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

C. Hopley
AlgoVoi

30 May 2026

Categorical Settlement Attestation Format for Agentic-Payment Flows
draft-hopley-x402-settlement-attestation-00

Abstract

This document specifies a categorical settlement attestation format for agentic-payment flows. The format records that a payment has reached a particular settlement state on a particular chain, at a particular instant, under the attesting party's risk model.

The receipt format uses a closed enumeration of categorical outcomes (SETTLED, PENDING_FINALITY, REVERSED). The categorical outcome is load-bearing for downstream regulatory obligations: SETTLED triggers settlement-finality record-keeping obligations under EU Markets in Crypto-Assets Regulation Article 80 and Anti-Money Laundering Regulation Article 56; the PSD2 (Directive 2015/2366) Article 89 unauthorised-payment refund-window clock begins at SETTLED. PENDING_FINALITY records an inclusion-without-finality state where the refund-window has not yet started. REVERSED records chain-reorganisation, fraud-reversal, or operator-initiated reversal events for chain-of-custody audit.

The format is multi-chain: a single settlement_chain string field identifies the chain on which settlement occurred, using a flat identifier convention (<chain_family> for default mainnets, <chain_family>:<network> for non-default networks). The receipt does not encode chain-specific finality semantics; those are applied at verification time by a verifier that knows the chain.

The format composes with the AlgoVoi-authored compliance receipt format ([I-D.hopley-x402-compliance-receipt]) and refund receipt format ([I-D.hopley-x402-refund-receipt]) under the same canonicalisation discipline ([I-D.hopley-x402-canonicalisation-jcs]). A verifier walking the audit chain confirms the full payment lifecycle from admission through settlement to refund (or non-refund) under one byte-deterministic canonicalisation pin.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 1 December 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Motivation

Agentic-payment flows generate three classes of categorical receipt across the payment lifecycle:

1. Admission-time receipts (compliance screening, payment authorisation, mandate verification).
2. Settlement-time receipts (this document).
3. Post-settlement receipts (refund, dispute resolution).

Settlement is the state transition at which a broadcast payment is considered final by the attesting party's risk model. The operational and regulatory significance is load-bearing:

* Under EU Markets in Crypto-Assets Regulation Article 80 (Regulation (EU) 2023/1114) and EU Anti-Money Laundering Regulation Article 56 (Regulation (EU) 2024/1624), operator records of settled crypto-asset transactions must be retained and re-verifiable for the statutory period.
* Under PSD2 (Directive 2015/2366) Article 89, the unauthorised-payment refund obligation has timing tied to when the payment was settled, not when it was broadcast. A PENDING_FINALITY state is operationally distinct from a SETTLED state for refund-window calculation purposes.
* Under EU Anti-Money Laundering Directives 5 and 6, a settled payment that is subsequently REVERSED (for example by chain reorganisation or court-ordered fraud reversal) requires an audit trail recording the reversal event independently of the original settlement event.

A receipt format that collapses these distinctions to a binary "settled / not settled" or to a continuous "confirmation depth" representation loses the load-bearing operational distinction.

This document specifies a settlement attestation format that preserves the categorical outcome at the canonical-bytes level and records the settlement chain explicitly, enabling multi-chain operator audit trails under one canonicalisation discipline.

1.2. Scope

This document specifies:

* The canonical JSON shape of the settlement attestation (Section 3).
* The reference to the canonicalisation rule applicable to the receipt (Section 4 -- normative reference to [I-D.hopley-x402-canonicalisation-jcs], not redefined inline).
* The audit chain composition under which settlement attestations compose with compliance receipts and refund receipts (Section 5).
* The year-N auditability properties the format pins (Section 6).
* Composition patterns across the AlgoVoi receipt-format suite (Section 7).
* Worked examples covering SETTLED, PENDING_FINALITY, and REVERSED outcomes (Appendix A).

This document does NOT specify:

* The chain-specific finality model. The receipt identifies the chain via settlement_chain; verifiers apply chain-specific finality semantics (Algorand's deterministic instant finality, Ethereum's probabilistic depth-based finality, Stellar's SCP, etc.) at verification time.
* Cryptographic settlement proofs. Cryptographically-strong post-quantum settlement proofs are an orthogonal mechanism specified elsewhere. The two layers compose: a categorical settlement attestation under this format MAY reference a cryptographic settlement proof in its settled_payment_ref or via a separate audit-chain row.
* Cross-chain bridge mechanics. The format supports cross-asset substitution (the settlement_amount.asset_id MAY differ from the original payment asset) but the bridge mechanism is out of scope.

