Independent Submission
Internet-Draft
Intended status: Informational
Expires: 26 November 2026

V. Research
Vauban Research

25 May 2026

x402 V2 Payment Lifecycle Finite State Machine
draft-vauban-x402-lifecycle-fsm-01

Abstract

This document defines a normative finite state machine (FSM) for x402 V2 Payment lifecycle transitions, covering the four canonical Payment Claim states: PaymentIntent (initialisation), SettlementReceipt (completion), RefundClaim (reversal), and DelegationGrant (bounded authority). Each transition is bound to the prior state by cryptographic linkage via the JCS canonical preimage discipline ([RFC8785]), enabling supervisor reconstruction of full payment histories from retained off-VM manifests. The FSM is compatible with the x402 STARK Receipt Format Extension [STARK-RECEIPTS] but applicable to any x402 receipt format that implements the canonical preimage discipline. Implementation evidence: 18 DelegationGrant vectors cross-validated byte-identical across five language runtimes ([X402-2432]), live on-chain anchor on Starknet Sepolia (PaymentDemoEmitter at 0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff).

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 26 November 2026.

Copyright Notice

Table of Contents

1. Introduction

The x402 protocol ([X402-V2]) defines HTTP-native payment flows using three messages: PAYMENT-REQUIRED (402 response), PAYMENT-SIGNATURE (client request), and PAYMENT-RESPONSE (facilitator confirmation). The base specification defines these three wire messages but does not define a normative lifecycle for the payment states that precede and follow a single exchange. In production systems, a payment passes through at least two states (intent to settle), and often more (reversal, agentic delegation). Receipt formats have individually attempted to capture lifecycle semantics, producing fragmented and mutually incompatible state representations.

This fragmentation creates a verifiability gap. A supervisor or auditor reconstructing a payment history from retained receipts cannot assert whether a refund was legitimately preceded by a settled intent, or whether a delegation authority was scoped correctly to the intents it subsequently spawned, unless each receipt format separately encodes and exposes the linkage. In practice, most do not.

This document addresses the gap by specifying the lifecycle FSM at a layer above receipt-format specifics. The FSM defines four canonical states and the legal transitions between them. Each transition carries a cryptographic linkage computed via the JSON Canonicalization Scheme ([RFC8785]) applied to the prior state’s canonical preimage. Any receipt format that emits these linkage digests can implement the FSM; no format is privileged by the FSM definition.

The x402 STARK Receipt Format Extension [STARK-RECEIPTS] is the first reference implementation of this FSM. Alternative receipt formats may bind the same FSM by satisfying the cryptographic discipline defined in Section 4.

This document is an Independent Submission. It is not the product of an IETF Working Group. It is published for community review and to establish a stable reference for implementors.

1.1. 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.

1.2. Terminology

Payment Claim: A cryptographically bound statement about a payment event in a specific lifecycle state. Each of the four states (PaymentIntent, SettlementReceipt, RefundClaim, DelegationGrant) is a Payment Claim variant. The term is generic; it does not refer to any specific receipt format or proof system.

Lifecycle State: One of the four canonical states defined in Section 2. A payment exists in exactly one Lifecycle State at any given instant. State transitions are irreversible.

State Transition: A directed edge in the FSM graph connecting two Lifecycle States. A transition is valid only when its preconditions (Section 3) are satisfied. Invalid transitions MUST be rejected by verifiers.

Verifier: An entity that checks whether a Payment Claim and its lifecycle linkage are well-formed and consistent. The Verifier may be the facilitator, the resource server, a compliance auditor, or an automated pipeline. Verifiers do not issue Payment Claims; they consume them.

JCS Preimage Hash: The SHA-256 digest of the UTF-8 encoding of the JCS canonical form ([RFC8785]) of a Payment Claim’s canonical preimage object. Used as the linkage token between consecutive lifecycle states. Encoded as "sha256:<lowercase-hex-64-chars>" in JSON contexts and as raw 32 bytes in CBOR contexts ([RFC8949]).

original_payment_ref: A JCS Preimage Hash carried by SettlementReceipt and RefundClaim that binds each downstream state to its upstream progenitor. For SettlementReceipt, original_payment_ref is the JCS Preimage Hash of the originating PaymentIntent. For RefundClaim, it is the JCS Preimage Hash of the SettlementReceipt being reversed.

