Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 5, 2026

Y. Wang
Independent

May 4, 2026

Judgment Event Protocol (JEP)
draft-wang-jep-judgment-event-protocol-06

Status of This Memo

Copyright Notice

Table of Contents

Judgment Event Protocol (JEP)
A Neutral Verifiable Event Format for Agent Decisions
draft-wang-jep-judgment-event-protocol-06

Author: Yuqiang Wang
Intended status: Experimental
Version: -06 working draft
Companion drafts:
- draft-wang-jep-profiles-00
- draft-wang-jep-conformance-00

Abstract

This document defines the Judgment Event Protocol (JEP), a neutral verifiable event format for decision-related operations in human, organizational, software, and autonomous agent systems.

JEP specifies four immutable event verbs: Judgment (J), Delegation (D), Termination (T), and Verification (V). It defines a signed JSON event

structure, anti-replay fields, detached JSON Web Signature (JWS) verification over JSON Canonicalization Scheme (JCS) canonicalized payloads, event hash and reference semantics, validation levels, structured validation results, failure codes, extension handling, trust-profile interfaces, and determinability boundaries.

JEP is intended to provide a global protocol-level interoperability layer for verifiable decision event records. JEP-Core does not define legal liability, authorization validity, regulatory compliance, organizational trust decisions, global identity, global truth, or mandatory support for any specific credential, identity, AI platform, agent framework, or blockchain system.

---

1. Introduction

Autonomous and semi-autonomous agent systems increasingly make, assist with, delegate, terminate, verify, or record decisions across organizational, platform, model, and jurisdictional boundaries. These systems need a minimal and interoperable way to record decision-related events so that later verifiers can determine whether a particular event payload existed, whether it was signed under an applicable trust profile, whether it was modified, whether it refers to other events or evidence, and which validation level has been reached.

JEP addresses this need by defining a compact signed JSON event format. The format is intentionally narrow: it records an event verb, actor identifier, timestamp, claim or digest, nonce, optional audience, references, extensions, and signature. More complex identity, credential, policy, archival, evidence, or causal-chain semantics are externalized to profiles, extensions, HJS-like archival layers, or JAC-like chain-composition layers.

JEP records verifiable protocol facts. It does not by itself prove external target facts. In partially observed systems, a signed event log can support audit and accountability workflows without guaranteeing complete or zero-error determination of external facts.

---

2. Protocol Objective and Non-Goals

2.1 Objective

JEP-Core defines a neutral event layer for verifiable judgment-related acts. A conforming JEP event can support determination, subject to the applicable validation mode and trust profile, that:

1. an event payload existed;
2. the event payload was signed as specified;
3. the payload was bound to an actor identifier under a trust profile;
4. the event carried a timestamp, nonce, optional audience, and optional references or extensions;
5. references, event hashes, and critical extensions were processed according to JEP-Core rules;
6. a verifier reached a declared validation level.

2.2 Non-Goals

JEP-Core does not define:

- legal liability;
- moral responsibility;
- regulatory compliance;

- global truth determination;
- external target-fact determinability;
- authorization delegation validity;
- permission-chain enforcement;
- lifecycle state-machine enforcement;
- a global identity framework;
- a global credential framework;
- a global trust framework;
- mandatory DID, VC, X.509, OAuth, RATS, blockchain, or AI-platform support;
- confidentiality for event content;
- long-term storage, redaction, retention, or disclosure policy;
- causal-chain or responsibility-graph computation.

A JEP event records claims about judgment-related acts. It MUST NOT be presented as proof that the underlying real-world assertion is true unless an applicable external profile and evidence policy makes that determination.

