`** (deterministic, key-sorted). The **`regulatoryContext`** block is INSIDE the signed payload — it is cryptographically bound. A verifier who omits it will get a signature mismatch.

B. Key Management

- Generate Ed25519 keypair using Node.js **`crypto.generateKeyPairSync("ed25519")`**
- Private key: **`STRIX_SIGNING_PRIVATE_KEY`** env var (base64 DER or PEM)
- Public key: served at **`GET /.well-known/strix-jwks.json`**
- JWKS format (minimal):

```
{
  "keys": [
    {
      "kty": "OKP",
      "crv": "Ed25519",
      "use": "sig",
      "kid": "strix-prod-2026-04",
      "x": "<base64url public key>"
    }
  ]
}
```

- JWKS is an array — multiple keys supported for rotation
- Historical keys are NEVER removed while signed records reference them

C. Database Migration

Add to **`governance_evidence`** table:

```
ALTER TABLE governance_evidence
  ADD COLUMN signature        varchar(200),
  ADD COLUMN signature_algorithm varchar(20),
  ADD COLUMN signing_key_id   varchar(60),
  ADD COLUMN signed_at        varchar(30),
  ADD COLUMN signed_payload   text,
  ADD COLUMN compliance_mode  varchar(30);
```

All columns nullable. Existing records remain valid. Migration is reversible.

Also add to **`drizzle/schema.ts`**:

```
signature:          varchar("signature", { length: 200 }),
signatureAlgorithm: varchar("signature_algorithm", { length: 20 }),
signingKeyId:       varchar("signing_key_id", { length: 60 }),
signedAt:           varchar("signed_at", { length: 30 }),
signedPayload:      text("signed_payload"),
complianceMode:     varchar("compliance_mode", { length: 30 }),
```

D. Signing Logic

Location: **`server/_core/governed-procedure.ts`**, inside **`recordEvidence()`**.

Call order (do not alter):

```
computeEvidenceHash()
→ insertGovernanceEvidence()   ← row.id and row.proofChainHash are now known
→ buildSignedPayload(row)      ← construct canonical payload with evidenceId bound
→ canonicalSerialize(payload)
→ crypto.sign("ed25519", buffer, privateKey)
→ db.update(governance_evidence).set({ signature, signedPayload, ... }).where(id = row.id)
→ return { evidenceId, evidenceHash, signature }
```

If **`STRIX_SIGNING_PRIVATE_KEY`** is missing:

- Standard mode: log warning, mark **`signatureAlgorithm = "failed"`**, proceed
- Compliance mode: log warning, mark **`signatureAlgorithm = "failed"`**,

set **`complianceMode = "COMPLIANCE_VIOLATION"`**, create compliance event, proceed (or block if **`STRIX_COMPLIANCE_BLOCK_ON_SIGN_FAILURE=true`**)

E. Verification Extension

Extend **`POST /api/proof/verify`** (**`server/proof-rest-api.ts`**):

- Fetch signature fields from DB
- If **`signature`** is present: fetch public key by **`signingKeyId`** from JWKS
- Reconstruct canonical signed payload from DB record fields
- Call **`crypto.verify("ed25519", canonicalSerialize(payload), publicKey, signature)`**
- Return expanded response (schema in Part V)

Same changes in **`proof.verifyRecord`** (**`server/proof-router.ts`**).

F. Compliance Flags in Verification Response

Compliance flags are derived from verification outcomes. Never set by assertion.

Flag	Is `true` when
`article12_traceable`	`hashValid: true` AND record has capabilityId, actorId, action, createdAt
`article12_tamper_resistant`	`hashValid: true` AND `chainValid: true` AND `signatureValid: true`
`article14_oversight_supported`	`action` field present AND capabilityId resolvable
`article28_audit_ready`	all above true AND `signatureValid: true`

If **`regulatoryContext.framework`** is **`null`** (standard mode), return **`compliance: null`** in response.