1.3. Relationship to other Internet-Drafts

This document normatively references:

* [I-D.hopley-x402-canonicalisation-jcs] -- the JCS canonicalisation discipline pinned in Section 4.

This document is complementary to:

* [I-D.hopley-x402-compliance-receipt] -- admission-time compliance screening receipts. A settlement attestation's settled_payment_ref MAY equal the content_hash of a compliance receipt.
* [I-D.hopley-x402-refund-receipt] -- post-settlement refund receipts. A refund receipt's original_payment_ref MAY equal the content_hash of a settlement attestation under this document.

2. Conventions and Definitions

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. Definitions

settlement attestation: a JSON object of the shape specified in Section 3, canonicalised under the discipline of [I-D.hopley-x402-canonicalisation-jcs].

content_hash: SHA-256, lowercase hex, of the JCS-canonical bytes of the attestation object.

settled_payment_ref: a string of the form sha256:<lowercase-hex-64> identifying the payment record being settled by content hash. The original record MAY be a compliance receipt, an operator-specific payment authorisation record, or a cryptographic settlement proof.

settlement_chain: a string identifier for the chain on which settlement occurred. See Section 3.5.

settlement_amount: a two-field object {amount_minor, asset_id} representing the settled value.

canon_version: an in-band string identifying the canonicalisation discipline. Fixed value jcs-rfc8785-v1 for this version.

3. Receipt Format Specification

A settlement attestation is a JSON object with the following eight fields. All fields are REQUIRED. The attestation is canonicalised under [I-D.hopley-x402-canonicalisation-jcs] per Section 4. Field names are sorted lexicographically by JCS during canonicalisation; the object itself uses arbitrary authoring order.

3.1. settlement_result

A string-valued field. The value MUST be one of:

* SETTLED -- payment confirmed on-chain with sufficient finality for the operator’s risk model. Funds transferred and irreversible under normal chain operation.
* PENDING_FINALITY -- payment broadcast and included in a block, awaiting the operator’s required confirmation depth. Operator has visibility of inclusion but does not yet assert finality.
* REVERSED -- payment previously SETTLED is now considered reversed (chain reorganisation, fraud reversal, double-spend resolution, operator-initiated under regulatory directive).

The three-element enumeration is closed. Implementations MUST reject any other value at validation time before canonicalisation. Score, depth, or confirmation-count representations are not acceptable substitutes for the categorical outcome.

The regulatory distinction is load-bearing. Under MiCA Article 80, settlement-finality records must be retained and re-verifiable; SETTLED is the operational state that triggers this obligation. Under PSD2 Article 89, refund-window timing is tied to settled state. Under AML Directives 5 and 6, reversed transactions require documented evidence chains independent of the original settlement.

3.2. jurisdiction_flags

An ordered array of string-valued ISO-3166-1 alpha-2 country codes or alpha-3 region codes identifying the applicable regulatory frameworks for the settlement event.

Authoring convention: primary jurisdiction first (where the operating entity is licensed), secondary jurisdictions in order of regulatory precedence.

Array element ORDER is SIGNIFICANT and load-bearing per [I-D.hopley-x402-canonicalisation-jcs] Section 4.3.

3.3. settled_payment_ref

A string-valued field of the form sha256:<lowercase-hex-64>. The hex digest is SHA-256 of the JCS-canonical bytes of the payment record being settled.

When settling a payment that was admitted under a compliance receipt (per [I-D.hopley-x402-compliance-receipt]), the settled_payment_ref MAY equal the content_hash of the compliance receipt itself. This is the conventional choice and enables the audit-chain composition described in Section 5.

When the original payment record is an operator-specific format (payment authorisation, mandate, settlement proof, etc.), the settled_payment_ref is the SHA-256 of that record’s JCS-canonical bytes.

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

3.4. settlement_amount

A sub-object with exactly two fields:

* amount_minor -- a string of decimal digits representing the settled value in the asset’s minor unit. String typing avoids float-precision loss and JavaScript-integer-overflow concerns for large values.

* asset_id -- a string identifying the asset and its decimal precision. Convention: <symbol>.<decimals> for chain-native assets (e.g. USDC.6, ALGO.6, ETH.18); <chain>:<asset_id>.<decimals> for ASA-style assets (e.g. algo:31566704.6).

The sub-object's keys are sorted lexicographically by RFC 8785 during canonicalisation: amount_minor then asset_id.

