

Verifiable AI Refusal Events using SCITT Transparency Logs  
draft-kamimura-scitt-refusal-events-01

## Abstract

This document describes a SCITT-based mechanism for creating verifiable records of AI content refusal events. It defines how refusal decisions can be encoded as SCITT Signed Statements, registered with Transparency Services, and verified by third parties using Receipts.

This specification provides auditability of refusal decisions that are logged, not cryptographic proof that no unlogged generation occurred. It does not define content moderation policies, classification criteria, or what AI systems should refuse; it addresses only the audit trail mechanism.

This revision (-01) incorporates lessons from the January 2026 Grok NCII incident, aligns with the CAP-SRP v1.0 specification, and addresses emerging regulatory requirements including EU AI Act Article 12/50 and the Korea AI Basic Act.

## About This Document

This note is to be removed before publishing as an RFC.

The latest version of this document, along with implementation resources and examples, can be found at [CAP-SRP].

Discussion of this document takes place on the SCITT Working Group mailing list (scitt@ietf.org).

The companion specification CAP-SRP v1.0 [CAP-SRP-SPEC] provides a complete domain profile for content/creative AI systems.

## Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 August 2026.

## Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	4
1.1. Motivation . . . . .	4
1.1.1. The Grok NCII Incident (January 2026) . . . . .	5
1.2. Regulatory Context . . . . .	5
1.2.1. EU AI Act . . . . .	5
1.2.2. Korea AI Basic Act . . . . .	6
1.2.3. US Regulatory Landscape . . . . .	6
1.3. Requirements Language . . . . .	6
1.4. Scope . . . . .	7
1.5. Limitations . . . . .	7
1.6. Relationship to CAP-SRP . . . . .	8
2. Terminology . . . . .	8
2.1. Mapping to SCITT Primitives . . . . .	9
3. Conformance Levels . . . . .	10
3.1. Bronze Level . . . . .	10
3.2. Silver Level . . . . .	11
3.3. Gold Level . . . . .	11
4. Use Cases . . . . .	12
4.1. Regulatory Audit . . . . .	12
4.2. Incident Investigation . . . . .	12

4.3.	Provider Accountability . . . . .	12
4.4.	Legal Proceedings . . . . .	13
4.5.	User Verification . . . . .	13
5.	Requirements . . . . .	13
5.1.	Completeness Invariant . . . . .	13
5.2.	Integrity . . . . .	14
5.3.	Privacy . . . . .	15
5.4.	Third-Party Verifiability . . . . .	15
5.5.	Timeliness . . . . .	16
5.6.	Conformance . . . . .	16
6.	Data Model . . . . .	17
6.1.	ATTEMPT Signed Statement Payload . . . . .	17
6.2.	DENY Signed Statement Payload . . . . .	19
6.3.	GENERATE Signed Statement Payload . . . . .	20
6.4.	ERROR Signed Statement Payload . . . . .	21
6.5.	Risk Categories (Non-Normative) . . . . .	22
7.	Evidence Pack . . . . .	23
7.1.	Directory Structure . . . . .	24
7.2.	Manifest Format . . . . .	24
7.3.	Verification Process . . . . .	25
8.	SCITT Integration . . . . .	26
8.1.	Encoding as Signed Statements . . . . .	26
8.2.	Registration . . . . .	26
8.3.	Registration Policy Guidance (Non-Normative) . . . . .	26
8.4.	Verification with Receipts . . . . .	27
8.5.	External Anchoring . . . . .	27
9.	C2PA Integration (Non-Normative) . . . . .	28
10.	IANA Considerations . . . . .	28
11.	Security Considerations . . . . .	29
11.1.	Threat Model . . . . .	29
11.2.	Omission Attacks . . . . .	29
11.3.	Log Equivocation . . . . .	30
11.4.	Split-View Between Event Types . . . . .	30
11.5.	Log Tampering . . . . .	30
11.6.	Replay Attacks . . . . .	30
11.7.	Key Compromise . . . . .	31
11.8.	Prompt Dictionary Attacks . . . . .	31
11.9.	Denial of Service . . . . .	31
12.	Privacy Considerations . . . . .	31
12.1.	Harmful Content Storage . . . . .	31
12.2.	Actor Identification . . . . .	32
12.3.	Correlation Risks . . . . .	32
12.4.	Data Subject Rights . . . . .	32
13.	Future Work (Non-Normative) . . . . .	32
13.1.	RATS/Attestation Integration . . . . .	32
13.2.	Batching and Scalability . . . . .	33
13.3.	Advanced Privacy Mechanisms . . . . .	33
13.4.	External Completeness Enforcement . . . . .	33

13.5. Post-Quantum Cryptography Migration . . . . .	33
14. References . . . . .	33
14.1. Normative References . . . . .	34
14.2. Informative References . . . . .	34
Appendix A. Example: Complete Refusal Event Flow . . . . .	35
A.1. Event Sequence . . . . .	35
A.2. Third-Party Verification . . . . .	36
A.3. Evidence Pack Verification . . . . .	36
Acknowledgements . . . . .	37
Author's Address . . . . .	37

## 1. Introduction

This document is NOT a content moderation policy. It does not prescribe what AI systems should or should not refuse to generate, nor does it define criteria for classifying requests as harmful. The mechanism described herein is agnostic to the reasons for refusal decisions; it provides only an interoperable format for recording that such decisions occurred. Policy decisions regarding acceptable content remain the domain of AI providers, regulators, and applicable law.

### 1.1. Motivation

AI systems capable of generating content increasingly implement safety mechanisms to refuse requests deemed harmful, illegal, or policy-violating. However, these refusal decisions typically leave no verifiable audit trail. When a system refuses to generate content, the event vanishes—there is no receipt, no log entry accessible to external parties, and no mechanism for third-party verification.

This creates several problems:

- \* Regulators cannot independently verify that AI providers enforce stated policies
- \* Providers cannot prove to external auditors that specific requests were refused
- \* Third parties investigating incidents have no way to establish refusal without trusting provider claims
- \* The completeness of audit logs cannot be verified externally

The SCITT architecture [I-D.ietf-scitt-architecture] provides primitives—Signed Statements, Transparency Services, and Receipts—that can address this gap. This document describes how these primitives can be applied to AI refusal events.

