Internet-Draft
draft-wang-jep-judgment-event-protocol-00
Intended status: Standards Track
Expires: September 19, 2026

Y. Wang
HJS Foundation Ltd.

March 19, 2026

JEP: A Judgment Event Protocol
draft-wang-jep-judgment-event-protocol-00

Abstract

This document defines the Judgment Event Protocol (JEP), an accountability attribution framework for distributed AI systems. JEP provides a responsibility tracing mechanism for judgment behaviors, operating with identity verification (OAuth/DID) and environmental trust (RATS).

Core Design:
- Four-primitive model: $Judge, $Delegate, $Terminate, $Verify, covering the complete judgment lifecycle.
- Three-tier privacy architecture: Public Governance Tier (Tier 1), Logical Accountability Tier (Tier 2), Private Payload Tier (Tier 3), balancing transparency and privacy.
- Minimal core fields: eight fields (verb, who, when, what, based_on, nonce, signature, time_anchor) form the atomic evidence set for accountability.
- Progressive verification strategy: from open mode (local verification) to platform mode (platform-enhanced), adapting to different scenarios.
- Compliance-friendly design: cryptographic erasure mechanism supports GDPR "right to be forgotten" and similar requirements, with specific policies configured via IANA寰喪egistered extensions.

JEP follows a "minimal intrusion, maximum compatibility" philosophy, adding verifiable accountability capabilities to critical judgment nodes without reconstructing AI systems, working with IETF standards including SCITT, RATS, and OAuth.

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 September 19, 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

As AI systems rapidly evolve from isolated black boxes to

interconnected decision networks, a fundamental governance gap has emerged in the Internet protocol stack: the semantic absence of judgment behavior accountability.

While existing infrastructure effectively proves data delivery and platform trust, it was not designed to securely convey judgment attribution across heterogeneous environments寰蚤 gap this protocol addresses.

Consider a representative multi-stage AI medical diagnosis chain:

- Stage 1: Primary screening AI flags potential cardiac anomalies
- Stage 2: Specialist AI confirms diagnosis and recommends treatment based on Stage 1 input
- Stage 3: Clinical Decision Support System (CDSS) generates final report

In post-incident tracing of misdiagnosis cases, audit agencies typically face:

1. Judgment silos: Stage 1 results exist only in private logs, lacking credentials that subsequent stages can reference and cryptographically verify.

2. Accountability chain break: Stage 2 records "external input" but cannot technically prove whether that input was legally delegated.

3. Governance cost: Audit teams spend excessive time reconciling incompatible proprietary log formats, yet still cannot form a closed-loop evidence chain.

The root cause is the absence of a standard, portable judgment event primitive, causing attribution break during cross-platform accountability information flow.

1.2. Related Work

SCITT [draft-ietf-scitt-architecture] provides transparent records for supply chain artifacts, forming an essential foundation for software supply chain security.

VAP [draft-ailex-vap-legal-ai-provenance] provides comprehensive provenance tracking across the AI lifecycle, addressing critical needs for AI transparency and auditability.

DMSC [draft-li-dmsc-inf-architecture] provides infrastructure support for dynamic multi-agent secured collaboration. JEP receipt chains may serve as accountability evidence for DMSC-orchestrated multi-agent workflows, particularly for high-risk AI decisions requiring post-hoc audit.

JEP addresses judgment accountability as a specific concern within the broader AI governance landscape, complementing the above work by focusing on the attribution of judgment behaviors across distributed systems.

1.3. Design Goals

JEP follows an "Atomically Complete" design philosophy (meaning judgment receipts contain the minimal non-divisible evidence set, not distributed transaction atomicity):

- Minimal Integration Cost: The protocol operates as a "governance plane," not requiring AI systems to reconstruct inference logic, only generating specification-compliant receipts at judgment

nodes.

- Semantic Neutrality: The protocol does not model judgment content, only modeling judgment attributes (actor, time, authorization, state).

- Non-repudiability: Judgment effectiveness exists independently of the originating system; auditors can verify responsibility chain integrity without accessing inference environments.

- Progressive Deployment: From minimal configuration (single-signature) to full functionality (three-tier architecture), scaling by requirements and capabilities.

- Scope Delimitation: JEP addresses accountability attribution for judgment events only. It does not provide data lineage, model provenance, training record verification, or end-to-end supply chain tracking. These are valid concerns addressed by other protocols; JEP's scope focuses on judgment accountability to enable minimal integration cost and clear deployment boundaries.

1.4. Relationship to IETF Ecosystem

JEP operates with IETF trustworthy architecture:

- SCITT [draft-ietf-scitt-architecture] may provide timestamp services for JEP receipts.
- RATS [RFC9334] attestations may be recorded in JEP Tier 2 contextual evidence.
- Identity frameworks (OAuth [RFC6749], DID [DID-CORE], AIP [AIP-CORE]) may provide identity verification services for JEP actors.

Table 1: JEP Positioning in Protocol Stack

Existing Layer	Function	JEP Usage
TLS/Transport	Data secure transport	No intervention
OAuth/DID/ Identity	Principal identity verification	Reference identity markers, add behavior responsibility trail
RATS [RFC9334]	Environmental trustworthiness proof	Optional environmental evidence in Tier 2
SCITT [draft-ietf-scitt-arch]	Artifact existence recording	Optional timestamp anchoring
JEP/Governance (New)	Judgment behavior responsibility attribution	Core protocol

1.5. Compatibility with EU Regulatory Frameworks

This protocol was designed with compatibility with major EU data protection and AI regulatory frameworks in mind, particularly:

- GDPR (General Data Protection Regulation) accountability principle, privacy by design, right to be forgotten, and data security requirements.

- EU AI Act (key effective dates in 2026) high-risk AI

transparency obligations and sensitive data processing requirements.

- 2026 Digital Omnibus Act unified requirements for AI training data legitimate interests, bias detection, and incident reporting.

JEP-01 achieves compliance-friendly design through:

- Cryptographic erasure mechanism (Section 7.3): Achieves "compliant forgetting" by destroying decryption keys while retaining auditable "logical tombstones," addressing the right to be forgotten through an approach distinct from append-only ledger technologies.

- Three-tier privacy architecture (Section 3): Tier 1 only exposes rotatable agent IDs, Tier 2 only accessible to auditors, Tier 3 only stores hash digests, naturally conforming to data minimization and privacy by design principles.

- TEE hardware binding (Section 5.3): Transforms hardware trust roots into logical accountability, meeting high-risk AI system security trust requirements.

- Configurable jurisdiction marking (Appendix G.5.1): Supports configuring retention periods and localization strategies according to GDPR, CCPA, PIPL, and other regulations.

Deployers can achieve more granular compliance proof through Appendix G compliance extensions. The cryptographic erasure mechanism of this protocol references ENISA "Data Protection Engineering" reports and EDPB "Right to be Forgotten Guidelines" in its design, and is designed to provide verifiable evidence of compliance with erasure obligations, though final recognition depends on specific jurisdictional interpretation.

