

Independent Submission
Internet-Draft
Intended status: Informational
Expires: 1 December 2026

C. Hopley
AlgoVoi
30 May 2026

Payment Evidence Frame for Agentic-Payment Lifecycle Receipts
draft-hopley-x402-payment-evidence-frame-00

Abstract

This document specifies a Payment Evidence Frame (PEF): a transport-agnostic envelope that wraps any payment lifecycle receipt from the x402 agentic-payment receipt stack under a named claim type, a deterministic frame identifier, and an optional transport-layer signature.

The frame introduces three properties that the inner receipt types do not individually provide: (1) a stable cross-system identifier (`frame_id`) derived deterministically from the receipt content by SHA-256 over the JCS-canonical preimage, so the same payment evidence can be referenced by identifier across HTTP headers, agent-to-agent task artifacts, audit logs, and on-chain memo fields without re-serialising the full receipt; (2) a taxonomy label (`claim_type`) that tells a consumer what class of evidence a frame carries before the inner receipt format is parsed; and (3) a receipt integrity hash (`receipt_hash`) that allows any downstream party to confirm the inner receipt is unaltered without deserialising its full structure.

The format uses the same JCS canonicalisation discipline (`draft-hopley-x402-canonicalisation-jcs`) as the five inner receipt formats it envelopes, so implementations require only one canonicalisation primitive for the full receipt stack.

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 1 December 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

| | | | |
|-------|-------|--|----|
| 1. | 1. | Introduction | 3 |
| 1.1. | 1.1. | Motivation | 3 |
| 1.2. | 1.2. | Scope | 4 |
| 1.3. | 1.3. | Relationship to Other Internet-Drafts | 5 |
| 2. | 2. | Conventions and Definitions | 6 |
| 2.1. | 2.1. | Notation | 6 |
| 2.2. | 2.2. | Definitions | 6 |
| 3. | 3. | Frame Format Specification | 7 |
| 3.1. | 3.1. | pef_version | 8 |
| 3.2. | 3.2. | claim_type | 8 |
| 3.3. | 3.3. | receipt_format | 9 |
| 3.4. | 3.4. | receipt | 10 |
| 3.5. | 3.5. | receipt_hash | 10 |
| 3.6. | 3.6. | frame_id | 11 |
| 3.7. | 3.7. | frame_provider_id | 11 |
| 3.8. | 3.8. | frame_timestamp_ms | 11 |
| 3.9. | 3.9. | canon_version | 11 |
| 3.10. | 3.10. | signature (OPTIONAL) | 12 |
| 4. | 4. | Frame ID Derivation | 12 |
| 4.1. | 4.1. | Verification | 13 |
| 4.2. | 4.2. | Stability Property | 13 |
| 5. | 5. | Receipt Hash Derivation | 13 |
| 6. | 6. | Canonicalisation | 14 |
| 7. | 7. | Composition with x402 Receipt Stack and Transport Protocols | 14 |
| 7.1. | 7.1. | Receipt Stack Composition | 14 |
| 7.2. | 7.2. | x402 Transport | 15 |
| 7.3. | 7.3. | Agent-to-Agent (A2A) Task Artifacts | 16 |
| 7.4. | 7.4. | MPP Payment Method Response | 17 |
| 7.5. | 7.5. | Audit Chain Integration | 17 |
| 8. | 8. | Year-N Auditability Properties | 17 |
| 9. | 9. | IANA Considerations | 18 |
| 9.1. | 9.1. | URN Namespace Registration | 18 |

| | | | |
|------------------|-------------|---|----|
| 9.2. | 9.2. | Frame Format Identifier | 18 |
| 9.3. | 9.3. | Claim Type Registry | 18 |
| 10. | 10. | Security Considerations | 19 |
| 10.1. | 10.1. | Frame Tampering | 19 |
| 10.2. | 10.2. | Preimage Exclusion | 19 |
| 10.3. | 10.3. | Frame Provider Identity | 19 |
| 10.4. | 10.4. | Claim Type Mismatch | 19 |
| 10.5. | 10.5. | Receipt Hash Collision | 19 |
| 10.6. | 10.6. | Signature Absence | 20 |
| 10.7. | 10.7. | Timestamp Integrity | 20 |
| 11. | 11. | References | 20 |
| 11.1. | 11.1. | Normative References | 20 |
| 11.2. | 11.2. | Informative References | 20 |
| Appendix A. | Appendix A. | Examples (Informative) | 21 |
| A.1. | A.1. | payment_admission -- compliance screen ALLOW | 21 |
| A.2. | A.2. | payment_settlement -- settled on-chain | 22 |
| A.3. | A.3. | payment_cancellation -- mandate terminated by compliance | 22 |
| A.4. | A.4. | payment_refund -- partial refund issued | 23 |
| A.5. | A.5. | composite_verdict -- verifier-of-verifier TRUSTED | 24 |
| A.6. | A.6. | Signed Frame (OPTIONAL signature field) | 24 |
| Appendix B. | Appendix B. | Reference Implementations (Informative) | 25 |
| Appendix C. | Appendix C. | Known Adopters (Informative) | 25 |
| Appendix D. | Appendix D. | Acknowledgments | 25 |
| Author's Address | | | 26 |

