Internet-Draft
Intended status: Experimental
Expires: October 29, 2026

Y. Wang
April 29, 2026

JAC: Declared Dependency Chains for Agent Receipts
A JEP/HJS Extension Profile for Chain Infrastructure
draft-wang-jac-02

Abstract

This document defines JAC v0.5, a minimal declared-dependency chain infrastructure for agent and AI-agent receipt systems. JAC is a profile over the Judgment Event Protocol (JEP) and, when AI-agent behavior receipts are used, over HJS. JAC does not define an independent event format, signature syntax, hash format, replay mechanism, identity framework, or governance framework.
JAC uses JEP extension fields to bind a signed event or receipt element to a declared parent dependency. The core is intentionally small: a single chain extension records a declared dependency link. Optional extensions can record state declarations, assignment references, handoff references, result references, capability claims, input/output references, and declared chain breaks.
JAC provides infrastructure only. It records signed dependency declarations; it does not determine factual causality, responsibility, fault, authorization validity, assignment validity, handoff validity, task correctness, fairness, entitlement, legal effect, or regulatory compliance.

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 October 29, 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Motivation

Agent systems increasingly create chains of decision-related events, behavior records, tool calls, handoffs, and verification reports across platforms and trust domains. Verifiers often need a way to determine whether one signed event declared a dependency on another signed event, behavior record, task descriptor, receipt manifest, or external digest-bound artifact.

JAC addresses this need by defining a minimal extension profile over JEP. JAC records declared dependency links in signed JEP events. When the linked objects are HJS behavior records or receipt manifests, JAC uses the HJS receipt model and validation rules.

JAC is not intended to prove real-world causality. A signed dependency link proves only that an issuer declared a dependency and that the declaration was cryptographically bound to a JEP event. The truth, completeness, exclusivity, or governance consequence of that dependency is outside JAC.

1.2. Relationship to JEP and HJS

JAC is a profile of the Judgment Event Protocol (JEP) [I-D.wang-jep-judgment-event-protocol]. JEP defines the event object, J/D/T/V verbs, algorithm-tagged digest strings, event hash semantics, detached JWS signing over JCS-canonicalized unsigned events, key resolution, validation modes, and the ext/ext_crit extension framework.

JAC does not redefine those JEP mechanisms. A JAC event is a JEP event that uses JAC-specific extension identifiers. JAC implementations MUST NOT introduce a separate signature syntax, hash syntax, event hash calculation, nonce mechanism, or critical-extension mechanism.

HJS [I-D.wang-hjs-accountability] defines accountability receipts and behavior records for AI-agent evidence. JAC can link HJS behavior records, receipt manifests, or receipt bundles by digest. When JAC is used with HJS objects, HJS validation applies before JAC dependency validation.

JEP: minimal verifiable event protocol
HJS: AI-agent receipt and behavior-record profile over JEP
JAC: declared-dependency chain profile over JEP/HJS

1.3. Scope

JAC defines:

* A minimal JEP extension profile for declared dependency links.
* The JAC-Core-1 chain extension identifier https://jac.org/chain.
* Core fields for a declared dependency link: based_on, based_on_type, and relation.
* Optional extension identifiers for state declarations, assignment references, handoff references, result references, capability claims, input/output references, and declared break references.
* Validation rules for checking that a JAC dependency declaration is structurally valid and digest-bound to a signed JEP event.
* Security and privacy guidance for interpreting chain declarations without treating them as causality, fault, authorization, governance, or legal conclusions.

JAC explicitly does not define:

* Factual causality, complete causal history, or exclusive causal explanation.
* Legal liability, culpability, fault, negligence, good faith, or responsibility assignment.
* Authorization validity, delegation validity, assignment validity, permission-chain correctness, consent validity, or lawful basis.
* Monitoring mandates, governance outcomes, escalation duties, sanctions, employment consequences, remedies, appeal rights, explanation rights, access rights, disclosure duties, legal admissibility, or regulatory compliance.
* A global identity, trust, credential, task-management, or workflow-enforcement framework.
* New JEP verbs. JAC uses the JEP verbs defined by JEP and expresses chain semantics through extensions.

1.4. Minimal Core and Optional Extensions

The JAC core is intentionally minimal. JAC-Core-1 defines only the chain extension needed to bind a JEP event to a declared parent dependency. All other capabilities are optional extensions or deployment profiles.