Note that compliance ultimately depends on deployment configuration and applicable legal interpretation; this protocol provides technical mechanisms to support, but does not guarantee, regulatory compliance.

1.6. Quick Start Guide

For readers new to JEP, this section provides a high-level overview to help understand the protocol’s core concepts and minimal implementation steps.

- Core Idea: JEP produces portable, cryptographically verifiable receipts that answer "who made what judgment, when, and based on what authority."

- Eight Core Fields: Every receipt contains exactly eight fields: verb, who, when, what, based_on, nonce, signature, time_anchor. These are the atomic evidence elements.

- Three Tiers: The fields are organized into three privacy tiers: Tier 1 (public), Tier 2 (audit-restricted), Tier 3 (private). This ensures minimal disclosure.

- Minimal Implementation: A basic JEP implementation can be built in 200寰00 lines of code, generating and verifying Judge receipts using SHA-256 and Ed25519. See Appendix E for pseudocode.

- Progressive Adoption: Start with simple receipts, then add chain verification, then enable extensions as needed.

The rest of this document provides detailed specifications for all aspects of the protocol.

2. Terminology and Conventions

2.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD 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.

2.2. Core Concepts

2.2.1. Judgment

An atomic declaration made by a judgment actor in a specific context, possessing accountability effectiveness. It represents the actor’s logical commitment to that declaration, not the absolute truth of the fact.

2.2.2. Receipt

A cryptographically verifiable judgment event record following JEP specification, serving as the minimal data unit for accountability tracing.

2.2.3. Judgment Actor

An entity (human or AI agent) initiating judgment behavior, whose identity is typically endorsed by hardware trust root.

2.2.4. Authorization Chain

A traceable path recording the flow of accountability sovereignty from mandator to delegatee, formed through parent receipt references.

2.2.5. Cryptographic Erasure

A technical mechanism rendering receipt ciphertext computationally unreadable by destroying decryption keys, used to satisfy compliance requirements such as GDPR "right to be forgotten."

2.3. Core Concepts Quick Reference

Table 2 provides a quick reference to the core fields and their roles.

Field	Tier	Necessity	Description
verb	1	Mandatory	Operation type
who	1	Mandatory	Actor identifier
when	1 or 2	Mandatory	Timestamp
what	3	Mandatory	Content hash
based_on	2	Mandatory	Parent receipt hash
nonce	1	Mandatory	Replay prevention
signature	(covers all)	Mandatory	Authenticity proof
time_anchor	2	Recommended	Time anchoring reference

3. Three-Tier Privacy Architecture

3.1. Design Principles

The three-tier privacy decoupled architecture achieves atomic completeness of four elements through physical and logical dual isolation, ensuring accountability transparency while maintaining individual privacy.

3.2. Public Governance Tier (Tier 1)

- Constituent Element: Identity Anchor
- Function: Proves "who" initiated the operation
- Technical Implementation: Based on DID, OAuth, or other interoperable identity frameworks, carrying judgment actor agent identity markers
- Privacy Property: Public. Used for global transparent auditing, replay attack prevention, and notary service mapping. This tier exposes only agent identity (AI Agent ID); human operator identity is protected through authorization chain limited disclosure in Tier 2 or privacy-preserving DID.

3.3. Logical Accountability Tier (Tier 2)

- Constituent Elements: Authorization Chain, Contextual Evidence
- Function: Proves "based on what legitimacy" and "in what environment" the judgment was made
- Technical Implementation:
  * Authorization Chain: Contains parent receipt reference, forming traceable responsibility chain, recording sovereignty flow path from mandator to delegatee
  * Contextual Evidence: Contains encrypted timestamp and environmental security measurements (e.g., RATS hardware proof or TEE measurement data)
- Privacy Property: Limited disclosure. Visible only to entities with audit permissions, ensuring auditors can verify responsibility flow path and judgment environment trustworthiness without exposing judgment content.

3.4. Private Payload Tier (Tier 3)

- Constituent Element: Logic Payload
- Function: Anchors judgment substantive content, ensuring results were not tampered with
- Technical Implementation: Encrypted hash digest of judgment content, with original text protected by out-of-band encrypted storage or independent keys
- Privacy Property: Strictly private. Receipt stores only content digest. During cryptographic erasure execution, only destroying keys associated with this tier achieves compliant "forgetting" without breaking Tier 1 and Tier 2 governance structure integrity.

3.5. Tier Relationships

Table 3: Four Elements Privacy Mapping

Element	Privacy Tier	Visibility	Core Function
Identity Anchor	Tier 1	Public	Establish responsibility principal
Authorization Chain	Tier 2	Limited	Trace responsibility flow
Contextual	Tier 2	Limited	Verify judgment

Evidence			environment
Logic Payload	Tier 3	Strict private	Anchor judgment content

3.6. Mapping to Core Fields

The three-tier architecture provides a natural organization for the core fields that constitute the minimal evidence set for accountability (see Section 4.4.1). Each core field is placed in the tier that corresponds to its intended visibility:

Core Field	Assigned Tier	Rationale
verb	Tier 1	Operation primitive MUST be publicly visible for basic verification.
who	Tier 1	Actor identity MUST be public to establish responsibility.
when	Tier 1 or 2	Timestamp MAY be public (Tier 1) or protected (Tier 2) depending on deployment needs.
what	Tier 3	Content hash MUST be private to protect the actual decision data.
based_on	Tier 2	Parent receipt hash is required for audit but need not be public.
nonce	Tier 1	Unique identifier MUST be public to prevent replay attacks.
signature	(covers all)	Signature covers the entire canonicalized core fields and is part of the JWS signature.
time_anchor	Tier 2	Time anchor reference is needed for audit but need not be public.

This mapping ensures that every JEP receipt, regardless of extensions, adheres to the same privacy and accountability principles. The core fields are immutable and mandatory; extensions may add fields to any tier but MUST NOT alter the semantics of the core.

3.7. Field Placement Rules

Implementations MUST follow these placement rules when constructing a JEP receipt:

- Fields assigned to Tier 1 MUST appear in the JWS Protected Header.
- Fields assigned to Tier 2 and Tier 3 MUST appear in the JWS Payload.
- The ‘when‘ field MAY be placed in either Tier 1 or Tier 2. If placed in Tier 2, a corresponding ‘time_anchor‘ SHOULD be provided to ensure timestamp integrity.
- Extensions MAY add fields to any tier, but MUST NOT alter the placement of core fields.

These rules guarantee that any JEP implementation can correctly parse and verify receipts without ambiguity.

4. JEP Protocol Specification

4.1. Four Primitives

JEP defines the complete judgment lifecycle governance through four core primitives.

4.1.1. $Judge Primitive

Judgment initiation primitive. Used to create new judgment events, encapsulating principal declarations into initial receipts.