2. Lifecycle States

The x402 V2 payment lifecycle comprises four canonical states. The three main-path states (PaymentIntent, SettlementReceipt, RefundClaim) form a linear sequence. The DelegationGrant state is a side channel that scopes the authority under which PaymentIntents may be spawned autonomously.

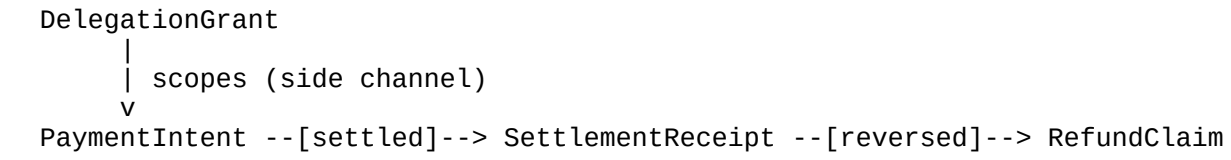

Figure 1: x402 V2 Payment Lifecycle FSM (canonical paths)

2.1. PaymentIntent

A PaymentIntent is the initiating state. It records the payer's commitment to transfer a specified amount of a specified currency to a specified payee, subject to the payment conditions expressed in the accompanying PAYMENT-SIGNATURE ([X402-V2]).

Required fields:

Field	Type	Description
id	string (UUID v4)	Unique identifier for this intent.
payer	string	Payer address or pseudonym.
payee	string	Payee (merchant) address or pseudonym.
amount	integer	Amount in the smallest indivisible unit of currency. No floating-point values are permitted.
currency	string	Token identifier (e.g., "USDC", "STRK").
issued_at	integer	Unix timestamp (seconds) of intent issuance.
expires_at	integer (optional)	Unix timestamp after which the intent is expired. Default window: 30 seconds from issued_at.
nonce	bytes (32)	Unique anti-replay token. MUST be generated from a cryptographically secure random number generator.

Table 1

JCS preimage rule: the canonical preimage object for a PaymentIntent is the JCS serialisation ([RFC8785]) of the JSON object with keys sorted lexicographically. All string fields MUST be Unicode Normalization Form C (NFC) before serialisation. The JCS Preimage Hash is SHA-256(UTF-8(JCS(preimage))).

2.2. SettlementReceipt

A SettlementReceipt is emitted by the Verifier (facilitator) once a PaymentIntent has been verified and the corresponding payment conditions satisfied. It represents the terminal successful state of the main path.

Required fields:

Field	Type	Description
payment_id	string (UUID v4)	The id of the PaymentIntent that was settled.
tx_hash	string	On-chain transaction reference where settlement was recorded. Format is receipt-format-specific.
block_number	integer	Block height at which settlement was confirmed.
settled_at	integer	Unix timestamp (seconds) of settlement confirmation.
original_payment_ref	string	JCS Preimage Hash of the originating PaymentIntent. Format: "sha256:<hex-64>". REQUIRED for FSM compliance.

Table 2

Cryptographic linkage to PaymentIntent: the original_payment_ref field MUST be set to the JCS Preimage Hash of the PaymentIntent whose id matches payment_id. A SettlementReceipt without a valid original_payment_ref is not FSM-compliant; verifiers MAY accept it for backward compatibility but MUST NOT treat it as FSM-grounded for audit purposes.

2.3. RefundClaim

A RefundClaim is emitted by the merchant to initiate a reversal of a settled payment. It is the terminal state of the reversal path. A RefundClaim MUST reference a prior SettlementReceipt; a refund without a prior settlement is a forbidden transition (Section 3.4).

Required fields:

Field	Type	Description
original_payment_id	string (UUID v4)	The id of the PaymentIntent / SettlementReceipt being refunded.
amount	integer	Refund amount in the smallest denomination of the original currency. MUST satisfy amount <= original_amount.
reason	string	Human-readable reason for the refund. Treated as disclosed to auditors.
issued_at	integer	Unix timestamp (seconds) of refund issuance.

Table 3

Partial vs. full refund semantics: a RefundClaim where amount equals the original settled amount is a full refund. A RefundClaim where amount is strictly less is a partial refund. In both cases the SettlementReceipt transitions to a Refunded sub-state. Only one RefundClaim per SettlementReceipt is permitted per default FSM rules; receipt formats that support multiple partial refunds MUST document their multi-refund policy.

Refund window: the FSM imposes a default validity window of 30 days from issued_at on RefundClaims. Receipt formats MAY shorten but MUST NOT extend this window without explicit merchant and facilitator agreement.