Optional JAC extensions provide technical mechanisms only. They do not define governance rules, fairness, authorization, fault, responsibility, access rights, disclosure duties, legal effect, or evidentiary sufficiency.

The absence of an optional JAC extension MUST NOT be interpreted by the protocol as agreement, waiver, admission, lack of objection, lack

of harm, lack of context, lack of external rights, or absence of external process.

1.5. 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.

1.6. Terminology

JAC event: A JEP event containing a JAC extension.
Dependency link: A signed declaration that one event or record is based on, derived from, related to, or otherwise linked to another artifact.
Declared dependency: A dependency link asserted by the issuer. It is not a protocol-level determination of factual causality.
Parent: The artifact referenced by a JAC based_on field.
Chain fragment: A set of JEP events or HJS records connected by JAC dependency links. A fragment need not contain all real-world predecessors.
Declared break: A signed declaration that an expected parent artifact is missing, unavailable, or not included in the exported fragment. A declared break is not a fault determination.

2. JAC Model

2.1. Design Goals

* Keep the core narrow: one required chain extension and no new event format.
* Reuse JEP signing, hashing, event hash, ext/ext_crit, and validation rules.
* Reuse HJS receipt and behavior-record semantics when AI-agent receipt objects are linked.
* Record declared dependencies without asserting causality, fault, authorization, governance, or legal effect.
* Allow optional deployment profiles for task state, assignment, handoff, results, capabilities, input/output references, and declared breaks.
* Support exportable chain fragments that remain verifiable even when not all related artifacts are disclosed.

2.2. Declared Dependency Links

A JAC declared dependency link is an assertion by the event issuer that the current event or receipt element is related to a parent artifact. The parent MAY be another JEP event, an HJS behavior record, an HJS receipt manifest, a task descriptor, an external digest-bound artifact, or another deployment-defined object.
JAC does not require the parent artifact to be available at acceptance time. A verifier can validate the structure and signature of the declaration even when the parent artifact is not present. Archival or bundle validation can later validate the referenced parent when it becomes available.
A JAC link is not a claim of complete or exclusive causation. It is a signed dependency declaration suitable for technical traceability and audit workflows.

2.3. Relationship between JEP ref and JAC Links

The JEP ref field is a general event reference defined by JEP. It normally references a related JEP event by event hash. JAC does not redefine ref.

The JAC based_on field, carried inside the https://jac.org/chain extension, identifies a parent dependency for chain analysis. It MAY reference a JEP event hash, an HJS behavior-record digest, an HJS receipt-manifest digest, a task descriptor digest, or another digest-bound artifact.

A deployment MAY use both ref and the JAC chain extension in the same JEP event. When both are present, ref remains the JEP event reference, while the JAC chain extension records the declared dependency used by JAC chain validation.

2.4. Infrastructure Positioning

JAC is chain infrastructure. It is not a task-management system, governance system, authorization system, liability engine, fairness engine, explanation-rights framework, appeal system, monitoring mandate, or legal evidence rulebook.

JAC enables parties to create, bind, export, and validate dependency declarations. External legal, organizational, human-rights, safety, operational, and governance frameworks decide what consequences, if any, follow from those declarations.

3. JAC-Core-1

3.1. Required Profiles

A JAC-Core-1 implementation MUST support JEP-Core-1 as defined by JEP [I-D.wang-jep-judgment-event-protocol]. If a JAC chain references HJS behavior records, receipt manifests, or receipt bundles, the implementation MUST process those objects according to HJS-Core-1 [I-D.wang-hjs-accountability].

JAC-Core-1 does not define independent cryptographic algorithms. Signature capability, canonicalization profile, hash family, key resolution, acceptance validation, archival validation, and critical extension handling are inherited from JEP and applicable JEP trust profiles.

3.2. Chain Extension

The JAC-Core-1 chain extension identifier is:
https://jac.org/chain

A JAC event MUST carry this extension in the JEP ext object when it intends to declare a JAC dependency link. If the chain declaration is critical to interpretation or validation, the extension identifier SHOULD also appear in the JEP ext_crit array.

The chain extension content MUST be included in the JEP signing payload because JEP signs all event members except sig, including ext and ext_crit.

3.3. Field Definitions

The https://jac.org/chain extension has the following fields:

Field	Type	Description
based_on	string/null	Algorithm-tagged digest or event hash of the declared parent. null indicates a declared chain root.
based_on_type	string	Type hint for the parent artifact.
relation	string	Relationship descriptor.
chain_id	string	OPTIONAL digest or opaque identifier for grouping a chain fragment.
sequence	integer	OPTIONAL deployment-defined sequence

		number within a chain fragment.