Cross-asset substitution is permitted. When settlement is in a different asset than the original payment (e.g. paid USDC on Base, settled USDC on Solana via a bridge), settlement_amount.asset_id reflects the SETTLED asset, not the original. The equivalence-of-value claim across the substitution is operator-layer and out of scope.

3.5. settlement_chain

A string-valued field identifying the chain on which settlement occurred. The string is in one of two forms:

1. Default mainnet of a chain family: <chain_family>. Examples:

    * algo -- Algorand mainnet
    * voi -- VOI mainnet
    * solana -- Solana mainnet
    * stellar -- Stellar Pubnet
    * hedera -- Hedera mainnet
    * base -- Base L2 mainnet (alias for ethereum:8453)

2. Non-default network of a chain family: <chain_family>:<network>. Examples:

    * algorand:testnet -- Algorand TestNet
    * ethereum:1 -- Ethereum mainnet
    * ethereum:8453 -- Base L2 (canonical by chainId)
    * tempo:mainnet -- Tempo mainnet
    * solana:devnet -- Solana devnet

The string is case-sensitive at the JCS layer. Implementations SHOULD canonicalise to lowercase before emission. Verifiers MUST treat Ethereum:8453 and ethereum:8453 as distinct canonical bytes per RFC 8785.

The receipt does not decompose the chain identifier into a sub-object. Operators requiring richer chain metadata (block height, confirmation count, validator quorum at settlement time) MUST emit those as separate operator-layer audit records, not as nested fields in this receipt.

3.6. settlement_provider_did

A string-valued DID URI identifying the entity attesting settlement (gateway, facilitator, oracle, or operator).

3.7. settlement_timestamp_ms

An integer-valued field carrying the epoch-millisecond timestamp of the settlement event 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 [I-D.hopley-x402-canonicalisation-jcs] Section 4.1.

3.8. canon_version

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

4. Canonicalisation

The settlement attestation MUST be canonicalised under the discipline pinned by [I-D.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) is in that document. This document does not redefine the discipline inline.

5. Audit Chain Composition

A settlement attestation MAY participate in an audit chain alongside compliance receipts, refund receipts, and other receipt classes that pin the same canonicalisation discipline.

5.1. Chain Row Shape

The audit chain row shape used by this document is identical to that specified in [I-D.hopley-x402-compliance-receipt] Section 5.1 and [I-D.hopley-x402-refund-receipt] Section 5.1:

```
{
  "row_number": 1,
  "content_hash": "<hex64>",
  "prev_hash": "<hex64>",
  "row_content_hash": "<hex64>"
}
```

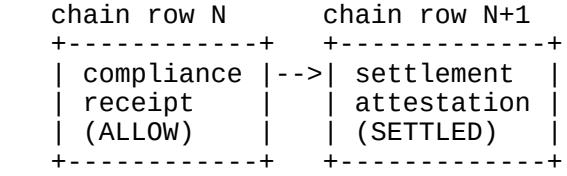
Row 1's prev_hash MUST be 64 zero hex characters. Row N's prev_hash MUST equal row N-1's row_content_hash.

5.2. Linkage Verification

Per [I-D.hopley-x402-compliance-receipt] Section 5.2: a verifier reading a chain segment recomputes each row's row_content_hash from its first three fields and confirms forward linkage via prev_hash. Any mismatch indicates tampering or chain integrity loss.

5.3. Composition with Compliance Receipts

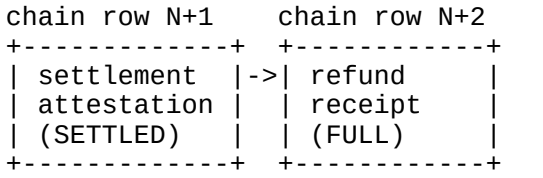
When a settlement attestation references a compliance receipt via settled_payment_ref, the audit-chain composition is:



Row N anchors the compliance receipt. Row N+1 anchors the settlement attestation. The settlement attestation's settled_payment_ref equals row N's content_hash. Chain linkage via prev_hash confirms ordering.

5.4. Composition with Refund Receipts

When a settled payment is subsequently refunded, the refund receipt's original_payment_ref MAY equal the content_hash of the settlement attestation. The chain extends:



A verifier walking the full chain confirms admission -> settlement -> refund under one canonicalisation pin.

6. Year-N Auditability Properties

The same six properties pinned by [I-D.hopley-x402-canonicalisation-jcs] Section 5 apply to the settlement attestation:

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 Section 7).
4. Tamper detection via per-row content_hash and chain prev_hash linkage.
5. Regulatory distinction preserved via closed enumeration.

