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

Y. Wang
April 29, 2026

Judgment Event Protocol (JEP)
A Minimal Verifiable Log Format for Agent Decisions
draft-wang-jep-judgment-event-protocol-05

Abstract

This document defines the Judgment Event Protocol (JEP), a minimal verifiable log format for decision-related operations in agent systems. JEP specifies four immutable event verbs (J, D, T, V), a signed JSON event structure, anti-replay protection, detached JSON Web Signature (JWS) verification over JSON Canonicalization Scheme (JCS) canonicalized payloads, event hash and reference semantics, validation modes, and a modular extension framework.

JEP is intended to provide global protocol-level interoperability for verifiable decision event records. It does not define legal liability, authorization validity, regulatory compliance, or organizational trust decisions.

Table of Contents

1. Introduction

1.1. Motivation

Autonomous and semi-autonomous agent systems increasingly make, assist with, or record decisions across organizational, platform, and jurisdictional boundaries. These systems need a minimal and interoperable way to record decision-related actions so that later verifiers can determine whether a particular event was emitted by a particular actor, whether the event has been modified, and whether it relates to other events in a verifiable chain.

JEP addresses this need by defining a compact signed JSON event format. The format is intentionally small: the protocol identifies a decision-related action, binds it to an actor, timestamp, nonce, optional audience, related content digest, related event reference, and signature.

JEP is designed to adapt to diverse cultural, legal, political, and operational environments. It achieves this by separating the stable event format from optional extension and trust profiles. This document defines the protocol-level mechanics; deployers and verifiers apply their own legal, organizational, and policy requirements.

JEP also does not assume that every external target fact is determinable from the available event log. In partially observed causal systems, logs, traces, measurements, or explanations can support verification and audit workflows without guaranteeing zero-error determination of external facts [TARGET-DETERMINABILITY]. JEP therefore records verifiable event claims rather than global fact determination.

1.2. Scope

JEP defines:

* Four immutable core decision event verbs: J, D, T, and V.

* A minimal signed JSON event object.

* Algorithm-tagged digest strings for content and event references.

* Event hash calculation and reference semantics.

* Detached JWS signature verification over JCS-canonicalized unsigned events.

* Anti-replay protection using nonce, timestamp, and optional audience binding.

* Acceptance validation and archival validation modes.

* A modular extension framework using "ext" and "ext_crit".

* Optional standard extensions for privacy, accountability distribution, lifecycle metadata, storage adaptation, subject reference, and cryptographic capability profiles.

JEP explicitly does not define:

* Legal liability or accountability determination rules.

* Authorization delegation validity or permission chain constraints.

* State machine or lifecycle enforcement logic.

* Regulatory, governance, or business compliance determination.

* Determinability of external target facts from partial observations.

* A global identity, credential, or trust framework.

* A confidentiality mechanism for event content.

1.3. 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.

1.4. Terminology

Event:
: A single observable record of a decision-related action.

Actor:
: The entity identified by "who" that claims to generate and sign a JEP event.

Subject:
: An entity referenced by an event, typically the person, agent, object, or system about which a decision is made. A subject is distinct from the actor.

Verifier:
: An entity that evaluates event structure, signatures, references, replay properties, extensions, and local policy.

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

JEP Signing Payload:
: The UTF-8 octets of the JCS-canonicalized unsigned event. This payload is supplied as the JWS Payload for detached JWS processing.

Event Hash:
: A protocol-level identifier calculated over the full signed event object, including the "sig" member.

Acceptance Validation:
: Verification performed when deciding whether to accept a newly received event. This mode includes replay and timestamp freshness checks.

Archival Validation:
: Verification performed when evaluating a previously recorded event. This mode does not reject an event solely because its timestamp is outside the current freshness window.

Trust Profile:
: A deployment-defined profile that binds actor identifiers and key identifiers to verification keys and defines key rotation, revocation, and historical validation rules.

Extension:
: A named object inside the "ext" member that adds deployment-specific or standardized semantics without changing the core JEP event structure.

2. Protocol Model

2.1. Design Goals

JEP has the following design goals:

* Minimality: the core event object is small and stable.

