Internet-Draft
draft-wang-jac-00
Intended status: Standards Track
Expires: September 26, 2026

Y. Wang
HJS Foundation Ltd.

March 26, 2026

JAC: Judgment Accountability Chain
draft-wang-jac-00

Abstract

This document defines the Judgment Accountability Chain (JAC) v0.0, a minimal infrastructure layer for tracking judgment chains in AI and agent systems. Built on JEP [draft-wang-jep-judgment-event-protocol-01] and HJS [draft-wang-hjs-accountability-02], JAC adds a single field— task_based_on— to establish verifiable causality between judgments across agents, platforms, and trust domains.

JAC follows the same narrow-waist design philosophy as JEP and HJS: a minimal mandatory core with all advanced features defined as optional extensions.

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 26, 2026.

Copyright Notice

Table of Contents

1. Core Accountability Model (JAC-Core)

1.1. Core Principles

JAC-Core is the minimal, non-negotiable judgment chain layer that all JAC-compliant systems MUST implement. It consists of:

1. JEP Event Format (as defined in [draft-wang-jep-judgment-event-protocol-01]) — provides event recording, signatures, and non-repudiation.

2. HJS Responsibility Chain (as defined in [draft-wang-hjs-accountability-02]) — provides responsibility tracking via based_on.

3. One additional field: task_based_on — establishes causality between judgments across agents.

No other fields or semantics are required for core compliance.

1.2. Core Field Definition

JAC does not define a new event format. Instead, it adds one field to the existing JEP+HJS event structure:

```
{
  "jep": "1",
  "verb": "E",
  "who": "did:example:agent-123",
  "when": 1742345678,
  "what": "hash-of-execution-result",
  "nonce": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "based_on": "hash-of-parent-event",
  "task_based_on": "hash-of-parent-task",
  "sig": "eyJhbGciOiJFZERTQSJ9..."
}
```

Core Field:

Field	Type	Description
task_based_on	string / null	Hash of the parent judgment event. null indicates the start of a judgment chain.

1.3. Core Semantics

JAC defines one new verb, "E" (Execute), with the following semantics:

Verb	Core Accountability Meaning
E	Execute a judgment. task_based_on MUST reference the parent judgment (if any).

Verb Semantics:

- "E" with task_based_on = null: Start of a new judgment chain.
- "E" with task_based_on pointing to an existing judgment: Continuation of a judgment chain.

1.4. Core Verification

```
def verify_jac_core(event):
    # Step 1: JEP signature verification (inherited)
    if not verify_jep_signature(event):
        return "INVALID"

    # Step 2: HJS chain verification (inherited)
    if event.based_on and not parent_exists(event.based_on):
        return "INVALID"
    if event.verb == "J" and event.based_on is not None:
        return "INVALID"

    # Step 3: JAC chain integrity
    if event.task_based_on and not parent_judgment_exists(event.task_based_on):
        return "INVALID"

    # Step 4: JAC chain head check
    if event.verb == "E" and event.task_based_on is not None:
        return "VALID"  # Chain continuation
    if event.verb == "E" and event.task_based_on is None:
        return "VALID"  # Chain start

    return "INVALID"
```

1.5. Core Guarantees

- Non-repudiation: Events are signed by the actor (inherited from JEP).
- Integrity: Any modification breaks the signature (inherited from JEP).
- Chain Integrity: task_based_on ensures causal ordering between judgments.
- Replay Protection: nonce prevents replay (inherited from JEP).
- Cross-Agent Traceability: Judgment chains can span multiple agents, platforms, and trust domains.

2. Optional Extension Modules

Extensions are independent, optional modules that add functionality to JAC-Core. All extensions MUST be placed in the extensions object at the top level of the JWS Payload.

2.1. Extension Framework

```
{
  "jep": "1",
  "verb": "E",
  "who": "did:example:agent-123",
  "when": 1742345678,
  "what": "hash-of-result",
  "nonce": "uuid",
  "based_on": "hash-of-parent-event",
  "task_based_on": "hash-of-parent-task",
  "extensions": {
    "https://jac.org/state": { ... },
    "https://jac.org/assign": { ... }
  },
  "sig": "..."
}
```

IANA Registration Requirements:

- Extension Identifier: URI (HTTPS recommended)
- Semantics Description: URL to specification
- Compatibility Level: "full" (ignorable), "wire" (may affect interoperability), "none" (required)
- Version: Semantic version

2.2. Task State Module (JAC-State)

Identifier: https://jac.org/state
Compatibility: full
Purpose: Track judgment lifecycle state.

```
{
  "extensions": {
    "https://jac.org/state": {
      "status": "completed",
      "progress": 0.75,
      "assigned_at": 1742345678,
      "started_at": 1742345700,
      "completed_at": 1742345800,
      "timeout_at": 1742345900,
      "retry_count": 2
    }
  }
}
```