1. 1. Introduction

1.1. 1.1. Motivation

The x402 agentic-payment receipt stack defines five categorical receipt formats for the distinct events in a payment lifecycle:

- * draft-hopley-x402-compliance-receipt -- pre-payment admission screening (ALLOW / REFER / DENY).
- * draft-hopley-x402-settlement-attestation -- per-execution settlement confirmation (SETTLED / PENDING_FINALITY / REVERSED).
- * draft-hopley-x402-cancellation-receipt -- mandate termination (USER_REQUESTED / MERCHANT_REQUESTED / COMPLIANCE_TERMINATED / EXPIRED).
- * draft-hopley-x402-refund-receipt -- post-settlement refund (FULL / PARTIAL / REJECTED).
- * draft-hopley-x402-composite-trust-query -- verifier-of-verifier conclusion over an audit chain composed of the above (TRUSTED / PROVISIONAL / INSUFFICIENT_EVIDENCE / UNTRUSTED).

Each receipt is a self-contained JCS-canonical JSON object with a content hash. Each is independently verifiable. Each is designed to be retained in an audit chain via hash-chain linkage.

A gap exists at the transport and interoperability layer. A compliance receipt emitted by system A and consumed by system B must be carried in some envelope that system B can inspect without knowing in advance which of the five receipt formats it will find. An agent-to-agent task artifact, an HTTP response extension field, a memo field on a blockchain transaction, and a webhook payload all face the same need: carry a receipt with enough metadata that the consumer can route it, verify its integrity, and correlate it with other events in the lifecycle -- without parsing the inner format first.

Three specific gaps motivate this document:

1. **Stable cross-system identifier.** A compliance receipt's content hash changes if any field changes. There is no stable identifier for "the payment evidence bundle associated with transaction T" that persists across the HTTP layer, the agent layer, and the audit-chain layer without carrying the full receipt bytes at every hop.
2. **Taxonomy label.** A consumer receiving a JSON blob over a generic channel (e.g. an A2A task artifact, an HTTP header) has no standard signal for which of the five receipt types it holds before deserialising the full struct. Dispatch logic cannot switch on receipt type without parsing.
3. **Integrity without deserialisation.** A consumer that only needs to confirm "this receipt has not been altered since the issuer signed the frame" must today fully parse the inner receipt and recompute its content hash. A receipt hash in the frame envelope allows integrity checks to be performed at the frame layer.

The Payment Evidence Frame specified in this document addresses all three gaps with a ten-field envelope that adds no new canonicalisation rules and no new cryptographic primitives.

1.2. 1.2. Scope

This document specifies:

- * The canonical JSON shape of the Payment Evidence Frame (Section 3).
- * The frame identifier derivation algorithm (frame_id, Section 4).
- * The receipt hash derivation algorithm (receipt_hash, Section 5).

- * The reference to the canonicalisation discipline applicable to the frame (Section 6 -- normative reference to draft-hopley-x402-canonicalisation-jcs, not redefined inline).
- * The claim type taxonomy (Section 3.2).
- * Composition of PEF with the five inner receipt formats and with transport protocols (Section 7).
- * Year-N auditability properties (Section 8).
- * Worked examples covering all five claim types (Appendix A).

This document does NOT specify:

- * The inner receipt formats. They are specified in their respective I-Ds (draft-hopley-x402-compliance-receipt, etc.). The PEF envelope is additive; it does not redefine any inner receipt field.
- * The transport protocol. PEF is a JSON shape; where it appears (HTTP response body, HTTP header, A2A task artifact, MPP method response, on-chain memo) is out of scope.
- * The optional signature value. The signature field carries a pre-computed RFC 9421 signature string when supplied; the signing procedure is defined in draft-hopley-x402-rfc9421-x402-binding, not redefined here.

1.3. 1.3. Relationship to Other Internet-Drafts

This document normatively references:

- * draft-hopley-x402-canonicalisation-jcs -- the JCS canonicalisation discipline pinned in Section 6.

This document is complementary to:

- * draft-hopley-x402-compliance-receipt -- provides the compliance-receipt-v1 inner receipt format for frames with claim_type: payment_admission.
- * draft-hopley-x402-settlement-attestation -- provides the settlement-attestation-v1 inner receipt format for frames with claim_type: payment_settlement.
- * draft-hopley-x402-cancellation-receipt -- provides the cancellation-receipt-v1 inner receipt format for frames with claim_type: payment_cancellation.
- * draft-hopley-x402-refund-receipt -- provides the refund-receipt-v1 inner receipt format for frames with claim_type: payment_refund.
- * draft-hopley-x402-composite-trust-query -- provides the composite-trust-query-v1 inner receipt format for frames with claim_type: composite_verdict.
- * draft-hopley-x402-rfc9421-x402-binding -- defines the signing procedure for the optional signature field.

PEF sits above all five receipt formats. It does not replace them; it envelopes them for transport and cross-system reference.

2. 2. Conventions and Definitions

2.1. 2.1. Notation

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.

2.2. 2.2. Definitions

***Payment Evidence Frame (PEF)*:** a JSON object of the shape specified in Section 3, wrapping one inner receipt from the x402 agentic-payment receipt stack under a named claim type and a deterministic frame identifier.