* Verifiability: event integrity is protected by a detached JWS signature over a JCS-canonicalized unsigned event.

* Algorithm agility: cryptographic algorithms are selected by JOSE headers, conformance profiles, and trust profiles rather than by event verbs.

* Global protocol interoperability: conforming implementations can parse, canonicalize, sign, reference, and verify the same event object.

* Policy neutrality: JEP records claims and evidence but does not decide legal, organizational, or regulatory consequences.

* Long-term operability: the protocol distinguishes real-time acceptance validation from archival validation.

2.2. Event Verbs

JEP defines four immutable core verbs:

Verb	Name	Semantics
J	Judge	Records the initiation or issuance of a

		decision-related action.
D	Delegate	Records a delegation claim or delegation-related event. JEP does not determine whether the delegation is legally, organizationally, or operationally valid.
T	Terminate	Records a termination claim for a referenced decision lifecycle or event chain. JEP does not itself enforce lifecycle closure.
V	Verify	Records a verification action or verification result concerning a referenced JEP event.

Algorithm choice, signature capability, privacy mode, retention policy, storage location, and identity resolution do not create new verbs. Those properties are represented by JOSE headers, trust profiles, or extensions.

2.3. Core Event Object

A JEP event is a JSON object. Producers MUST emit event objects that are compatible with I-JSON [RFC7493] and JSON [RFC8259]. Producers MUST NOT emit duplicate JSON member names. Verifiers MUST reject events containing duplicate JSON member names.

The "jep" member identifies the wire-format major version and is fixed as the string "1" for this specification. The Internet-Draft revision number is not the wire-format version.

The top-level extensibility mechanism is the "ext" member. Producers SHOULD NOT define new top-level members outside this specification unless those members are defined by a future JEP revision or a registered JEP extension that explicitly updates the top-level event object.

The following top-level members are used by JEP:

Member	Status	Description
jep	REQUIRED	Wire-format version. MUST be "1".
verb	REQUIRED	One of "J", "D", "T", or "V".
who	REQUIRED	Actor identifier.
when	REQUIRED	Unix timestamp in seconds.
what	Conditional	Digest of related content, descriptor, or report.
nonce	REQUIRED	UUIDv4 nonce.
aud	RECOMMENDED	Intended recipient or audience.
ref	Conditional	Reference to a related JEP event or chain.
ext	OPTIONAL	Extension object.
ext_crit	OPTIONAL	Critical extension identifier list.
sig	REQUIRED	Detached JWS signature.

The "when" value is a JSON integer representing seconds since the Unix epoch. It MUST be within the exact integer range permitted by I-JSON.

The "what" member:

* MUST be a non-null algorithm-tagged digest string for J, D, and T events.

* MAY be null for V events when the event records only that a verification action occurred.

* SHOULD be a non-null algorithm-tagged digest string for V events that record a verification result or report.

The "ref" member:

* SHOULD be null for root J events.

* SHOULD reference the relevant prior event or lifecycle event for D and T events.

* MUST be non-null for V events and SHOULD contain the event hash of the event being verified.

The "aud" member is RECOMMENDED for acceptance validation. If present, it MUST be included in the signed payload and MUST be checked by an acceptance validator against the expected recipient or audience.

2.4. Algorithm-Tagged Digest Strings

JEP uses algorithm-tagged digest strings for "what", "ref", event hashes, and extension integrity fields.

The syntax is:

```
<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-1 MUST support "sha256". A sha256 digest string contains exactly 64 hexadecimal characters after the colon.

Example:

```
sha256:3a6eb0790f39ac87c94f3856b2dd2c5d110e6811602261a9a923d3bb23adc8b7
```

Deployments MAY define additional digest algorithms, such as regional or post-quantum-adjacent profiles, if the algorithm identifier, digest length, and verification rules are specified by the applicable trust profile or registered extension.

This syntax is defined by this document for JEP interoperability.

2.5. Event Hashes and Reference Semantics

The event hash of a JEP event is calculated as follows:

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

where "full_signed_event" is the complete JEP event object including the "sig" member.

The textual representation of the event hash is the algorithm-tagged digest string:

```
sha256:<lowercase-hex-digest>
```

The signing input and the event hash are intentionally different:

* The signing input is the unsigned event with "sig" omitted.

* The event hash identifies the full signed event with "sig" included.

The "ref" member SHOULD contain the event hash of the referenced JEP event unless a registered extension or trust profile explicitly defines another reference scheme.

A reference chain proves cryptographic linkage between event objects. It does not by itself prove that the chain is complete, append-only, or free from omission. Append-only log semantics, transparency logs, or storage-level guarantees are outside the JEP core scope.

2.6. Signing Input and Signature Container

JEP uses JSON Canonicalization Scheme (JCS) [RFC8785] and JSON Web Signature (JWS) [RFC7515].

To create a JEP signature, a producer MUST perform the following steps:

1. Construct the unsigned event object with all top-level members except "sig".

2. Canonicalize the unsigned event object using JCS.

3. Encode the canonicalized unsigned event as UTF-8 octets. These octets are the JEP signing payload.

4. Supply the JEP signing payload as the JWS Payload.

5. Create a detached JWS over that payload according to JWS rules.

6. Insert the detached JWS into the "sig" member.

For JWS Compact Serialization, the detached payload form is used by replacing the second segment with the empty string. The resulting compact representation has the form:

```
BASE64URL(Protected) || "." || "." || BASE64URL(Signature)
```

For JWS JSON Serialization, the detached payload form is used by omitting the "payload" member. The omitted payload is the JEP signing payload.

For single-signature JEP-Core-1 events, "sig" SHOULD be a detached JWS Compact Serialization string. For multi-signature or composite signature profiles, "sig" MAY be a detached JWS JSON Serialization object or another registered signature container defined by an applicable profile.

The JWS Protected Header MUST contain "alg". The "alg" value "none" MUST NOT be used. The JWS Protected Header SHOULD contain "kid".

A JEP-Core-1 producer MUST use the fully specified JOSE algorithm identifier "Ed25519" for Ed25519 signatures. A JEP-Core-1 verifier MUST support "Ed25519". A verifier MAY accept legacy "EdDSA" under a local compatibility policy, but producers SHOULD NOT emit "EdDSA" for JEP-Core-1 events.

A verifier MUST reject an event if the JOSE algorithm is unsupported, prohibited by local policy, inconsistent with the resolved verification key, or inconsistent with a critical cryptographic extension.

2.7. Key Resolution and Trust Profiles

JEP does not define a global identity or trust framework.

The "who" member identifies the actor that claims to have generated the event. The JWS "kid" header identifies the verification key used for the signature.

A verifier MUST resolve "who" and "kid" to a verification key

according to an applicable trust profile. A trust profile MAY use DID resolution, JWKS, X.509 certificates, public key hashes, local configuration, or another mechanism.

A verifier MUST reject an event if the resolved key is not bound to the actor identified by "who" under the applicable trust profile.

Trust profiles SHOULD define:

* Supported actor identifier forms.

* Key identifier syntax and key discovery.

* Binding rules between "who" and "kid".

* Key rotation handling.

* Revocation handling.

* Historical key validity for archival validation.

* Accepted algorithms and algorithm downgrade policy.

2.8. Anti-Replay and Validation Modes

Event initiators MUST generate a fresh UUIDv4 nonce [RFC9562] for each event. Nonces MUST be generated using a cryptographically secure random source.

For acceptance validation, a receiver MUST maintain a replay cache. The replay cache scope MUST include "who" and "nonce". If "aud" is present, the replay cache scope SHOULD include "aud" as well.

Unless otherwise configured by a trust profile, a freshness window of plus or minus 300 seconds is RECOMMENDED for acceptance validation.

2.8.1. Acceptance Validation

Acceptance validation is used when a receiver decides whether to accept a newly received event.

A verifier performing acceptance validation MUST:

1. Parse and validate the JSON event structure, including rejection of duplicate member names.

2. Validate top-level field requirements for the event verb.

3. Reconstruct the unsigned event by omitting "sig".

4. Canonicalize the unsigned event using JCS.

5. Verify the detached JWS signature using the reconstructed JEP signing payload.

6. Resolve "who" and "kid" according to the applicable trust profile.

7. Verify nonce uniqueness within the replay cache.

8. Verify that "when" falls within the configured freshness window.

9. Verify "aud" if present.

10. Verify "what" and "ref" syntax if present.

11. Process all extensions listed in "ext_crit".

12. Apply local verifier policy.

2.8.2. Archival Validation

Archival validation is used when a verifier evaluates a previously recorded event.

A verifier performing archival validation MUST:

1. Parse and validate the JSON event structure, including rejection of duplicate member names.

2. Validate top-level field requirements for the event verb.

3. Reconstruct the unsigned event by omitting "sig".

4. Canonicalize the unsigned event using JCS.

5. Verify the detached JWS signature using the reconstructed JEP signing payload.

6. Verify the event hash if an expected event hash is supplied.

7. Verify "ref" chain integrity if applicable.

8. Resolve historical key validity according to the applicable trust profile, if such information is available.

9. Process all extensions listed in "ext_crit".

10. Apply local verifier policy.

Archival validation MUST NOT reject an event solely because "when" is outside the current freshness window.

2.9. Extension Framework

Extensions are represented by the optional "ext" member. The "ext" member is a JSON object. Each member name inside "ext" is an extension identifier.

The optional "ext_crit" member is an array of extension identifiers that are critical to validation or interpretation.

The "ext" and "ext_crit" members, when present, are included in the JEP signing payload.

If a verifier encounters an extension listed in "ext_crit" that it does not understand or cannot process, it MUST reject the event.

If a verifier encounters an extension not listed in "ext_crit" that it does not understand, it MAY ignore that extension.

Extension identifiers are stable identifiers. They are not required to be dereferenceable.

Example:

```
{
  "ext": {
    "https://jep.org/ttl": {
      "expires_at": 1774567890,
      "expiry_action": "delete_plaintext_retain_hash"
    }
  },
```

```
  "ext_crit": [
    "https://jep.org/ttl"
  ]
}
```

2.10. Standard Extensions

This section defines standard JEP extensions. Each extension is optional unless listed in "ext_crit" or required by a trust profile.

2.10.1. Digest-Only Anonymity Extension

Identifier: https://jep.org/priv/digest-only

Purpose: support pseudonymous actor or subject representation while enabling traceability under a deployment-defined dispute or audit process.

Example:

```
{
  "https://jep.org/priv/digest-only": {
    "identity_digest": "sha256:0123456789abcdef0123456789abcdef0123456789abcdef0123456
789abcdef",
    "digest_alg": "sha256",
    "salt_holder": "did:example:escrow-1",
    "salt_policy": "profile:example-dispute-resolution-v1",
    "domain": "actor"
  }
}
```

The "identity_digest" field MUST be an algorithm-tagged digest string. The "domain" field SHOULD identify the digest domain, such as "actor" or "subject", to support domain separation.

This extension supports pseudonymization. It does not guarantee legal anonymity. Privacy depends on salt confidentiality, salt entropy, domain separation, and the entropy of the underlying identifier.

2.10.2. Multi-Signature Extension

Identifier: https://jep.org/multisig

Purpose: record collaborative decisions or approvals involving multiple parties, enabling distributed accountability.

Example:

```
{
  "https://jep.org/multisig": {
    "mode": "threshold",
    "threshold": 2,
    "participants": [
      "did:example:a#key-1",
      "did:example:b#key-1",
      "did:example:c#key-1"
    ]
  }
}
```

If this extension is listed in "ext_crit", a verifier MUST verify that at least "threshold" valid signatures over the same JEP signing payload correspond to the participant list.

A critical multisig event MUST use JWS JSON Serialization or another registered signature container capable of representing multiple

signatures over the same JEP signing payload.

2.10.3. Time-to-Live Extension

Identifier: https://jep.org/ttl

Purpose: express lifecycle metadata for plaintext data or linked evidence.

Example:

```
{
  "https://jep.org/ttl": {
    "expires_at": 1774567890,
    "expiry_action": "delete_plaintext_retain_hash",
    "retained_evidence": "sha256:0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef"
  }
}
```

The "expires_at" field is a Unix timestamp in seconds. The "expiry_action" field describes a declared action, such as "delete_plaintext_retain_hash" or "anonymize_plaintext_retain_hash".

TTL metadata can support data-retention workflows. JEP does not guarantee physical deletion, legal compliance, or fulfillment of any particular data-subject right.

2.10.4. Sovereign Storage Extension

Identifier: https://jep.org/storage

Purpose: identify where related content or evidence may be stored, while decoupling JEP from any particular storage vendor.

Example:

```
{
  "https://jep.org/storage": {
    "adapter_type": "uri",
    "storage_address": "https://storage.example/object/123",
    "integrity_hash": "sha256:0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "jurisdiction": "optional-string"
  }
}
```

This extension identifies storage metadata. JEP does not endorse or validate any storage provider, jurisdiction, or data sovereignty claim.

2.10.5. Subject Reference Extension

Identifier: https://jep.org/subject

Purpose: identify the subject of a decision to improve traceability transparency.

Example:

```
{
  "https://jep.org/subject": {
    "id_type": "did",
    "id": "did:example:user-123",
    "privacy_mode": "plaintext"
  }
```

```
}
```

Example using pseudonymous representation:

```
{
  "https://jep.org/subject": {
    "id_type": "salted_digest",
    "id": "sha256:0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "privacy_mode": "pseudonymous"
  }
}
```

A subject reference is informational. It does not imply consent, authorization, legal responsibility, or correctness of the referenced identity.

2.10.6. Cryptographic Profile Extension

Identifier: https://jep.org/crypto/profile

Purpose: declare cryptographic capabilities or preferences of an issuer, system, or event so that verifiers can discover signature surface, canonicalization profile, and hash family without changing event verbs.

Example:

```
{
  "https://jep.org/crypto/profile": {
    "scope": "issuer",
    "profile_id": "did:example:agent-789#crypto-profile-2026-04",
    "signature_capability": "composite",
    "signature_schemes": [
      "Ed25519",
      "ML-DSA-65"
    ],
    "composite_rule": "all-of",
    "canonicalization_profile": "jcs-rfc8785",
    "hash_family": [
      "sha256"
    ],
    "valid_from": 1742340000,
    "valid_until": null
  }
}
```

The "signature_capability" value MUST be one of the values registered in the JEP Signature Capability Values Registry. Initial values are "classical", "post_quantum", and "composite".

Absence of this extension means that cryptographic capability is undeclared. Security-sensitive verifiers SHOULD apply local policy rather than assuming undeclared capability is equivalent to classical capability.

The cryptographic profile extension is descriptive unless it is listed in "ext_crit" or made normative by an applicable trust profile.

JEP does not require post-quantum cryptography. JEP does not prescribe which signature capability a verifier must accept.

"composite" is distinct from "post_quantum". The distinction reflects a different cryptographic failure model, not merely a nominal security level.

Multi-signature and composite signature are distinct mechanisms.

Multi-signature distributes accountability across multiple actors or keys. Composite signature combines multiple cryptographic schemes for a single issuer or signing identity. Implementations MUST NOT treat a multisig threshold rule as equivalent to a composite cryptographic rule unless an applicable profile explicitly defines such equivalence.

2.11. Conformance Profiles

2.11.1. JEP-Core-1

A JEP-Core-1 producer MUST generate events with:

* "jep" equal to "1".
* A verb equal to "J", "D", "T", or "V".
* I-JSON compatible JSON.
* JCS canonicalization.
* Detached JWS Compact Serialization for single signatures.
* JOSE "alg" equal to "Ed25519".
* Algorithm-tagged digest strings using "sha256".
* UUIDv4 nonces.
* Correct event hash and "ref" semantics when references are used.
* "ext" and "ext_crit" semantics when extensions are present.

A JEP-Core-1 verifier MUST support:

* Parsing and validation of JEP event objects.
* Duplicate JSON member rejection.
* JCS canonicalization.
* Detached JWS Compact Serialization verification.
* JOSE "alg" equal to "Ed25519".
* Algorithm-tagged digest strings using "sha256".
* UUIDv4 nonce syntax.
* Event hash calculation.
* Acceptance validation.
* Archival validation.
* Unknown critical extension rejection.

2.11.2. JEP-Multisig-1

A JEP-Multisig-1 implementation MUST support:

* The https://jep.org/multisig extension.
* JWS JSON Serialization for multiple signatures over the same JEP signing payload, or another registered equivalent container.
* Threshold validation.

* Participant key binding according to the applicable trust profile.

2.11.3. JEP-Crypto-Profile-1

A JEP-Crypto-Profile-1 implementation MUST support:

* The https://jep.org/crypto/profile extension.

* "signature_capability".

* "signature_schemes".

* "canonicalization_profile".

* "hash_family".

* "scope".

2.11.4. JEP-PQC-1

JEP-PQC-1 is an optional deployment profile. It is not part of JEP-Core-1.

A JEP-PQC-1 profile SHOULD use registered JOSE algorithm identifiers for post-quantum signatures, such as ML-DSA algorithm identifiers registered for JOSE. The profile MUST define verification rules, accepted algorithms, key representation, and downgrade policy.

2.11.5. Regional Cryptographic Profiles

Regional cryptographic profiles MAY define additional algorithms, digest identifiers, or key resolution rules, such as SM2/SM3 profiles. Such profiles MUST define their algorithm identifiers, digest syntax, key representation, and verifier policy sufficiently for independent implementation.

3. Security and Privacy Considerations

3.1. Core Security Properties

A valid JEP signature proves that the signed event payload was signed by a key resolved under the applicable trust profile. It does not prove that the event content is true, legally effective, authorized, compliant, or complete.

The "sig" member protects event integrity for all signed fields in the unsigned event, including "ext" and "ext_crit" when present. Any modification to those fields invalidates the signature.

The "event_hash" identifies a particular full signed event object. It does not prove that no other event was omitted from a log.

JEP does not provide confidentiality. Sensitive data requires transport protection, object encryption, JWE [RFC7516], or another confidentiality control.

3.2. Threat Model and Non-Goals

Replay attacks:
: Acceptance validators MUST enforce nonce uniqueness within the configured replay cache and freshness window.

Archival validation:
: Archival validators MUST NOT reject an event solely because it is

outside the current freshness window.

Algorithm downgrade:
: Verifiers MUST enforce local algorithm policy and reject prohibited or unsupported algorithms.

Extension downgrade:
: Critical extensions listed in "ext_crit" MUST be understood and processed. Unknown critical extensions MUST cause rejection.

Key compromise:
: Key compromise, rotation, and revocation are handled by trust profiles. JEP does not define a global revocation system.

Identity substitution:
: Verifiers MUST ensure that the resolved "kid" is bound to "who" according to the applicable trust profile.

Log truncation and omission:
: JEP references can link events but do not guarantee append-only logging, completeness, or non-equivocation.

Timestamp manipulation:
: The "when" value is an actor-supplied timestamp. Trust profiles or external timestamping systems can provide stronger time evidence.

Legal and policy decisions:
: JEP records verifiable event evidence. It does not determine legal liability, consent, authorization validity, delegation validity, or regulatory compliance.

3.3. Verification Limits under Partial Observation