Cryptographic linkage to SettlementReceipt: the RefundClaim MUST carry an original_payment_ref field set to the JCS Preimage Hash of the SettlementReceipt being reversed. This creates a two-hop chain: RefundClaim references SettlementReceipt; SettlementReceipt

references PaymentIntent. A supervisor can reconstruct the full three-state chain from the two linkage digests and the retained manifests.

2.4. DelegationGrant

A DelegationGrant is a side-channel state that authorises an agent (the delegatee) to spawn PaymentIntents on behalf of a principal (the delegator), subject to scoped constraints. The DelegationGrant does not itself initiate a payment; it bounds the authority under which subsequent PaymentIntents may be autonomously issued.

Required fields:

Field	Type	Description
delegator	string	Address or pseudonym of the principal granting authority.
delegatee	string	Address or pseudonym of the agent receiving authority.
scope	string[]	Ordered list of scope identifiers constraining what the agent may do (e.g., ["payment:usdc", "merchant:0x04..."]).
expires_at	integer (optional)	Unix timestamp after which the grant is expired. If absent, the grant is perpetual until explicitly revoked.

Table 4

Binding to PaymentIntent: any PaymentIntent spawned under a DelegationGrant MUST carry a reference to the grant's content digest in a receipt-format-specific field. The FSM does not mandate the exact field name; receipt formats MUST document their DelegationGrant binding mechanism.

Scope enforcement: a Verifier receiving a PaymentIntent that claims to operate under a DelegationGrant MUST verify that the intent's currency and payee are within the grant's scope list. Intents that exceed the grant's scope MUST be rejected with an explicit scope-violation error.

3. State Transitions

3.1. Canonical Path: Intent to Settlement

Trigger: the payer submits a PAYMENT-SIGNATURE ([X402-V2]) referencing a PaymentIntent, and the Verifier confirms that all payment conditions are satisfied.

Preconditions (all MUST hold):

1. A PaymentIntent with the referenced id exists and has not expired (issued_at + expiry_window > now).

2. The payment conditions (amount, currency, payee) in the PAYMENT-SIGNATURE match the PaymentIntent fields.

3. The anti-replay nonce has not been previously consumed for this payment pair.

4. The payer attestation is valid per the receipt format’s verification rules.

Postconditions:

1. A SettlementReceipt is emitted with payment_id set to the PaymentIntent id.

2. original_payment_ref is set to the JCS Preimage Hash of the PaymentIntent.

3. settled_at is set to the current time.

4. The anti-replay nonce is marked as consumed and MUST NOT be accepted again.

5. The PaymentIntent transitions to the Settled sub-state; no further transitions from this PaymentIntent are permitted.

3.2. Reversal Path: Settlement to Refund

Trigger: the merchant or facilitator initiates a refund against a prior SettlementReceipt.

Preconditions (all MUST hold):

1. A SettlementReceipt with the referenced payment_id exists and is in the Settled state (not already Refunded).

2. The refund amount satisfies amount <= original_settled_amount.

3. The current time is within the refund window: now <= settlement.settled_at + refund_window_seconds.

4. The requesting party (merchant or facilitator) is authorised to issue refunds for this payment pair.

Postconditions:

1. A RefundClaim is emitted with original_payment_id set to the PaymentIntent id.

2. The RefundClaim MUST carry an original_payment_ref set to the JCS Preimage Hash of the SettlementReceipt being reversed.

3. issued_at is set to the current time.

4. The SettlementReceipt transitions to the Refunded sub-state; no further refunds against this receipt are permitted under default FSM rules.

Partial refund semantics: when amount < original_settled_amount, the RefundClaim represents a partial reversal. The remaining balance is settled and not subject to further refund unless the receipt format explicitly supports multi-refund sequences. Implementations supporting partial refunds MUST document their sequencing rules.

3.3. Bounded Authority Path: Delegation to Intent

Trigger: an agent (the delegatee) spawns a PaymentIntent under a previously issued DelegationGrant.

Preconditions (all MUST hold):

1. A DelegationGrant exists for the delegatee with a non-expired expires_at (or no expiry).

2. The intended payment’s currency is listed in the grant’s scope.

3. The intended payment’s payee is listed in the grant’s scope (if the scope constrains payees).

4. The DelegationGrant’s chain depth (number of intermediate delegation hops) does not exceed max_chain_length (default: 1 direct hop; see Section 3.4).