***frame_id*:** a string of the form sha256:<lowercase-hex-64> derived from the JCS-canonical bytes of the PEF preimage (the frame object with frame_id and signature fields excluded). See Section 4 for the derivation algorithm. The frame_id is stable: it does not change if the signature field is subsequently added, and it does not change across independent re-derivations from identical content.

***receipt_hash*:** a string of the form sha256:<lowercase-hex-64> equal to the SHA-256 of the JCS-canonical bytes of the receipt field value. See Section 5.

***claim_type*:** a string-valued taxonomy label drawn from the closed five-element enumeration in Section 3.2. Identifies the lifecycle event the frame represents and the expected inner receipt format.

***receipt_format*:** a string-valued identifier for the inner receipt format, drawn from the closed mapping in Section 3.3.

***frame_provider_did*:** a DID URI identifying the party that constructed and issued the PEF frame.

***frame_timestamp_ms*:** an integer Unix epoch millisecond timestamp at which the frame was constructed.

***preimage*:** the PEF frame object with the frame_id and signature fields removed, used as input to the frame_id derivation algorithm. The preimage is itself a valid JSON object and is canonicalised under draft-hopley-x402-canonicalisation-jcs before hashing.

canon_version: an in-band string identifying the canonicalisation discipline. For this version of the format, the full URN is `urn:x402:canonicalisation:jcs-rfc8785-v1`.

3. 3. Frame Format Specification

A Payment Evidence Frame is a JSON object with ten fields, of which nine are REQUIRED and one (signature) is OPTIONAL. The frame is canonicalised under `draft-hopley-x402-canonicalisation-jcs` per Section 6. Field names sort lexicographically by JCS during canonicalisation; the object uses arbitrary authoring order.

The ten fields are:

| Field | Type | Required | Description |
|---------------------------------|---------|----------|---|
| <code>canon_version</code> | string | REQUIRED | Canonicalisation URN. Fixed value for this version. |
| <code>claim_type</code> | string | REQUIRED | Lifecycle event taxonomy. Closed five-element enum. |
| <code>frame_id</code> | string | REQUIRED | SHA-256 of JCS(preimage). Stable cross-system identifier. |
| <code>frame_provider_did</code> | string | REQUIRED | DID URI of the party that constructed the frame. |
| <code>frame_timestamp_ms</code> | integer | REQUIRED | Unix epoch milliseconds at which the frame was constructed. |
| <code>pef_version</code> | string | REQUIRED | Format version. Fixed value 1 for this version. |
| <code>receipt</code> | object | REQUIRED | The inner receipt, embedded verbatim. |
| <code>receipt_format</code> | string | REQUIRED | Inner receipt |

| | | | |
|--------------|--------|----------|---|
| | | | format identifier. Closed mapping per Section 3.3. |
| receipt_hash | string | REQUIRED | SHA-256 of JCS(receipt). Integrity check without deserialisation. |
| signature | string | OPTIONAL | Pre-computed RFC 9421 transport- layer signature, when present. |

Table 1

3.1. 3.1. pef_version

A string-valued field. The value MUST be 1 for this version of the format. Implementations MUST reject any other value at validation time.

3.2. 3.2. claim_type

A string-valued field drawn from the following closed five-element enumeration:

| | | |
|----------------------|---|---------------------------|
| claim_type | Lifecycle event | Expected receipt_format |
| payment_admission | Pre-payment compliance screen | compliance-receipt-v1 |
| payment_settlement | Per-execution settlement confirmation | settlement-attestation-v1 |
| payment_cancellation | Mandate or order termination | cancellation-receipt-v1 |
| payment_refund | Post-settlement refund | refund-receipt-v1 |
| composite_verdict | Verifier-of-verifier audit chain conclusion | composite-trust-query-v1 |

Table 2

The enumeration is closed. Implementations MUST reject any value not in this set at validation time before canonicalisation. The claim_type field is the primary dispatch signal: a consumer routing incoming frames by lifecycle event MUST switch on claim_type before inspecting the inner receipt.

3.3. receipt_format

A string-valued field drawn from the following closed mapping, keyed by claim_type:

| | | |
|----------------------|---------------------------|--|
| claim_type | receipt_format | |
| payment_admission | compliance-receipt-v1 | |
| payment_settlement | settlement-attestation-v1 | |
| payment_cancellation | cancellation-receipt-v1 | |
| payment_refund | refund-receipt-v1 | |
| composite_verdict | composite-trust-query-v1 | |

Table 3

The `receipt_format` value MUST match the inner receipt's self-declared format identifier. Implementations MUST verify this mapping at validation time. The mapping is closed and version-pinned; future versions of either the frame or the inner receipt formats would advance the version indicator.

3.4. 3.4. receipt

An object-valued field containing the inner receipt, embedded verbatim. The object MUST conform to the shape specified by the format identified in `receipt_format`. The object MUST NOT be modified, re-ordered, or re-encoded relative to its canonical form before embedding; consumers SHOULD re-canonicalise and re-hash the embedded receipt to verify `receipt_hash`.

3.5. 3.5. receipt_hash

A string-valued field of the form `sha256:<lowercase-hex-64>`. The hex digest is the SHA-256 of the JCS-canonical bytes of the receipt field value. See Section 5 for the derivation algorithm.

`receipt_hash` allows a consumer to confirm the inner receipt is unaltered without fully deserialising it. A consumer that only needs integrity confirmation computes `SHA-256(JCS(receipt))` and compares against `receipt_hash`. A mismatch indicates tampering.