---

3. Design Principles

JEP-Core follows these principles:

1. Core minimality: JEP-Core defines the stable narrow-waist event layer.
2. Identity-system neutrality: JEP-Core MUST NOT require any specific identity system.
3. Credential-system neutrality: JEP-Core MUST NOT require VC or any other credential model.
4. Platform neutrality: JEP-Core MUST NOT require any AI platform, agent framework, cloud provider, or blockchain network.
5. Profile-based interoperability: identity, credentials, attestation, authorization, and archival policy are handled by optional profiles.
6. Layered verification: syntax, cryptographic validity, actor-binding, chain validation, and policy validation are separate.
7. Explicit determinability boundary: protocol validity is not the same as external truth.
8. Privacy by minimization: sensitive evidence SHOULD be referenced by digest or controlled evidence mechanisms rather than embedded in event payloads.
9. Extension without semantic capture: extensions MUST NOT redefine JEP-Core semantics.
10. Algorithm agility: cryptographic algorithms are selected by JOSE headers, conformance profiles, and trust profiles rather than by event verbs.

---

4. 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 when, and only when, they appear in all capitals.

---

5. Terminology

Actor: The entity identified by who that is claimed by the event. An actor MAY be a human, organization, model, agent, tool, service, device, workflow, committee, session, swarm, human-agent composite, or organization-agent composite.

Signer: The key holder that produces the event signature. The signer is not necessarily identical to the actor. The binding between signer and actor is defined by a trust profile.

Subject: The entity or object about which a decision, delegation, termination, or verification is made. A subject is distinct from the actor.

Event: A single immutable signed JSON object representing one judgment-related act.

Unsigned Event: A JEP event object with the sig member omitted.

JEP Signing Payload: The UTF-8 octets of the JCS-canonicalized unsigned event.

Event Hash: An algorithm-tagged digest over the full signed event.

Claim: Semantic content carried by what or by an extension.

Reference: A cryptographic or typed pointer to another event, chain, digest, credential, policy, evidence, context, archive receipt, or external object.

Trust Profile: A profile that defines actor identifier forms, signing-key resolution, actor/key binding, revocation, historical validity, acceptable algorithms, and related evidence policy.

Validation Level: The highest validation stage completed by a verifier.

Validation Mode: The context in which validation is performed, such as acceptance, archival, chain, or policy review.

Verification Scope: The declared scope of a V event, such as syntax, cryptographic, actor-binding, chain-integrity, credential-status, policy compliance, human review, external evidence, factual claim, or archival integrity.

---

6. Core Event Object

A JEP event is a JSON object. Producers MUST emit I-JSON-compatible JSON and MUST NOT emit duplicate JSON member names. Verifiers MUST reject events containing duplicate JSON member names.

The top-level members are:

Member	Status	Description
jep	REQUIRED	Wire-format major version. For this draft, "1".
verb	REQUIRED	One of "J", "D", "T", "V".
who	REQUIRED	Actor identifier claimed by the event.
when	REQUIRED	Actor-supplied event timestamp, Unix seconds.
what	Conditional	Claim object, descriptor, or algorithm-tagged digest.
nonce	REQUIRED	Replay-protection nonce.
aud	RECOMMENDED	Intended audience or validation context.
ref	Conditional	Typed reference or event hash reference.
ext	OPTIONAL	Extension object.
ext_crit	OPTIONAL	Critical extension identifier list.
sig	REQUIRED	Detached signature container.

The top-level extensibility mechanism is ext. Producers SHOULD NOT add new top-level members outside this specification unless defined by a

future JEP revision or a registered extension that explicitly updates the top-level object.

---

7. Field Semantics

7.1 jep

The jep member identifies the wire-format major version. For this draft, the value is "1".

JEP-Core-0.6 identifies the specification release version. It is not the wire-format version. The wire-format value jep: "1" can remain stable across multiple draft revisions and release versions when the core event object remains wire-compatible.

A verifier MUST NOT infer Internet-Draft revision number or release maturity solely from the jep member.

7.2 verb

The verb member identifies the event verb. It MUST be one of J, D, T, or V.

Event verbs do not determine cryptographic algorithms, storage policy, privacy mode, identity method, or legal effect.

7.3 who

who identifies the actor claimed by the event.

who is not necessarily:

- the signer;
- a legal person;
- the controller of the key;
- a real-world identity;
- the subject of the decision.

The binding between who and the signing key is determined by the applicable trust profile.

