Network Working Group
Internet-Draft
Intended status: Informational
Expires: 18 October 2026

Y.C. Chang
OIA Lab

16 April 2026

Delta-1: A SCITT Profile for Binary Settlement Receipts in Agentic AI Accountability

draft-chang-delta1-settlement-00

Abstract

This document defines Delta-1, a profile of the SCITT (Supply Chain Integrity, Transparency, and Trust) architecture applied to AI agent decision accountability.

Delta-1 specializes the SCITT receipt model for a specific use case: proving that an individual AI decision met three accountability conditions simultaneously:

C1: Evidence of the decision process was consumed and sealed. C2: Strategic intent was isolated from inference providers. C3: An authorized party recorded the settlement.

The verdict is binary (VALID or INVALID) with no intermediate states. Receipts are independently verifiable without the issuing infrastructure being online, subject to the trust and attestation assumptions defined in this document.

By building on the SCITT framework, Delta-1 inherits transparency-log semantics, receipt conventions, and verification patterns, while extending them with domain-specific requirements for AI decision closure.

This profile is intended for regulated industries, including finance, healthcare, and legal, operating under accountability and record-retention obligations.

Note to RFC Editor

This document is submitted as an individual Internet-Draft. Remove this note prior to publication as an RFC.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 18 October 2026.

Copyright Notice

Table of Contents

1. Introduction

As AI agents increasingly make or assist in consequential decisions, such as financial trades, clinical recommendations, procurement approvals, and legal assessments, organizations need a mechanism to prove that each decision met defined accountability conditions at the time it was made.

Current approaches often provide authorization semantics ("this AI action was approved") but not settlement semantics ("the decision’s accountability conditions can be cryptographically proven after the fact"). This gap creates regulatory, legal, and operational risk.

1.1. Relationship to SCITT

The SCITT architecture defines a framework for creating transparent, tamper-evident signed statements registered on append-only transparency services, with receipts proving inclusion.

Delta-1 maps SCITT concepts to AI decision accountability:

* SCITT Signed Statement = Delta-1 condition evaluation result

* SCITT Transparency Service = append-only evidence chain

* SCITT Receipt = Delta-1 settlement receipt (binary verdict)

* SCITT Issuer = AI decision pipeline being attested

* SCITT Verifier = Customer, auditor, or regulator

Delta-1 extends SCITT with:

* Binary verdict semantics (VALID or INVALID, with no scoring)

* Three-condition conjunction (C1 AND C2 AND C3)

* Transcript binding to prevent receipt transplantation

* Anti-replay protection via run identifiers and monotonic counters

* Optional hardware attestation to upgrade trust from software self-report to hardware-attested execution

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

Settlement Receipt A cryptographically signed document attesting to the binary verdict of a Delta-1 evaluation.

Delta-1 (D1) The conjunction of three conditions (C1 AND C2 AND C3) that, when all are true, indicates accountability closure for a single AI decision.

C1 (Evidence Consumed) The decision’s evidence chain has been recorded in an append-only, tamper-evident store and sealed.

C2 (Intent Isolated) Strategic intent was fragmented or transformed before reaching the inference provider, preventing the provider from reconstructing the decision-maker’s strategy under the threat model defined by the deployment.

C3 (Settlement Recorded) An authorized party has reviewed and signed off on the decision’s closure.

BBOX Black-box evidence chain: an append-only, hash-linked log of decision-process steps.

Transcript Binding A canonical digest that cryptographically binds the receipt to the pipeline execution context.

3. Protocol Overview

The Delta-1 protocol operates as the final step in an AI decision accountability pipeline:

1. AI query is processed through privacy and isolation layers.
2. Evidence of each processing step is recorded in the evidence chain.
3. The evidence chain is sealed.
4. The Delta-1 validator evaluates C1, C2, and C3.
5. A binary receipt is produced: VALID or INVALID.
6. The receipt is signed using the customer’s signing key.
7. Optional hardware attestation evidence is attached.
8. The receipt is delivered for archival and later verification.