Implementations MUST NOT strip the `sha256:` prefix during canonicalisation or verification.

3.6. 3.6. frame_id

A string-valued field of the form sha256:<lowercase-hex-64>. The hex digest is SHA-256 of the JCS-canonical bytes of the PEF preimage. See Section 4 for the derivation algorithm.

frame_id is the stable cross-system identifier for this payment evidence bundle. It is deterministic: independent parties holding the same inner receipt and the same frame metadata MUST derive the same frame_id. It is stable across signature addition: adding the signature field does not change the frame_id because signature is excluded from the preimage.

Implementations MUST NOT strip the sha256: prefix during canonicalisation or verification.

3.7. 3.7. frame_provider_did

A string-valued DID URI identifying the party that constructed and issued the PEF frame. The DID method is not constrained; the field is treated as opaque identifier-bytes under JCS canonicalisation.

The frame provider is typically the platform or agent that built the inner receipt and is issuing it to a downstream consumer. The DID MAY be the same as the compliance_provider_did, settlement_provider_did, or verifier_did in the inner receipt if the same party is acting in both capacities.

3.8. 3.8. frame_timestamp_ms

An integer-valued field carrying the Unix epoch millisecond timestamp at which the frame was constructed, in UTC.

This field MUST be an integer. RFC 3339 string forms (e.g. "2026-05-30T12:00:00Z") MUST be rejected at the validation layer before canonicalisation. This is Substrate Rule 2, normatively specified in draft-hopley-x402-canonicalisation-jcs Section 4.1.

3.9. 3.9. canon_version

A string-valued in-band canonicalisation rule pin. For this version of the frame format the value MUST be urn:x402:canonicalisation:jcs-rfc8785-v1.

The full URN form is used here (rather than the short form jcs-rfc8785-v1 used in the inner receipt formats) to distinguish the frame-level canonicalisation pin from the inner receipt's own canon_version field. The underlying discipline is identical.

3.10. 3.10. signature (OPTIONAL)

A string-valued field carrying a pre-computed RFC 9421 HTTP Message Signature string (draft-hopley-x402-rfc9421-x402-binding) when present.

The signature field is OPTIONAL. Its absence does not affect frame validity or frame_id derivation. When present, it binds the frame to a transport-layer identity claim. The signing procedure, verification algorithm, and key material requirements are defined in draft-hopley-x402-rfc9421-x402-binding.

The signature field is excluded from the preimage used to derive frame_id (Section 4). This exclusion means frame_id is stable before and after signature addition, enabling a two-phase workflow: (1) build frame and derive frame_id; (2) sign and attach signature without invalidating any prior references to the frame_id.

4. 4. Frame ID Derivation

The frame_id is derived as follows:

1. *Construct the preimage.* Start with the full PEF frame object. Remove the frame_id field. Remove the signature field if present. The resulting object is the preimage. The preimage is a valid JSON object containing all fields except frame_id and signature.
2. *Canonicalise the preimage.* Apply the JCS canonicalisation discipline per draft-hopley-x402-canonicalisation-jcs. This produces a deterministic byte sequence. Field names are sorted lexicographically; whitespace is removed; numeric values are normalised per RFC 8785 Section 3.2.2.3; the receipt object is recursively canonicalised.
3. *Hash.* Compute SHA-256 of the canonical byte sequence.
4. *Encode.* Encode as the lowercase hex string of the 32-byte digest, prefixed with sha256:. The result is a 71-character string.

Implementations MUST follow these steps in order. In particular:

- * The frame_id field itself MUST be absent from the preimage. If a frame is received with a frame_id field present, the verifier MUST remove it before computing the expected value and comparing.
- * The signature field MUST be absent from the preimage regardless of whether it is present in the received frame.

- * The receipt object inside the preimage is canonicalised as part of the preimage, not separately. The receipt is NOT independently re-canonicalised and substituted; it is included in the preimage object as-is and the whole preimage is canonicalised in one pass.

4.1. 4.1. Verification

To verify a received frame's `frame_id`:

1. Remove `frame_id` and signature from the frame object.
2. Canonicalise the result per draft-hopley-x402-canonicalisation-jcs.
3. Compute SHA-256 of the canonical bytes.
4. Compare `sha256:<hex>` against the received `frame_id`.
5. If they match, the `frame_id` is valid. If they differ, the frame has been tampered with or the `frame_id` was incorrectly derived.

4.2. 4.2. Stability Property

The `frame_id` is stable across the signature field lifecycle. A frame built without a signature and a frame with an identical content but an added signature field produce the same `frame_id`. This enables the following pattern:

1. Build frame, derive `frame_id`.
2. Reference `frame_id` in downstream systems (audit chain, A2A task artifact, HTTP header, memo field).
3. Later, sign the frame and attach signature.
4. Downstream references to `frame_id` remain valid.

5. 5. Receipt Hash Derivation

The `receipt_hash` is derived as follows:

1. Extract the receipt object from the frame.
2. Canonicalise the receipt object per draft-hopley-x402-canonicalisation-jcs.
3. Compute SHA-256 of the canonical byte sequence.
4. Encode as `sha256:<lowercase-hex-64>`.