JAC-Core-1 implementations MUST support based_on_type values of jep-event, hjs-behavior-record, hjs-receipt-manifest, task-descriptor, external-digest, and unknown. Additional values MAY be defined by deployment profiles.

JAC-Core-1 implementations MUST support relation values of chain-root and declared-dependency. Additional relation values are deployment-defined unless registered or specified by a future document. JAC does not assign governance, authorization, or legal meaning to relation values.

The based_on value MUST be null when relation is chain-root. When relation is not chain-root, based_on SHOULD contain an algorithm-tagged digest string or JEP event hash as defined by JEP.

3.4. Core Example


```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:agent-789",
  "when": 1742345700,
  "what": "sha256:4bf5122f344554c53bde2ebb8cd2b7e3d1600ad6318a57c91189c74a9efc52d3",
  "nonce": "b7f3e5a6-1d44-4a22-8d72-7ef7f2d9c9aa",
  "aud": "https://platform.example.com",
  "ref": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a1b7c",
  "ext": {
    "https://jac.org/chain": {
      "based_on": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a
1b7c",
      "based_on_type": "jep-event",
      "relation": "declared-dependency",
      "chain_id": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23ad
c8b7",
      "sequence": 2
    }
  },
  "ext_crit": ["https://jac.org/chain"],
  "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6YWdlbnQtNzg5I2tleS0xIn0..exa
mple-detached-signature"
}
```


4. Optional JAC Extensions

4.1. Extension Principles

JAC optional extensions provide reference and declaration mechanisms. They do not define the truth, sufficiency, governance meaning, legal effect, or operational consequence of the referenced material.

Optional extensions MAY be omitted. When present, extension content that affects chain interpretation, evidence binding, privacy, export, or validation MUST be included in the signed JEP event payload or referenced by a digest-bound record.

If an optional JAC extension is listed in JEP ext_crit, a verifier that does not understand or cannot process that extension MUST reject the event or receipt according to JEP critical-extension rules.

4.2. State Declaration Extension

Identifier: https://jac.org/state

Purpose: Record a deployment-defined state declaration for a chain element.


```
"https://jac.org/state": {
  "status": "completed",
  "status_taxonomy": "https://example.org/task-state-v1",
  "declared_at": 1742345800,
```

```
    "state_ref": "sha256:b94d27b9934d3e08a52e52d7da7dabfadebc7d1532a914b0d3c7d2b2d8f6a6e
4"
  }
```

State values are deployment-defined unless an external profile gives them meaning. JAC does not enforce lifecycle state, task completion, cancellation, or workflow correctness.

4.3. Assignment Reference Extension

Identifier: https://jac.org/assign

Purpose: Record a digest-bound assignment reference or assignment claim.

```
  "https://jac.org/assign": {
    "assignment_ref": "sha256:ef92b778bafe771e89245b89ecbc08a44a4e166c0665995c7f69a7649c
2b8d30",
    "assigner_ref": "did:example:manager-456",
    "assignee_ref": "did:example:worker-789",
    "deadline": 1742346000
  }
```

An assignment reference is not an authorization decision. JAC does not determine whether an assignment is valid, accepted, lawful, fair, or enforceable.

4.4. Handoff Reference Extension

Identifier: https://jac.org/handoff

Purpose: Record a digest-bound handoff reference or handoff declaration.

```
  "https://jac.org/handoff": {
    "handoff_ref": "sha256:b94d27b9934d3e08a52e52d7da7dabfadebc7d1532a914b0d3c7d2b2d8f6a
6e4",
    "from_ref": "did:example:agent-a",
    "to_ref": "did:example:agent-b",
    "declared_at": 1742345750
  }
```

A handoff reference is not a governance action. JAC does not determine whether a handoff is valid, complete, accepted, authorized, or operationally sufficient.

4.5. Result Reference Extension

Identifier: https://jac.org/result

Purpose: Record result and verification-report references for a chain element.

```
  "https://jac.org/result": {
    "output_digest": "sha256:4bf5122f344554c53bde2ebb8cd2b7e3d1600ad6318a57c91189c74a9ef
c52d3",
    "output_schema": "https://example.org/schema/result-v1",
    "verification_report_ref": "sha256:5c9c720c81ef7a09d86d4930a2ce3ef9a503c3b8b8d4e2cc6
a1b7081aa01",
    "verifier_ref": "did:example:verifier-123"
  }
```

A result reference is not a correctness determination. A verification report reference proves only digest binding to external material; its meaning and sufficiency are outside JAC.