7.4 when

when is an actor-supplied event timestamp in Unix seconds. JEP-Core does not by itself prove wall-clock time. Stronger time evidence requires a timestamping, receipt, archival, transparency, or storage profile.

Implementations SHOULD distinguish:

- event time;
- signature time;
- receipt time;
- archive time;
- verification time;
- policy-evaluation time.

7.5 what

what carries the event claim, digest, descriptor, or report. It records what the actor asserted, judged, delegated, terminated, or verified. It does not by itself prove external truth.

When represented as an object, what SHOULD use semantic subfields such as:

- claim;
- subject;
- scope;
- result;
- evidence;
- context.

When represented as a digest, the digest MUST be an algorithm-tagged digest string.

7.6 nonce

nonce supports replay protection. Nonce uniqueness is evaluated within an actor, audience, trust-profile, and replay-window context. JEP-Core does not require global nonce uniqueness.

7.7 aud

aud indicates intended audience or validation context. It does not by itself enforce access control. Access control, retention, redaction, and disclosure policy are outside JEP-Core.

7.8 ref

ref is a reference field. A reference does not by itself imply endorsement, truth, authorization validity, or causality.

References MAY identify:

- event;
- chain;
- digest;
- credential;
- policy;
- evidence;
- context;
- archive receipt;
- external object.

A simple event reference SHOULD use the event hash of the referenced JEP event.

7.9 ext and ext_crit

ext contains named extension objects. ext_crit contains the identifiers of critical extensions. Unknown critical extensions MUST cause validation failure. Unknown non-critical extensions MAY be ignored.

7.10 sig

sig carries the detached signature container. JEP-Core uses detached JWS over the JCS-canonicalized unsigned event unless another registered signature profile applies.

---

8. Event Verb Semantics

8.1 J - Judgment

A Judgment event records that an actor made, accepted, produced, approved, selected, rejected, ranked, classified, or otherwise committed to a decision-related claim.

A J event MUST NOT be interpreted as proof that the judged claim is true.

Typical uses include:

- model output approval;
- human approval;
- risk classification;
- tool-selection decision;
- policy decision;
- recommendation acceptance;
- evidence assessment.

8.2 D - Delegation

A Delegation event records that an actor delegated a task, authority, responsibility, capability, or decision context to another actor or system.

A D event SHOULD identify, directly or by reference:

- delegator;
- delegatee;
- scope;
- constraints;
- expiry;
- context;
- termination conditions.

A D event records a delegation claim. It does not by itself prove legal authorization or external enforceability.

8.3 T - Termination

A Termination event records that a previous delegation, authority, session, capability, workflow context, verification context, or chain reliance is ended, revoked, expired, superseded, or no longer valid for future reliance.

A T event MUST identify the terminated target and termination scope.

A T event does not delete historical events, erase past facts, or automatically revoke external legal authority. It affects future reliance according to JEP-Core rules and applicable profiles.

8.4 V - Verification

A Verification event records that an actor performed a validation, audit, review, confirmation, rejection, or verification action over an event, chain, digest, subject, credential, policy, evidence, or external object.

A V event MUST declare its verification scope. A V event MUST NOT imply verification beyond its declared scope.

Initial verification scopes include:

- syntax;
- cryptographic;
- actor_binding;
- chain_integrity;
- extension_processing;
- credential_status;
- policy_compliance;
- human_review;
- external_evidence;
- factual_claim;
- archival_integrity.

---

9. References and Chains

A JEP reference proves only that one event payload cryptographically or semantically referenced another object. It does not prove causality, endorsement, truth, authorization, or completeness.

A JEP reference does not by itself imply causality. Causal interpretation is defined by JAC or another chain profile.

A reference chain proves linkage between observed event objects. It does not prove that the observed log is complete, append-only, or free from omission unless a profile explicitly declares and supports a complete-log assumption.

---

10. Algorithm-Tagged Digest Strings

JEP uses algorithm-tagged digest strings for event hashes, content digests, and reference digests.

Syntax:

```
<hash-algorithm>:<lowercase-hex-digest>
```

The hash algorithm identifier MUST be lower-case ASCII. The digest value MUST be lower-case hexadecimal.