#### 1.1.1. The Grok NCII Incident (January 2026)

The January 2026 Grok incident exposed the critical need for verifiable refusal mechanisms. xAI's generative AI system produced approximately 4.4 million images in 9 days, with external analysis indicating at least 41% were sexualized images and 2% depicted minors. This triggered unprecedented multi-jurisdictional enforcement:

- \* EU Digital Services Act investigation (potential fine up to 6% of global revenue)
- \* 35-state US coalition demanding elimination of harmful content capabilities
- \* UK Ofcom Online Safety Act investigation (potential fine up to 10% of global revenue)
- \* Brazil joint regulatory action with 30-day compliance deadline
- \* Indonesia temporary service block

When xAI asserted that moderation systems were functioning, no external party could verify this claim. The absence of verifiable refusal records meant regulators had to rely on provider self-reports, AI Forensics external testing, and user complaints rather than cryptographic proof.

This incident demonstrates that the problem is not detection accuracy alone, but verification capability. Even if an AI system has effective content classifiers, the inability to prove refusals occurred creates an accountability gap that this specification addresses.

#### 1.2. Regulatory Context

Multiple jurisdictions are implementing AI transparency and logging requirements that this specification can help satisfy:

##### 1.2.1. EU AI Act

The EU AI Act (Regulation 2024/1689) establishes comprehensive logging requirements:

- \* Article 12 mandates automatic recording of events for high-risk AI systems, with minimum 6-month retention
- \* Article 50 requires AI-generated content to be marked in machine-readable format, with detection tools available
- \* High-risk AI obligations become applicable August 2, 2026

This specification's event model directly supports Article 12 compliance by providing tamper-evident logging with external anchoring for independent verification.

#### 1.2.2. Korea AI Basic Act

The Korea AI Basic Act (AI), effective January 22, 2026, requires:

- \* Mandatory labeling for generative AI outputs
- \* Meaningful explanations for high-impact AI decisions
- \* Domestic representatives for foreign AI businesses exceeding specified thresholds

The completeness invariant defined in this specification provides a mechanism for demonstrating that AI systems make consistent decisions that can be explained and audited.

#### 1.2.3. US Regulatory Landscape

US regulations relevant to AI content provenance include:

- \* Colorado AI Act (SB24-205, effective June 30, 2026): requires impact assessments and 3-year document retention
- \* California SB 942 (effective August 2, 2026): requires provenance metadata and detection tools
- \* TAKE IT DOWN Act (platform compliance by May 19, 2026): requires 48-hour NCII removal with documentation

#### 1.3. Requirements Language

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.

#### 1.4. Scope

This document describes:

- \* Terminology for refusal events mapped to SCITT primitives
- \* A data model for ATTEMPT and DENY events as Signed Statement payloads
- \* A completeness invariant for audit trail integrity checking
- \* An integration approach with SCITT Transparency Services
- \* Evidence Pack format for regulatory submission
- \* Conformance levels (Bronze/Silver/Gold) for graduated implementation

This document does NOT define:

- \* Content moderation policies (what should be refused)
- \* Classification algorithms or risk scoring methods
- \* Thresholds or criteria for refusal decisions
- \* General SCITT architecture (see [I-D.ietf-scitt-architecture])
- \* Legal or regulatory requirements for specific jurisdictions

#### 1.5. Limitations

This specification provides auditability of refusal decisions that are logged, not cryptographic proof that no unlogged generation occurred. An AI system that bypasses logging entirely cannot be detected by this mechanism alone. Detection of such bypass requires external enforcement mechanisms (e.g., trusted execution environments, attestation) which are outside the scope of this document.

This profile does not require Transparency Services to enforce completeness invariants; such checks are performed by verifiers using application-level logic.

## 1.6. Relationship to CAP-SRP

This Internet-Draft provides the IETF-track specification for verifiable AI refusal events using SCITT primitives. The companion CAP-SRP specification [CAP-SRP-SPEC] published by the VeritasChain Standards Organization provides:

- \* Complete domain profile for content/creative AI systems
- \* Integration with the VAP (Verifiable AI Provenance) framework
- \* Detailed regulatory compliance mapping
- \* Reference implementation guidance
- \* C2PA integration for content provenance

Implementations may conform to this Internet-Draft alone for basic SCITT interoperability, or additionally conform to CAP-SRP for comprehensive content AI audit trail capabilities.

## 2. Terminology

This document uses terminology from [I-D.ietf-scitt-architecture]. The following terms are specific to this profile:

**Generation Request** A request submitted to an AI system to produce content. May include text prompts, reference images, or other inputs.

**Refusal Event** A decision by an AI system to decline a generation request based on safety, policy, or other criteria. Results in no content being produced.

**ATTEMPT (GEN\_ATTEMPT)** A Signed Statement recording that a generation request was received. Does not indicate the outcome. CAP-SRP uses the term GEN\_ATTEMPT for this event type.

**DENY (GEN\_DENY)** A Signed Statement recording that a generation request was refused. References the corresponding ATTEMPT via attemptId. CAP-SRP uses the term GEN\_DENY for this event type.

**GENERATE (GEN)** A Signed Statement recording that content was successfully generated in response to a request. References the corresponding ATTEMPT via attemptId. CAP-SRP uses the term GEN for this event type.

**ERROR (GEN\_ERROR)** A Signed Statement indicating that no content was



generated due to system failure (e.g., timeout, resource exhaustion, model error) rather than a policy decision. ERROR does not constitute a refusal and does not indicate policy enforcement. References the corresponding ATTEMPT via attemptId.

**Outcome** A Signed Statement recording the result of a generation request: DENY (refusal), GENERATE (successful generation), or ERROR (system failure).

**Verifiable Refusal Record** An auditable record consisting of an ATTEMPT Signed Statement, a DENY Signed Statement, and Receipts proving their inclusion in a Transparency Service. This provides evidence that a refusal decision was logged, but does not cryptographically prove that no unlogged generation occurred.

**Completeness Invariant** The property that every logged ATTEMPT has exactly one corresponding Outcome. Formally:  $\Box \text{ ATTEMPT} = \Box \text{ GENERATE} + \Box \text{ DENY} + \Box \text{ ERROR}$ . This invariant is checked by verifiers at the application level; it is not enforced by Transparency Services.

**Evidence Pack** A self-contained, cryptographically verifiable collection of events suitable for regulatory submission or third-party audit. Includes events, anchor records, Merkle proofs, and verification metadata.