- Initiated by judgment actor
- Establishes Tier 1 identity anchor and initial sovereignty
- Generates initial receipt with no parent reference
- Post-state: ACTIVE

4.1.2. $Delegate Primitive

Delegation primitive. Used to execute responsibility transfer, recording accountability sovereignty flow path from mandator to delegatee.

- Phase One (Initiation): Mandator generates delegation proposal, state becomes DELEGATING
- Phase Two (Acceptance): Delegatee generates acceptance signature, state becomes DELEGATED
- Timeout Rollback: If acceptance signature not received within specified time, automatic rollback to ACTIVE
- Supports multi-level delegation
- Pre-state MUST be ACTIVE or DELEGATED
- Mandator MUST exist in current Tier 2 authorization chain

4.1.3. $Terminate Primitive

Termination primitive. Used to formally declare the end of judgment accountability lifecycle. Signed by authorized principal, ensuring termination behavior itself is non-repudiable. Supports "archived state" and "cryptographic erasure" modes.

- ACTIVE state: Judgment actor MAY directly terminate, generating termination receipt
- DELEGATED state: Mandator and Delegatee MUST execute dual signature to achieve closed-loop consensus, preventing unilateral responsibility evasion
- DELEGATING state: Mandator MAY unilaterally cancel incomplete delegation, state rolls back to ACTIVE
- Termination mode marking: ARCHIVED (retain all three tiers) or ERASED (erase Tier 3)
- Post-state: TERMINATED_ARCHIVED or TERMINATED_ERASED

4.1.4. $Verify Primitive

Verification primitive. Non-state-changing operation for integrity, authorization legitimacy, and timeliness verification of receipts and their responsibility chains. Executable at any point during active period and post-termination audit period.

- Probe Mode: Immediate feedback verification result, no trace generated
- Audit Mode: Generates verification trace record, produces Verify receipt

4.2. State Machine

4.2.1. State Definitions

Table 4: Judgment Lifecycle States

State	Meaning
INITIATED	Initial state, no receipt. Supports pre-authorization and bootstrap scenarios.
ACTIVE	Judgment created, responsibility not yet transferred.
DELEGATING	Delegation initiated, awaiting Delegatee acceptance signature.
DELEGATED	Valid acceptance signature received, responsibility transferred to Delegatee.
TERMINATED_ARCHIVED	Terminated state, all three tiers retained for long-term compliance audit.
TERMINATED_ERASED	Terminated state, Tier 3 logic payload cryptographically erased (satisfying right to be forgotten).
DISPUTED_FROZEN	Dispute detected, chain frozen pending resolution
RESOLVED_CONTINUE	Dispute resolved, chain operational with resolution mark
RESOLVED_INVALID	Dispute resolved by invalidation, re-initiation required
VERIFIED	Meta-state indicating logical chain legitimacy confirmed (read-only result).

4.2.2. State Transition Rules

$Judge Transition:
- Initiated by judgment actor, establishes Tier 1 identity anchor and initial sovereignty
- Generates initial receipt, no parent reference
- Post-state: ACTIVE

$Delegate Transition:
- Phase One (Initiation): Mandator generates delegation proposal, state becomes DELEGATING
- Phase Two (Acceptance): Delegatee generates acceptance signature, state becomes DELEGATED
- Timeout Rollback: If acceptance signature not received within specified time, automatic rollback to ACTIVE
- Supports multi-level delegation
- Pre-state MUST be ACTIVE or DELEGATED
- Mandator MUST exist in current Tier 2 authorization chain

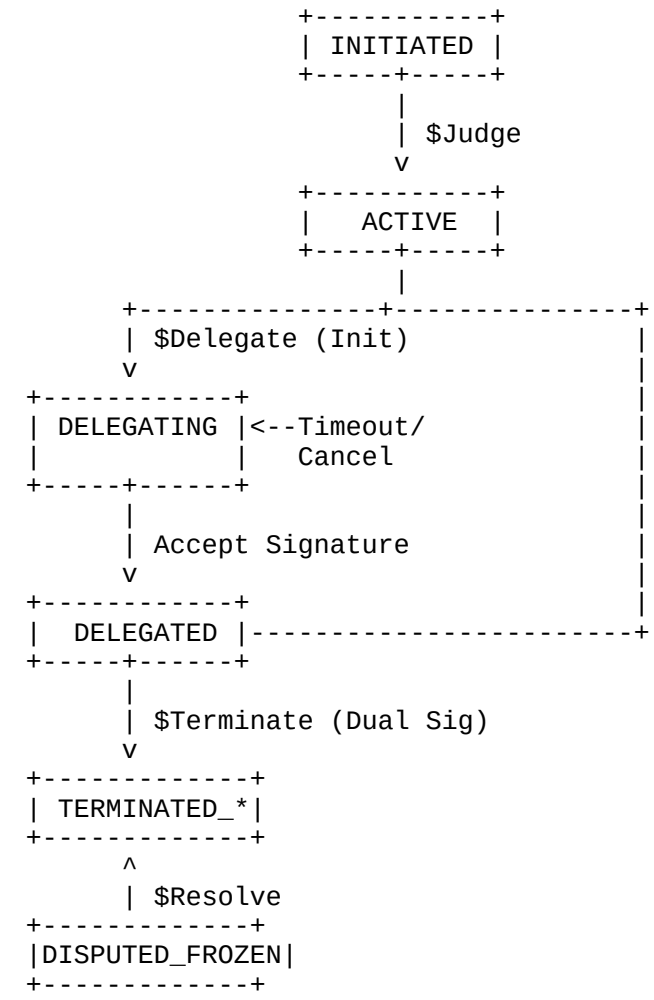
$Terminate Transition:
- ACTIVE state: Judgment actor MAY directly terminate
- DELEGATED state: Mandator and Delegatee MUST execute dual signature
- DELEGATING state: Mandator MAY unilaterally cancel, state rolls back to ACTIVE
- Termination mode marking: ARCHIVED or ERASED
- Post-state: TERMINATED_ARCHIVED or TERMINATED_ERASED

$Verify Execution:
- Non-state-changing operation
- Probe Mode: Immediate feedback, no trace
- Audit Mode: Generates verification trace

$Resolve Transition (see Section 4.5.5):
- Pre-state: DISPUTED_FROZEN
- Post-state: RESOLVED_CONTINUE or RESOLVED_INVALID

4.2.3. Delegation Flow

Figure 1: Delegation State Machine



4.3. Event Format

4.3.1. Common Fields

All JEP events MUST contain the following fields:

- jep_version: Protocol version (e.g., "1.0")
- event_id: Globally unique event identifier (UUID [RFC9562])
- event_type: Event type (Judge, Delegate, Terminate, Verify, Resolve)
- actor_id: Judgment actor identifier (typically DID)
- timestamp: Unix timestamp (seconds since epoch)
- prev_hash: Previous receipt hash (null for initial)
- primitive: Primitive-specific data
- proof: Cryptographic proof (signature)

4.3.2. Lifecycle Events

Judge Event:

- decision_hash: Hash of judgment content
- authority_scope: Authority scope URI
- valid_from/valid_until: Validity period
- state: "A" (ACTIVE)

Delegate Event:
- parent_receipt: Parent receipt reference
- delegatee_id: Delegatee identifier
- acceptance_signature: Delegatee acceptance (required for completion)
- delegation_scope: Delegated authority scope

Terminate Event:
- termination_mode: "ARCHIVED" or "ERASED"
- pkd: Proof of Key Destruction (for ERASED mode)
- consensus_signatures: Dual signatures (for DELEGATED state)

Resolve Event:
- disputed_receipt: ID of disputed receipt
- resolution_type: "RESOLVED_CONTINUE" or "RESOLVED_INVALID"
- resolution_statement: Human-readable explanation
- governance_signatures: Multi-signature from governance authority
- new_anchor: ID of new anchor receipt (for RESOLVED_INVALID)

4.3.3. Verify Events

- target_receipt: Target receipt ID
- verification_mode: "open", "platform", or "dual"
- verification_result: "VALID", "INVALID", or "DISPUTED"
- verification_timestamp: Verification time
- verifier_id: Verifier identifier

4.4. Receipt Format

4.4.1. Core Fields

Every JEP receipt MUST contain the following core fields, which constitute the minimal nondivisible evidence set for accountability attribution. These fields are immutable across protocol versions and MUST be implemented by all conforming implementations.

Field	Description
verb	Operation primitive: "J" (Judge), "D" (Delegate), "T" (Terminate), "V" (Verify).
who	Actor identifier URI (typically a DID) that uniquely identifies the judgment actor.
when	Unix timestamp (seconds since epoch) when the judgment was made.
what	Hash of the judgment content (e.g., SHA-256 of the decision payload). The content itself is stored externally as per Tier 3 of the privacy architecture.
based_on	Hash of the parent receipt (null for initial judgments). This provides chain integrity and links to the authorization source.
nonce	Globally unique identifier (UUID [RFC9562]) to prevent replay attacks.
signature	Digital signature over the canonical JSON

		representation of all preceding core fields, using the algorithm indicated in the JWS header.
time_anchor		(Optional but RECOMMENDED) Reference to an external time anchoring service (e.g., SCITT entry ID, blockchain transaction ID) that cryptographically proves the timestamp's integrity.

These eight fields answer the fundamental questions of accountability: who (who), when (when), what (what), why (verb), based on what (based_on), how to prevent replay (nonce), how to prove authenticity (signature), and how to secure time (time_anchor). Any fewer fields would leave a critical gap in the accountability chain.

4.4.2. JWS Serialization

JEP receipts use JWS Compact Serialization [RFC7515]:

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature)
```

Protected Header (Tier 1 - Public):
- alg: Signature algorithm (e.g., "EdDSA")
- typ: "JEP-Receipt"
- jep_v: JEP version
- jep_aid: Actor ID (maps to the core field 'who')
- jep_op: Operation code (maps to the core field 'verb')
- jep_rid: Receipt ID (maps to the core field 'nonce')
- (Optional) Additional Tier 1 fields from extensions

Payload (Tier 2 & 3):
- Contains the remaining core fields: when, based_on, what, time_anchor
- May also contain Tier 2 and Tier 3 extension fields
- The payload MAY be encrypted (e.g., using JWE [RFC7516]) to protect Tier 2 and Tier 3 data.

Signature: Covers the entire JWS Protected Header and Payload.

4.4.3. Hash Computation

Receipt canonicalization MUST use JCS (JSON Canonicalization Scheme) [RFC8785] before hashing.

Hash algorithm: SHA-256 [RFC6234] (MUST be supported)
Alternative: SHA3-256 (MAY be supported)

The hash for the 'based_on' field is computed as the SHA-256 of the canonicalized parent event (without its proof field).

4.4.4. Verification Algorithm

1. Parse JWS structure
2. Verify protected header signature using actor's public key
3. Decrypt payload (if required for verification depth)
4. Verify parent hash chain integrity (check that the computed hash of the parent event matches based_on)
5. Verify authorization chain signatures (if present in Tier 2)
6. Verify timestamp within tolerance window (using time_anchor if present)
7. Return verification result

4.4.5. JWS Construction Example

Below is a concrete example of a Judge receipt serialized as JWS.

For readability, the header and payload are shown decoded; actual receipts use base64url encoding.

Header (Tier 1):

```
{
  "alg": "EdDSA",
  "typ": "JEP-Receipt",
  "jep_v": "1.0",
  "jep_aid": "did:example:agent123",
  "jep_op": "J",
  "jep_rid": "f47ac10b-58cc-4372-a567-0e02b2c3d479"
}
```

Payload (Tier 2 & 3):