G. ProofReceipt Type Update

In **`shared/governance-types.ts`**, add to **`ProofReceipt`**:

```
signed: boolean;
signingKeyId: string | null;
complianceMode: string | null;
```

H. Proof Trust Center UI

In **`ProofTrustCenter.tsx`**, show three separate verification badges per record:

- **Hash verified ✓ / ✗**

- **Chain verified ✓ / ✗**
- **Signature verified ✓ / ✗** / Not applicable (legacy)

When **`compliance`** block is present in verification response:

- Show "EU AI Act Ready" badge only when **`article28_audit_ready: true`**
- Show individual article mappings: Articles 12 / 14 / 28
- NEVER show compliance badges on **`LEGACY_UNSIGNED`** records

Legacy unsigned records: label clearly as **`LEGACY_UNSIGNED`** with tooltip: "This record predates cryptographic signing. Hash and chain integrity still apply."

I. JWKS and Well-Known Route

Register in **`server/_core/index.ts`**:

```
app.get("/.well-known/strix-jwks.json", (req, res) => {
  res.json({ keys: getPublicKeys() }); // returns array from env/config
});
```

J. Compliance Event Record (compliance mode only)

When **`STRIX_COMPLIANCE_MODE=eu_ai_act`** and signing fails, insert a record into a new **`compliance_events`** table (or reuse **`metadata`** JSONB on governance_evidence with a **`complianceViolation: true`** flag). This record must be visible in the governance admin panel.

K. Test Suite

Add to existing governance test suite:

Happy path:

- [ ] Valid signature passes verification
- [ ] Signed payload is deterministic for identical inputs (golden vector)

Tamper matrix:

- [ ] Tampered **`evidenceHash`** → signature fails
- [ ] Tampered **`proofChainHash`** → signature fails
- [ ] Tampered **`createdAt`** → signature fails
- [ ] Tampered **`capabilityId`** → signature fails
- [ ] Tampered **`action`** → signature fails
- [ ] Tampered **`actorId`** → signature fails
- [ ] Tampered **`regulatoryContext`** → signature fails

Key failure matrix:

- [ ] Wrong public key → signature fails
- [ ] Unresolvable **`kid`** → **`verificationStatus: "UNVERIFIABLE"`**, not **`"VERIFIED"`**
- [ ] Unsupported algorithm → fails

- [ ] Missing **`STRIX_SIGNING_PRIVATE_KEY`** (standard mode) → **`"UNSIGNED"`**, mutation proceeds
- [ ] Missing **`STRIX_SIGNING_PRIVATE_KEY`** (compliance mode) → **`"COMPLIANCE_VIOLATION"`**, mutation proceeds

Legacy:

- [ ] Pre-migration record → **`verificationStatus: "LEGACY_UNSIGNED"`**, no compliance flags

Compliance flags:

- [ ] All verification passes → all four compliance flags **`true`**
- [ ] Hash fails → **`article12_tamper_resistant: false`**, **`article28_audit_ready: false`**
- [ ] Signature missing → **`article12_tamper_resistant: false`**, **`article28_audit_ready: false`**

Rotation:

- [ ] Record signed with old key ID still verifies after new key added to JWKS

Part V — Verification Response Schemas

Fully verified, compliance mode:

```
{
  "evidenceId": 12345,
  "recordType": "signed_evidence_v1",
  "hashValid": true,
  "chainValid": true,
  "signaturePresent": true,
  "signatureValid": true,
  "signatureAlgorithm": "Ed25519",
  "signingKeyId": "strix-prod-2026-04",
  "verificationStatus": "VERIFIED",
  "verificationReason": "All verification checks passed",
  "compliance": {
    "framework": "EU_AI_ACT",
    "systemMode": "compliance_enforced",
    "article12_traceable": true,
    "article12_tamper_resistant": true,
    "article14_oversight_supported": true,
    "article28_audit_ready": true
  }
}
```

Legacy unsigned record:

```
{
  "evidenceId": 12001,
  "recordType": "legacy_unsigned_evidence",
  "hashValid": true,
  "chainValid": true,
  "signaturePresent": false,
  "signatureValid": false,
  "verificationStatus": "LEGACY_UNSIGNED",
  "verificationReason": "Record predates signed evidence rollout",
  "compliance": null
```

```
}
```

Compliance violation (compliance mode, signing failed):

```
{
  "evidenceId": 12346,
  "recordType": "compliance_violation_evidence",
  "hashValid": true,
  "chainValid": true,
  "signaturePresent": false,
  "signatureValid": false,
  "verificationStatus": "COMPLIANCE_VIOLATION",
  "verificationReason": "Signing failed in compliance-enforced mode",
  "compliance": {
    "framework": "EU_AI_ACT",
    "systemMode": "compliance_enforced",
    "article12_traceable": true,
    "article12_tamper_resistant": false,
    "article14_oversight_supported": true,
    "article28_audit_ready": false
  }
}
```

Standard mode, signing failed:

```
{
  "evidenceId": 12347,
  "recordType": "unsigned_evidence",
  "hashValid": true,
  "chainValid": true,
  "signaturePresent": false,
  "signatureValid": false,
  "verificationStatus": "UNSIGNED",
  "verificationReason": "Signing key unavailable at record creation time",
  "compliance": null
}
---
```

Part VI — Global Invariants

Apply all of these. Do not skip.