**promptHash** A cryptographic hash of the generation request content. Enables verification that a specific request was processed without storing the potentially harmful prompt text.

**Refusal Receipt** A cryptographic token provided to users proving their request was processed and refused. Enables user-side verification without exposing internal audit details.

This document focuses on refusal events because successful generation is already observable through content existence and downstream provenance mechanisms (e.g., C2PA manifests, watermarks). Refusal events, by contrast, are negative events that leave no external artifact unless explicitly logged. The GENERATE and ERROR outcomes are defined for completeness invariant verification but are not the primary focus of this specification.

## 2.1. Mapping to SCITT Primitives

This profile maps refusal event concepts directly to SCITT primitives, minimizing new terminology:

=====	=====	=====
This Document	CAP-SRP Term	SCITT Primitive
=====	=====	=====
ATTEMPT	GEN_ATTEMPT	Signed Statement
-----	-----	-----
DENY	GEN_DENY	Signed Statement
-----	-----	-----
GENERATE	GEN	Signed Statement
-----	-----	-----
ERROR	GEN_ERROR	Signed Statement
-----	-----	-----
AI System	Issuer	Issuer
-----	-----	-----
Inclusion Proof	Receipt	Receipt
-----	-----	-----

Table 1

Refusal events are registered with a standard SCITT Transparency Service; this document does not define a separate log type.

This document uses domain-agnostic event type names (ATTEMPT, DENY, GENERATE, ERROR) to enable application across multiple AI domains. CAP-SRP uses domain-specific prefixes (GEN\_ATTEMPT, GEN\_DENY, GEN, GEN\_ERROR) appropriate for content generation. Implementations targeting CAP-SRP conformance SHOULD use CAP-SRP event type names in the eventType field; implementations targeting broader SCITT interoperability MAY use the names defined in this document.

### 3. Conformance Levels

This specification defines three conformance levels to accommodate different organizational capabilities and regulatory requirements. These levels align with the VAP Framework v1.2 [VAP-FRAMEWORK] conformance structure.

#### 3.1. Bronze Level

Minimum requirements for basic conformance:

- \* MUST: Log all ATTEMPT events before safety evaluation
- \* MUST: Log corresponding Outcome for each ATTEMPT
- \* MUST: Hash prompt content (promptHash), never store cleartext
- \* MUST: Sign all events using COSE\_Sign1

- \* MUST: Use SHA-256 for hashing
- \* MUST: Use Ed25519 for signatures
- \* MUST: Include ISO 8601 timestamps with timezone
- \* SHOULD: Use UUIDv7 for event identifiers
- \* SHOULD: Implement hash chain linking (PrevHash)
- \* Retention: Minimum 6 months

Bronze level is suitable for voluntary transparency and early adopters.

### 3.2. Silver Level

All Bronze requirements, plus:

- \* MUST: Register events with SCITT Transparency Service
- \* MUST: Obtain and store Receipts for all events
- \* MUST: Implement external anchoring (minimum daily)
- \* MUST: Verify Completeness Invariant continuously
- \* MUST: Support Evidence Pack generation
- \* MUST: Implement Merkle tree construction
- \* SHOULD: Provide third-party verification endpoint
- \* Retention: Minimum 2 years

Silver level is recommended for organizations subject to EU AI Act Article 12 or similar regulations.

### 3.3. Gold Level

All Silver requirements, plus:

- \* MUST: Implement real-time anchoring (within 1 hour)
- \* MUST: Use HSM for signing key storage
- \* MUST: Provide real-time audit API

- \* MUST: Support 24-hour incident response evidence preservation
- \* SHOULD: Integrate with SCITT Transparency Service for continuous monitoring
- \* SHOULD: Implement geographic redundancy
- \* Retention: Minimum 5 years

Gold level is required for Very Large Online Platforms (VLOPs) under DSA Article 37 and high-risk AI systems requiring maximum assurance.

#### 4. Use Cases

##### 4.1. Regulatory Audit

A regulatory authority investigating AI system compliance needs to verify that a provider's stated content policies are actually enforced. Without verifiable refusal events, the regulator must trust provider self-reports. With this mechanism, regulators can request Evidence Packs for specified time ranges, verify ATTEMPT/Outcome completeness for logged events, confirm refusal decisions are anchored in an append-only log, and compare refusal statistics against external incident reports.

This directly addresses the verification gap exposed by the Grok incident, where regulators had no mechanism to independently verify provider claims about safety system effectiveness.

##### 4.2. Incident Investigation

When investigating whether an AI system refused a specific request, investigators need to establish provenance. A Verifiable Refusal Record (ATTEMPT + DENY + Receipts) demonstrates that a specific request was received, classified as policy-violating, refused, and the refusal was logged with external timestamp anchoring.

##### 4.3. Provider Accountability

AI service providers may need to demonstrate to stakeholders that safety mechanisms function as claimed. Verifiable refusal events enable statistical reporting on logged refusal rates, third-party verification of safety claims, auditable proof that specific requests were refused, and comparison against industry benchmarks.

#### 4.4. Legal Proceedings

In legal proceedings concerning AI-generated content, parties may need evidence that a system declined a request. Verifiable Refusal Records provide such evidence, subject to the limitation that they demonstrate logged refusals, not the absence of unlogged generation. Evidence Packs provide court-admissible documentation with cryptographic integrity proofs.

#### 4.5. User Verification

Users who receive refusals may need proof that their request was processed. Refusal Receipts enable users to verify their request was logged, appeal refusal decisions with evidence, and demonstrate to third parties that they attempted but were refused (useful for content creators documenting compliance efforts).

### 5. Requirements

This section defines requirements for implementations. To maximize interoperability while allowing implementation flexibility, a small set of core requirements use MUST; other requirements use SHOULD or MAY.