```
{
  "when": 1712345678,
  "based_on": null,
  "what": "sha256:3a7bd3e2360a3d29eea436fcfb7e44c735d117c42d1c1835420b6b9942dd4f1b",
  "time_anchor": "scitt:entry:123456"
}
```

The signature is computed over the canonicalized concatenation of the header and payload.

4.5. Verification Modes

To accommodate different trust assumptions and performance requirements, JEP supports three verification modes:

Table 5: Verification Modes

Mode	Verification Content	Latency	Output
open	Local cryptographic proof (hash chain + signature)	<1ms	VALID/ INVALID
platform	Platform signature + local proof	<10ms	VALID/ INVALID
dual	Simultaneous platform and open verification	<100ms	VALID/ INVALID/ DISPUTED

4.5.1. open Mode

- Pure local cryptographic verification
- No external dependencies
- Suitable for cross-platform offline audit
- Minimal trust assumption: only cryptographic primitives

4.5.2. platform Mode

- Platform signature + local proof
- Suitable for internal monitoring, high-trust environments
- Higher performance than dual mode
- Trust assumption: platform not compromised

4.5.3. dual Mode

- Simultaneous platform and open verification
- Suitable for critical decisions, intrusion detection
- Conflict detection capability

4.5.4. DISPUTED State Handling

DISPUTED state occurs when platform verification passes but open verification fails (or vice versa), indicating:

- Platform may be compromised (signature key leaked)
- Implementation bug (hash chain calculation error)
- Network partition causing timestamp inconsistency

Handling RECOMMENDED:
1. Trigger manual review
2. Freeze related responsibility chain
3. Initiate emergency response

4.5.5. Dispute Resolution and Recovery

JEP provides a structured recovery path from DISPUTED state through authorized third-party intervention. This is defined in the Resolution Event extension (see Appendix G.2.2).

Resolution types:
- "RESOLVED_CONTINUE": Dispute resolved, chain MAY continue normal operation
- "RESOLVED_INVALID": Dispute resolved by invalidating disputed receipts, chain MUST be re-initiated from last known good state

Resolution receipts MUST include governance signatures meeting configured threshold.

4.6. Error Handling

Table 6: JEP Error Codes

Error Code	Scenario
INVALID_SIGNATURE	Signature verification failed
BROKEN_CHAIN	Parent reference broken or tampered
UNAUTHORIZED_DELEGATION	Delegating principal not in authorization chain
INVALID_TERMINATION	Insufficient termination permission (e.g., missing dual signature in DELEGATED state)
PAYLOAD_ERASED	Tier 3 payload compliance-erased, only PKD provided
CHAIN_TOO_DEEP	Delegation level exceeds limit (10 levels)
EXPIRED_RECEIPT	Receipt timestamp outside trusted window
DISPUTED	platform and open verification conflict (dual mode)
RESOLUTION_REQUIRED	Governance resolution applied, chain status changed per resolution type

4.7. Time Trust Anchor Mechanism

To mitigate risks from single time source failures or NTP deviations, JEP implementations MAY implement a Time Trust Anchor (TTA) mechanism using multiple independent time sources.

TTA-enabled implementations MUST configure at least two of the following time sources:
- Local NTP with authenticated NTS (Network Time Security)
- SCITT transparency service timestamp (if SCITT usage enabled)
- Hardware-backed trusted clock (e.g., TPM RTC, TEE secure clock)

When multiple time sources are configured, implementations MUST

apply the following consensus algorithm:
1. Collect timestamps from all available sources
2. Discard outliers exceeding 2x the configured tolerance window
3. Select the median timestamp as authoritative
4. If sources disagree beyond tolerance window, enter SUSPECTED state and trigger manual review

Example Deployment:
- In a sovereign cloud, configure:
  - NTP from national time service (e.g., PTB in Germany)
  - SCITT anchoring on a national blockchain
  - TPM hardware clock for backup
- Tolerance: 5 seconds
- If SCITT is unavailable, fallback to NTP + hardware clock

If all configured sources become unavailable, implementations MAY use a local monotonic clock but MUST mark the receipt as "time_unverified" in the verification result.

5. Extensions and Integration

5.1. Extension Mechanism

JEP supports protocol extensions through IANA registration. Extensions MUST NOT violate core protocol semantics. All extensions are optional and independent of the core 8 fields. Extensions are organized by recommendation level for different deployment scenarios. Implementers may choose any combination of extensions based on their requirements; the following grouping is advisory only.

For new deployments, it is RECOMMENDED to implement at least the Priority 1 extensions to benefit from enhanced security and interoperability. See Section 5.6 for suggested extension sets for common scenarios.

5.2. SCITT Usage

SCITT [draft-ietf-scitt-architecture] may provide timestamp services for JEP receipts. This is optional; JEP verification does not require SCITT integration.

5.3. RATS Usage

RATS [RFC9334] attestations may be recorded in Tier 2 contextual evidence. This is optional. TEE evidence encoding formats are defined in Appendix G.2.1.

5.4. Identity Framework Usage

Table 7: Identity Framework Mapping

Identity Standard	Tier 1 Mapping	Tier 2 Mapping
DID	jep_aid	Authorization chain DID resolution
OAuth 2.0	jep_aid + token ref	Token scope verification
AIP	jep_aid + capability declaration	Agent capability chain

X.509	Certificate fingerprint	Certificate chain verification path

5.5. Extension Specification Requirements

JEP extensions MUST comply with the following mandatory specification requirements:

- semantics: Textual description of behavioral changes introduced
- compatibility: "full", "wire", or "none"
- version: Semantic version string
- requires: Array of prerequisite extension identifiers, if any

Extensions MUST be registered via the IANA registry and MUST NOT be deployed in production prior to registration completion.

5.6. Recommended Extension Combinations

To help implementers choose an appropriate set of extensions, the following combinations are recommended for common deployment scenarios:

- Minimal (IoT / Edge): Core fields + Lightweight Mode (G.3.1)
- Standard (Internal audit): Core + P1 (TEE evidence, multi-source time)
- Compliance (EU GDPR / AI Act): Core + P1 + P4 (jurisdiction, GDPR, AI Act)
- High-Security (Financial): Core + P1 + P3 (enhanced hardware binding)
- Cross-Border Governance: Core + P1 + P5 (federal resolution)

These are recommendations only; implementers may adapt based on specific requirements.

6. Security Considerations

6.1. Threat Model

JEP security design assumes a semi-trusted distributed network environment. The protocol MUST defend against:

- Insider Attackers: Legitimate judgment actors may maliciously delegate to evade responsibility, or exploit controlled environments to forge/tamper contextual evidence.

- External Attackers: Third parties may intercept and replay legitimate receipts, or perform traffic analysis through Tier 1 public metadata to infer judgment behavior frequency and business logic.

- Computational Threats: With quantum computing development, existing elliptic curve-based digital signatures may face security degradation or instant compromise.

Trust Assumptions:
- Judgment actors are responsible for private key security
- Time sources are partially trusted (within tolerance window)
- Optional anchoring services (e.g., SCITT) are trusted

6.2. Key Management

- Signature keys MUST be stored in hardware trust root (HSM/TEE)
- Rotation period: RECOMMENDED 1 year
- Compromise response: Immediate $Terminate with logical invalidation

- Encryption keys (Tier 3) MAY be stored in external KMS

- Erasure execution: Deterministic destruction with PKD generation

6.3. Cryptographic Mechanisms

- Ed25519: High performance, high security, constant-time execution
- SHA-256: Broad support, sufficient security margin
- ECDSA with P-256: Alternative for FIPS 140-2 compliance
- SM2: Chinese commercial cryptography compliance

Algorithm agility: JWS Protected Header alg field supports future migration.

6.4. Attack Analysis

Table 8: Attack Matrix

Attack Type	Description	Mitigation
Signature Forgery	Attacker forges Ed25519 signature	Key in HSM/TEE
Hash Collision	Find different judgments with same hash	Use SHA-256
Replay Attack	Reuse old receipts	UUID+timestamp window
Man-in-the-Middle	Tamper with receipts in transit	TLS+signature
Denial of Service	Construct deep delegation chains	10-level depth limit
Time Tampering	Forge judgment timestamps	SCITT anchoring (optional)
Responsibility Dumping	Mandator unilateral delegation without consent	Mandatory dual-signature

6.5. Post-Quantum Migration

JEP Tier 1 Header alg field reserves algorithm agility for migration to post-quantum signatures (Dilithium, Falcon, or SPHINCS+). Ed25519 remains secure until NIST post-quantum standards are widely deployed. Implementations SHOULD monitor NIST standardization progress.

