JEP: Judgment Event Protocol

Minimal Verifiable Log Format for Agent Decisions

IETF Internet-Draft | draft-wang-jep-judgment-event-protocol-04 | March 2026

Every decision leaves a trace

What is JEP?

JEP (Judgment Event Protocol) is a minimal, verifiable log format protocol for recording and verifying decision-making actions of AI agents.

Minimal

Only defines necessary fields, no business rules presupposed

Verifiable

Cryptographic signatures + anti-replay + tamper-proof

Interoperable

Any platform, any agent can use it

Neutral

Records facts, does not judge right or wrong

JEP vs. Ordinary Logs

This is the most critical page for understanding JEP

Dimension	Ordinary Log	JEP
Trustworthiness	Anyone can modify	Cryptographically signed, tamper-proof
Verifiability	Cannot verify authenticity	Anyone can verify signatures
Replay Protection	None	Nonce uniqueness guarantee
Cross-Platform	Inconsistent formats	Unified format, standard interoperability
Causality Chain	No links	`ref` field supports chain linking
Provenance	Cannot prove who wrote it	Signature proves event origin

JEP Core Components

Four verbs + event format + verification rules

Actor

Agent or system component executing the decision

```
did:agent:assistant-001
```

Verb

Four standardized action types

```
J | D | T | V
```

Object

Resource or action being operated on

```
action:send-email:draft-123
```

Timestamp

Precise time in RFC 3339 format

```
2025-05-15T10:30:00Z
```

Four Core Verbs

Why these four?

J Judge

Initiate/make a decision

Example: AI makes a diagnosis recommendation

D Delegate

Transfer decision authority to another entity

Example: Supervisor delegates approval to assistant

T Terminate

End decision lifecycle

Example: Manual stop of automated trading

V Verify

Verify authenticity of existing events

Example: Auditor checks historical records

JEP Can Be Integrated by Any System

Design principle: zero-barrier integration, cross-platform interoperability

No external dependencies

No reliance on specific databases, message queues, blockchains, or cloud services

Pure text format

JSON-based, any programming language can parse and generate

Standard cryptography

Uses JWS (RFC 7515), mature library support in all languages

No mandatory external services

Can run locally, no remote API calls required

Self-contained events

Each event is independent and complete, no external context needed

Minimum Requirements for JEP Integration

Only three conditions needed

Requirement	Description
Can generate JSON	All modern programming languages support it
Can compute SHA-256	Standard library in all major languages
Can generate Ed25519 signatures	Mature libraries available in most languages

Cross-Agent Tracking and Propagation

Core mechanism: `ref` field builds event chains

Forward Tracking

Start from Event 1, find all subsequent events via `ref`

Backward Tracing

Start from Event 3, trace back to original decision via `ref`

Cross-Agent

Chain remains intact regardless of how many agents it passes through

Tamper-Proof

If any intermediate event is modified, all subsequent signatures become invalid

Cross-Agent Propagation Example

Scenario: Medical diagnosis cross-agent collaboration

Agent A (Radiology AI) Initial diagnosis

```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:radiology-ai",
  "when": 1742345678,
  "what": "sha256:imaging_diagnosis_hash",
  "nonce": "uuid-001",
  "ref": null,  //! Chain start
  "sig": "signature_A"
}
```

Agent B (Specialist AI) Further diagnosis

```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:specialist-ai",
  "when": 1742345700,
  "what": "sha256:specialist_opinion_hash",
  "nonce": "uuid-002",
  "ref": "sha256:event1_hash",  //! Points to A
  "sig": "signature_B"
}
```

Agent C (Review AI) Verify chain

```
{
  "jep": "1",
  "verb": "V",
  "who": "did:example:audit-ai",
  "when": 1742345720,
  "what": null,
  "nonce": "uuid-003",
  "ref": "sha256:event2_hash",  //! Points to B
  "sig": "signature_C"
}
```

What Does a JEP Event Look Like?

Complete JEP event example

```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:agent-789",
  "when": 1742345678,
  "what": "sha256:e8878aa9a38f4d123456789abcdef01234",
  "nonce": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "aud": "https://platform.example.com",
  "ref": null,
  "sig": "eyJhbGciOiJFZERTQSJ9..."
}
```

How Does JEP Ensure Trustworthiness?

Four-layer security guarantees

Mechanism	Function	Implementation
Digital Signature	Non-forgeable, non-tamperable	JWS (RFC 7515)
Nonce Uniqueness	Prevents replay attacks	UUIDv4
Timestamp Validation	Rejects stale events	Unix timestamp ±5 min
JSON Canonicalization	Cross-platform consistency	JCS (RFC 8785)

Primary Use Cases

Core positioning: Any scenario requiring "proof of decision traceability"

Healthcare AI

AI diagnosis + doctor review

Prove doctor saw and judged

Financial Trading

Algorithmic trading + risk approval

Prove transaction underwent human review

Autonomous Driving

Vehicle decision + human takeover

Prove authenticity of takeover timing

Content Moderation

AI flagging + moderator judgment

Prove moderator participated in judgment

Recruitment Screening

AI recommendation + HR decision

Prove screening process is traceable

Legal Compliance

Any regulated AI decision

Meet EU AI Act recording obligations

How to Use JEP?

Three-step integration

Step 1: Generate Event

- Set verb (J/D/T/V)
- Compute SHA-256 hash of decision content, put in 'what'
- Generate UUIDv4 nonce
- Sign with private key, generate 'sig'
- If parent event exists, put parent hash in 'ref'

Step 2: Store/Transmit

- Store in audit database (append-only)
- Or transmit via HTTPS to audit service
- Encrypt sensitive data with TLS/JWE

Step 3: Verify

- Retrieve event
- Verify signature with actor's public key
- Check nonce uniqueness
- Validate timestamp within window
- Verify ref chain integrity (recursively trace parents)

Presentation Formats

Three presentation formats for JEP events

JSON

Storage/transmission/API

Machine-readable, for system integration

Compact JWS

HTTP Headers

Convenient for REST API transmission

Visual Report

Human audit

Render JEP events into human-readable audit reports

Extension Framework

Modular design - enable as needed

Extension	ID	Purpose
Digest Anonymity	https://jep.org/priv/digest-only	Use hashed identities for routine logs, traceable via trusted party during disputes
Multi-Signature	https://jep.org/multisig	Multi-party collaborative decisions, only effective when threshold met
TTL Lifecycle	https://jep.org/ttl	Set data expiration time, auto-anonymization upon expiry
Sovereign Storage	https://jep.org/storage	Decouple from storage vendors, users choose data location independently
Subject Reference	https://jep.org/subject	Explicitly identify decision-affected subjects, enhance traceability transparency

Extension Capability

Serving More Complex Scenarios

Scenario	Problem	Extension Solution
Privacy Protection	Protect human participant identities	Digest Anonymity + TTL
Accountability Chain	Build complete responsibility traceability chain	`ref` chain + Multi-Signature
Causality Tracking	Record causal relationships between decisions	Extended fields + chain references

Thank You

JEP: Judgment Event Protocol

IETF:	draft-wang-jep-judgment-event-protocol-04
GitHub:	https://github.com/hjs-spec
Email:	signal@humanjudgment.org

Every decision leaves a trace