The `receipt_hash` is a commitment to the inner receipt content at the time the frame was built. If the inner receipt is subsequently modified (e.g. a field is altered in transit), the recomputed hash will differ from `receipt_hash`, and the consumer will detect tampering without needing to re-derive `frame_id`.

Implementations MUST verify `receipt_hash` as part of frame validation. The two-step verification is:

1. Verify receipt_hash: recompute SHA-256(JCS(receipt)) and compare.
2. Verify frame_id: per Section 4.1.

Both checks MUST pass for a frame to be considered valid.

6. 6. Canonicalisation

The PEF frame (excluding the signature field when present) MUST be canonicalised under the discipline pinned by draft-hopley-x402-canonicalisation-jcs and identified by the URN:

urn:x402:canonicalisation:jcs-rfc8785-v1

The full normative specification of the discipline (JCS RFC 8785 plus the schema-normalisation requirements including Substrate Rule 2 for integer timestamps) is in that document. This document does not redefine the discipline inline.

The signature field, when present, MUST be excluded from canonicalisation for the purposes of frame_id derivation (Section 4). A frame with a signature field is canonical in its full form for transport; the frame_id preimage is canonical without it.

7. 7. Composition with x402 Receipt Stack and Transport Protocols

7.1. 7.1. Receipt Stack Composition

A PEF frame envelopes exactly one inner receipt. The five inner receipt formats and their expected claim types are:

| | | |
|---------------------------|----------------------|--|
| receipt_format | claim_type | Inner I-D |
| compliance-receipt-v1 | payment_admission | draft-hopley-x402-compliance-receipt |
| settlement-attestation-v1 | payment_settlement | draft-hopley-x402-settlement-attestation |
| cancellation-receipt-v1 | payment_cancellation | draft-hopley-x402-cancellation-receipt |
| refund-receipt-v1 | payment_refund | draft-hopley-x402-refund-receipt |
| composite-trust-query-v1 | composite_verdict | draft-hopley-x402-composite-trust-query |

Table 4

A payment lifecycle generates a sequence of PEF frames: one `payment_admission` frame when the compliance receipt is emitted, one `payment_settlement` frame when the settlement attestation is emitted, zero or one `payment_cancellation` frame, zero or one `payment_refund` frame, and zero or more `composite_verdict` frames when a verifier queries the resulting audit chain.

Each frame's `frame_id` is an independent stable identifier for that lifecycle event. Downstream systems correlating events across a lifecycle do so by collecting `frame_id` values from each frame as it is emitted, rather than by walking the underlying audit chain.

7.2. 7.2. x402 Transport

In the x402 HTTP protocol, a PEF frame MAY be included in the verify response body as an extension field alongside the existing payment confirmation fields:

```
POST /verify HTTP/1.1
...
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "x402Version": 1,
  "settlement_evidence": {
    "pef_version": "1",
    "claim_type": "payment_settlement",
    "receipt_format": "settlement-attestation-v1",
    "frame_id": "sha256:<hex64>",
    ...
  }
}
```

A PEF frame MAY also be carried in an X-Payment-Evidence response header, with the frame_id value as the header value and the full frame retrievable at a content-addressed endpoint. The full frame transport via header is implementation-defined.

7.3. 7.3. Agent-to-Agent (A2A) Task Artifacts

In A2A payment agent flows, a PEF frame MAY be attached as a task artifact after a payment task completes. The standard artifact shape is:

```
{
  "type": "payment_evidence",
  "mimeType": "application/json",
  "data": {
    "pef_version": "1",
    "claim_type": "payment_settlement",
    "frame_id": "sha256:<hex64>",
    "receipt_format": "settlement-attestation-v1",
    "receipt_hash": "sha256:<hex64>",
    "receipt": { ... }
  }
}
```

The frame_id is the stable correlation handle for the orchestrator managing a multi-step payment workflow. Agents reference the frame_id in subsequent task messages without carrying the full receipt bytes.

7.4. 7.4. MPP Payment Method Response

In Multi-Party Payment (MPP) flows, a PEF frame MAY be included in the payment method authorisation response body, providing the payer agent with a portable receipt envelope that references the compliance-screen and settlement events in a single content-addressed form.

7.5. 7.5. Audit Chain Integration

A PEF frame MAY be anchored into an audit chain as a chain row. The chain row shape per draft-hopley-x402-compliance-receipt Section 5.1 uses the frame's `receipt_hash` (the SHA-256 of the inner receipt) as the `row_content_hash`. This enables an audit chain that interleaves PEF frames with raw receipt records: a chain walker can verify both the inner receipt and the frame envelope from a single retained chain.

Alternatively, the `frame_id` itself (the SHA-256 of the JCS preimage) MAY be used as the `row_content_hash` when the audit chain is a chain of frames rather than a chain of raw receipts. In this case the chain references frame envelopes, and each frame is independently verifiable.

8. 8. Year-N Auditability Properties

The six properties pinned by draft-hopley-x402-canonicalisation-jcs Section 5 apply to the PEF frame:

1. Self-describing canonicalisation pin via `canon_version`.
2. No external rule registry required.
3. Cross-implementation verifiability (8-implementation matrix per the discipline I-D).
4. Tamper detection via `receipt_hash` (inner receipt) and `frame_id` (frame envelope).
5. Regulatory lifecycle coverage via closed `claim_type` enumeration spanning all five payment lifecycle events.