For high-security environments requiring protection against "harvest now, decrypt later" attacks, hybrid signature schemes (e.g., Ed25519 + Dilithium) are RECOMMENDED during the transition period. Hybrid signatures can be encoded by using a composite algorithm identifier (e.g., "Ed25519-Dilithium3") and including both signatures in the JWS signature field concatenated or structured as defined by the composite scheme. Implementations SHOULD verify at least one of the signatures according to their security policy.

Once NIST finalizes post-quantum standards, JEP will update the recommended algorithm set accordingly. The core protocol remains unaffected by algorithm changes.

7. Privacy Considerations

7.1. Privacy Risk Analysis

- Observability Risk: Tier 1 actor ID may enable long-term tracking
- Disclosure Risk: Tier 2 authorization chain may expose organizational internal structure
- Identification Risk: Behavior patterns may infer real identity
- Aggregation Risk: Multiple receipt correlation may reconstruct complete business graph

7.2. Data Minimization

Receipts MUST NOT contain non-essential fields. All fields serve explicit accountability purposes. The three-tier architecture ensures that only the minimal necessary information is exposed to each party.

7.3. Cryptographic Erasure Mechanism

Execution $Terminate (ERASED mode) leaves "logical tombstones" in responsibility chain:

- Tier 3 payload content computationally unrecoverable through key destruction
- Retains undecryptable hash anchor
- Preserves Tier 1 (public governance) and Tier 2 (accountability path)

This design proves "compliant erasure obligation fulfilled" during regulatory audit while retaining minimal accountability evidence. For guidance on interacting with regulators and demonstrating compliance, see Appendix M.

7.4. Deployment Recommendations

- Public Deployment: Use ephemeral DID, rotation period RECOMMENDED 24 hours
- Private Deployment: MAY use fixed DID, relying on network isolation
- Cross-border Deployment: Consider data localization requirements

7.5. EU Regulatory Compliance Analysis

JEP was designed with compatibility with EU major regulatory frameworks in mind. See Appendix J for a detailed compliance quick reference guide.

8. IANA Considerations

8.1. Media Type Registration

IANA is requested to add the following entry to the "Media Types" registry:

Type name: application
Subtype name: jep+json
Required parameters: N/A
Optional parameters: charset (default is UTF-8)
Encoding considerations: Follows RFC 8259 (JSON)
Security considerations: See Section 6 of this document
Interoperability considerations: JEP receipts support JWS Compact Serialization (RFC 7515)
Published specification: This document
Applications that use this media type: Judgment Event Protocol implementations, AI governance systems, compliance audit platforms
Fragment identifier considerations: N/A

Additional information: None
Person and email address to contact for further information:
Yuqiang Wang <signal@humanjudgment.org>
Intended usage: COMMON
Restrictions on usage: None
Author: Yuqiang Wang
Change controller: IETF

8.2. JWS Header Parameters Registration

IANA is requested to add the following entries to the "JSON Web Signature and Encryption Header Parameters" registry per RFC 7515 Section 9.1:

Parameter Name	Description	Header Parameter Location
jep_v	JEP protocol version (e.g., "1.0")	JOSE Header
jep_aid	Actor identifier URI (maps to core field ‘who‘)	JOSE Header
jep_op	Operation code: "J", "D", "T", "V" (maps to ‘verb‘)	JOSE Header
jep_rid	Receipt identifier (UUID, maps to ‘nonce‘)	JOSE Header

Change controller: IETF
Specification document(s): This document, Section 4.4.2

8.3. JEP Error Code Registry

IANA is requested to create a new sub-registry named "Judgment Event Protocol Error Codes" under "Protocol Registries".

Registration procedure: Specification Required (RFC 8126)
Change controller: IETF

Initial registrations: See Table 6 in Section 4.6.

8.4. JEP Extension Registry

IANA is requested to create a new sub-registry named "JEP Extensions" under "Protocol Registries".

Registration procedure: Specification Required (RFC 8126)
Change controller: IETF

Registry fields:
- Extension Identifier: Unique URI (e.g., urn:ietf:jep:ext:NAME)
- Semantics Description: URL to specification document
- Compatibility Level: "full" | "wire" | "none"
- Version: Semantic version string
- Registration Date: ISO 8601 date
- Contact: Email of registrant

Initial registrations: See Appendix G for the initial set.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, May 2015.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, May 2015.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, June 2017.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020.

[RFC9334] Birkholz, H., et al., "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, January 2023.

[RFC9562] Davis, K., et al., "Universally Unique IDentifiers (UUIDs)", RFC 9562, May 2024.

9.2. Informative References

[HJS-00] Wang, Y., "HJS: A Judgment Event Protocol", draft-wang-hjs-judgment-event-00, March 2026.

[draft-ietf-scitt-architecture] Birkholz, H., et al., "Supply Chain Integrity, Transparency, and Trust (SCITT) Architecture", Work in Progress.

[DID-CORE] Sporny, M., et al., "Decentralized Identifiers (DIDs) v1.0", W3C Recommendation, July 2022.

[GDPR-2016] European Parliament, "General Data Protection Regulation (GDPR)", 2016.

[EU-AI-ACT] European Commission, "Artificial Intelligence Act", 2024.

[draft-ailex-vap-legal-ai-provenance] Ailin, A., et al., "Verifiable AI Provenance (VAP) Framework", Work in Progress.

[draft-li-dmsc-inf-architecture] Li, Y., Wang, A., et al., "Dynamic Multi-agent Secured Collaboration (DMSC) Infrastructure Architecture", Work in Progress.

[AIP-CORE] Agent Identity Protocol Working Group, "Agent Identity Protocol (AIP) Core Specification", 2025, <https://aip.dev/>.

[ENISA-DPE] European Union Agency for Cybersecurity, "Engineering Personal Data Sharing - How to support GDPR compliance

through technological tools", ENISA Report, 2023.

[EDPB-RTBF] European Data Protection Board, "Guidelines 5/2019 on the criteria of the Right to Be Forgotten in the search engines cases under the GDPR (Version 2.0)", 2020.

Appendix A. Performance Benchmarks

This section summarizes performance benchmarks from JEP Reference Implementation v1.0. Tests conducted on:

Constrained Node:
- CPU: Intel Xeon E5-2680 v4 @ 2.4 GHz (single-core)
- Memory: 512MB DDR4-2400
- Storage: SATA SSD

Standard Server:
- CPU: Intel Xeon Gold 6248R @ 3.0 GHz (16 cores)
- Memory: 64GB DDR4-2933
- Storage: NVMe SSD

Results:
- $Judge generation latency: average 4.5 ms
- Throughput: 12,000+ receipts/second (16-core)
- open mode verification: <1 ms
- platform mode verification: <10 ms
- dual mode verification: <100 ms
- 10-level chain verification: median 120 ms (asynchronous)
- Receipt size: 1.2 - 2.5 KB (core only), up to 5 KB with extensions

Appendix B. Dispute Handling Framework

When $Verify produces disputed results, the following multi-tier handling flow is RECOMMENDED:

B.1. Logical Degradation: If receipt is TERMINATED_ERASED, accept PKD as legitimate explanation for Tier 3 absence.

B.2. Conflict Detection: If dual mode returns DISPUTED:
1. Immediate freeze of responsibility chain
2. Root cause analysis
3. Emergency response

B.3. Evidence Re-evaluation: If TEE binding fails, trigger SUSPECTED mode and re-evaluate RATS measurements.

B.4. Sovereignty Traceback: Trace through Tier 2 authorization chain to initial human principal for legal attribution.

Appendix C. Implementation Risks and Mitigations

Risk Category	Risk Description	Mitigation
Clock Drift	Node desynchronization causing EXPIRED_RECEIPT	NTS sync, tolerance ±5s
State Inconsistency	Asynchronous $Terminate propagation delay	Backward probing
Key Loss	Actor loses private key	Emergency termination by parent

Storage Explosion	Massive receipt chains	Tombstone compression
Verification Mode Misuse	Misusing platform mode in high-risk scenarios	Dual mode for critical ops

Appendix D. Compatibility with HJS-00

HJS-00 Feature	JEP-01 Status	Migration Notes
Four primitives	Fully retained	No modification required
platform/open/ dual modes	Fully retained	Semantically consistent
Pending state	Clarified as DELEGATING	Clearer state machine
Acceptance sig	Strengthened as dual-sign	Stricter security
10-level depth	Retained	From recommendation to SHOULD-level
Three-tier arch	Evolved from LPV/SEN/GEI	Clearer concepts, functional consistency

Compatibility Commitment: All HJS-00 receipts are verifiable in JEP-01; JEP-01 new features are optional extensions.

Appendix E. Implementation Guide

E.1. Reference Implementation

Available at https://github.com/hjs-spec/jep-reference-impl (Rust, Apache-2.0)

E.2. Minimal Viable Implementation (Pseudocode)

