Internet-Draft
Intended status: Informational
Expires: 21 August 2026

Y. Wang
HUMAN JUDGMENT SYSTEMS FOUNDATION LTD.

21 February 2026

HJS: An Accountability Layer for AI Agents

draft-wang-hjs-accountability-00

Abstract

This document defines HJS (Human Judgment System), a standalone accountability layer for AI agents. HJS provides four primitives (Judgment/Delegation/Termination/Verification) for recording complete responsibility chains, and uses OTS (Open Timestamp) proofs anchored to the Bitcoin blockchain to create immutable, court-admissible evidence. While HJS can complement other protocols (e.g., AIIP) by providing the evidence they require, it is designed to be independent and usable with any AI agent communication framework.

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 21 August 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. The Four Primitives 3
  3.1. Judgment 3
  3.2. Delegation 4
  3.3. Termination 4
  3.4. Verification 5
  3.5. Responsibility Chain 5
4. OTS Evidence Chain 6
  4.1. OTS Proof Overview 6

4.2. Locking the Chain 6
5. Evidence Format 7
5.1. Complete Evidence Structure 7
5.2. Verification Process 8
6. Relationship with Other Protocols 8
6.1. Complementing AIIP 8
6.2. Independence 9
7. Use Cases 9
7.1. Regulatory Compliance 9
7.2. Dispute Resolution 10
7.3. Audit Trail 10
8. Security Considerations 10
9. IANA Considerations 11
10. Normative References 11
Author's Address 12

1. Introduction

As AI agents become increasingly autonomous, the need for a comprehensive accountability layer becomes critical. Current protocols focus on how agents communicate and execute tasks, but they do not address the fundamental question: who is responsible for what, and how can it be proven?

This document defines HJS (Human Judgment System), a standalone accountability layer for AI agents that provides:

1. Four primitives (Judgment, Delegation, Termination, Verification) for recording complete responsibility chains. These capture not just what happened, but why it happened, who was involved, and how it was verified.

2. OTS evidence chain that anchors cryptographic hashes of the primitives to the Bitcoin blockchain, creating immutable, court-admissible evidence.

3. Complete evidence locking that binds all components together into a verifiable package.

HJS is designed to be independent - it does not require any specific communication protocol and can be used with any AI agent framework. It can also complement existing protocols (such as AIIP [draft-sogomonian-aiip-architecture-03]) by providing the accountability layer they lack.

2. Terminology

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.

3. The Four Primitives

HJS defines four primitives that together form a complete responsibility chain for any AI agent action.

3.1. Judgment

The Judgment primitive records the basis on which an AI agent made

its decision. This includes:

- Input data (or cryptographic commitment to input)
- Model identifier and version
- Parameters and configuration
- Reasoning path (optional)

Example:

```
{
  "judgment": {
    "input_commitment": "base64url(sha256(input))",
    "model_id": "gpt-4-2026-02",
    "parameters": { "temperature": 0.7, "max_tokens": 500 }
  }
}
```


3.2. Delegation

The Delegation primitive records the chain of responsibility from the original requester to the executing agent:

- Delegator identifier
- Agent identifier
- Delegation time
- Scope and constraints

Example:

```
{
  "delegation": {
    "from": "aiip://user.phone",
    "to": "aiip://agent.service",
    "time": 1734149100000,
    "scope": "make_call",
    "constraints": { "max_duration": 30000 }
  }
}
```

3.3. Termination

The Termination primitive records the final outcome of the task:

- Status (success/failure/abort)
- Reason (if failure)
- Output commitment
- Duration

Example:

```
{
  "termination": {
    "status": "success",
    "output_commitment": "base64url(sha256(output))",
    "duration_ms": 2345
  }
}
```

3.4. Verification

The Verification primitive records how the result was verified:

- Verifier identifier
- Verification method
- Verification time
- Result

Example:

```
{
  "verification": {
    "verifier": "aiip://audit.service",
    "method": "ots_verify",
    "time": 1734149110000,
    "result": "valid"
  }
}
```

3.5. Responsibility Chain

The four primitives, when combined, form a complete responsibility chain that answers:

- Why was this done? (Judgment)
- Who requested it? (Delegation)
- What happened? (Termination)
- How was it verified? (Verification)

Implementations MUST include all four primitives to form a complete chain. Each primitive MAY be hashed individually to allow selective disclosure while maintaining chain integrity.

4. OTS Evidence Chain

While the four primitives provide structure for accountability, they must be made immutable and verifiable to serve as evidence. HJS uses OTS (Open Timestamp) proofs anchored to the Bitcoin blockchain for this purpose.