Plus three PEF-specific properties:

6. *Stable cross-system identifier.* `frame_id` is derivable by any party holding the frame bytes using only SHA-256 and the JCS discipline. A consumer reading a retained `frame_id` reference years after emission can re-derive the identifier from the stored frame bytes without access to any AlgoVoi infrastructure.

7. *Inner receipt integrity without deserialisation.* receipt_hash is SHA-256(JCS(receipt)). A consumer verifying a stored frame confirms inner receipt integrity with one hash comparison. The inner receipt remains independently verifiable at its own content address.
8. *Signature stability.* The optional signature field, if subsequently added to a stored frame, does not alter frame_id. Audit chain entries referencing a frame_id remain valid after the frame is signed. This allows post-hoc signature enrichment of stored frames without invalidating existing references.

9. IANA Considerations

9.1. URN Namespace Registration

This document references the URN urn:x402:canonicalisation:jcs-rfc8785-v1 registered by draft-hopley-x402-canonicalisation-jcs Section 10.1. No additional URN namespace registration is required by this document.

9.2. Frame Format Identifier

This document defines the frame format identifier:

urn:x402:frame:payment-evidence-v1

This identifier MAY be used by PEF implementations to refer to Payment Evidence Frames as a class. Registration with IANA is requested under the x402 URN namespace.

9.3. Claim Type Registry

This document defines the claim_type closed enumeration:

payment_admission
payment_settlement
payment_cancellation
payment_refund
composite_verdict

These values are defined for the urn:x402:frame:payment-evidence-v1 frame format. Future additions to the enumeration would require a new I-D revising this document.

10. 10. Security Considerations

10.1. 10.1. Frame Tampering

A PEF frame's `frame_id` is the SHA-256 of its JCS-canonical preimage. Tampering with any field in the preimage (any field other than signature) produces a different hash; the tampered frame fails verification against any stored `frame_id` reference. Tampering with the inner receipt is detected by `receipt_hash` mismatch before `frame_id` is even computed. Implementations MUST verify both `receipt_hash` and `frame_id` before accepting a frame.

10.2. 10.2. Preimage Exclusion

The `frame_id` excludes signature from its preimage (Section 4). This is an intentional design choice that enables signature addition after `frame_id` derivation. A consumer MUST NOT include signature in the preimage when verifying `frame_id`; doing so would make unsigned and signed frames verify differently, breaking stable cross-system references.

10.3. 10.3. Frame Provider Identity

The `frame_provider_did` field is a self-asserted DID. It identifies the party that constructed the frame but does not by itself prove that party's identity to a consumer. Consumers MUST establish trust in the frame provider through out-of-band means (DID method resolution, certificate authority, mutual-trust agreement, or the optional signature field per draft-hopley-x402-rfc9421-x402-binding).

10.4. 10.4. Claim Type Mismatch

A malicious party could embed a `compliance-receipt-v1` inner receipt in a frame with `claim_type: payment_settlement`. Implementations MUST verify that `claim_type` and `receipt_format` are consistent per the mapping in Section 3.3, and MUST verify that the inner receipt's self-declared format (its own version field or equivalent) matches `receipt_format`. A mismatch MUST cause the frame to be rejected.

10.5. 10.5. Receipt Hash Collision

`receipt_hash` is SHA-256, which is collision-resistant to currently known attacks. A consumer SHOULD NOT accept a frame whose `receipt_hash` is all-zeros or any other degenerate value. The concern is not SHA-256 weakness but implementation errors that emit a zero hash when the inner receipt is empty or malformed; the receipt field MUST be a non-empty object.

10.6. 10.6. Signature Absence

The signature field is OPTIONAL. Its absence does not indicate a frame is invalid or untrustworthy; many legitimate frames are unsigned. Consumers that require a transport-layer identity guarantee (e.g. for regulatory-grade evidence submission) SHOULD require the signature field to be present and verify it per draft-hopley-x402-rfc9421-x402-binding before accepting the frame as authoritative evidence.

10.7. 10.7. Timestamp Integrity

frame_timestamp_ms is self-asserted by the frame provider. It records the claimed time at which the frame was constructed, not a verified time. Consumers relying on frame timestamps for time-sensitive decisions (e.g. refund window eligibility) SHOULD verify the timestamp against independently-derived evidence (settlement chain timestamps, HTTP Date header, blockchain block timestamp).

11. 11. References

11.1. 11.1. 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.
- * [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.
- * [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017.
- * [RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020.

11.2. 11.2. Informative References

- * draft-hopley-x402-canonicalisation-jcs, Hopley, C., "JCS Canonicalisation Discipline for Agentic-Payment Receipts", May 2026.
- * draft-hopley-x402-compliance-receipt, Hopley, C., "Categorical Compliance Screening Receipt Format for Agentic-Payment Flows", May 2026.
- * draft-hopley-x402-settlement-attestation, Hopley, C., "Categorical Settlement Attestation Format for Agentic-Payment Flows", May 2026.