Verification is performed by the receipt holder. Offline verification of the receipt signature and transcript binding is REQUIRED. Verification of optional external artifacts, such as TEE quote status or trusted time evidence, MAY depend on cached trust anchors or online validation services, depending on local policy.

4. Receipt Structure

A Delta-1 receipt is a JSON object with the following fields.

4.1. Required Fields

schema_version String. MUST be "delta1-v1".

receipt_id String. Globally unique identifier for this receipt. RECOMMENDED format: "EP-" followed by 8 or more hexadecimal characters.

session_id String. Identifier of the decision session.

run_id String. Globally unique nonce for anti-replay protection. MUST NOT be reused across receipts.

sequence_number Unsigned integer. Monotonically increasing counter per issuing validator instance.

delta1_valid Boolean. True if and only if C1, C2, and C3 are all true.

conditions_met Object containing boolean fields c1, c2, and c3.

timestamp String. RFC 3339 timestamp of receipt issuance.

proof_hash String. SHA-256 digest of the canonical proof input defined in Section 8.1.

signature String. Ed25519 signature over the proof_hash, produced by the customer's signing key.

public_key_ref String. Reference to the public key used for signature verification.

4.2. Optional Fields

transcript_digest String. Canonical digest of the pipeline execution context. REQUIRED for production deployments.

attestation Object. Hardware attestation proof, if a TEE is used.

bbox_seal_hash String. SHA-256 digest of the sealed evidence chain.

trusted_time Object. Optional trusted time evidence, such as RFC 3161 or Roughtime-derived proof.

5. Condition Evaluation

The Delta-1 validator MUST evaluate three conditions. All three MUST be true for the receipt to be VALID.

5.1. C1: Evidence Consumed

C1 is true when all of the following hold:

a. An evidence chain exists for the session.

b. The chain contains at least the minimum required evidence packets; implementation-defined, RECOMMENDED value is 3 or greater.

c. Required processing layers are represented in the chain.

d. Chain integrity is valid and hash linkage is unbroken.

e. The chain has been sealed and no further entries are accepted.

5.2. C2: Intent Isolated

C2 is true when all of the following hold:

a. Intent fragmentation or transformation was applied before transmission to the inference provider.

b. Training-signal blocking mechanisms were active.

c. Confirmation-denial or equivalent response screening verified that provider responses did not reveal protected original entities.

d. Under the deployment threat model, the original intent is not reconstructable from the transformed query material delivered to providers.

Deployments claiming C2 compliance SHOULD document the threat model, attacker vantage point, and reconstruction criteria used for validation.

5.3. C3: Settlement Recorded

C3 is true when all of the following hold:

a. A settlement record exists for the session.

b. The settlement was produced by an authorized signer.

c. The signature is cryptographically valid.

d. The settlement has not expired.

e. Closure was explicitly acknowledged and was not purely automatic.

5.4. Binary Conjunction

The verdict is computed as follows:

delta1_valid = c1 AND c2 AND c3

There is no partial validity, no scoring, and no weighted average. Two out of three conditions being true still yields INVALID.

6. Transcript Binding

To prevent receipt transplantation, the receipt SHOULD include a transcript binding digest.

The transcript_digest is computed as follows:

```
transcript_digest = SHA-256(
  original_request_digest ||
  transformed_request_digests ||
  provider_dispatch_map ||
  provider_response_digests ||
  trt_verdict_digest ||
  bbox_seal_hash ||
  policy_versions ||
  model_identities
)
```

6.1. Binding Fields

original_request_digest SHA-256 of the user’s original query before transformation.

transformed_request_digests SHA-256 of each transformed fragment sent to providers.

provider_dispatch_map Ordered pairs of fragment identifier and provider identifier.

provider_response_digests SHA-256 of each provider’s raw response.

trt_verdict_digest SHA-256 digest of the transformation or sanitization verdict.

bbox_seal_hash SHA-256 digest of the sealed evidence chain.

policy_versions Identifiers of active policy versions at processing time.

model_identities Provider and model identifiers used in the pipeline.