A JEP event can be structurally valid and signature-valid while still being insufficient to determine an external target fact. In partially observed causal systems, multiple real histories can yield the same observed log projection while differing on the target fact of interest [TARGET-DETERMINABILITY].

Verifiers therefore need external evidence policies, trust profiles, and domain-specific procedures when using JEP events for audit, accountability, or dispute-resolution decisions. Archival validation checks the integrity and linkage of recorded events; it does not establish that the available observations are complete or determinative.

3.4. Extension Security Considerations

Digest-only anonymity:
: Privacy depends on salt confidentiality, salt entropy, domain separation, and identifier entropy. Low-entropy identifiers can be vulnerable to dictionary attacks.

Multi-signature:
: Implementations need to protect against malicious key substitution, participant-list ambiguity, and threshold downgrade.

TTL:
: TTL declares lifecycle metadata but does not enforce physical deletion. Deletion workflows can retain verifiable hashes or logs, depending on policy.

Storage:
: Security of external storage is outside the JEP core scope. Storage integrity SHOULD be checked using digest fields.

Subject reference:
: Subject references can expose personal or sensitive information. Deployers SHOULD minimize subject identifiers and use pseudonymous representations when appropriate.

Cryptographic profile:
: A cryptographic profile can help verifiers discover algorithm capability, but it does not replace actual signature verification or local algorithm policy.

3.5. Privacy and Data Minimization

Deployers SHOULD minimize directly identifying information in event objects. When direct identifiers are not required, deployers SHOULD consider digest-only or pseudonymous extensions.

The subject reference extension SHOULD be enabled only when necessary to support transparency, auditability, or dispute resolution.

TTL metadata can support retention workflows but MUST NOT be described as a guarantee of legal compliance or physical deletion.

4. IANA Considerations

4.1. JEP Verbs Registry

IANA is requested to create the "JEP Verbs Registry".

The registration policy is IETF Review.

New verb registrations are NOT RECOMMENDED. A new verb registration requires an RFC that updates this specification and explains why the semantics cannot be represented by an existing verb plus a registered extension.

Registration template:

* Value:
* Name:
* Description:
* Reference:
* Change controller:
* Security considerations:
* Privacy considerations:

Initial registrations:

Value	Name	Description	Reference
J	Judge	Decision-related action	This document
D	Delegate	Delegation claim	This document
T	Terminate	Termination claim	This document
V	Verify	Verification action or result	This document

4.2. JEP Extensions Registry

IANA is requested to create the "JEP Extensions Registry".

The registration policy is Specification Required [RFC8126].

Registration template:

* Extension identifier:

* Short name:

* Description:

* Specification URI:

* Criticality allowed: yes/no

* Required fields:

* Optional fields:

* Validation impact:

* Security considerations:

* Privacy considerations:

* Change controller:

Initial registrations:

Extension Identifier	Short Name	Reference
https://jep.org/priv/digest-only	digest-only	This document
https://jep.org/multisig	multisig	This document
https://jep.org/ttl	ttl	This document
https://jep.org/storage	storage	This document
https://jep.org/subject	subject	This document
https://jep.org/crypto/profile	crypto-profile	This document

4.3. JEP Signature Capability Values Registry

IANA is requested to create the "JEP Signature Capability Values Registry".

The registration policy is Specification Required [RFC8126].

Registration template:

* Value:

* Description:

* Reference:

* Security considerations:

Initial registrations:

Value	Description	Reference
classical	Uses non-post-quantum signature schemes.	This document
post_quantum	Uses post-quantum signature schemes.	This document
composite	Combines multiple cryptographic	This document

	schemes under a defined verification rule.	

4.4. JEP Conformance Profiles Registry

IANA is requested to create the "JEP Conformance Profiles Registry".

The registration policy is Specification Required [RFC8126].

Registration template:

* Profile name:

* Description:

* Required features:

* Specification URI:

* Security considerations:

* Privacy considerations:

Initial registrations:

Profile Name	Description	Reference
JEP-Core-1	Core event interoperability	This document
JEP-Multisig-1	Multisignature extension	This document
JEP-Crypto-Profile-1	Crypto profile extension	This document

5. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015.

[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>.

[RFC7493] Bray, T., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC8037] Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, DOI 10.17487/RFC8037, January 2017, <https://www.rfc-editor.org/info/rfc8037>.

[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>.

[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>.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

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

[RFC9864] Jones, M. B. and O. Steele, "Fully-Specified Algorithms for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE)", RFC 9864, DOI 10.17487/RFC9864, October 2025, <https://www.rfc-editor.org/info/rfc9864>.

6. Informative References

[DID-CORE] W3C, "Decentralized Identifiers (DIDs) v1.0", W3C Recommendation, July 2022, <https://www.w3.org/TR/did-core/>.

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", FIPS PUB 204, DOI 10.6028/NIST.FIPS.204, August 2024.

[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>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Examples and Test Vectors

This appendix provides a signed JEP-Core-1 example. The test vector uses fixed Ed25519 private keys for reproducibility. Production systems MUST NOT use these keys.

A.1. Public Keys

Actor public JWK:

{

