TELOS Record of Trust Protocol
Receipt Semantics, Verification Boundaries, and Deployment Evidence
TELOS AI Labs Inc.
August 2026

Purpose

The Record of Trust is the evidence layer produced around TELOS measurement. It is intended to let a reviewer connect three things:

- the authority declared before work began;
- the actions submitted for measurement;
- the contemporaneous score and decision recorded for each submitted action.

The current implementation does not put all three inside one self-contained receipt. This protocol therefore distinguishes the signed action receipt from the separate deployment evidence needed to establish authorization, ordering, human handling, identity, and time.

Evidence model

Record set

A reviewable record set may contain:

Evidence object	Function	Current status
Approved Manifest and Purpose Anchor	Defines the authority and specification against which work is measured	Deployment artifact
Configuration record	Identifies scorer version, embedding model, boundary corpus version, weights, thresholds, optional components, gate mode, and fail policy	Must be preserved separately
Signed action receipt	Covers submitted action, measurement fields, runtime decision, tool name, and timestamp	Available through the opened session signing path
Session export	Collects receipts associated with a session	Available as a list, not a signed hash-linked chain
Adapter event log	Shows that a result was surfaced, blocked, allowed, or escalated	Deployment-specific evidence
Human decision record	Shows approval, rejection, override, resolution, or termination	Must be preserved separately if needed
Key binding evidence	Connects a verification key to an issuer, deployment, and validity period	Must be pinned independently
Trusted time evidence	Establishes time beyond the receipt's own timestamp field	Not provided by the public prototype

The word record refers to this set. It must not be used to imply that the present receipt schema alone proves the entire authorization and handling history.

Signed receipt path

When an integration submits a scored result through GovernanceSessionContext.sign_result(), the signed payload covers the fields emitted by that path. According to the opened implementation and whitepaper disclosure, those fields include:

- action text;
- decision point;
- five dimension values;
- composite fidelity;
- effective fidelity;
- runtime decision;
- boundary flag;
- tool name;
- timestamp;
- cryptographic envelope.

The five reported dimensions are purpose, scope, tool, chain, and boundary. A verifier should use the exact schema and canonicalization identified in the deployment's verification material.

What the signature establishes

An Ed25519 verification result can establish that:

- the covered bytes match the signed payload;
- the signature was created by the private key corresponding to the supplied public key;
- alteration to a covered field after signing is detectable.

This statement assumes the verifier obtained the correct public key through an independent and trustworthy channel. A signature does not, by itself, establish who controlled the key, whether the key was valid at the claimed time, or whether the timestamp came from a trusted clock.

Session co-signature

The disclosed production path describes a session-bound co-signature that binds a receipt to a governed session and requires session verification material. The exact composition and implementation remain held under NDA. A deployment claiming this property should publish or privately furnish a verifier with the applicable schema, covered fields, key-binding evidence, and validation procedure.

Fields not presently bound in the opened receipt

The opened receipt schema does not include:

- Manifest identity or version;
- approval root;
- principal identity;
- predecessor receipt hash;

- configuration fingerprint;
- human approval or rejection;
- specification change;
- override or resolution event;
- termination event;
- trusted timestamp authority evidence.

These omissions determine the claim boundary. Authorization, chronological sequence, human handling, and trusted time require separately preserved evidence.

Public benchmark receipt path

The benchmark projections published in the public repository use integrity hashes and predecessor links. They do not include signatures or a public verification key. The bundled verifier accordingly reports that signature verification is not possible from the public commitments alone.

The public evidence can demonstrate:

- that the published payload matches its integrity hash;
- that predecessor references form the published order;
- that a changed covered payload would no longer match its recorded hash.

The public evidence cannot, by itself, demonstrate:

- who authored or issued the record;
- that the stated timestamp is trustworthy;
- that a Manifest was approved before the action;
- that the scoring configuration matches a production deployment;
- that the record was complete when published;
- that a human saw or resolved a surfaced event.