4.6. Capability Claim Reference Extension

Identifier: https://jac.org/capability

Purpose: Record a capability claim or capability attestation reference.

```
  "https://jac.org/capability": {
    "capability_ref": "sha256:ef92b778bafe771e89245b89ecbc08a44a4e166c0665995c7f69a7649c
2b8d30",
    "attestation_issuer_ref": "did:example:certifier",
    "valid_until": 1750000000
```

```
}
```

A capability claim reference does not prove competence, authorization, certification validity, or fitness for a task. Those determinations belong to external trust and governance profiles.

4.7. Input/Output Reference Extension

Identifier: https://jac.org/io

Purpose: Record digest-bound input and output references for a chain element.

```
"https://jac.org/io": {
  "input_digest": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23adc8b7",
  "input_schema": "https://example.org/schema/input-v1",
  "output_digest": "sha256:4bf5122f344554c53bde2ebb8cd2b7e3d1600ad6318a57c91189c74a9efc52d3",
  "output_location_ref": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a1b7c"
}
```

Input and output references can support audit and replay analysis, but JAC does not provide confidentiality. Sensitive content requires external access control, encryption, minimization, or redaction.

4.8. Declared Break Reference Extension

Identifier: https://jac.org/break

Purpose: Record that an expected parent dependency is missing, unavailable, withheld, or not included in an exported chain fragment.

```
"https://jac.org/break": {
  "expected_parent": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a1b7c",
  "expected_parent_type": "jep-event",
  "declaration_ref": "sha256:b94d27b9934d3e08a52e52d7da7dabfadebc7d1532a914b0d3c7d2b2d8f6a6e4",
  "declared_at": 1742345900,
  "declared_by": "did:example:orchestrator-agent"
}
```

A declared break is not a fault determination. It does not prove that an agent failed, acted negligently, acted in bad faith, or caused the missing parent. It records only a signed declaration about a missing or unavailable dependency.

5. Validation

5.1. Validation Inputs

JAC validation operates on a JEP event, optional HJS records or receipt manifests, and an optional set of parent artifacts. A verifier MAY validate a single JAC event, a chain fragment, or an exported receipt bundle.

JAC validation does not require a verifier to retrieve missing parent artifacts over the network. Deployment profiles MAY define discovery mechanisms, but such mechanisms are outside JAC-Core-1.

5.2. JEP and HJS Validation

A verifier MUST perform JEP validation before JAC dependency validation. If the JEP event fails signature, structure, critical-extension, or applicable validation-mode checks, the JAC dependency declaration MUST NOT be treated as valid.

If the JAC event references HJS behavior records, receipt manifests, or receipt bundles, the verifier SHOULD validate those HJS objects according to HJS before using them as JAC parent or child artifacts.

5.3. JAC Dependency Validation

To validate a JAC dependency declaration, a verifier performs the following steps:

* Locate the https://jac.org/chain extension in the JEP ext object.
* If the extension is listed in ext_crit and is not understood, reject the event according to JEP rules.
* Validate that based_on, based_on_type, relation, chain_id, and sequence have the expected JSON types.
* If relation is chain-root, verify that based_on is null.
* If relation is not chain-root and a parent artifact is available, compute the parent artifact digest or event hash according to the applicable profile and compare it to based_on.
* If relation is not chain-root and the parent artifact is unavailable, return a missing-parent result rather than asserting causality or fault.
* If a declared break extension is present, validate that expected_parent matches based_on or the missing parent being declared.

5.4. Missing Parent and Declared Break Handling

A missing parent does not automatically make a signed JAC event false. It means the verifier cannot complete dependency validation for that link with the supplied materials.

When https://jac.org/break is present and structurally valid, the verifier MAY return DECLARED_BREAK_PRESENT. This result indicates only that a signed declared-break reference is present and digest-bound. It MUST NOT be interpreted by the protocol as fault, excuse, waiver, admission, negligence, system failure, or chain validity in a legal or governance sense.