#### 5.1. Completeness Invariant

The completeness invariant is the central requirement of this profile:

Formal definition:

$$\boxed{\times} \text{ ATTEMPT} = \boxed{\times} \text{ GENERATE} + \boxed{\times} \text{ DENY} + \boxed{\times} \text{ ERROR}$$

For any time window  $[T, T]$ :

$$\begin{aligned} \text{count(ATTEMPT where T timestamp T)} = \\ \text{count(GENERATE where T timestamp T + grace\_period)} + \\ \text{count(DENY where T timestamp T + grace\_period)} + \\ \text{count(ERROR where T timestamp T + grace\_period)} \end{aligned}$$

- \* Every logged ATTEMPT Signed Statement MUST have exactly one corresponding Outcome Signed Statement (DENY, GENERATE, or ERROR).
- \* Outcome Signed Statements MUST reference their corresponding ATTEMPT via the attemptId field.
- \* Prompt content MUST NOT be stored in cleartext in Signed Statements; implementations MUST use cryptographic hashes (promptHash) instead.

- \* The ATTEMPT event MUST be logged BEFORE any safety evaluation begins (pre-evaluation logging).

Verifiers SHOULD flag any logged ATTEMPT without a corresponding Outcome as potential evidence of incomplete logging or system failure.

Violation detection:

Condition	Meaning	Implication
Attempts > Outcomes	Unmatched attempts	System may be hiding results
Outcomes > Attempts	Orphan outcomes	System may have fabricated refusals
Duplicate outcomes	Multiple outcomes per attempt	Data integrity failure

Table 2

This completeness invariant is defined at the event semantics level and applies only to logged events. It cannot detect ATTEMPT events that were never logged. Cryptographic detection of invariant violations depends on the properties of the underlying Transparency Service and verifier logic.

This profile does not require Transparency Services to enforce completeness invariants; such checks are performed by verifiers using application-level logic.

## 5.2. Integrity

To protect against tampering, implementations SHOULD:

- \* Include a cryptographic hash of event content in each Signed Statement (EventHash)
- \* Digitally sign all Signed Statements
- \* Chain events via PrevHash fields to detect deletion or reordering
- \* Register Signed Statements with a SCITT Transparency Service and obtain Receipts

PrevHash chaining is RECOMMENDED but not required because append-only guarantees are primarily provided by the Transparency Service. PrevHash provides an additional, issuer-local integrity signal that can detect tampering even before Transparency Service registration.

SHA-256 for hashing and Ed25519 for signatures are RECOMMENDED. Other algorithms registered with COSE MAY be used. Implementations SHOULD support algorithm agility for future post-quantum cryptography migration.

### 5.3. Privacy

Refusal events may be triggered by harmful or sensitive content. To avoid the audit log becoming a repository of harmful content, implementations SHOULD:

- \* Replace prompt text with promptHash
- \* Replace reference images with cryptographic hashes
- \* Ensure refusal reasons do not quote or describe prompt content in detail
- \* Pseudonymize actor identifiers where appropriate (ActorHash)

The hash function SHOULD be collision-resistant to prevent an adversary from claiming a benign prompt hashes to the same value as a harmful one.

Hashing without salting may be vulnerable to dictionary attacks if an adversary has a list of candidate prompts. Mitigations include access controls on event queries, time-limited retention policies, and monitoring for bulk query patterns. Salting may provide additional protection but introduces complexity; if used, implementations must ensure verification remains possible without requiring disclosure of the salt to third-party verifiers.

### 5.4. Third-Party Verifiability

To enable external verification without access to internal systems, implementations SHOULD:

- \* Ensure verification requires only the Signed Statement and Receipt
- \* Publish Issuer public signing keys or certificates
- \* Make Transparency Service logs queryable by authorized auditors

- \* Support offline verification given the necessary cryptographic material
- \* Provide Evidence Pack export in standardized format

### 5.5. Timeliness

To maintain audit trail integrity, implementations SHOULD:

- \* Create ATTEMPT Signed Statements promptly upon request receipt (within 100ms)
- \* Create Outcome Signed Statements promptly upon decision (within 1 second for automated decisions)
- \* Register Signed Statements with the Transparency Service within a reasonable window (within 60 seconds)

External anchoring frequency requirements by conformance level:

Level	Minimum Frequency	Maximum Delay
Bronze	Optional	N/A
Silver	Daily	24 hours
Gold	Hourly	1 hour

Table 3

Some operational scenarios may require delayed outcomes:

- \* Human review processes may take minutes, hours, or days
- \* System crashes may delay outcome logging until recovery
- \* Network failures may delay Transparency Service registration

Implementations SHOULD document expected latency bounds in their Registration Policy. Extended delays SHOULD trigger monitoring alerts.

### 5.6. Conformance

An implementation conforms to this specification if it satisfies the following requirements:



- \* MUST: Every logged ATTEMPT has exactly one Outcome
- \* MUST: Outcomes reference ATTEMPTs via attemptId
- \* MUST: Prompt content is hashed, not stored in cleartext
- \* MUST: Signed Statements are encoded as COSE\_Sign1
- \* MUST: ATTEMPT is logged before safety evaluation

All other requirements (SHOULD, RECOMMENDED, MAY) are guidance for interoperability and security best practices but are not required for conformance.

Implementations MAY extend the data model with additional fields provided the core conformance requirements are satisfied.

Implementations claiming a specific conformance level (Bronze/Silver/Gold) MUST satisfy all requirements for that level as defined in Section 3.

## 6. Data Model

This section defines example payloads for ATTEMPT, DENY, GENERATE, and ERROR Signed Statements. These are encoded as JSON payloads. This data model is non-normative; implementations MAY extend or modify these structures provided the conformance requirements in Section 5.6 are satisfied.

### 6.1. ATTEMPT Signed Statement Payload

An ATTEMPT records that a generation request was received:

```
{
  "eventType": "ATTEMPT",
  "eventId": "019467a1-0001-7000-0000-000000000001",
  "chainId": "019467a0-0000-7000-0000-000000000000",
  "timestamp": "2026-01-29T14:23:45.100Z",
  "issuer": "urn:example:ai-service:img-gen-prod",
  "promptHash": "sha256:7f83b1657ff1fc53b92dc18148a1d65d...",
  "inputType": "text+image",
  "referenceInputHashes": [
    "sha256:9f86d081884c7d659a2feaa0c55ad015..."
  ],
  "sessionId": "019467a1-0001-7000-0000-000000000000",
  "actorHash": "sha256:e3b0c44298fc1c149afbf4c8996fb924...",
  "modelId": "img-gen-v4.2.1",
  "policyId": "content-safety-v2",
  "policyVersion": "2026-01-01",
  "hashAlgo": "SHA256",
  "signAlgo": "ED25519",
  "prevHash": "sha256:00000000000000000000000000000000...",
  "eventHash": "sha256:alb2c3d4e5f6alb2c3d4e5f6alb2c3d4..."
}
```