Plus one settlement-specific property:

6. *Chain-aware finality semantics applied at verification time.* The settlement_chain string identifies which chain's finality model applies. A verifier years after the event can apply the correct chain-specific finality semantics to re-evaluate the original SETTLED claim against retained chain state (block height, confirmation depth, etc.).

7. Composition with Other x402 Substrate

7.1. Compliance Receipt Linkage

See Section 5.3. The conventional settled_payment_ref target for an admission-then-settlement flow is the content_hash of the compliance receipt that admitted the original payment.

7.2. Refund Receipt Linkage

See Section 5.4. A refund receipt's original_payment_ref MAY equal a settlement attestation content_hash rather than the underlying compliance receipt, producing a precise "the settled payment was reversed" audit trail.

7.3. Cryptographic Settlement Proofs

Cryptographically-strong settlement proofs (post-quantum proofs of payment conditions, validator signatures, STARK proofs of inclusion, etc.) are orthogonal to the categorical settlement attestation specified by this document. The two layers compose:

* A categorical settlement attestation records the operator's categorical claim (SETTLED / PENDING_FINALITY / REVERSED) at the canonical-bytes level.
* A cryptographic settlement proof provides byte-level evidence of the underlying on-chain condition.

Both MAY be retained alongside one another, or chained via settled_payment_ref references, depending on operator requirements.

8. IANA Considerations

8.1. URN Namespace Registration

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

8.2. Receipt Format Identifier

This document defines the receipt format identifier:

urn:x402:receipt:settlement-attestation-v1

This identifier MAY be used by composite-trust-query implementations to refer to settlement attestations as a class. Registration with IANA is requested under the x402 URN namespace.

9. Security Considerations

9.1. Attestation Tampering

A settlement attestation's content_hash is the SHA-256 of its JCS-canonical bytes. Tampering with any field produces a different hash; the tampered attestation fails verification against any audit-chain row referencing the original content_hash.

9.2. Chain Reorganisation

A SETTLED attestation may later become incorrect if the underlying chain reorganises and the settled transaction is no longer canonical. The REVERSED outcome records this event. Operators SHOULD emit a REVERSED attestation when a previously SETTLED state is invalidated; the audit chain extends rather than overwrites.

A malicious operator could emit a SETTLED attestation for a payment that never reaches chain-finality, then fail to emit the corresponding REVERSED. Detection is verifier-side: a verifier SHOULD periodically re-check chain state against retained SETTLED attestations and flag discrepancies.

9.3. Chain Identifier Spoofing

The settlement_chain field is operator-asserted. A verifier MUST NOT trust the chain identifier alone; verification against actual chain state (via the verifier’s own chain client) is required to confirm settlement.

9.4. Cross-Asset Substitution

When settlement_amount.asset_id differs from the original payment asset, the equivalence-of-value claim is operator-asserted. A verifier cannot determine equivalence from the receipt alone; external evidence (price oracle attestations at settlement time, bridge proofs) is required and out of scope of this document.

9.5. Operator Continuity Loss

If the original attesting operator becomes unavailable, the audit chain and its attestations MUST remain independently verifiable from retained bytes plus the reference implementations cited in [I-D.hopley-x402-canonicalisation-jcs] Section 7.

Appendix A. References

A.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.
* [I-D.hopley-x402-canonicalisation-jcs] Hopley, C., "JCS Canonicalisation Discipline for Agentic-Payment Receipts", draft-hopley-x402-canonicalisation-jcs-v1, May 2026.

A.2. Informative References

* [I-D.hopley-x402-compliance-receipt] Hopley, C., "Categorical Compliance Screening Receipt Format for Agentic-Payment Flows", draft-hopley-x402-compliance-receipt-00, May 2026.
* [I-D.hopley-x402-refund-receipt] Hopley, C., "Categorical Refund Receipt Format for Agentic-Payment Flows", draft-hopley-x402-refund-receipt-00, May 2026.
* [AlgoVoi-Substrate-Authorship] AlgoVoi, "Substrate Authorship and Provenance", target="https://docs.algovoi.co.uk/substrate-authorship-provenance" (https://docs.algovoi.co.uk/substrate-authorship-provenance)
* 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), Article 89.
* EU Anti-Money Laundering Directive 5 (Directive (EU) 2018/843).
* EU Anti-Money Laundering Directive 6 (Directive (EU) 2018/1673).
* EU Digital Operational Resilience Act (DORA, Regulation (EU) 2022/2554), Article 14.