A hash is an integrity commitment. It is not a digital signature and must not be described as one.

Chain semantics

The opened generate_proof() method exports a list of receipts. It does not sign the list as a chain and the signed receipt does not carry a predecessor hash. Chain-level signing therefore remains COMING.

Where a deployment needs ordered, tamper-evident session evidence, the complete design should bind at least:

where:

- is the current chain commitment;
- is the prior commitment;
- is a canonical representation of the event and receipt;
- is the policy, Manifest, and configuration identity applicable to the event;
- is a documented cryptographic hash function;
- denotes unambiguous concatenation.

The terminal commitment should be signed by a key whose deployment identity and validity period can be checked independently. This equation describes the protocol target. It does not assert that the opened implementation currently performs it.

Recommended deployment evidence profile

Authority record

Preserve before activation:

- Manifest content and semantic version;
- immutable content hash;
- Purpose Anchor or reproducible compilation commitment;
- named approving principal;
- approval decision and effective time;
- validity window;
- superseded version, if any.

Measurement configuration record

Preserve for every effective configuration:

- TELOS engine version;
- embedding model and model artifact hash;
- normalization procedure;
- boundary corpus version and commitment;
- centroid or prototype construction method version;
- dimension weights;
- decision and escalation thresholds;
- optional classifier and micro-judge state;
- gateMode and failPolicy;
- canonicalization and signing profile;
- key identifier.

This configuration is necessary because a score cannot be interpreted or reproduced without knowing the measurement conditions.
Per-action record
Preserve:

- canonical submitted action;
- action and session identifiers;
- tool and decision point;
- dimension scores;
- composite and effective fidelity;
- boundary flag and runtime decision;
- event time and clock source;
- Manifest and configuration identifiers;
- predecessor commitment;
- signature and key identifier;
- handling result from the adapter;
- human resolution where applicable.

Some of these fields are current receipt fields and some are target fields. Implementations must mark the difference explicitly.
Offline verification procedure
A reviewer should perform the following checks:

- Obtain the Manifest, configuration, issuer identity, public key, and validity material through an independently trusted channel.
- Confirm that the Manifest and configuration hashes match the commitments named in the record set.
- Recreate the canonical payload using the declared schema version.
- Verify the Ed25519 signature over the documented covered bytes.
- Recompute each payload hash and, where implemented, each predecessor or chain commitment.
- Check key validity, revocation status, and deployment binding for the claimed period.
- Check the timestamp against independently preserved clock or timestamp evidence if time is material.
- Reproduce scores only if the applicable embedding, boundary, compilation, and calibration artifacts are available.
- Compare the runtime decision with the declared thresholds and policy.
- Inspect adapter and human event records before claiming that an event was surfaced, blocked, approved, or resolved.

The verifier should return separate results for payload integrity, signature validity, key identity, ordering, authorization binding, score reproducibility, handling evidence, and trusted time. A single valid flag conceals material distinctions.
How the record is used: underwriting and claims
At underwriting time, agent deployments are largely opaque to a pricing decision. An underwriter sees a vendor questionnaire and a policy narrative. Manifest-bound deployment replaces that with reviewable artifacts: each material agent's Manifest, meaning its scope of authority in writing and approved by a named human; the escalation policy and its named recipient; and a standing record of whether past work stayed within scope, with departures visible.
At claims time, the questions turn concrete:

- Was the loss-generating action within approved scope?
- Was it detected as a departure at the time?
- Did it surface to the accountable human, and when?