- * draft-hopley-x402-cancellation-receipt, Hopley, C., "Categorical Mandate Cancellation Receipt Format for Agentic-Payment Flows", May 2026.
- * draft-hopley-x402-refund-receipt, Hopley, C., "Categorical Refund Receipt Format for Agentic-Payment Flows", May 2026.
- * draft-hopley-x402-composite-trust-query, Hopley, C., "Composite Trust Query Response Format for Agentic-Payment Audit Chains", May 2026.
- * draft-hopley-x402-rfc9421-x402-binding, Hopley, C., "RFC 9421 HTTP Message Signatures Binding for x402 Payment Frames", May 2026.
- * AlgoVoi, "Substrate Authorship and Provenance", <https://docs.algovoi.co.uk/substrate-authorship-provenance> (<https://docs.algovoi.co.uk/substrate-authorship-provenance>)
- * AlgoVoi, "Payment Evidence Frame reference implementation", <https://pypi.org/project/algovoi-pef/> (<https://pypi.org/project/algovoi-pef/>)
- * EU Markets in Crypto-Assets Regulation (MiCA, Regulation (EU) 2023/1114), Article 80.
- * EU Anti-Money Laundering Regulation (AMLR, Regulation (EU) 2024/1624), Article 56.
- * EU Payment Services Directive 2 (PSD2, Directive 2015/2366), Articles 64, 72, 89.

Appendix A. Appendix A. Examples (Informative)

A.1. A.1. payment_admission -- compliance screen ALLOW

```
{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "payment_admission",
  "frame_id": "sha256:9badca886409ed26d09adfe6ce133a53100909dd4544d4ad160e130b6a755f29
",
  "frame_provider_did": "did:key:z6MkgExzvcpvxrghf4Q3285xqSdenhRZHcP6wc5UvY6VVaz5",
  "frame_timestamp_ms": 1780143974835,
  "pef_version": "1",
  "receipt": {
    "canon_version": "jcs-rfc8785-v1",
    "jurisdiction_flags": ["UK", "EU"],
    "payer_ref": "sha256:e0f023d54479255752bac099d0565984b5884afec0f1a1e27e0eaf70a20
5ba",
    "screen_provider_did": "did:key:z6MkgExzvcpvxrghf4Q3285xqSdenhRZHcP6wc5UvY6VVaz5",
    "screen_result": "ALLOW",
    "screen_timestamp_ms": 1780143974835
  },
  "receipt_format": "compliance-receipt-v1",
  "receipt_hash": "sha256:bc7a68b64925b8a76109d35e89cca4c7ae04073fa686844975a5b5f4410a
fa27"
}
```

A compliance screen result of ALLOW. The frame_id is stable: any downstream consumer, agent, or audit system holding the same inner receipt and frame metadata MUST derive the identical frame_id.

A.2. A.2. payment_settlement -- settled on-chain

```
{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "payment_settlement",
  "frame_id": "sha256:3c4f1e2a8b7d6e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f
",
  "frame_provider_did": "did:web:api.algovoi.co.uk",
  "frame_timestamp_ms": 1780144012000,
  "pef_version": "1",
  "receipt": {
    "amount_microunits": 1000000,
    "canon_version": "jcs-rfc8785-v1",
    "jurisdiction_flags": ["UK"],
    "settled_payment_ref": "sha256:0dd5d0b76c9b9281fdeb2509ad38ab132b16a17385ca01d976f
f9e6e12563a0f",
    "settlement_chain": "base_mainnet",
    "settlement_provider_did": "did:web:api.algovoi.co.uk",
    "settlement_result": "SETTLED",
    "settlement_timestamp_ms": 1780144010000
  },
  "receipt_format": "settlement-attestation-v1",
  "receipt_hash": "sha256:a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0
a1b2"
}
```

A confirmed on-chain settlement on base_mainnet. The frame_id can be passed in the x402 /verify response body as settlement_evidence.frame_id, allowing the merchant backend to reference the payment evidence without embedding the full receipt.

A.3. A.3. payment_cancellation -- mandate terminated by compliance

```

{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "payment_cancellation",
  "frame_id": "sha256:b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3",
  "frame_provider_did": "did:web:api.algovoi.co.uk",
  "frame_timestamp_ms": 1780145000000,
  "pef_version": "1",
  "receipt": {
    "cancellation_reason": "COMPLIANCE_TERMINATED",
    "cancellation_timestamp_ms": 1780144990000,
    "canon_version": "jcs-rfc8785-v1",
    "issuer_did": "did:web:api.algovoi.co.uk",
    "jurisdiction_flags": ["UK", "EU"],
    "mandate_ref": "sha256:c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4",
    "receipt_format": "cancellation-receipt-v1",
    "receipt_hash": "sha256:d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5"
  }
}

```

A mandate terminated by compliance action. The `claim_type` signals `payment_cancellation` without the consumer needing to inspect the inner receipt's `cancellation_reason` field to classify the event.

A.4. A.4. payment_refund -- partial refund issued

```

{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "payment_refund",
  "frame_id": "sha256:e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6",
  "frame_provider_did": "did:web:api.algovoi.co.uk",
  "frame_timestamp_ms": 1780146000000,
  "pef_version": "1",
  "receipt": {
    "canon_version": "jcs-rfc8785-v1",
    "issuer_did": "did:web:api.algovoi.co.uk",
    "jurisdiction_flags": ["UK"],
    "refund_amount_microunits": 500000,
    "refund_result": "PARTIAL",
    "refund_timestamp_ms": 1780145990000,
    "settlement_ref": "sha256:a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2",
    "receipt_format": "refund-receipt-v1",
    "receipt_hash": "sha256:f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7"
  }
}

```

A partial refund. Note the `settlement_ref` in the inner receipt linking back to the settlement attestation frame -- the full lifecycle is traceable by `frame_id` at each stage.