Postconditions:

1. The spawned PaymentIntent MUST carry a receipt-format-specific reference to the DelegationGrant content digest.

2. The Verifier MUST verify scope compliance before accepting the PaymentIntent for settlement.

3. The DelegationGrant remains active; a single grant may spawn multiple PaymentIntents within its scope and expiry.

3.4. Forbidden Transitions

The following state pairs MUST be rejected by Verifiers. Acceptance of a forbidden transition is a security violation; implementations MUST emit an explicit rejection error rather than silently ignoring the inconsistency.

Transition	Reason	Error signal
RefundClaim without a prior SettlementReceipt	A refund requires a preceding settlement.	RefundWithoutSettlement
SettlementReceipt referencing a non-existent or expired PaymentIntent	No settlement without a valid intent.	IntentNotFound or IntentExpired
Second SettlementReceipt for an already-settled PaymentIntent	Intents settle exactly once.	AlreadySettled
Second RefundClaim for an already-refunded SettlementReceipt (default FSM)	Receipts refund at most once unless the format documents multi-refund support.	AlreadyRefunded
DelegationGrant chain depth exceeding max_chain_length	Unbounded delegation chains enable privilege escalation.	DelegationDepthExceeded
PaymentIntent issued after DelegationGrant expiry	Expired grants do not authorise new intents.	GrantExpired

Table 5

The default max_chain_length for DelegationGrant chains is 1 (one direct delegation from principal to agent, no sub-delegation). Receipt formats MAY define a larger value but MUST document the maximum explicitly. The recommended upper bound is 8 hops; values above 32 SHOULD be rejected as operationally unsafe.

4. Cryptographic Linkage

4.1. JCS Canonical Preimage Discipline

The linkage between consecutive lifecycle states is a JCS Preimage Hash: the SHA-256 digest of the UTF-8 encoding of the JCS ([RFC8785]) canonical form of the prior state’s canonical preimage object.

The canonical preimage object is a JSON object containing the fields defined for each state in Section 2. Keys MUST be sorted lexicographically per [RFC8785] Section 3.2.3. String values MUST be in Unicode Normalization Form C before serialisation. Float values for integer-typed fields (such as amount or issued_at) MUST be rejected before canonicalisation.

The JCS Preimage Hash is encoded as:

```
original_payment_ref = "sha256:" || lowercase_hex(SHA-256(UTF-8(JCS(preimage))))
```

This encoding is identical to the action_ref derivation defined in [STARK-RECEIPTS] and in the shared canonicalisation discipline coordinated at the x402 working group ([X402-2326]).

4.2. Chain Reconstruction Algorithm

A supervisor reconstructing a full payment history from retained off-VM manifests applies the following algorithm:

1. Collect all Payment Claims associated with a payment session (identified by the PaymentIntent id or a session token established by the receipt format).

2. For each SettlementReceipt, verify that SHA-256(UTF-8(JCS(PaymentIntent_preimage))) equals the claim’s original_payment_ref. If the digests do not match, reject the chain as tampered.

3. For each RefundClaim, verify that SHA-256(UTF-8(JCS(SettlementReceipt_preimage))) equals the claim’s original_payment_ref. If the digests do not match, reject the chain as tampered.

4. Verify that all state transitions in the reconstructed chain are legal per Section 3. Any forbidden transition (Section 3.4) encountered during reconstruction MUST cause the supervisor to mark the chain as invalid.

The chain is valid if and only if all digest checks pass and no forbidden transition appears. A valid chain provides a tamper-evident audit trail from PaymentIntent to SettlementReceipt and (if applicable) to RefundClaim.

4.3. Resistance to Forking Attacks

A forking attack occurs when an adversary attempts to produce two valid SettlementReceipts for the same PaymentIntent (double settlement), or two valid RefundClaims for the same SettlementReceipt (double refund). The FSM resists forking by the following mechanisms:

1. The anti-replay nonce in PaymentIntent MUST be marked as consumed on first settlement (Section 3.1 postcondition 4). A second PAYMENT-SIGNATURE presenting the same nonce MUST be rejected.

2. The original_payment_ref in SettlementReceipt is deterministic: any two valid SettlementReceipts for the same PaymentIntent will carry the same original_payment_ref. A Verifier maintaining a original_payment_ref-indexed store can detect and reject a second settlement attempt.

3. The same property applies to RefundClaim: any two RefundClaims for the same SettlementReceipt share the same original_payment_ref (the hash of the SettlementReceipt). A Verifier that records issued RefundClaims by original_payment_ref can detect duplicate refund attempts.