```
  "kty": "OKP",
  "crv": "Ed25519",
  "kid": "did:example:agent-789#key-1",
  "x": "A6EHv_POEL4dcN0Y50vAmWfk1jCbpQ1fHdyGZBJVMbg"
}
```

Verifier public JWK:

```
{
  "kty": "OKP",
  "crv": "Ed25519",
  "kid": "did:example:verifier-123#key-1",
  "x": "Kay64UG8yvCyLhqU000LxzYeUm0L_hLIl5S8kyKWbdc"
}
```

A.2. Signed Judgment Event

The following event uses detached JWS Compact Serialization in "sig":

```
{
  "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,
  "ext": {
    "https://jep.org/crypto/profile": {
      "scope": "issuer",
      "signature_capability": "classical",
      "signature_schemes": [
        "Ed25519"
      ],
      "canonicalization_profile": "jcs-rfc8785",
      "hash_family": [
        "sha256"
      ]
    }
  },
  "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6YWdlbnQtNzg5I2tleS0xIn0..toFm0e
HSPXiSdy3CM6U0HXM93b3JACjyOM6hjdxMG4NXB85mcdxUS74a4GO9bZ-ahFKNVDoKTEXWK9UhPfYRBA"
}
```

Event hash:

sha256:1ea7989431a7f21cfcd5300284c4f6dcdcff885ba004942654aeb5916ddf2558

A.3. Signed Verification Event

The following V event references the event hash of the judgment event:

```
{
  "jep": "1",
  "verb": "V",
  "who": "did:example:verifier-123",
  "when": 1742345680,
  "what": "sha256:11433ad3dd3794f3607bef1c13b5a68939df8753dbcf4b79ebe66f64010b8e25",
  "nonce": "a1b2c3d4-5678-4abc-8ef0-123456789abc",
  "aud": "https://platform.example.com",
  "ref": "sha256:1ea7989431a7f21cfcd5300284c4f6dcdcff885ba004942654aeb5916ddf2558",
  "sig": "eyJhbGciOiJFZDI1NTE5Iiwia2lkIjoiZGlkOmV4YW1wbGU6dmVyaWZpZXItMTIzI2tleS0xIn0..Up
B0JKwtHYXvcrGeBd8EFJsEN0iE5c_s9YyO5QRdqboDDysq5EFPd3G42OxbSRSVn4z4h4aWtKN0kYK7mIqnDQ"
}
```

Event hash:

sha256:34affe990f7f09e5a623f66f80d318fad861346fc2064d8a454ff512a30738c8

A.4. Canonicalization Note

To verify the examples, remove the "sig" member, canonicalize the remaining event object using JCS, and supply the resulting UTF-8 octets as the detached JWS payload. Reconstruct the JWS signing input according to RFC 7515.

Appendix B. Changes from -04

The following major changes were made from draft-wang-jep-judgment-event-protocol-04 to this revision:

* Clarified the JEP signing input and adopted detached JWS semantics.
* Defined event hash calculation and reference semantics.
* Split validation into acceptance validation and archival validation.
* Clarified required, recommended, optional, and verb-specific fields.
* Replaced ambiguous multihash language with algorithm-tagged digest strings.
* Added key resolution and trust profile requirements.
* Added "ext" and "ext_crit" extension container semantics.
* Added minimal schemas for standard extensions.
* Added the cryptographic profile extension for signature capability, canonicalization profile, and hash family.
* Distinguished multi-signature from composite signature.
* Added conformance profiles.
* Expanded security and privacy considerations.
* Added an informative reference and security-boundary note on target determinability under partial causal observation.
* Expanded IANA registration templates and expert guidance.
* Updated normative and informative references.

Author’s Address

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