Field	Type	Description
status	string	assigned, executing, completed, failed, cancelled
progress	number	Optional, 0-1
assigned_at	integer	Unix timestamp
started_at	integer	Unix timestamp
completed_at	integer	Unix timestamp
timeout_at	integer	Unix timestamp
retry_count	integer	Number of retries attempted

2.3. Task Assignment Module (JAC-Assign)

Identifier: https://jac.org/assign
Compatibility: wire
Purpose: Record judgment assignment between agents.

```
{
  "extensions": {
    "https://jac.org/assign": {
      "assigner": "did:example:manager-agent",
      "assignee": "did:example:worker-agent",
      "capability_required": "diagnosis.radiology",
      "deadline": 1742346000,
      "priority": "high"
    }
  }
}
```

Field	Type	Description
assigner	string	DID of agent assigning judgment
assignee	string	DID of agent receiving judgment
capability_required	string	Capability required to execute
deadline	integer	Unix timestamp
priority	string	low, normal, high, critical

2.4. Task Handoff Module (JAC-Handoff)

Identifier: https://jac.org/handoff
Compatibility: wire
Purpose: Record judgment handoff between agents.

```
{
  "extensions": {
    "https://jac.org/handoff": {
      "from": "did:example:agent-a",
      "to": "did:example:agent-b",
      "reason": "specialization",
      "context": "hash-of-context-data"
    }
  }
}
```

Field	Type	Description
from	string	DID of agent handing off
to	string	DID of agent receiving
reason	string	specialization, overload, failure, escalation
context	string	Hash of handoff context

2.5. Result Verification Module (JAC-Result)

Identifier: https://jac.org/result

Compatibility: full
Purpose: Record and verify judgment execution results.

```
{
  "extensions": {
    "https://jac.org/result": {
      "output_hash": "hash-of-result",
      "output_schema": "https://schema.org/MedicalDiagnosis",
      "verifier": "did:example:validator-agent",
      "verification_sig": "base64...",
      "confidence": 0.95,
      "human_reviewed": true,
      "human_reviewer": "did:example:doctor-456"
    }
  }
}
```

Field	Type	Description
output_hash	string	Hash of judgment output
output_schema	string	URI of output schema
verifier	string	DID of entity verifying result
verification_sig	string	Signature of verification
confidence	number	0-1 confidence score
human_reviewed	boolean	Whether human reviewed
human_reviewer	string	DID of human reviewer

2.6. Agent Capability Module (JAC-Capability)

Identifier: https://jac.org/capability
Compatibility: full
Purpose: Attest to agent capabilities.

```
{
  "extensions": {
    "https://jac.org/capability": {
      "required": ["diagnosis.radiology", "report.generation"],
      "attestation": "did:example:certifier",
      "attestation_sig": "base64...",
      "expires_at": 1750000000
    }
  }
}
```

Field	Type	Description
required	array	List of required capabilities
attestation	string	DID of attestation issuer
attestation_sig	string	Signature of attestation
expires_at	integer	Expiration timestamp

2.7. Task Input/Output Module (JAC-IO)

Identifier: https://jac.org/io
Compatibility: full
Purpose: Reference judgment inputs and outputs.

```
{
  "extensions": {
    "https://jac.org/io": {
      "input_hash": "hash-of-input",
      "input_schema": "https://schema.org/PatientData",
      "output_location": "s3://bucket/result.json",
      "output_encrypted": true
    }
  }
}
```

Field	Type	Description
input_hash	string	Hash of judgment input
input_schema	string	URI of input schema
output_location	string	URI of output storage
output_encrypted	boolean	Whether output is encrypted

2.8. Fault Recording Module (JAC-Fault)

Identifier: https://jac.org/fault
Compatibility: full
Purpose: Record chain breaks caused by missing parent judgments.

In dynamic multi-agent environments, it is possible that a parent judgment fails to be signed (e.g., due to network failure, agent crash, or timeout), causing the logical chain to break. This module allows the downstream agent to record the break and provide auditable evidence of the expected but missing parent.

```
{
  "extensions": {
    "https://jac.org/fault": {
      "expected_parent": "hash-of-expected-parent-task",
      "fault_type": "timeout",
      "fault_reason": "Agent did not respond within 30s",
      "fault_detected_at": 1742345900,
      "detected_by": "did:example:orchestrator-agent"
    }
  }
}
```

Field	Type	Description
expected_parent	string	Hash of the parent judgment that was expected but missing
fault_type	string	timeout, agent_unavailable, signature_failure, unknown
fault_reason	string	Humanreadable reason
fault_detected_at	integer	Unix timestamp when the fault was detected
detected_by	string	DID of the agent or system that

		detected the fault

Verification Semantics:

When a JAC event includes this extension and the core verification would otherwise return ‘INVALID‘ due to a missing parent judgment (i.e., ‘task_based_on‘ points to a nonexistent event), the verifier SHOULD treat the event as ‘VALID_WITH_FAULT‘ if the following conditions hold:

- The ‘expected_parent‘ field matches the value of ‘task_based_on‘.
- The ‘fault_type‘ and ‘fault_reason‘ provide a plausible explanation for the absence.
- The signature on the event is valid.