The current record set answers these from signed per-action payloads together with companion deployment evidence identifying the constraint version in force.
Those associations must not be presented as chain-level cryptographic proof. Signed-field binding of the Manifest version, approval-root signing, and signing the attestation sequence as one chain are all COMING, not WIRED. The answers above therefore rest partly on separately preserved deployment evidence, and their strength is the strength of that preservation. Coverage also extends only to actions submitted through an instrumented path.
For the deployer, the record is the difference between asserting diligence and producing it: a per-action account of oversight that exists before anyone asks for it.
Retention and confidentiality
The intelligence-layer configuration exposes a default retention_days value of 90, but automatic expiry is not implemented in the opened source. Clearing is manual and removes all telemetry for an agent rather than only aged records.
Encryption at rest is optional. When configured, session record lines can be encrypted. The session header and aggregate file remain cleartext, and encryption setup failure can fall back to plaintext. Deployers must therefore assess the actual stored content, metadata, failure behavior, access controls, backup handling, and deletion process.
Retention periods must be set from the governing legal, contractual, safety, and quality obligations of the deployment. For example, EU AI Act Article 19 establishes a minimum period for certain automatically generated logs under a high-risk AI provider's control, subject to applicable Union or national law. TELOS does not automatically implement or prove that retention obligation.
Threat model
The receipt protocol is intended to detect after-the-fact alteration of covered data and to support reconstruction when all necessary evidence is preserved. It does not, without additional controls, prevent or resolve:

- compromise of the signing key;
- signing by a malicious authorized process;
- omission of events before signing;
- false timestamps from an untrusted host;

- use of an unapproved Manifest or scorer;
- manipulation of the action before it reaches the measurement adapter;
- incomplete preservation or selective disclosure;
- a reviewer receiving the wrong public key;
- errors in the scoring model or boundary corpus.

Mitigations include hardware-backed keys, append-only storage, remote attestation, independent timestamping, key transparency, configuration commitments, completeness monitors, dual control over Manifest approval, and independent audit.

Capability status

Capability	Status	Defensible claim
Per-result Ed25519 signing path	WIRED in opened session code	Covered-field alteration is detectable against the correct pinned key
Session-bound co-signature	Disclosed production path, implementation held under NDA	Requires deployment-specific verification evidence
Public benchmark integrity hashes	WIRED	Payload-to-hash and published predecessor integrity
Public signature verification	NOT AVAILABLE from public commitments	No public authorship claim
Manifest version bound into receipt	COMING	Must currently be supplied by separate deployment evidence
Approval root bound into receipt	COMING	Must currently be supplied separately
Signed hash-linked session chain	COMING	Current export is a receipt list
Automatic retention expiry	COMING	Current clearing is manual
Independent trusted time	Deployment-specific	Receipt timestamp alone is insufficient
Legal compliance certification	NOT CLAIMED	Evidence relevance only

Claim language for external use

The accurate statement of what this protocol supports:

TELOS can produce signed per-action measurement receipts. When verified against independently pinned deployment keys, the receipts make alteration of covered fields detectable. Authorization, sequence, human handling, and trusted time depend on the additional deployment evidence preserved with the receipts.

Claims of immutability, of a blockchain-like proof, or of independent certification that every agent action is compliant are not supported. Such language overstates immutability, chain implementation, authorship, authorization binding, and legal sufficiency.

Related documents

- Core whitepaper
- Technical method
- Validation report
- Regulatory alignment map

Appendix: receipt exhibit, demonstration key, and cold verifier

Demonstration boundary. This is a standalone demonstration fixture, not a public benchmark receipt, not a production-chain entry, and not evidence of issuer provenance. It verifies payload integrity against the included demo key. Checking a signed chain needs the signed payloads themselves, plus verification material you pinned independently; because the included key travels with the fixture, it is not that independent pin. The verifier checks the thirteen canonical payload fields, payload hash, and Ed25519 signature; it does not verify the included HMAC value, session history, approval root, a chain-level signature, authorship, or time. The demonstration script exits 0 in both genuine and tampered modes; its semantic output, not process exit status, distinguishes genuine=True from one-field-tamper=False.

The full receipt shown in III.2, with its complete signature envelope, a standalone cold verifier (Python plus cryptography), and the verification transcript. Signed with a fresh demo Ed25519 keypair generated only for this exhibit; not entered into a production chain.