4.1. OTS Proof Overview

OTS proofs work as follows:

1. Compute SHA-256 hash of the data to be proven
2. Submit the hash to the Bitcoin network via Open Timestamp
3. A Bitcoin transaction includes the hash, creating an immutable timestamp
4. Generate an OTS proof file for verification

Key properties:

- Decentralized: No trusted third party required
- Immutable: Once anchored, cannot be altered
- Publicly verifiable: Anyone can verify with Bitcoin data
- Court-admissible: Recognized in multiple jurisdictions

4.2. Locking the Chain

HJS does not simply prove individual primitives - it locks the entire responsibility chain together:

1. Hash each primitive individually (allowing selective disclosure)
2. Combine the four hashes into a chain hash
3. Generate an OTS proof of the chain hash
4. The OTS proof becomes the evidence lock that binds all components together

This ensures that:

- Any tampering with any primitive breaks the chain
- Partial disclosure is possible without breaking trust
- The entire chain has a single, verifiable timestamp

5. Evidence Format

5.1. Complete Evidence Structure

The complete HJS evidence package includes:

```
{
  "hjs": {
    "version": "1.0",
    "primitives": {
      "judgment": { ... },
      "delegation": { ... },
      "termination": { ... },
      "verification": { ... }
    },
    "hashes": {
      "judgment_hash": "base64url(sha256(judgment))",
      "delegation_hash": "base64url(sha256(delegation))",
      "termination_hash": "base64url(sha256(termination))",
      "verification_hash": "base64url(sha256(verification))",
      "chain_hash": "base64url(sha256(judgment_hash + delegation_hash + termination_ha
sh + verification_hash))"
    },
    "evidence": {
      "ots_proof": "base64url(ots-proof-data)",
      "bitcoin_txid": "txid",
      "timestamp": 1734149123000
    }
  }
}
```


5.2. Verification Process

To verify an HJS evidence package:

1. Verify the OTS proof using standard OTS tools
2. Confirm the OTS proof matches the chain_hash
3. Recompute each primitive’s hash and verify against stored hashes
4. Optionally, verify individual primitives without accessing others
5. Check that the chain_hash matches recomputed combination

This process can be performed by anyone with access to Bitcoin blockchain data, without requiring access to the original AI system.

6. Relationship with Other Protocols

6.1. Complementing AIIP

The AIIP architecture [draft-sogomonian-aiip-architecture-03] defines execution receipts and requires attestation evidence. HJS naturally complements AIIP by providing exactly what AIIP requires but does not specify:

AIIP Requirement	HJS Solution
Attestation evidence	OTS proofs anchored to Bitcoin
Execution receipt	Can be extended with HJS primitives
Auditability	Complete responsibility chain
External anchoring	Bitcoin blockchain

However, HJS is not a subset or extension of AIIP. It is a complete, standalone accountability layer that can:

- Be used with AIIP by embedding HJS evidence in AIIP receipts
- Be used with any other communication protocol
- Be used entirely independently for logging and audit purposes

6.2. Independence

HJS makes no assumptions about:

- How agents communicate (HTTP, AIIP, custom protocols)
- What AI models are used
- Where execution happens (cloud, edge, TEE)
- Which jurisdiction applies

It provides a jurisdiction-agnostic, protocol-independent accountability layer that can be added to any AI system.

7. Use Cases

7.1. Regulatory Compliance

Organizations subject to regulations (finance, healthcare, EU AI Act) can use HJS to demonstrate compliance by maintaining verifiable records of AI agent decisions, including:

- What data was used (Judgment)
- Who authorized the action (Delegation)
- What happened (Termination)
- How it was verified (Verification)
- Immutable proof of all of the above (OTS evidence)

7.2. Dispute Resolution

When disputes arise about AI agent actions, HJS provides:

- Cryptographic proof of what happened
- Court-admissible evidence recognized in multiple jurisdictions
- Ability to verify without accessing the original AI system

7.3. Audit Trail

HJS creates a complete, immutable audit trail for AI agent operations, enabling:

- Internal audits
- Third-party audits
- Regulatory examinations
- Forensic analysis

8. Security Considerations

The security of HJS evidence depends on:

- Bitcoin blockchain: As of this writing, Bitcoin has never been successfully reorganized to alter confirmed transactions, providing strong assurance of immutability.

- Hash functions: Implementations MUST use SHA-256 or stronger hash functions.

- Key management: If signing is used (e.g., for verifier identity), implementations MUST follow best practices for key security.

- Selective disclosure: The ability to verify individual primitive hashes without revealing their content enables privacy-preserving audits.

9. IANA Considerations

This document has no IANA actions.

10. Normative References

[draft-sogomonian-aiip-architecture-03]
Sogomonian, A., "Artificial Intelligence Internet Protocol (AIIP) Architecture", Work in Progress, draft-sogomonian-aiip-architecture-03, December 2025.

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

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

[opentimestamp]
Open Timestamp Protocol, "Open Timestamp Protocol Specification", <https://opentimestamps.org/>, 2016.

[hjs-github]
Wang, Y., "HJS Protocol Implementation", MIT License, <https://github.com/schchit/hjs-api>.

Author’s Address

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