```python
class Receipt:
    def __init__(self, event_id, actor, decision_hash):
        self.event_id = event_id
        self.actor = actor
        self.decision_hash = decision_hash
        self.prev_hash = None
        self.signature = None
    def sign(self, private_key):
        data = self.canonicalize()
        self.signature = sign(private_key, data)
    def verify(self, public_key):
        data = self.canonicalize()
        return verify(public_key, data, self.signature)
    def canonicalize(self):
        return jcs(self.to_dict())
```

E.3. Implementation Checklist

- [ ] All four primitives implemented
- [ ] State machine (8 states) supported
- [ ] Three verification modes
- [ ] Chain depth limit (10 levels)
- [ ] Dual-signature enforcement

```
        - [ ] Cryptographic erasure (optional)
        - [ ] IANA registry usage
```

Appendix F. Test Vectors

F.1. Judge Receipt Example

```
‘‘‘json
{
  "hjs": "1.0",
  "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "actor": "did:example:agent123",
  "decision_hash": {
    "hash": "3a7bd3e2360a3d29eea436fcfb7e44c735d117c42d1c1835
             420b6b9942dd4f1b",
    "alg": "sha256"
  },
  "authority_scope": "https://hospital.example/policy/diagnosis/v1",
  "valid": {"from": 1712345678, "until": 1712349278},
  "state": "A",
  "prev_hash": null,
  "primitive": {"type": "J"},
  "proof": {"signature": "base64...", "alg": "Ed25519"}
}
‘‘‘
```

F.2. Delegate Receipt Example

```
(Assuming parent receipt ID "f47ac10b-...")
‘‘‘json
{
  "hjs": "1.0",
  "id": "a1b2c3d4-...",
  "actor": "did:example:agent123",
  "decision_hash": {...},
  "authority_scope": "...",
  "valid": {...},
  "state": "D",
  "prev_hash": "sha256:3a7bd3e2360a3d29eea436f...",
  "primitive": {
    "type": "D",
    "payload": {
      "to": "did:example:agent456",
      "acceptance_proof": "base64...",
      "delegation_scope": "..."
    }
  },
  "proof": {...}
}
‘‘‘
```

F.3. Terminate Receipt Example (ERASED mode)

```
‘‘‘json
{
  "hjs": "1.0",
  "id": "...",
  "actor": "did:example:agent456",
  "decision_hash": {...},
  "authority_scope": "...",
  "valid": {...},
  "state": "T",
  "prev_hash": "sha256:a1b2c3d4...",
  "primitive": {
    "type": "T",
```

```
      "payload": {
        "mode": "ERASED",
        "pkd": "base64-pkd-hash"
      }
    },
    "proof": {...}
  }
  ```
```

### F.4. Verify Receipt Example

```
  ```json
  {
    "hjs": "1.0",
    "verify": {
      "event_id": "f47ac10b-...",
      "method": "dual",
      "attestation": {...},
      "verifier": "did:example:auditor",
      "timestamp": 1712345680
    },
    "proof": {...}
  }
  ```
```

### F.5. Extended Receipt with TEE Evidence

This example shows a Judge receipt with TEE evidence in Tier 2.

```
Header (Tier 1):
{
  "alg": "EdDSA",
  "typ": "JEP-Receipt",
  "jep_v": "1.0",
  "jep_aid": "did:example:agent123",
  "jep_op": "J",
  "jep_rid": "f47ac10b-58cc-4372-a567-0e02b2c3d479"
}

Payload (Tier 2 & 3):
{
  "when": 1712345678,
  "based_on": null,
  "what": "sha256:3a7bd3e2360a3d29eea436f...",
  "time_anchor": "scitt:entry:123456",
  "contextual_evidence": {
    "tee_type": "Intel-TDX",
    "format": "eat",
    "evidence": "base64-encoded-EAT",
    "timestamp": 1712345678
  }
}
```

F.6. Negative Test Cases

```
- Modify any byte of the signed content → expect INVALID_SIGNATURE
- Change prev_hash to an incorrect value → BROKEN_CHAIN
- Set timestamp to 5 seconds before current time → VALID (within window)
- Set timestamp to 10 seconds after current time → EXPIRED_RECEIPT
- Construct an 11-level delegation chain → CHAIN_TOO_DEEP
- Omit required field (e.g., 'who') → parse error
```

Appendix G. Protocol Extensions

This appendix defines optional extensions to the core JEP
```

protocol. Extensions are organized by recommendation level for different deployment scenarios. Implementers may choose any combination of extensions based on their requirements; the following grouping is advisory only.

G.1. Extension Mechanism Principles

All extensions MUST NOT modify core protocol semantics (four primitives, three modes, three-tier architecture). Extensions MUST be registered via IANA (Section 8.4) and SHOULD follow reverse domain naming (e.g., org.example.jep.extension). Extensions MAY declare compatibility levels (full|wire|none).

G.2. Priority 1 Extensions (Strongly Recommended)

G.2.1. TEE Evidence Format

Purpose: Standardize TEE evidence encoding in Tier 2 for cross-platform interoperability.

Identifier: org.ietf.jep.ext.tee-evidence
Compatibility: wire

Specification:
- tee_type: REQUIRED. String identifying TEE type.
- format: REQUIRED. String. Values: "eat", "raw"
- evidence: REQUIRED. Base64-encoded attestation evidence
- verifier: OPTIONAL. URI of external verification service
- timestamp: REQUIRED. Unix timestamp of attestation generation

Example:

```json
{
  "contextual_evidence": {
    "tee_type": "Intel-TDX",
    "format": "eat",
    "evidence": "base64...",
    "timestamp": 1712345678
  }
}
```

G.2.2. Resolution Event ($Resolve)

Purpose: Provide automated recovery from DISPUTED state.

Identifier: org.ietf.jep.ext.resolution
Compatibility: full

Specification:
- Adds new primitive: $Resolve
- Adds new states: DISPUTED_FROZEN, RESOLVED_CONTINUE, RESOLVED_INVALID
- Resolution event format includes disputed_receipt, resolution_type, governance_signatures, etc.

G.2.3. Multi-Source Time Validation

Purpose: Enhance time reliability through multiple independent time sources.

Identifier: org.ietf.jep.ext.multi-time
Compatibility: full

Specification: See Section 4.7 for the core algorithm; this extension adds explicit encoding of multiple time sources in Tier 2.

G.3. Priority 2 Extensions (Optional)

G.3.1. Lightweight Mode

Purpose: Reduce receipt size and verification overhead for constrained environments.

Identifier: org.ietf.jep.ext.lightweight
Compatibility: wire

Specification:
- jep_version: "1.0-lite"
- compression: OPTIONAL (gzip, zstd)
- chain_proof.type: "partial" (only last N ancestors)
- verification_mode: MUST be "open"

G.3.2. AIP Usage

Purpose: Deep integration with Agent Identity Protocol.

Identifier: org.ietf.jep.ext.aip
Compatibility: wire

Adds aip_session and aip_capability fields to Tier 1 header.

G.3.3. Batch Operations

Purpose: Atomic batch processing of multiple primitives.

Identifier: org.ietf.jep.ext.batch
Compatibility: none

Adds $Batch primitive with operations array.

G.3.4. Refusal Event

Purpose: Support accountability for AI system "refusal to generate" scenarios.

Identifier: org.ietf.jep.ext.refusal
Compatibility: wire

Adds $Refuse primitive with refusal.reason, policy_ref, input_hash.

G.4. Priority 3 Extensions (Security Enhancement)

G.4.1. Key Evolution

Purpose: Provide forward secrecy through cryptographically verifiable key rotation.

Identifier: org.ietf.jep.ext.key-evolution
Compatibility: wire

Adds ROTATE_KEY event type, with rotation_proof from old key.

G.4.2. Enhanced Hardware Binding

Purpose: Provide physical binding strength through VDF and biometrics.

Identifier: org.ietf.jep.ext.enhanced-hardware
Compatibility: wire

Supports binding_level (standard/enhanced/maximum), VDF proofs, biometric binding.

G.5. Priority 4 Extensions (Compliance Support)

G.5.1. Jurisdiction Marking

Purpose: Mark receipts with applicable legal jurisdiction.

Identifier: org.ietf.jep.ext.jurisdiction
Compatibility: full

Fields: governing_law, data_localization, retention_period_days, erasure_proof_required.

G.5.2. Legal Validity Proof

Purpose: Enhance courtroom admissibility through notary and regulatory witness signatures.

Identifier: org.ietf.jep.ext.legal-proof
Compatibility: full

Includes witness objects with signatures.

G.5.3. GDPR Compliance Proof

Purpose: Provide cryptographic proof of GDPR compliance.

Identifier: org.ietf.jep.ext.gdpr-proof
Compatibility: full

Includes erasure_proof (PKD, erasure_timestamp, witness), dpo_signature, audit_log.

G.5.4. AI Act Transparency Marking

Purpose: Support EU AI Act transparency obligations.

Identifier: org.ietf.jep.ext.ai-act-transparency
Compatibility: wire

Fields: risk_level, transparency_obligation, synthetic_content, human_oversight, conformity_assessment.

G.6. Priority 5 Extensions (Governance Flexibility)

G.6.1. Federal Resolution

Purpose: Enable configurable multi-party governance for dispute resolution.

Identifier: org.ietf.jep.ext.federal-resolution
Compatibility: full

Supports governance_model (single/federal), participants with weights, threshold_policy. Resolution requires signatures meeting threshold.

Appendix H. TEE Evidence Examples

Appendix I. Dispute Resolution Workflow Examples

Table K.1: Tier Naming Conventions

Tier	Functional Name	Primary Content
Tier 1	Public Governance Tier	Identity Anchor
Tier 2	Logical Accountability Tier	Authorization Chain, Contextual Evidence
Tier 3	Private Payload Tier	Logic Payload (hash)

All references to "Tier X" in this document map to the above functional names.

Appendix L. Formal Justification of Core Fields

This appendix provides a formal argument for why the eight core fields constitute a minimal complete set for accountability attribution.

Let an accountability event be defined as a tuple (A, T, D, R, P) where:
- A: Actor (who)
- T: Time (when)
- D: Decision content (what)
- R: Reason/Authorization (based_on)
- P: Proof (signature)

Additionally, we require:
- A unique identifier to prevent replay (nonce)
- A secure timestamp anchor to prevent time tampering (time_anchor)
- An action type to distinguish different primitives (verb)

The eight fields correspond to these essential components. Removing any field would leave at least one component missing, breaking the ability to fully attribute the event. For example, without ‘who‘, the actor is unknown; without ‘based_on‘, the authorization chain cannot be verified; without ‘nonce‘, receipts can be replayed.

This minimal set has been validated against numerous real-world accountability scenarios, including medical diagnosis chains, financial transactions, and autonomous agent workflows. No additional field has been found necessary for basic accountability in any of these scenarios.

Future extensions may add fields for specific use cases (e.g., TEE evidence, compliance markings), but the core eight remain unchanged and universally applicable.

Appendix M. Regulatory Interaction Guide for Cryptographic Erasure

This guide provides deployers with practical advice on how to present cryptographic erasure evidence to regulators and auditors.

M.1. Preparing for Audit

- Maintain a log of all $Terminate events with ERASED mode, including the PKD and timestamp.
- Store the PKD in a secure, long-term repository (e.g., a signed audit log).
- Ensure that the time_anchor for the termination event is from a trusted source (e.g., SCITT, national TSP).

M.2. Responding to a Deletion Request (GDPR Art. 17)

1. Execute $Terminate with ERASED mode on the relevant receipts.
2. Provide the requestor with the PKD and the termination receipt ID.
3. If the regulator requests proof, present the termination receipt, PKD, and time_anchor evidence.

M.3. Communicating with Regulators

- Explain that the original content is irrecoverable because the decryption key has been destroyed.
- Emphasize that Tier 1 and Tier 2 data remain to prove that the erasure was performed correctly and at a specific time.
- Reference ENISA and EDPB guidelines that support this technique.

M.4. Legal Disclaimer

While this mechanism is designed to comply with deletion obligations, final recognition depends on the interpretation of the applicable law in each jurisdiction. Deployers should consult with legal counsel.

Author’s Address

Yuqiang Wang
HUMAN JUDGMENT SYSTEMS FOUNDATION LTD.
Email: signal@humanjudgment.org
GitHub: https://github.com/hjs-spec