Implementations conforming to JEP-Core-0.6 MUST support sha256.

Example:

```
sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23adc8b7
```

Additional digest algorithms MAY be defined by conformance profiles, trust profiles, or registered extensions.

---

11. Event Hashes and Signing Input

The JEP signing input is the unsigned event object with sig omitted, canonicalized using JCS and encoded as UTF-8 octets.

The event hash identifies the full signed event object, including sig.

For the default hash profile:

```
event_hash = sha256(UTF8(JCS(full_signed_event)))
```

The signing input and event hash are intentionally different:

- signing input excludes sig;
- event hash includes sig.

---

12. Signature, Hash, and Algorithm Agility

JEP-Core preserves algorithm agility.

A JEP event is protected by a detached JWS signature over a JCS-canonicalized unsigned event payload.

JEP-Core does not assign cryptographic algorithms to event verbs. J, D, T, and V share the same cryptographic processing model.

The concrete signature algorithm, key type, hash algorithm, and algorithm acceptability policy are determined by JOSE headers, conformance profiles, and trust profiles.

JEP-Core semantics are algorithm-neutral. A JEP-Core implementation MUST NOT assign algorithms to event verbs or treat algorithm choice as part of the semantics of J, D, T, or V.

A baseline conformance class MAY define a required-to-implement algorithm set for interoperability. For compatibility with the -05 baseline, an Ed25519/JWS/JCS conformance class MAY be defined by the companion conformance draft. Such a conformance class does not make Ed25519 the only algorithm allowed by JEP-Core semantics.

A trust profile MUST define which algorithms are acceptable for a given deployment context.

A verifier MUST reject an event if the declared algorithm is unsupported, prohibited by the applicable profile, inconsistent with the resolved key type, inconsistent with the signature container, or inconsistent with a critical cryptographic extension.

A verifier SHOULD distinguish real-time acceptance validation from archival validation. An algorithm MAY be acceptable for archival validation while being prohibited for new event signatures.

---

13. Trust Profile Interface

JEP-Core does not define a global identity or trust framework.

A trust profile MUST define:

- supported actor identifier forms;
- key identifier syntax and discovery;
- binding rules between who and signing keys;
- accepted signature algorithms;
- downgrade policy;
- key rotation handling;
- revocation handling;
- historical key validity;
- credential or attestation use, if any;
- policy evaluation hooks, if any.

Support for DID, VC, X.509, OAuth, RATS, blockchain anchoring, or any specific identity system is OPTIONAL and MUST NOT be required for JEP-Core conformance.

---

14. Validation Model

14.1 Validation Levels

JEP validation is divided into five levels:

Level	Name	Meaning
0	Syntax	JSON, I-JSON, required fields, schema shape.
1	Cryptographic	Canonicalization, event hash, detached signature.
2	Actor Binding	Key bound to actor under trust profile.
3	Chain	References, critical extensions, replay, termination effects, chain integri

ty. |
| 4 | Policy | Domain, legal, organizational, regulatory, or deployment policy. |

A verifier MUST report the highest successfully completed validation level. A verifier MUST NOT present Level 1 cryptographic validity as Level 4 policy validity.

A JEP event MAY be cryptographically valid but policy-invalid.

14.2 Validation Modes

Initial validation modes are:

- acceptance;
- archival;
- chain;
- policy.

Acceptance validation checks freshness and replay. Archival validation MUST NOT reject an event solely because its timestamp is outside the current freshness window.

14.3 Deterministic Validation Algorithm

A verifier SHOULD process an event in this order:

1. parse JSON;
2. reject duplicate JSON member names;
3. check required top-level fields;
4. check verb-specific field requirements;
5. remove sig to construct unsigned event;
6. canonicalize unsigned event using JCS;
7. verify detached signature;
8. compute event hash if needed;
9. resolve actor/key under trust profile;
10. check actor/key binding;
11. check nonce and freshness for acceptance mode;
12. validate aud if present and relevant;
13. resolve and validate references if applicable;
14. process critical extensions;
15. apply chain rules, including termination effects;
16. apply optional policy rules;
17. return structured validation result.