5.5. Validation Result Labels

JAC implementations MAY expose validation result labels. The following non-exhaustive labels are defined for interoperability:

Label	Meaning
VALID_ROOT	A signed chain-root declaration is valid.
VALID_LINK	A signed dependency link matches an available parent artifact.
VALID_FRAGMENT	The supplied chain fragment is internally consistent, but may not include all parents.
MISSING_PARENT	A parent referenced by based_on is not available to the verifier.
DECLARED_BREAK_PRESENT	A signed break declaration is present for a missing or unavailable parent.
INVALID_SIGNATURE	JEP signature validation failed.
INVALID_EXTENSION	The JAC extension is malformed or unsupported while marked critical.
PARENT_MISMATCH	The available parent digest does not match the based_on value.

These labels are technical validation labels only. They do not assign responsibility, prove factual causality, or determine legal or governance outcomes.

6. Security and Privacy Considerations

6.1. Declared Dependency Is Not Causality

A valid JAC link proves that an issuer signed a declaration of dependency. It does not prove that the referenced parent was the complete, exclusive, sufficient, or actual cause of the later event. Multiple real histories may be consistent with the same chain fragment.

6.2. Assignment Is Not Authorization

Assignment references can support traceability, but JAC does not determine whether an assignment was authorized, accepted, lawful, fair, or enforceable. Verifiers need external authorization and governance profiles for those determinations.

6.3. Handoff Is Not Governance

Handoff references record technical declarations or references. They do not prescribe escalation duties, operational responsibility, employment consequences, monitoring duties, or governance outcomes.

6.4. Declared Break Is Not Fault Determination

The declared break extension records a signed declaration that an expected parent is missing, unavailable, withheld, or omitted from a fragment. JAC does not determine whether the break resulted from negligence, malice, system failure, privacy minimization, redaction, network failure, or any other cause.

6.5. Partial Observation Limits

JAC chains are projections of observed or exported records. They may omit relevant events, external facts, human context, private records, redacted materials, or alternative causal paths. Verifiers should treat JAC chains as verifiable evidence structures, not complete causal models.

This limitation is particularly important in AI-agent and multi-agent environments where logs, traces, and receipts may capture only part of the real execution environment. External evidence policies and domain-specific review procedures remain necessary.

6.6. Privacy and Data Minimization

JAC inherits privacy and minimization considerations from JEP and HJS. JAC extensions SHOULD avoid plaintext personal data when a digest, opaque reference, pseudonymous identifier, or privacy-preserving receipt view is sufficient.

A JAC chain fragment MAY be exported with redactions or minimized views. Such exports MUST NOT misrepresent the remaining signature, digest, and dependency relationships. When a parent is omitted for privacy, safety, confidentiality, or legal reasons, deployments MAY use the declared break extension or an external profile to describe the omission without revealing sensitive content.

6.7. Governance Neutrality

JAC is infrastructure for creating, binding, exporting, and validating dependency declarations. It is not an accountability tribunal, governance framework, compliance authority, monitoring mandate, fairness engine, appeal system, explanation-rights framework, or liability determination system.

The absence of a JAC optional extension or parent artifact MUST NOT be interpreted by the protocol as agreement, waiver, admission, lack of objection, lack of harm, lack of relevant context, or absence of external rights.

7. IANA Considerations

7.1. JAC Extension Identifier Registrations

This document requests registration of the following JAC extension identifiers in the JEP Extensions Registry defined by JEP. JAC does not request creation of a separate JAC Extensions Registry.

Extension Identifier	Description
https://jac.org/chain	Declared dependency link
https://jac.org/state	State declaration reference
https://jac.org/assign	Assignment reference
https://jac.org/handoff	Handoff reference
https://jac.org/result	Result reference
https://jac.org/capability	Capability claim reference
https://jac.org/io	Input/output reference
https://jac.org/break	Declared break reference

These identifiers are stable identifiers and are not required to be dereferenceable. Future JAC-specific extensions should be registered in the JEP Extensions Registry using the registration template and policy defined by JEP.

7.2. No JAC Verbs Registry

JAC does not define new JEP verbs and does not request a JAC Verbs Registry. JAC uses the J/D/T/V verbs defined by JEP and expresses dependency semantics through JEP extensions.

7.3. No Separate JAC Extensions Registry

A separate JAC Extensions Registry would duplicate the JEP extension registry and weaken interoperability. JAC-specific extensions are therefore registered under the JEP Extensions Registry.

8. References

8.1. 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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[I-D.wang-jep-judgment-event-protocol] Wang, Y., "Judgment Event Protocol (JEP)", Work in Progress, Internet-Draft, draft-wang-jep-judgment-event-protocol-05, April 2026.