7. Anti-Replay Protection

7.1. Run ID Nonce

Each receipt MUST contain a globally unique run_id. The validator MUST reject any validation request whose run_id has been previously used.

Implementations SHOULD use cryptographically random identifiers of at least 128 bits.

7.2. Monotonic Sequence Number

Each receipt MUST contain a sequence_number that is strictly greater than the sequence_number of any previous receipt from the same validator instance.

Gaps are permitted, such as after rejected validations, but decreases are not permitted.

8. Signature and Verification

8.1. Proof Construction and Signing

Before computing proof_hash, implementations MUST construct a canonical proof input object containing the following fields: session_id, run_id, sequence_number, conditions_met, transcript_digest, and timestamp.

The proof input object MUST be serialized using the JSON Canonicalization Scheme (JCS) defined in [RFC8785]. The proof_hash is the SHA-256 digest of that canonicalized byte string.

The receipt signature MUST use Ed25519 [RFC8032]. The signed payload is the proof_hash.

The signing key MUST be controlled by the customer, not by the infrastructure vendor.

8.2. Verification

Verification requires, at minimum:

a. The receipt JSON document.

b. The customer’s Ed25519 public key.

Verification procedure:

1. Recompute proof_hash from the canonical proof input.

2. Verify the Ed25519 signature against proof_hash.

3. If transcript_digest is present, verify that it matches the expected canonical digest.

4. If sequence tracking is implemented, verify that sequence_number exceeds the last known value for the validator instance.

Verification of receipt signature and transcript binding MUST NOT require network access, vendor APIs, or online services.

9. Hardware Attestation (Optional)

When the Delta-1 validator executes inside a Trusted Execution Environment (TEE), the receipt MAY include attestation evidence.

9.1. Attestation Proof Structure

tee_type TEE technology identifier, such as sgx_dcap, sev_snp, cca, or nitro.

quote The TEE attestation artifact appropriate to tee_type.

measurement Measurement of the trusted execution payload, such as MRENCLAVE or equivalent.

signer_identity Identity of the signing authority for the measured payload, if applicable.

tcb_level Status of the TEE trusted computing base.

report_data_hash The receipt proof_hash bound into the attestation evidence.

mode Deployment mode. Simulation values MUST NOT be trusted in production.

9.2. Attestation Verification

Attestation verification depends on tee_type and local trust policy. Implementations MAY validate attestation online, or offline using cached collateral and trust anchors.

At minimum, verifiers SHOULD:

A. Verify the attestation evidence against the relevant TEE trust chain.

B. Verify that the measured payload matches an expected published measurement.

C. Verify that report_data_hash equals receipt.proof_hash.

D. Verify that tcb_level is acceptable under local policy.

9.3. Trust Model

Without attestation, the customer trusts that vendor-operated software ran correctly. With attestation, trust shifts toward the TEE platform and its measurement chain. This is a substantial trust upgrade, but not a zero-trust guarantee.

10. Receipt Lifecycle

10.1. Immutability

A receipt, once issued, MUST NOT be modified. If a condition is later found to have been recorded incorrectly, a new receipt MUST be issued with a new receipt_id and the updated verdict.

10.2. Archival

Receipts SHOULD be archived by the customer for the duration required by applicable regulation, contract, or internal retention policy.

10.3. Chain of Custody

To prove accountability over a period of time, implementations MAY chain receipts by incorporating the previous receipt’s proof_hash into the next receipt’s transcript binding.

11. Transport Considerations

Delta-1 receipts are transport-agnostic. They MAY be delivered via:

* HTTPS API response for real-time systems

* File download, including compliance or legal export flows

* Message queue for asynchronous processing pipelines

* Blockchain anchoring for optional public verifiability

The receipt format is JSON. Implementations SHOULD support both JSON and CBOR [RFC8949] encodings where bandwidth-constrained environments require compact encoding.