This allows the chain to continue while preserving a complete audit trail of the break.

3. Implementation Guide

3.1. Minimal Implementation

```
class JACReceipt:
    def __init__(self, verb, who, when, what, nonce, based_on, task_based_on):
        self.verb = verb
        self.who = who
        self.when = when
        self.what = what
        self.nonce = nonce
        self.based_on = based_on
        self.task_based_on = task_based_on

    def verify(self, public_key):
        # Step 1: JEP signature verification
        if not self.verify_signature(public_key):
            return "INVALID"

        # Step 2: HJS chain integrity
        if self.based_on and not self.parent_exists():
            return "INVALID"
        if self.verb == "J" and self.based_on is not None:
            return "INVALID"

        # Step 3: JAC chain integrity
        if self.task_based_on and not self.parent_judgment_exists():
            # Check for fault extension
            if self.extensions and "https://jac.org/fault" in self.extensions:
                fault = self.extensions["https://jac.org/fault"]
                if fault.get("expected_parent") == self.task_based_on:
                    return "VALID_WITH_FAULT"
            return "INVALID"

        # Step 4: JAC chain head check
        if self.verb == "E" and self.task_based_on is not None:
            return "VALID"
        if self.verb == "E" and self.task_based_on is None:
            return "VALID"

        return "INVALID"
```

3.2. Error Code Reference

Code	Description
INVALID_SIGNATURE	Signature verification failed
BROKEN_CHAIN	Parent hash mismatch (HJS chain broken)
BROKEN_TASK_CHAIN	Parent task hash mismatch
INVALID_TASK_HEAD	E with task_based_on not null not allowed
EXPIRED_RECEIPT	Timestamp outside window

4. IANA Considerations

4.1. JAC Extensions Registry

This document requests IANA to create a new registry titled "JAC Extensions". Initial entries:

Extension Identifier	Compatibility	Reference
https://jac.org/state	full	Section 2.2
https://jac.org/assign	wire	Section 2.3
https://jac.org/handoff	wire	Section 2.4
https://jac.org/result	full	Section 2.5
https://jac.org/capability	full	Section 2.6
https://jac.org/io	full	Section 2.7
https://jac.org/fault	full	Section 2.8

5. Security Considerations

JAC inherits the security properties of JEP and HJS:

- Signature Forgery: Root signature prevents unauthorized modification of core fields.
- Chain Tampering: Each hop signature covers all previous hops; modification of any hop invalidates all subsequent signatures.
- Replay Attacks: nonce prevents replay.
- Key Management: Private keys SHOULD be stored in HSMs or secure enclaves. Key rotation is supported via extensions.
- Offline Verification: Core verification requires only the issuer’s public key. No network calls are required.

6. References

6.1. Normative References

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

[RFC7515] Jones, M., "JSON Web Signature (JWS)", RFC 7515,

DOI 10.17487/RFC7515, May 2015,
https://www.rfc-editor.org/info/rfc7515.

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

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

[RFC9562] Davis, D., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, https://www.rfc-editor.org/info/rfc9562.

[draft-wang-jep-judgment-event-protocol-01]
Wang, Y., "Judgment Event Protocol (JEP)", Work in Progress, Internet-Draft, draft-wang-jep-judgment-event-protocol-01, March 2026.

[draft-wang-hjs-accountability-02]
Wang, Y., "HJS: An Accountability Layer for AI Agents", Work in Progress, Internet-Draft, draft-wang-hjs-accountability-02, March 2026.

6.2. Informative References

[I-D.ietf-scitt-architecture]
Birkholz, H., Delignat-Lavaud, A., Deshpande, Y., and Y. Wang, "Supply Chain Integrity, Transparency, and Trust (SCITT) Architecture", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-05, March 2026.

[DID-CORE] Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and C. Allen, "Decentralized Identifiers (DIDs) v1.0", W3C Recommendation, July 2022, https://www.w3.org/TR/did-core/.

Author’s Address

Yuqiang Wang
HJS Foundation Ltd.
Email: signal@humanjudgment.org
GitHub: https://github.com/hjs-spec