[I-D.wang-hjs-accountability] Wang, Y., "HJS: Accountability Receipts for AI Agents", Work in Progress, Internet-Draft, draft-wang-hjs-accountability-05, April 2026.

8.2. Informative References

[TARGET-DETERMINABILITY] Wang, Y., "Target Determinability under Partial Causal Observation: A Faithful Reduction Framework", Zenodo, Version v1, DOI 10.5281/zenodo.19678205, April 2026, <https://doi.org/10.5281/zenodo.19678205>.

[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/>.

Appendix A. Non-Normative Examples

A.1. Chain Start Example


```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:agent-789",
  "when": 1742345678,
  "what": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23adc8b7",
  "nonce": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "aud": "https://platform.example.com",
  "ref": null,
  "ext": {
    "https://jac.org/chain": {
      "based_on": null,
      "based_on_type": "unknown",
      "relation": "chain-root",
      "chain_id": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23ad
c8b7",
      "sequence": 1
    }
  },
  "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6YWdlbnQtNzg5I2tleS0xIn0..exa
mple-detached-signature"
}
```


A.2. Chain Continuation Example


```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:agent-456",
  "when": 1742345700,
  "what": "sha256:4bf5122f344554c53bde2ebb8cd2b7e3d1600ad6318a57c91189c74a9efc52d3",
  "nonce": "b7f3e5a6-1d44-4a22-8d72-7ef7f2d9c9aa",
  "aud": "https://platform.example.com",
  "ref": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a1b7c",
  "ext": {
    "https://jac.org/chain": {
      "based_on": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a
1b7c",
      "based_on_type": "jep-event",
      "relation": "declared-dependency",
      "chain_id": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23ad
c8b7",
      "sequence": 2
    },
    "https://jac.org/io": {
      "input_digest": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb
23adc8b7",
      "output_digest": "sha256:4bf5122f344554c53bde2ebb8cd2b7e3d1600ad6318a57c91189c74
a9efc52d3"
    }
  },
  "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6YWdlbnQtNDU2I2tleS0xIn0..exa
```

```
mple-detached-signature"
   }
```

A.3. Declared Break Example

```
   {
     "jep": "1",
     "verb": "J",
     "who": "did:example:orchestrator-agent",
     "when": 1742345900,
     "what": "sha256:ef92b778bafe771e89245b89ecbc08a44a4e166c0665995c7f69a7649c2b8d30",
     "nonce": "3f94ff2c-b5b8-4e76-bbc6-9d0462ac8ea5",
     "aud": "https://platform.example.com",
     "ref": null,
     "ext": {
       "https://jac.org/chain": {
         "based_on": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fca2f2e0a
1b7c",
         "based_on_type": "jep-event",
         "relation": "declared-dependency",
         "chain_id": "sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23ad
c8b7",
         "sequence": 3
       },
       "https://jac.org/break": {
         "expected_parent": "sha256:0f343b0931126a20f133d67c2b018a3b4c8d9a756d7e1f8fd94fc
a2f2e0a1b7c",
         "expected_parent_type": "jep-event",
         "declaration_ref": "sha256:b94d27b9934d3e08a52e52d7da7dabfadebc7d1532a914b0d3c7d
2b2d8f6a6e4",
         "declared_at": 1742345900,
         "declared_by": "did:example:orchestrator-agent"
       }
     },
     "ext_crit": ["https://jac.org/chain"],
     "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6b3JjaGVzdHJhdG9yI2tleS0xIn0.
.example-detached-signature"
   }
```

Appendix B. Changes from draft-wang-jac-01

- Aligned JAC with JEP v0.5 and HJS v0.5.
- Changed intended status to Experimental.
- Removed task_based_on as a top-level field; replaced it with the https://jac.org/chain extension carried in JEP ext.
- Changed causality language to declared dependency language.
- Removed the suggestion to use a new E verb; JAC defines no new verbs.
- Replaced extensions with JEP ext/ext_crit processing.
- Renamed fault handling to declared break handling and removed protocol-level plausible-explanation evaluation.
- Removed the independent JAC Extensions Registry and moved JAC extension registration under the JEP Extensions Registry.
- Clarified that JAC inherits signature, hash, canonicalization, key-resolution, validation-mode, and cryptographic-profile rules from JEP.
- Added governance-neutrality, partial-observation, privacy, and non-inference boundaries.

Author’s Address

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