12. Security Considerations

12.1. Self-Attestation Risk

Without TEE support, the receipt is ultimately a software self-report. A compromised validator can produce fraudulent VALID receipts.

12.2. Key Management

The customer’s Ed25519 signing key is a root of trust. Compromise of this key permits forged receipts. Implementations SHOULD use hardware security modules or equivalent protections.

12.3. Replay Attacks

run_id nonces and monotonic sequence_number values are used to mitigate replay attacks. Validators MUST reject duplicate run_id values.

12.4. Transcript Transplantation

Without transcript binding, a valid receipt could be attached to a different decision. Production deployments SHOULD include transcript binding.

12.5. Side-Channel Attacks on TEE

TEE technologies remain exposed to side-channel and implementation-level attacks. Current mitigations reduce, but do not eliminate, residual risk.

12.6. Timing Information

The timestamp in the receipt is produced by the validator’s local clock. High-assurance deployments SHOULD bind receipt issuance time to a trusted external time source, such as RFC 3161 or Roughtime-based evidence.

12.7. JSON Canonicalization

Receipt proof construction MUST use JCS canonicalization before hashing. Without deterministic serialization, semantically identical content can yield different proof_hash values.

12.8. Key Rotation and Revocation

Implementations MUST support key rotation. Historical receipts signed by an old key remain valid if the corresponding public key and validity period are preserved.

12.9. Validator Integrity

The Delta-1 validator is a single point of trust when no TEE is used. Reproducible builds, transparency logs, and multi-party validation are RECOMMENDED mitigations for high-assurance deployments.

12.10. TEE Vendor Generalization

Implementations SHOULD support multiple TEE families and MUST distinguish them using tee_type.

12.11. Quote Freshness

Attestation artifacts can be replayed. Binding proof_hash into report_data_hash ensures that an attestation artifact covers exactly one receipt instance.

13. IANA Considerations

This document requests registration of the "application/ delta1-receipt+json" media type in the "Media Types" registry.

13.1. Media Type Registration

Type name application

Subtype name delta1-receipt+json

Required parameters None.

Optional parameters version, default value "delta1-v1".

Encoding considerations Binary. Delta-1 receipt documents are UTF-8 encoded JSON texts.

Security considerations See Section 12.

Interoperability considerations Interoperability depends on consistent canonicalization, proof construction, and algorithm support as defined in this specification.

Published specification This document.

Applications that use this media type AI accountability systems, evidence archival systems, regulatory reporting systems, and audit verification tools.

Fragment identifier considerations Same as for application/json.

Additional information Magic number: N/A. File extension: .delta1.json. Macintosh file type code: N/A.

Person and email address to contact for further information YC Chang, yc@oia-lab.com

Intended usage Common.

Restrictions on usage None.

Author YC Chang

Change controller IETF

14. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

[SCITT-ARCH] Birkholz, H., "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture, <https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture/>.

15. Informative References

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9530] Polli, R. and L. Pardue, "Digest Fields", RFC 9530, DOI 10.17487/RFC9530, February 2024, <https://www.rfc-editor.org/info/rfc9530>.

[P11] Chang, Y., "Authorization Is Not Settlement: Why AI Accountability Requires Binary Closure", DOI 10.5281/zenodo.19338864, March 2026, <https://doi.org/10.5281/zenodo.19338864>.

[EUAIA] Council, E. P. A., "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act)", 2024.

[COAIA] Colorado, S. O., "SB 24-205 Concerning Consumer Protections for Artificial Intelligence", 2024.

[FORESHADOW] Bulck, J. V., "Foreshadow: Extracting the Keys to the Intel SGX Kingdom", 2018.