---

15. Validation Result Object

A verifier SHOULD return a structured validation result.

Example:

```
{
  "valid": true,
  "level": 3,
  "mode": "archival",
  "profile": "jep-core-0.6",
  "scopes": ["syntax", "cryptographic", "actor_binding", "chain_integrity"],
  "event_hash": "sha256:...",
  "warnings": [],
  "errors": []
}
```

Validation results MUST distinguish:

- cryptographically valid but policy-invalid;

- archivally valid but not acceptable for new reliance;
- valid under profile A but invalid under profile B;
- signature-valid but actor-binding invalid;
- chain-valid under observed-log assumptions only.

---

16. Failure Codes

A conforming validator SHOULD return structured failure codes.

16.1 Syntax Errors

- ERR_INVALID_JSON
- ERR_DUPLICATE_MEMBER
- ERR_UNSUPPORTED_JEP_VERSION
- ERR_UNKNOWN_VERB
- ERR_MISSING_REQUIRED_FIELD
- ERR_INVALID_FIELD_TYPE
- ERR_INVALID_TIMESTAMP

16.2 Cryptographic Errors

- ERR_CANONICALIZATION_FAILED
- ERR_CANONICALIZATION_VERSION_UNSUPPORTED
- ERR_INVALID_EVENT_HASH
- ERR_UNSUPPORTED_SIGNATURE_ALG
- ERR_PROHIBITED_SIGNATURE_ALG
- ERR_ALG_KEY_TYPE_MISMATCH
- ERR_ALG_PROFILE_MISMATCH
- ERR_HASH_ALG_UNSUPPORTED
- ERR_SIGNATURE_CONTAINER_INVALID
- ERR_SIGNATURE_MISSING
- ERR_SIGNATURE_INVALID
- ERR_DIGEST_MISMATCH
- ERR_ARCHIVAL_ALG_STATUS_UNKNOWN
- ERR_ALG_DEPRECATED_FOR_NEW_EVENTS

16.3 Actor and Trust Errors

- ERR_ACTOR_UNRESOLVED
- ERR_KEY_UNRESOLVED
- ERR_KEY_NOT_BOUND_TO_ACTOR
- ERR_KEY_REVOKED
- ERR_KEY_NOT_VALID_AT_EVENT_TIME
- ERR_TRUST_PROFILE_UNSUPPORTED

16.4 Replay and Freshness Errors

- ERR_NONCE_REPLAY
- ERR_EVENT_EXPIRED
- ERR_TIMESTAMP_OUT_OF_WINDOW

16.5 Reference and Chain Errors

- ERR_REF_UNRESOLVED
- ERR_REF_HASH_MISMATCH
- ERR_CHAIN_BROKEN
- ERR_DELEGATION_SCOPE_EXCEEDED
- ERR_TERMINATED_REFERENCE_REUSED
- ERR_CYCLE_DETECTED
- ERR_COMPLETE_LOG_ASSUMPTION_UNSATISFIED

16.6 Extension Errors

- ERR_UNKNOWN_CRITICAL_EXTENSION

- ERR_EXTENSION_SCHEMA_INVALID
- ERR_EXTENSION_VALIDATION_FAILED
- ERR_EXTENSION_CONFLICT

16.7 Policy Errors

- ERR_POLICY_REJECTED
- ERR_AUTHORIZATION_CONTEXT_MISSING
- ERR_DOMAIN_REQUIREMENT_UNSATISFIED

---

17. Extension Rules and Conflict Handling

An extension MUST declare:

1. extension identifier;
2. extension version;
3. JSON schema or equivalent data model;
4. whether it may be critical;
5. validation requirements;
6. security considerations;
7. privacy considerations;
8. interaction with event hashes and signatures;
9. interaction with other known extensions, if applicable.

Extensions MUST NOT redefine the semantics of core JEP members.

Unknown critical extensions MUST cause rejection. Unknown non-critical extensions MAY be ignored.