A.5. A.5. composite_verdict -- verifier-of-verifier TRUSTED

```

{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "composite_verdict",
  "frame_id": "sha256:a7b8c9d0elf2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8",
  "frame_provider_did": "did:web:api.algovoi.co.uk",
  "frame_timestamp_ms": 1780147000000,
  "pef_version": "1",
  "receipt": {
    "canon_version": "jcs-rfc8785-v1",
    "chain_ref": "sha256:0dd5d0b76c9b9281fdeb2509ad38ab132b16a17385ca01d976ff9e6e12563a0f",
    "ctq_timestamp_ms": 1780146990000,
    "jurisdiction_flags": ["UK", "EU"],
    "query_ref": "sha256:8b7df143d91c716ecfa5fc1730022f6b421b05cedee8fd52b1fc65a96030ad52",
    "trust_outcome": "TRUSTED",
    "verifier_did": "did:web:api.algovoi.co.uk"
  },
  "receipt_format": "composite-trust-query-v1",
  "receipt_hash": "sha256:b8c9d0elf2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9"
}

```

A composite trust query verdict of TRUSTED, wrapped in a PEF frame for transport. The frame_id can be passed to an orchestrating agent as the correlation handle for the full audit conclusion, without embedding the full CTQ response in every downstream message.

A.6. A.6. Signed Frame (OPTIONAL signature field)

```

{
  "canon_version": "urn:x402:canonicalisation:jcs-rfc8785-v1",
  "claim_type": "payment_settlement",
  "frame_id": "sha256:3c4f1e2a8b7d6e9f0a1b2c3d4e5f6a7b8c9d0elf2a3b4c5d6e7f8a9b0c1d2e3f",
  "frame_provider_did": "did:web:api.algovoi.co.uk",
  "frame_timestamp_ms": 1780144012000,
  "pef_version": "1",
  "receipt": { "...": "..." },
  "receipt_format": "settlement-attestation-v1",
  "receipt_hash": "sha256:a1b2c3d4e5f6a7b8c9d0elf2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2",
  "signature": "sig=:base64url-encoded-signature;; keyid=\"...\"; created=1780144012"
}

```

The frame_id is identical to the unsigned example in A.2. Adding signature does not change the frame_id because signature is excluded from the preimage. Downstream references to the frame_id derived before signing remain valid.

Appendix B. Appendix B. Reference Implementations (Informative)

The following open-source implementations conform to this format. Both implement `build_pef()`, `verify_pef()`, and `pef_frame_id()` with byte-for-byte cross-language parity.

- * `algovoi-pef` (Python) on PyPI: <https://pypi.org/project/algovoi-pef/> (<https://pypi.org/project/algovoi-pef/>) Depends on `algovoi-substrate` for the JCS canonicalisation primitive. Apache 2.0 licensed.
- * `@algovoi/pef` (TypeScript) on npm: <https://www.npmjs.com/package/@algovoi/pef> (<https://www.npmjs.com/package/@algovoi/pef>) Byte-for-byte parity with the Python sibling. Depends on `@algovoi/substrate` for the JCS canonicalisation primitive. Apache 2.0 licensed.

A live production instance of PEF emission is accessible at:

POST <https://api.algovoi.co.uk/compliance/screen>

The `compliance_receipt_pef` field in the response carries a `payment_admission` PEF frame when the screen result is `ALLOW` or `DENY`. The `frame_id` is derivable by any party holding the returned frame using only SHA-256 and the JCS discipline.

Appendix C. Known Adopters (Informative)

| Adopter | Surface | Claim types emitted |
|--|---|---------------------------------|
| AlgoVoi (api.algovoi.co.uk) | <code>/compliance/screen</code> response | <code>payment_admission</code> |
| AlgoVoi (api.algovoi.co.uk) | <code>/checkout/*/verify</code> response | <code>payment_settlement</code> |

Table 5

Appendix D. Acknowledgments

This document, the frame format it specifies, and the reference implementations derived from it are AlgoVoi work under AlgoVoi authorship. Substrate authorship history is catalogued at <https://docs.algovoi.co.uk/substrate-authorship-provenance> (<https://docs.algovoi.co.uk/substrate-authorship-provenance>).

The canonicalisation discipline pinned by Section 6 is normatively specified in draft-hopley-x402-canonicalisation-jcs under the same authorship.

This document closes the transport layer gap in the agentic-payment receipt stack. Below it sit five AlgoVoi-authored receipt formats: compliance, settlement, cancellation, refund, and composite trust query. PEF sits above all five and provides the stable cross-system identifier and taxonomy label that enables receipt bytes to travel across HTTP, agent-to-agent, and audit-chain boundaries without losing their identity or requiring the consumer to parse the inner format before routing.

The author acknowledges Anders Rundgren as the editor of RFC 8785, the JSON Canonicalization Scheme on which the canonicalisation discipline builds.

Author's Address

Christopher Hopley
AlgoVoi
United Kingdom
Email: chopmob@gmail.com