Receipt as emitted:


```
{
  "decision_point": "tool_execute",
  "action_text": "Draft appointment confirmation email for existing customer C-10472 for Tuesday 2026-07-07 at 10:30 PT; do not quote pricing; route pricing questions to the account owner.",
  "decision": "escalate",
  "effective_fidelity": 0.818,
  "composite_fidelity": 0.818,
  "boundary_triggered": true,
  "tool_name": "customer_outreach.email_draft",
  "timestamp": 1783462200.0,
  "purpose_fidelity": 1.0,
  "scope_fidelity": 1.0,
  "boundary_violation": 0.817,
  "tool_fidelity": 1.0,
  "chain_continuity": 0.0,
  "ed25519_signature":
"9cd8debe1f1e4bdf27a75387cd5d952034a05a1dfe41cfb7073419aea3a85f2dd5e14a42d3f188cdfa33857131bbcb7fa191930829279a9a4a249bc51353a90d",
  "hmac_signature":
"fdc3e9e88e6db1d55493ca38cacbb4150610692d0604d929c5ac1e46901c994fddf1d19c40384faaf760c745c5ffb891cf6601058cfa08f532b6407dc7196f9a",
  "public_key": "71720f9e7a0bd1f2609b64ad5eef23afc730474bd468baedb096b8ce6a0ecfee",
```

```
  "payload_hash": "8d4b907e5f6ffbe256f3cc9eaae97063686d330cbb9326219c675fa01f46fbe0"
}
```

Demo Ed25519 public key:

```
71720f9e7a0bd1f2609b64ad5eef23afc730474bd468baedb096b8ce6a0ecfee
```

Standalone cold verifier, requiring only Python stdlib plus cryptography:

```
import argparse
import copy
import hashlib
import json

from cryptography.exceptions import InvalidSignature
from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PublicKey


CANONICAL_FIELDS = [
    "decision_point",
    "action_text",
    "decision",
    "effective_fidelity",
    "composite_fidelity",
    "boundary_triggered",
    "tool_name",
    "timestamp",
    "purpose_fidelity",
    "scope_fidelity",
    "boundary_violation",
    "tool_fidelity",
    "chain_continuity",
]


def canonicalize(receipt):
    payload = {field: receipt[field] for field in CANONICAL_FIELDS}
    return json.dumps(payload, sort_keys=True, separators=(",", ":")).encode("utf-8")


def verify(receipt, public_key_hex):
    expected_hash = hashlib.sha256(canonicalize(receipt)).digest()
    if expected_hash.hex() != receipt.get("payload_hash"):
        return False
    try:
        public_key = Ed25519PublicKey.from_public_bytes(bytes.fromhex(public_key_hex))
        public_key.verify(bytes.fromhex(receipt["ed25519_signature"]), expected_hash)
        return True
    except (InvalidSignature, ValueError, KeyError):
        return False


parser = argparse.ArgumentParser()
parser.add_argument("receipt_json")
parser.add_argument("public_key_hex")
parser.add_argument("--tamper", action="store_true")
args = parser.parse_args()

with open(args.receipt_json, "r", encoding="utf-8") as f:
    receipt = json.load(f)

if args.tamper:
    receipt = copy.deepcopy(receipt)
    receipt["scope_fidelity"] = 0.287
    print(f"one-field-tamper={verify(receipt, args.public_key_hex)}")
else:
    print(f"genuine={verify(receipt, args.public_key_hex)}")
```

Cold verification transcript:

```
$ python cold_verify_receipt.py receipt.json 71720f9e7a0bd1f2609b64ad5eef23afc730474bd468baedb096b8ce6a0ecfee
genuine=True

$ python cold_verify_receipt.py receipt.json 71720f9e7a0bd1f2609b64ad5eef23afc730474bd468baedb096b8ce6a0ecfee --tamper
one-field-tamper=False
```