Appendix A. JSON Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Delta1Receipt",
  "type": "object",
  "required": [
    "schema_version",
    "receipt_id",
    "session_id",
    "run_id",
    "sequence_number",
    "delta1_valid",
    "conditions_met",
    "timestamp",
    "proof_hash",
    "signature",
    "public_key_ref"
  ],
  "properties": {
    "schema_version": {
      "type": "string",
      "const": "delta1-v1"
    },
    "receipt_id": { "type": "string" },
    "session_id": { "type": "string" },
    "run_id": { "type": "string" },
    "sequence_number": {
      "type": "integer",
      "minimum": 1
    },
    "delta1_valid": { "type": "boolean" },
    "conditions_met": {
      "type": "object",
      "required": ["c1", "c2", "c3"],
      "properties": {
        "c1": { "type": "boolean" },
        "c2": { "type": "boolean" },
        "c3": { "type": "boolean" }
      }
    },
    "timestamp": {
      "type": "string",
      "format": "date-time"
    },
    "proof_hash": { "type": "string" },
    "signature": { "type": "string" },
    "public_key_ref": { "type": "string" },
    "transcript_digest": { "type": "string" },
    "bbox_seal_hash": { "type": "string" },
```

```
      "trusted_time": {
        "type": "object",
        "properties": {
          "type": { "type": "string" },
          "value": { "type": "string" }
        }
      },
      "attestation": {
        "type": "object",
        "properties": {
          "tee_type": { "type": "string" },
          "quote": { "type": "string" },
          "measurement": { "type": "string" },
          "signer_identity": { "type": "string" },
          "tcb_level": { "type": "string" },
          "report_data_hash": { "type": "string" },
          "mode": { "type": "string" }
        }
      }
    }
  }
```

Appendix B. Example Receipts

VALID Receipt (All Conditions Met)

```
{
  "schema_version": "delta1-v1",
  "receipt_id": "EP-mnu2gmo2-df9ba6d5",
  "session_id": "session-20260414-e3f1",
  "run_id": "run-20260414-a7f3b2c1d4e5",
  "sequence_number": 42,
  "delta1_valid": true,
  "conditions_met": { "c1": true, "c2": true, "c3": true },
  "timestamp": "2026-04-14T10:24:15Z",
  "proof_hash": "3fa2b7c1e8d9f4a5b6c7d8e9f0a1b2c3...",
  "signature": "ed25519:MIGHAoGBANr5nD1y4X...",
  "public_key_ref": "customer_hsm_key_001",
  "transcript_digest": "a1b2c3d4e5f6a7b8c9d0e1f2...",
  "bbox_seal_hash": "9a8b7c6d5e4f3a2b1c...",
  "attestation": {
    "tee_type": "sgx_dcap",
    "quote": "SIMULATION:abc123...",
    "measurement": "f7a2b1c3d4e5f6a7b8c9d0e1...",
    "signer_identity": "e1f2a3b4c5d6e7f8a9b0c1d2...",
    "tcb_level": "simulation",
    "report_data_hash": "3fa2b7c1e8d9f4a5b6c7d8e9f0a1b2c3...",
    "mode": "simulation"
  }
}
```


INVALID Receipt (C2 Failed)


```
{
  "schema_version": "delta1-v1",
  "receipt_id": "EP-x9k3m7p2-c4a1b8f3",
  "session_id": "session-20260414-b2c3",
  "run_id": "run-20260414-f8e7d6c5b4a3",
  "sequence_number": 43,
  "delta1_valid": false,
  "conditions_met": { "c1": true, "c2": false, "c3": true },
  "timestamp": "2026-04-14T10:25:02Z",
  "proof_hash": "7b8c9d0e1f2a3b4c5d6e7f8a...",
  "signature": "ed25519:KJHFaskjdfh...",
  "public_key_ref": "customer_hsm_key_001",
  "transcript_digest": ""
}
```


Author’s Address

YC Chang
OIA Lab
Email: yc@oia-lab.com
URI: https://oia-lab.com