Appendix B. Appendix A. Examples (Informative)

B.1. A.1. SETTLED on Base under UK + EU joint jurisdiction


```
{
  "canon_version": "jcs-rfc8785-v1",
  "jurisdiction_flags": ["UK", "EU"],
  "settled_payment_ref": "sha256:0dd5d0b76c9b9281fdeb2509ad38ab132b16a17385ca01d976ff9
e6e12563a0f",
  "settlement_amount": {"amount_minor": "100000", "asset_id": "USDC.6"},
  "settlement_chain": "ethereum:8453",
  "settlement_provider_did": "did:web:api.algovoi.co.uk",
  "settlement_result": "SETTLED",
  "settlement_timestamp_ms": 1716494400000
}
```

Records that 0.1 USDC was settled on Base mainnet (chainId 8453) under joint UK + EU jurisdiction. The PSD2 Article 89 refund-window clock begins at settlement_timestamp_ms. MiCA Article 80 record-keeping obligations apply to retention of this attestation.

B.2. A.2. PENDING_FINALITY on Algorand


```
{
  "canon_version": "jcs-rfc8785-v1",
  "jurisdiction_flags": ["UK"],
  "settled_payment_ref": "sha256:0dd5d0b76c9b9281fdeb2509ad38ab132b16a17385ca01d976ff9
e6e12563a0f",
  "settlement_amount": {"amount_minor": "100000", "asset_id": "USDC.6"},
  "settlement_chain": "algo",
  "settlement_provider_did": "did:web:api.algovoi.co.uk",
  "settlement_result": "PENDING_FINALITY",
  "settlement_timestamp_ms": 1716494400000
}
```


Records broadcast-and-inclusion on Algorand mainnet without yet asserting operator-required confirmation depth.

B.3. A.3. REVERSED on Ethereum mainnet after chain reorg


```
{
  "canon_version": "jcs-rfc8785-v1",
  "jurisdiction_flags": ["UK", "EU"],
  "settled_payment_ref": "sha256:0dd5d0b76c9b9281fdeb2509ad38ab132b16a17385ca01d976ff9
e6e12563a0f",
  "settlement_amount": {"amount_minor": "100000", "asset_id": "USDC.6"},
  "settlement_chain": "ethereum:1",
  "settlement_provider_did": "did:web:api.algovoi.co.uk",
  "settlement_result": "REVERSED",
  "settlement_timestamp_ms": 1716494400000
}
```


Records that a previously SETTLED payment on Ethereum mainnet is now considered reversed. The audit chain extends with this row; the prior SETTLED row remains as historical record.

Appendix C. Appendix B. Reference Implementations (Informative)

The following open-source implementations conform to this format:

* algovoi-settlement-attestation (Python) on PyPI: target="https://pypi.org/project/algovoi-settlement-attestation/" (https://pypi.org/project/algovoi-settlement-attestation/) Provides build_settlement_attestation(). Depends on algovoi-substrate for the JCS canonicalisation primitive. Apache 2.0 licensed.

* @algovoi/settlement-attestation (TypeScript) on npm: target="https://www.npmjs.com/package/@algovoi/settlement-attestation" (https://www.npmjs.com/package/@algovoi/settlement-attestation) Byte-for-byte parity with the Python sibling. Depends on @algovoi/substrate for the JCS canonicalisation primitive. Apache 2.0 licensed.

Conformance vectors:

https://github.com/chopmob-cloud/algovoi-jcs-conformance-vectors/tree/main/vectors/settlement_attestation_v1

8 byte-level reference vectors + 5 pair invariants + 3 chain invariants pinning the closed-enumeration, jurisdiction-array-order, canon_version pin, and audit-chain linkage properties.

Appendix D. Appendix C. Acknowledgments

This document, the receipt format it specifies, and the conformance vectors derived from it are AlgoVoi work under sole AlgoVoi authorship. Substrate authorship history is catalogued at target="https://docs.algovoi.co.uk/substrate-authorship-provenance" (https://docs.algovoi.co.uk/substrate-authorship-provenance).

The canonicalisation discipline pinned by Section 4 is normatively specified in [I-D.hopley-x402-canonicalisation-jcs] under the same authorship.

This document closes the lifecycle gap between the admission-time compliance receipt format ([I-D.hopley-x402-compliance-receipt]) and the post-settlement refund receipt format ([I-D.hopley-x402-refund-receipt]). The three formats share the same canonicalisation pin, audit-chain row shape, and integer- millisecond timestamp encoding so that a verifier walking the full payment lifecycle requires only one implementation of the canonicalisation discipline.

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

Author’s Address

Christopher Hopley
AlgoVoi
Email: chopmob@gmail.com