Receipt formats implementing this FSM MUST maintain an indexed store of consumed nonces and issued linkage digests. The store indexing strategy is format-specific; the FSM only requires that the store enables O(1) lookup at state transition time.

5. Compatibility with Receipt Formats

Any receipt format that emits JCS canonical preimage hashes for each lifecycle state can implement this FSM. The format is not required to use the field names original_payment_ref defined here; it MUST document the mapping from its own field names to the FSM linkage fields. Verifiers that support multiple receipt formats MUST resolve the mapping before applying the chain reconstruction algorithm (Section 4.2).

The x402 STARK Receipt Format Extension [STARK-RECEIPTS] is the first reference implementation of this FSM. The SettlementReceipt.original_payment_ref field defined in that extension is the canonical spelling; other formats SHOULD use the

same field name for interoperability. Formats that use an alternative field name MUST register the mapping in the x402 Receipt Format registry or document it in a published specification.

Other formats may bind the FSM via signature-based linkage mechanisms (such as HMAC-SHA256 or a digital signature over the canonical preimage bytes), provided the binding is deterministic and produces a 32-byte digest that satisfies the preimage rules of Section 4.1. Signature-based binding does not replace the JCS canonical preimage requirement; it is an additional layer on top.

6. Security Considerations

6.1. State Spoofing

An adversary may attempt to submit a SettlementReceipt for a PaymentIntent that was never created, or a RefundClaim for a SettlementReceipt that was never issued. Verifiers MUST validate the original_payment_ref linkage against retained manifests before accepting any downstream state transition. Accepting a receipt without verifying its original_payment_ref against the prior state's known-good digest enables state-spoofing attacks.

Mitigation: chain reconstruction (Section 4.2) is the normative defence. Verifiers that cannot retain Payment Claim manifests SHOULD use an external audit log maintained by a trusted third party with its own chain-verification capability.

6.2. Replay Across Linkage Horizons

The deterministic nature of JCS Preimage Hashes means that a SettlementReceipt for a given PaymentIntent always carries the same original_payment_ref. This property is exploited by forking defences (Section 4.3), but it also means that an adversary in possession of a valid SettlementReceipt can derive the original_payment_ref for a future RefundClaim without ever seeing the SettlementReceipt preimage, provided the PaymentIntent preimage is known.

Mitigation: anti-replay stores MUST be indexed by both the linkage digest and the specific state type (Settlement vs. Refund). An original_payment_ref that has been used as the basis of a RefundClaim MUST be marked as refunded in the store; a second RefundClaim presenting the same original_payment_ref MUST be rejected.

Receipt formats MUST enforce temporal constraints on refund acceptance; the default 30-day refund window (Section 2.3) limits the replay horizon.

6.3. Cross-Format Interleaving Attacks

A cross-format interleaving attack occurs when a SettlementReceipt emitted by format A is presented as a valid predecessor to a RefundClaim emitted by format B, where the two formats use incompatible preimage schemas. If a Verifier accepts mixed-format chains without verifying preimage schema compatibility, the JCS digest check may pass spuriously (if the two schemas happen to produce the same bytes for different semantic content) or fail with an opaque error (masking a tampered chain).

Mitigation: Verifiers MUST reject mixed-format chains unless both formats implement the canonical preimage discipline of Section 4.1 with identical field definitions and encoding rules for the overlapping fields. Receipt formats MUST declare their preimage schema version in each emitted claim, enabling Verifiers to detect schema mismatches before applying the digest check.

6.4. DelegationGrant Scope Inflation

A delegatee that has received a DelegationGrant may attempt to spawn PaymentIntents whose scope exceeds the grant’s constraints (currency, payee, or amount cap). If the Verifier does not check scope compliance before settling, the delegation mechanism provides no meaningful access control.

Mitigation: scope enforcement is normative (Section 3.3 precondition 3). Verifiers MUST check scope compliance at settlement time, not only at grant issuance time. The max_chain_length parameter (Section 3.4) bounds privilege escalation via sub-delegation chains; the default value of 1 permits only direct principal-to-agent delegation with no intermediate nodes.

7. IANA Considerations

This document defines no new IANA registries. Implementations bind to existing x402 receipt-format registries. The FSM state names (PaymentIntent, SettlementReceipt, RefundClaim, DelegationGrant) and transition error tokens defined in Section 3.4 are informational identifiers; they do not require IANA registration.