Field definitions:

eventType "ATTEMPT" (or "GEN\_ATTEMPT" for CAP-SRP alignment)

eventId Unique identifier (UUID v7 [RFC9562] RECOMMENDED for temporal ordering)

chainId Identifier for the event chain (enables multiple independent chains per issuer)

timestamp ISO 8601 timestamp of request receipt with timezone

issuer URN identifying the AI system

promptHash Hash of the textual prompt (if any)

inputType Type of input: "text", "image", "text+image", "audio", "video"

referenceInputHashes Array of hashes for non-text inputs

sessionId Session identifier for correlation

actorHash Pseudonymized hash of the requesting user/system

modelId Identifier and version of the AI model

policyId Identifier of the content policy applied

policyVersion Version of the policy (enables policy change tracking)

hashAlgo Hash algorithm used (default: "SHA256")

signAlgo Signature algorithm used (default: "ED25519")

prevHash Hash of the previous event (for chain integrity)

eventHash Hash of this event's canonical form

## 6.2. DENY Signed Statement Payload

A DENY records that a request was refused:

```
{
  "eventType": "DENY",
  "eventId": "019467a1-0001-7000-0000-000000000002",
  "chainId": "019467a0-0000-7000-0000-000000000000",
  "timestamp": "2026-01-29T14:23:45.150Z",
  "issuer": "urn:example:ai-service:img-gen-prod",
  "attemptId": "019467a1-0001-7000-0000-000000000001",
  "riskCategory": "NCII_RISK",
  "riskSubCategories": ["REAL_PERSON", "CLOTHING_REMOVAL_REQUEST"],
  "riskScore": 0.94,
  "refusalReason": "Non-consensual intimate imagery request detected",
  "modelDecision": "DENY",
  "humanOverride": false,
  "escalationId": null,
  "hashAlgo": "SHA256",
  "signAlgo": "ED25519",
  "prevHash": "sha256:alb2c3d4e5f6alb2c3d4e5f6alb2c3d4...",
  "eventHash": "sha256:e5f6g7h8i9j0e5f6g7h8i9j0e5f6g7h8..."
}
```

Field definitions:

eventType "DENY" (or "GEN\_DENY" for CAP-SRP alignment)

eventId Unique identifier

chainId Must match the corresponding ATTEMPT's chainId

timestamp ISO 8601 timestamp of refusal decision

attemptId Reference to the corresponding ATTEMPT (required for completeness invariant)

**riskCategory** Category of policy violation detected. See Section 6.5 for non-normative taxonomy.

**riskSubCategories** Array of sub-categories for detailed classification

**riskScore** Confidence score (0.0 to 1.0) if available. Scoring methodology is implementation-defined.

**refusalReason** Human-readable reason (SHOULD NOT contain prompt content). The taxonomy of reasons is implementation-defined.

**modelDecision** The action taken: "DENY", "WARN", "ESCALATE", "QUARANTINE"

**humanOverride** Boolean indicating if a human reviewer was involved

**escalationId** Reference to escalation record if human review was triggered

**prevHash** Hash of the previous event

**eventHash** Hash of this event's canonical form

This specification does not standardize content moderation categories, risk taxonomies, or refusal reason formats. These are policy decisions that remain the domain of AI providers and applicable regulations.

### 6.3. GENERATE Signed Statement Payload

A GENERATE records that content was successfully produced. This document focuses on refusal events; GENERATE is included for completeness invariant verification:

```
{
  "eventType": "GENERATE",
  "eventId": "019467a1-0001-7000-0000-000000000004",
  "chainId": "019467a0-0000-7000-0000-000000000000",
  "timestamp": "2026-01-29T14:23:46.500Z",
  "issuer": "urn:example:ai-service:img-gen-prod",
  "attemptId": "019467a1-0001-7000-0000-000000000001",
  "outputHash": "sha256:b2c3d4e5f6a7b2c3d4e5f6a7b2c3d4e5...",
  "outputType": "image/png",
  "c2paManifestId": "urn:c2pa:manifest:...",
  "hashAlgo": "SHA256",
  "signAlgo": "ED25519",
  "prevHash": "sha256:a1b2c3d4e5f6a1b2c3d4e5f6a1b2c3d4...",
  "eventHash": "sha256:c3d4e5f6a7b8c3d4e5f6a7b8c3d4e5f6..."
}
```

Field definitions specific to GENERATE:

`outputHash` Hash of the generated content (enables verification without storing content)

`outputType` MIME type of generated content

`c2paManifestId` OPTIONAL. Reference to C2PA manifest if content provenance is embedded in output

GENERATE events are typically not the focus of regulatory audits since successful generation is observable through content existence and downstream provenance (e.g., C2PA manifests, SynthID watermarks). They are included here to enable completeness invariant verification.

#### 6.4. ERROR Signed Statement Payload

An ERROR records that a request failed due to system issues:

```
{
  "eventType": "ERROR",
  "eventId": "019467a1-0001-7000-0000-000000000003",
  "chainId": "019467a0-0000-7000-0000-000000000000",
  "timestamp": "2026-01-29T14:23:45.200Z",
  "issuer": "urn:example:ai-service:img-gen-prod",
  "attemptId": "019467a1-0001-7000-0000-000000000001",
  "errorCode": "TIMEOUT",
  "errorMessage": "Model inference timeout after 30s",
  "hashAlgo": "SHA256",
  "signAlgo": "ED25519",
  "prevHash": "sha256:e5f6g7h8i9j0e5f6g7h8i9j0e5f6g7h8...",
  "eventHash": "sha256:h8i9j0k1l2m3h8i9j0k1l2m3h8i9j0k1..."
}
```

ERROR events indicate system failures, not policy decisions. A high ERROR rate may indicate operational issues or potential abuse (e.g., adversarial inputs designed to crash the system). Implementations SHOULD monitor ERROR rates and investigate anomalies.