If two critical extensions impose inconsistent requirements, validation MUST fail with ERR_EXTENSION_CONFLICT.

Extension identifiers SHOULD be collision-resistant. Supported forms include:

- reverse-DNS identifiers, e.g. org.example.jep.profile.vc;
- URI identifiers;
- registered short names;
- experimental x-* identifiers.

---

18. Conformance Requirements

A JEP-Core-0.6 producer MUST support:

- I-JSON-compatible event construction;
- required top-level fields;
- JCS canonicalization of unsigned payloads;
- detached JWS signature generation under at least one conformance class;
- algorithm-tagged digest strings;
- nonce generation;
- ext and ext_crit semantics.

A JEP-Core-0.6 verifier MUST support:

- duplicate-member rejection;
- core field validation;
- JCS canonicalization;
- detached signature verification under at least one conformance class;
- event hash calculation;
- validation levels;
- structured validation result object;
- unknown critical extension rejection;

- acceptance and archival validation modes.

A JEP-Core-0.6 implementation MUST NOT require support for any optional identity, credential, attestation, blockchain, AI platform, or agent framework profile.

18.1 Baseline Algorithm Conformance

For interoperability with earlier JEP drafts, a baseline conformance class MAY require support for detached JWS using JCS canonicalization, sha256 algorithm-tagged digest strings, and Ed25519 verification.

This baseline is a conformance-class requirement, not a JEP-Core semantic requirement. Other profiles MAY define additional or alternative algorithm suites, including regional, enterprise, COSE/CBOR, composite, or post-quantum profiles, provided that their algorithm identifiers, key representations, downgrade policies, and validation behavior are specified.

---

19. Determinability Boundary

JEP distinguishes observable protocol facts from external target facts.

19.1 Observable Protocol Facts

JEP can support determination of protocol-level facts such as:

- whether a payload was signed by a key;
- whether a key is acceptable under a trust profile;
- whether an event hash matches an event payload;
- whether an event references another event;
- whether a nonce has been seen within a replay window;
- whether a critical extension was processed;
- whether a delegation was later terminated in the observed log.

19.2 External Target Facts

JEP alone does not determine external target facts such as:

- whether a real-world statement is true;
- whether a model internally understood a request;
- whether an actor is legally liable;
- whether a delegation is legally enforceable;
- whether a human actually read a document;
- whether a physical-world action occurred outside the logged system.

A profile MAY define evidence rules for external target facts. Such rules are outside JEP-Core.

---

20. Observed Log Assumptions

An observed JEP log is not necessarily a complete log.

Absence of an event in an observed log MUST NOT be interpreted as proof that the event did not occur unless a complete-log assumption is explicitly declared by the deployment profile.

Profiles MAY define:

- complete-log profile;
- partial-log profile;
- selective-disclosure log;

- redacted log;
- archive-backed log;
- transparency-backed log.

A chain reconstruction result MUST declare whether it relies on complete or partial log assumptions.

---

21. Relationship to HJS and JAC

JEP defines atomic signed judgment events.

HJS-like systems manage storage, receipt, archival context, retention, redaction, selective disclosure, privacy policy, and evidence lifecycle for JEP events. HJS MUST NOT redefine JEP-Core event semantics, signature semantics, event hash semantics, validation levels, or failure codes.

JAC-like systems compose JEP events into causality chains, responsibility chains, delegation paths, verification paths, and workflow accountability graphs. JAC MUST NOT redefine JEP-Core event format, signature semantics, event hash semantics, or validation levels.

A JEP reference does not by itself imply causality. Causal interpretation is defined by JAC or another chain profile.

---

22. Security Considerations

JEP-Core mitigates:

- payload tampering;
- signature forgery under supported algorithms;
- replay within configured windows;
- reference hash mismatch;
- unknown critical extension processing;
- basic chain integrity failures.

JEP-Core does not by itself prevent:

- compromised signing keys;
- false claims signed by legitimate actors;
- omission of relevant events from a log;
- collusion among actors;
- legal or organizational misuse;
- incorrect external evidence;
- malicious trust profiles;
- timestamp manipulation without external time evidence.