Kernel

K-1 through K-6 (see full gate system document)

Evidence

EVI-1 through EVI-7 (see full gate system document)

Proof

P-1 through P-4 (see full gate system document)

Reliability

R-1 through R-5 (see full gate system document)

Governance-of-Governance

GOG-1 through GOG-3 (see full gate system document)

Signed Evidence (feature-specific)

SE-1 through SE-18 (see v1 — all apply unchanged)

Compliance (new — adopted in Revision 2)

CI-1 — Traceability Every governed action must produce a record reconstructible by a third party without internal DB access.

CI-2 — Tamper Resistance SHA-256 hash + chain + Ed25519 signature together constitute tamper resistance. Neither hash alone nor signature alone is sufficient for the full claim.

CI-3 — Provenance The system must prove which system produced the record, which key signed it, and that the signer is trusted. Symmetric keys (HMAC) do not satisfy CI-3.

CI-4 — Independent Verifiability A third party must be able to verify using only: JWKS endpoint, verification API, documented payload schema. No internal access required.

CI-5 — Truthful Representation Compliance flags are derived from verification outcomes. Never set by assertion. A failed check must produce **`false`** on every compliance flag that depends on it.

CI-6 — Oversight Support Evidence records must distinguish decision outcome from execution outcome. Approval paths and intervention points must be exposed.

Part VII — Gate Criteria

Gate D — Architectural Coherence

PASS requires:

- [ ] Signed payload schema is defined, versioned, and locked
- [ ] **`regulatoryContext`** is inside the signed payload (cryptographically bound)
- [ ] Compliance flags are derived from verification — not asserted
- [ ] Key lifecycle documented (Option A confirmed)
- [ ] Failure modes defined for both standard and compliance mode
- [ ] Legacy handling explicit
- [ ] External verification contract defined

Gate E — Enforcement / Integrity

PASS requires:

- [ ] Signing occurs after **`insertGovernanceEvidence()`** returns (evidenceId and proofChainHash bound)
- [ ] No path skips signing silently on a required path

- [ ] Signed payload includes all required fields
- [ ] All tamper matrix tests pass
- [ ] All key failure matrix tests pass
- [ ] Compliance mode failure is observable (not silent)

Gate F — Integration / Embodiment

PASS requires:

- [ ] Migration adds all columns
- [ ] Schema.ts updated
- [ ] **`recordEvidence()`** signs and persists
- [ ] **`proof-rest-api.ts`** verifies signature + returns compliance flags
- [ ] **`proof-router.ts`** verifies signature + returns compliance flags
- [ ] **`ProofReceipt`** type updated
- [ ] Outbox (**`evidence_outbox`**) preserves signature fields on sync
- [ ] **`/.well-known/strix-jwks.json`** live
- [ ] Proof Trust Center shows three-layer badges + compliance badges
- [ ] **`STRIX_SIGNING_PRIVATE_KEY`**, **`STRIX_COMPLIANCE_MODE`**,

`STRIX_COMPLIANCE_BLOCK_ON_SIGN_FAILURE` in **`.env.example`**

Gate G — Reliability / Failure Modes

PASS requires all scenarios defined (Part III Decision 2a/2b) plus:

- [ ] **`STRIX_COMPLIANCE_MODE`** missing → standard mode (no crash)
- [ ] JWKS endpoint unreachable at verify time → **`verificationStatus: "UNVERIFIABLE"`**
- [ ] Key rotation test: old record signed with previous **`kid`** still verifies
- [ ] Outbox retry preserves signature bytes exactly (no re-signing)
- [ ] Mixed deployment: unsigned legacy + new signed records verified correctly

Gate H — Observability / External Proof

PASS requires:

- [ ] Verification response matches schemas in Part V
- [ ] Compliance flags computed from verification, tested
- [ ] EU AI Act badges on Trust Center only when **`article28_audit_ready: true`**
- [ ] Legacy records labeled **`LEGACY_UNSIGNED`**, never shown as compliance-failed
- [ ] JWKS publicly accessible, CORS-open
- [ ] An external party can verify independently using only public material

Gate J — Crypto Primitive Integrity

PASS requires:

- [ ] Ed25519 used (not ECDSA or RSA)
- [ ] `schemaVersion` in every signed payload
- [ ] `kid` mandatory — no unsigned-kid records possible
- [ ] Golden test vector committed
- [ ] Full tamper + key-failure regression matrix
- [ ] Key rotation historical verification vector

Part VIII — Required Output Format

Return this structure when implementation is complete.

Agent: Date: Branch / PR:

Gate Results

Gate	Status	Notes
D — Architectural Coherence	PASS / FAIL / PARTIAL	
E — Enforcement / Integrity	PASS / FAIL / PARTIAL	
F — Integration / Embodiment	PASS / FAIL / PARTIAL	
G — Reliability / Failure Modes	PASS / FAIL / PARTIAL	
H — Observability / External Proof	PASS / FAIL / PARTIAL	
J — Crypto Primitive Integrity	PASS / FAIL / PARTIAL	

Overall Status: READY TO MERGE / BLOCKED

Required Artifacts Checklist

- [ ] Migration with all signature + compliance columns
- [ ] Signed payload schema locked and versioned
- [ ] JWKS endpoint live and CORS-open
- [ ] Signing logic in `recordEvidence()` with correct call order
- [ ] Verification in `proof-rest-api.ts` and `proof-router.ts`
- [ ] Compliance flags derived (not asserted)
- [ ] `ProofReceipt` type updated
- [ ] Proof Trust Center: three-layer badges + EU AI Act compliance badges

- [ ] Full test suite including tamper matrix, key failure matrix, compliance flags
- [ ] Golden test vector committed
- [ ] Env vars documented in `.env.example`
- [ ] Key rotation test passing

Invariant Check

Invariant	Status	Evidence
CI-1 Traceability	PASS / FAIL	
CI-2 Tamper Resistance	PASS / FAIL	
CI-3 Provenance	PASS / FAIL	
CI-4 Independent Verifiability	PASS / FAIL	
CI-5 Truthful Representation	PASS / FAIL	
CI-6 Oversight Support	PASS / FAIL	
SE-1 through SE-18	(list each)	
P-1 Proof endpoints never overstate	PASS / FAIL	
GOG-3 Claims do not exceed proven	PASS / FAIL	

Failure Modes Defined

Scenario	Behavior
`STRIX_SIGNING_PRIVATE_KEY` missing — standard mode	
`STRIX_SIGNING_PRIVATE_KEY` missing — compliance mode	
Signing throws at runtime	
JWKS endpoint unreachable at verify time	
Key rotation — old record verification	
Mixed-version deployment	
Outbox retry with signature	
Tampered field at verify time	
`STRIX_COMPLIANCE_BLOCK_ON_SIGN_FAILURE=true`, signing fails	

Verification Response Samples

(Paste live examples from running environment)

Signed, compliance mode, fully verified:

Legacy unsigned:

Compliance violation:

Standard unsigned:

Trust Claim Impact

State exactly how this feature changes Strix's external trust claim. Confirm claim does not exceed what verification can prove.

Final Decision

APPROVED / APPROVED WITH FOLLOW-UPS / NOT APPROVED

Next Actions

(In implementation order)

Strix Agent Update — Standard Format v1

1. Summary

2. Technical Changes

- Files updated:
- Functions modified:
- Schema changes:
- New endpoints:
- New env vars:

3. Impacts

- Trust claim change:
- Compliance surface:
- Dependencies affected:
- Risks:

4. Next Steps

5. Notion Entries

Build Log

- Agent:
- Date:

- Summary:
- Files Changed:
- Status:

Architecture Log

- Decision: Add Ed25519 cryptographic signatures with EU AI Act compliance mode to governance evidence records
- Reason: Upgrade trust claim from hash integrity to provenance + non-repudiation; enable regulatory-grade auditability for Articles 12, 14, 28
- Impact: Proof Trust Center, Investor Lab, external verification API, compliance positioning