#### 6.5. Risk Categories (Non-Normative)

The following risk categories are provided as a non-normative reference taxonomy. Implementations MAY use different categories based on their policies and applicable regulations:

Category	Description	Example
CSAM_RISK	Child sexual abuse material	Minor sexualization request
NCII_RISK	Non-consensual intimate imagery	Deepfake pornography
MINOR_SEXUALIZATION	Content sexualizing minors	Age-inappropriate requests
REAL_PERSON_DEEPFAKE	Unauthorized realistic depiction	Celebrity face swap
VIOLENCE_EXTREME	Graphic violence	Gore, torture
HATE_CONTENT	Discriminatory content	Racist imagery
TERRORIST_CONTENT	Terrorism-related	Propaganda, recruitment
SELF_HARM_PROMOTION	Self-harm encouragement	Suicide methods
COPYRIGHT_VIOLATION	Clear IP infringement	Trademarked characters
COPYRIGHT_STYLE_MIMICRY	Artist style imitation	Protected style requests
OTHER	Other policy violations	Custom policies

Table 4

## 7. Evidence Pack

An Evidence Pack is a self-contained, cryptographically verifiable collection of events suitable for regulatory submission or third-party audit.

### 7.1. Directory Structure

```
evidence_pack/
├── manifest.json          # Pack metadata and integrity info
├── events/
│   ├── events_001.jsonl   # Event batch 1 (JSON Lines format)
│   ├── events_002.jsonl   # Event batch 2
│   └── ...
├── anchors/
│   ├── anchor_001.json    # External anchor records
│   └── ...
├── merkle/
│   ├── tree_001.json      # Merkle tree structure
│   └── proofs/            # Selective disclosure proofs
├── keys/
│   └── public_keys.json    # Public keys for verification
└── signatures/
    └── pack_signature.json # Pack-level signature
```

### 7.2. Manifest Format



```
{
  "packId": "019467b2-0000-7000-0000-000000000000",
  "packVersion": "1.0",
  "generatedAt": "2026-01-29T15:00:00Z",
  "generatedBy": "urn:example:ai-service:img-gen-prod",
  "conformanceLevel": "Silver",
  "eventCount": 150000,
  "timeRange": {
    "start": "2026-01-01T00:00:00Z",
    "end": "2026-01-29T14:59:59Z"
  },
  "checksums": {
    "events/events_001.jsonl": "sha256:...",
    "events/events_002.jsonl": "sha256:...",
    "anchors/anchor_001.json": "sha256:..."
  },
  "completenessVerification": {
    "totalAttempts": 145000,
    "totalGenerate": 140000,
    "totalDeny": 4500,
    "totalError": 500,
    "invariantValid": true,
    "verificationTimestamp": "2026-01-29T15:00:00Z"
  },
  "externalAnchors": [
    {
      "anchorId": "019467b0-0000-7000-0000-000000000000",
      "anchorType": "RFC3161",
      "anchorTimestamp": "2026-01-29T00:00:00Z"
    }
  ]
}
```

### 7.3. Verification Process

Third-party verification of an Evidence Pack involves:

1. Verify pack signature against published public key
2. Verify all file checksums in manifest
3. Verify hash chain integrity across all events
4. Verify individual event signatures
5. Verify Completeness Invariant
6. Verify external anchor records against TSA/SCITT

## 7. Generate verification report

## 8. SCITT Integration

### 8.1. Encoding as Signed Statements

ATTEMPT, DENY, GENERATE, and ERROR events are encoded as SCITT Signed Statements:

- \* The event JSON is the Signed Statement payload
- \* The Issuer is the AI system's signing identity
- \* The Content Type MAY use "application/vnd.scitt.refusal-event+json"
- \* The Signed Statement is wrapped in COSE\_Sign1 per [RFC9052]

The JSON payload is canonicalized per [RFC8785] and signed as the COSE\_Sign1 payload bytes. This ensures deterministic serialization for signature verification.

### 8.2. Registration

After creating a Signed Statement, the Issuer SHOULD register it with a SCITT Transparency Service:

1. Submit the Signed Statement via SCRAPI [I-D.ietf-scitt-scrapi]
2. Receive a Receipt proving inclusion
3. Store the Receipt for future verification requests

The Transparency Service's Registration Policy MAY verify that required fields are present and timestamps are within acceptable bounds.

Registration may fail due to network issues, service unavailability, or policy rejection. Implementations SHOULD implement retry logic with exponential backoff. Persistent registration failures SHOULD be logged locally and trigger operational alerts.

### 8.3. Registration Policy Guidance (Non-Normative)

A Transparency Service operating as a refusal event log MAY implement a Registration Policy that validates:

- \* Signature validity (COSE\_Sign1 verification)

- \* Required fields present (eventType, eventId, timestamp, issuer)
- \* Timestamp sanity (not in the future, not unreasonably old)
- \* Issuer authorization (if the TS restricts which issuers may register)

This profile does not require Transparency Services to enforce completeness invariants. A TS accepting refusal events is not expected to verify that every ATTEMPT has an Outcome; such verification is performed by auditors and verifiers at the application level.

#### 8.4. Verification with Receipts

A complete Verifiable Refusal Record consists of:

1. The ATTEMPT Signed Statement and its Receipt
2. The corresponding DENY Signed Statement and its Receipt
3. Verification that attemptId in DENY matches eventId in ATTEMPT

Verifiers can confirm that a refusal was logged by validating both Receipts and checking the ATTEMPT/DENY linkage. This demonstrates that the refusal decision was recorded in the Transparency Service, but does not prove that no unlogged generation occurred.

#### 8.5. External Anchoring

For additional assurance, implementations MAY periodically anchor Merkle tree roots to external systems:

- \* RFC 3161 Time Stamping Authority (TSA)
- \* Multiple independent SCITT Transparency Services
- \* Public blockchains (Bitcoin, Ethereum)
- \* Regulatory authority registries

External anchoring provides defense against a compromised Transparency Service and satisfies regulatory requirements for independent timestamp verification.