22.1 Downgrade Resistance

A verifier MUST reject algorithms prohibited by the applicable profile. A verifier MUST NOT accept a weaker algorithm merely because it is syntactically valid in JOSE.

22.2 Human-in-the-Loop Semantics

A human-review event records that a human actor emitted or endorsed a review-related claim. It does not prove that the human fully understood the underlying material, that the decision was correct, or that legal compliance was satisfied.

22.3 AI Actor Semantics

JEP-Core does not mandate any specific AI actor identity scheme. AI actor identity, model identity, tool identity, service identity, and session identity are defined by trust profiles or extensions.

---

23. Privacy Considerations

JEP events may reveal actor identity, subject identity, decision timing, delegation structure, organizational workflow, tool usage, and audit relationships.

Deployments SHOULD minimize personal data in what and extensions. When possible, external evidence SHOULD be referenced by digest rather than embedded directly.

Digest references may enable correlation, confirmation attacks, or dictionary attacks. Sensitive evidence references MAY require salted digests, commitment schemes, access-controlled evidence stores, audience-bound references, selective disclosure, or redaction.

JEP signatures and references may create linkability across contexts. Implementations SHOULD avoid reusing actor identifiers across unrelated audiences unless required by the trust profile.

JEP is an accountability protocol component. It SHOULD NOT be deployed as a general monitoring mechanism without data-minimization, retention, access-control, and redaction policies.

---

24. Registry Considerations

JEP registries SHOULD cover:

- verbs;
- extension identifiers;
- verification scopes;
- validation modes;
- error codes;
- trust profile identifiers;
- conformance class identifiers;
- algorithm policy labels.

New verb registrations are NOT RECOMMENDED. New verbs require an update to JEP-Core explaining why existing verbs plus extensions are insufficient.

---

25. Versioning and Compatibility

Minor versions MAY add optional fields or optional extensions. Major versions MAY change validation requirements.

Unknown critical extensions MUST cause validation failure. Unknown non-critical extensions MAY be ignored.

Unknown required core fields MUST cause validation failure unless defined by a future compatible revision.

---

26. Examples

26.1 Minimal Judgment Event Shape

```
{
  "jep": "1",
  "verb": "J",
  "who": "did:example:agent-789",
  "when": 1742345678,
  "what": "sha256:aa55ad4393538f14e6b4961de1a29216eed93517cb6c2631a56a5ee75edb3b7a",
  "nonce": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "aud": "https://platform.example.com",
  "ref": null,
  "sig": "..."
}
```

Full test vectors are defined in the companion conformance draft.

---

27. Changes from -05

Major changes from draft-wang-jep-judgment-event-protocol-05:

- Reframed JEP-Core as a neutral narrow-waist event infrastructure.
- Added explicit protocol objective and non-goals.
- Added design principles.
- Added detailed field semantics for who, when, what, nonce, aud, ref, ext, and sig.
- Expanded J/D/T/V semantics.
- Added verification-scope requirements for V events.
- Added validation levels.
- Added structured validation result object.
- Added structured failure-code taxonomy.
- Added deterministic validation algorithm.
- Added extension conflict handling.
- Added trust profile interface.
- Added determinability boundary.
- Added observed-log assumptions.
- Added relationship to HJS and JAC.
- Added human-in-the-loop and AI actor semantic boundaries.
- Added downgrade-resistance language.
- Moved concrete DID/VC/X.509/OAuth/RATS profile material to draft-wang-jep-profiles-00.
- Moved schema, test vectors, and reference validator behavior to draft-wang-jep-conformance-00.

Author’s Address

Yuqiang Wang
Email: signal@humanjudgment.org
URI: https://github.com/hjs-spec

v0.6 Direct Upgrade Note

This document is part of the JEP v0.6 draft set. The v0.6 set directly incorporates standardization structure, interoperability profiles, conformance artifacts, schemas, signed test vectors, invalid cases, and multi-language validator seeds without changing the JEP-Core narrow-waist semantics.