Anchor record format:

```
{
  "anchorId": "019467b0-0000-7000-0000-000000000000",
  "anchorType": "RFC3161",
  "merkleRoot": "sha256:abcd1234...",
  "eventCount": 1000,
  "firstEventId": "019467a0-0000-7000-0000-000000000001",
  "lastEventId": "019467a0-0000-7000-0000-000001000000",
  "timestamp": "2026-01-29T00:00:00Z",
  "anchorProof": "MIIHkwyJKoZIhvc...",
  "serviceEndpoint": "https://timestamp.digicert.com"
}
```

## 9. C2PA Integration (Non-Normative)

The Coalition for Content Provenance and Authenticity (C2PA) provides standards for content provenance that complement this specification:

- \* C2PA proves "this content was generated"
- \* This specification proves "this request was blocked"

Generated content MAY include a C2PA manifest with a reference to the corresponding SCITT events:

```
{
  "c2pa:assertions": {
    "scitt:reference": {
      "eventId": "019467a1-...",
      "chainId": "019467a0-...",
      "verificationEndpoint": "https://api.example.com/verify"
    }
  }
}
```

This cross-reference enables verifiers to trace from the content artifact back to the complete audit trail including any prior refusal attempts in the same session.

## 10. IANA Considerations

This document has no IANA actions at this time.

Future revisions may request registration of:

- \* Media type "application/vnd.scitt.refusal-event+json"
- \* Registry for standardized event type values

- \* Registry for risk category codes

## 11. Security Considerations

### 11.1. Threat Model

This specification assumes the following threat model:

- \* The AI system (Issuer) is partially trusted: it is expected to log events but may have bugs or be compromised
- \* The Transparency Service is partially trusted: it provides append-only guarantees but may be compromised or present split views
- \* Verifiers are trusted to perform completeness checks correctly
- \* External parties (regulators, auditors) have access to Receipts and can query the Transparency Service

This specification does NOT protect against:

- \* An AI system that bypasses logging entirely (no ATTEMPT logged)
- \* Collusion between the Issuer and Transparency Service
- \* Compromise of all verifiers

### 11.2. Omission Attacks

An adversary controlling the AI system might attempt to omit refusal events to hide policy violations or, conversely, omit GENERATE events to falsely claim content was refused. The completeness invariant provides detection for logged events: auditors can identify ATTEMPT Signed Statements without corresponding Outcomes. Hash chains detect deletion of intermediate events.

However, if an ATTEMPT is never logged, this specification cannot detect the omission. Complete prevention of omission attacks is beyond the scope of this specification and would require external enforcement mechanisms such as trusted execution environments, RATS attestation, or real-time external monitoring.

The requirement that ATTEMPT be logged BEFORE safety evaluation (pre-evaluation logging) prevents selective logging where only "safe" requests are recorded.

### 11.3. Log Equivocation

A malicious Transparency Service might present different views of the log to different parties (equivocation). For example, it might show auditors a log containing DENY events while providing a different view to other verifiers. Mitigations include:

- \* Gossiping of Signed Tree Heads between verifiers to detect inconsistencies
- \* Registration with multiple independent Transparency Services
- \* External anchoring to public ledgers that provide global consistency
- \* Auditor comparison of Receipts for the same time periods

Detection of equivocation requires coordination between verifiers; a single verifier in isolation cannot detect it.

### 11.4. Split-View Between Event Types

A malicious Issuer might maintain separate logs for refusals and generations, showing only the refusal log to auditors. The completeness invariant mitigates this by requiring every logged ATTEMPT to have an Outcome; if the GENERATE outcomes are hidden, auditors will observe orphaned ATTEMPTs.

### 11.5. Log Tampering

Direct modification of log entries is prevented by cryptographic signatures on Signed Statements, hash chain linking, Merkle tree inclusion proofs in Receipts, and the append-only structure enforced by the Transparency Service.

### 11.6. Replay Attacks

An attacker might attempt to replay old refusal events to inflate refusal statistics or create false alibis. Mitigations include: UUID v7 provides temporal ordering (events with earlier timestamps have smaller UUIDs), timestamps are verified against Transparency Service registration time, and prevHash chaining detects out-of-order insertion or duplicate events.

### 11.7. Key Compromise

If an Issuer's signing key is compromised, an attacker could create fraudulent Signed Statements. Previously signed Signed Statements remain valid. Implementations SHOULD support key rotation and revocation. Transparency Service timestamps provide evidence of when Signed Statements were registered, which can help bound the impact of a compromise.

Gold level conformance requires HSM storage for signing keys, significantly reducing key compromise risk.

### 11.8. Prompt Dictionary Attacks

Although prompts are stored as hashes, an adversary with a dictionary of known prompts could attempt to identify which prompt was used by computing hashes and comparing. Mitigations include access controls on event queries, time-limited retention policies, monitoring for bulk query patterns, and rate limiting.

Salted hashing may provide additional protection but introduces operational complexity. If salting is used, the salt must be managed such that verification remains possible without disclosing the salt to third parties. This specification does not mandate salting.

### 11.9. Denial of Service

An attacker could flood the system with generation requests to create a large volume of ATTEMPT Signed Statements, potentially overwhelming the Transparency Service or obscuring legitimate events. Standard rate limiting and access controls at the AI system level can mitigate this. The Transparency Service MAY implement its own admission controls.

## 12. Privacy Considerations

### 12.1. Harmful Content Storage

This profile requires that harmful content not be stored. Prompt text is replaced with promptHash, reference images are replaced with hashes, and refusal reasons SHOULD NOT quote or describe prompt content in detail. This prevents the audit log from becoming a repository of harmful content.

## 12.2. Actor Identification

Actor identification creates tension between accountability and privacy. Implementations SHOULD use pseudonymous identifiers (ActorHash) by default, maintain a separate access-controlled mapping from pseudonyms to identities, define clear policies for de-pseudonymization, and support erasure of the mapping while preserving audit integrity (crypto-shredding).

## 12.3. Correlation Risks

Event metadata may enable correlation attacks. Timestamps could reveal user activity patterns, SessionIDs link multiple requests, and ModelIDs reveal which AI systems a user interacts with. Implementations SHOULD apply appropriate access controls and MAY implement differential privacy techniques for aggregate statistics.

## 12.4. Data Subject Rights

Where personal data protection regulations apply (e.g., GDPR), implementations SHOULD support data subject access requests, erasure requests via crypto-shredding (destroying encryption keys for personal data while preserving cryptographic integrity proofs), and purpose limitation.

Crypto-shredding architecture:

- \* Sensitive fields encrypted with per-user symmetric keys
- \* Key deletion renders personal data unrecoverable
- \* Hash chain integrity preserved (hashes remain, content inaccessible)
- \* Completeness invariant remains verifiable

## 13. Future Work (Non-Normative)

This section describes potential extensions and research directions that are outside the scope of this specification but may be addressed in future work.

### 13.1. RATS/Attestation Integration

Integration with Remote ATtestation procedures (RATS) [RFC9334] could provide stronger guarantees that the AI system is operating as expected and logging all events. Hardware-backed attestation could reduce the trust assumptions on the Issuer.



### 13.2. Batching and Scalability

High-volume AI systems may generate millions of events per day. Future work could explore batching mechanisms, rolling logs, and hierarchical Merkle structures to improve scalability while maintaining verifiability.

### 13.3. Advanced Privacy Mechanisms

More sophisticated privacy mechanisms could be explored, including:

- \* Commitment schemes that allow selective disclosure
- \* Zero-knowledge proofs for aggregate statistics without revealing individual events
- \* Homomorphic encryption for privacy-preserving audits

These mechanisms would add complexity and are not required for the core auditability goals of this specification.

### 13.4. External Completeness Enforcement

Stronger completeness guarantees could be achieved through external enforcement mechanisms such as:

- \* Trusted execution environments (TEEs) that guarantee logging before generation
- \* Hardware security modules (HSMs) that control signing keys
- \* Real-time monitoring by independent observers
- \* Blockchain-based commitment schemes

These approaches involve significant architectural changes and are outside the scope of this specification.

### 13.5. Post-Quantum Cryptography Migration

The current specification uses Ed25519 signatures which are vulnerable to quantum attacks. Future revisions should address migration to post-quantum algorithms (e.g., ML-DSA/Dilithium) as NIST standards mature. The hashAlgo and signAlgo fields support algorithm agility for this transition.

## 14. References

#### 14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8785] Rundgren, A., "JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020, <<https://www.rfc-editor.org/rfc/rfc8785>>.
- [RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", RFC 9052, August 2022, <<https://www.rfc-editor.org/rfc/rfc9052>>.
- [I-D.ietf-scitt-architecture] Birkholz, H., Delignat-Lavaud, A., and C. Fournet, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-22, 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture-22>>. In RFC Editor Queue as of October 2025
- [I-D.ietf-scitt-scrapi] Steele, O., "SCITT Reference APIs", Work in Progress, Internet-Draft, draft-ietf-scitt-scrapi-06, 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-scitt-scrapi-06>>. In Working Group Last Call as of December 2025

#### 14.2. Informative References

- [RFC6962] Laurie, B., "Certificate Transparency", RFC 6962, June 2013, <<https://www.rfc-editor.org/rfc/rfc6962>>.
- [RFC9334] Birkholz, H., "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.
- [RFC9562] Davis, K., "Universally Unique IDentifiers (UUIDs)", RFC 9562, May 2024, <<https://www.rfc-editor.org/rfc/rfc9562>>.

[CAP-SRP] VeritasChain Standards Organization, "CAP-SRP Reference Implementation", <https://github.com/veritaschain/cap-srp>, 2026.

[CAP-SRP-SPEC] VeritasChain Standards Organization, "CAP-SRP: Content/Creative AI Profile - Safe Refusal Provenance Technical Specification v1.0", <https://github.com/veritaschain/cap-spec>, January 2026.

[VAP-FRAMEWORK] VeritasChain Standards Organization, "Verifiable AI Provenance Framework (VAP) Specification v1.2", <https://veritaschain.org/specs/vap>, January 2026.

[EU-AI-ACT] European Parliament and Council, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act)", Official Journal of the European Union L 2024/1689, July 2024.

[KOREA-AI-ACT] Republic of Korea National Assembly, "Framework Act on Artificial Intelligence (AI)", January 2026. Effective January 22, 2026

## Appendix A. Example: Complete Refusal Event Flow

This appendix illustrates a complete flow from request receipt to Verifiable Refusal Record verification.

### A.1. Event Sequence

1. User submits generation request to AI system
2. AI system creates ATTEMPT Signed Statement (computes promptHash = SHA256(prompt), generates UUID v7 EventId, signs as COSE\_Sign1)
3. AI system registers ATTEMPT with Transparency Service
4. Transparency Service returns Receipt\_ATTEMPT
5. AI system evaluates request against content policy
6. Policy classifier determines refusal is required

7. AI system creates DENY Signed Statement (sets attemptId = ATTEMPT.eventId, records riskCategory and RefusalReason, signs as COSE\_Sign1)
8. AI system registers DENY with Transparency Service
9. Transparency Service returns Receipt\_DENY
10. AI system generates Refusal Receipt for user
11. User receives refusal response with Receipt

#### A.2. Third-Party Verification

An auditor verifying the Verifiable Refusal Record:

1. Obtains ATTEMPT Signed Statement and Receipt\_ATTEMPT
2. Obtains DENY Signed Statement and Receipt\_DENY
3. Verifies Issuer signature on both Signed Statements
4. Verifies both Receipts against Transparency Service public key
5. Confirms DENY.attemptId equals ATTEMPT.eventId
6. Confirms DENY.Timestamp is after ATTEMPT.Timestamp
7. Verifies external anchor if available
8. Concludes: The request identified by ATTEMPT.promptHash was refused and the refusal was logged at DENY.Timestamp

This verification confirms that a refusal was logged, but does not prove that no unlogged generation occurred.

#### A.3. Evidence Pack Verification

A regulator verifying an Evidence Pack:

1. Verify pack signature against published issuer key
2. Verify all checksums in manifest
3. Load all events from events/\*.jsonl files
4. Verify hash chain integrity

5. Verify each event signature
6. Verify Completeness Invariant:  $\text{count}(\text{ATTEMPT}) = \text{count}(\text{GENERATE}) + \text{count}(\text{DENY}) + \text{count}(\text{ERROR})$
7. Verify external anchors against TSA/SCITT services
8. Generate verification report with statistics

#### Acknowledgements

The authors thank the members of the SCITT Working Group for developing the foundational architecture. This work builds upon the transparency log concepts from Certificate Transparency [RFC6962].

The January 2026 Grok incident, while harmful, provided critical motivation for this specification by demonstrating the real-world consequences of unverifiable AI safety claims.

Thanks to the VeritasChain Standards Organization for developing the CAP-SRP specification that informed this Internet-Draft.

#### Author's Address

TOKACHI KAMIMURA  
VeritasChain Standards Organization  
Japan  
Email: [standards@veritaschain.org](mailto:standards@veritaschain.org)  
URI: <https://veritaschain.org>