Receipt formats implementing this FSM and registering in the x402 Receipt Format registry (established by [STARK-RECEIPTS]) SHOULD include in their registration a reference to this document to indicate FSM compliance.

Acknowledgments

The authors thank the x402 Foundation coalition contributors who established the shared canonicalisation discipline through the three-implementation consensus thread [X402-2357] and the multi-runtime conformance work in [X402-2432]. The 18 DelegationGrant vectors cross-validated byte-identical across five language runtimes in [X402-2432] provided the empirical basis for the bounded authority path definition in Section 3.3. The shared canonicalisation discipline [X402-2326] grounded the JCS preimage linkage rules in Section 4.1.

References

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[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/rfc/rfc8785>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

Informative References

[STARK-RECEIPTS] "x402 STARK Receipt Format Extension", Work in Progress, Internet-Draft, draft-vauban-x402-stark-receipts-00, n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-stark-receipts/>.

[X402-2326] "Shared canonicalisation discipline (privacy_class and receipt-format extensions)", n.d., <https://github.com/x402-foundation/x402/issues/2326>.

[X402-2357]
"x402 STARK Receipts Extension Proposal", n.d.,
<https://github.com/x402-foundation/x402/issues/2357>.

[X402-2432]
"Bounded-spend authorization sample: 18-vector cross-runtime conformance matrix", n.d.,
<https://github.com/x402-foundation/x402/pull/2432>.

[X402-V2] "x402 Linux Foundation V2 Working Group", n.d.,
<https://github.com/x402-foundation/x402>.

Appendix A. FSM State Summary

The following table summarises the four canonical states, their roles, and their linkage requirements.

State	Role	Required linkage field
PaymentIntent	Initiating state; payer commitment	None (origin state)
SettlementReceipt	Terminal success state; facilitator confirmation	original_payment_ref = JCS Preimage Hash of PaymentIntent
RefundClaim	Terminal reversal state; merchant-initiated	original_payment_ref = JCS Preimage Hash of SettlementReceipt
DelegationGrant	Side-channel authority; scopes future intents	Receipt-format-specific binding to spawned PaymentIntents

Table 6

Appendix B. Legal Transition Table

The following table is a machine-readable summary of legal and forbidden transitions for use in conformance testing. Verifiers MAY generate test cases from this table.

From state	To state	Legal?	Condition
(none)	PaymentIntent	Yes	Origin; no precondition.
PaymentIntent	SettlementReceipt	Yes	All preconditions in Section 3.1.
SettlementReceipt	RefundClaim	Yes	All preconditions in Section 3.2.
(none)	DelegationGrant	Yes	Origin side-channel; no prior state required.
DelegationGrant	PaymentIntent	Yes	All preconditions in Section 3.3.
PaymentIntent	RefundClaim	No	RefundWithoutSettlement; settlement must precede refund.
PaymentIntent	PaymentIntent	No	Intents do not chain; each is an independent origin.
SettlementReceipt	SettlementReceipt	No	AlreadySettled; one settlement per intent.
RefundClaim	RefundClaim	No	AlreadyRefunded (default FSM; format may override for partial refunds).
RefundClaim	SettlementReceipt	No	Refunds are terminal; no re-settlement after reversal.
DelegationGrant	SettlementReceipt	No	Settlement requires an intervening PaymentIntent.
DelegationGrant	RefundClaim	No	Refund requires an intervening PaymentIntent and SettlementReceipt.

Table 7

Appendix C. Conformance Checklist

A receipt format claiming FSM compliance MUST:

* Emit original_payment_ref (or an equivalently named field with documented mapping) in every SettlementReceipt, set to the JCS Preimage Hash of the originating PaymentIntent.

* Emit original_payment_ref (or equivalent) in every RefundClaim, set to the JCS Preimage Hash of the SettlementReceipt being reversed.

* Reject all forbidden transitions listed in Section 3.4.

* Maintain an anti-replay store sufficient to detect duplicate settlements and duplicate refunds.

* Enforce the default 30-day refund window or document an explicit alternative.

* Declare a max_chain_length for DelegationGrant chains; if not declared, the default value of 1 applies.

* Enforce scope compliance for DelegationGrant-spawned PaymentIntents at settlement time.

* Declare its preimage schema version in each emitted claim to enable cross-format interleaving detection.

A Verifier claiming FSM compliance MUST:

* Apply the chain reconstruction algorithm (Section 4.2) before accepting any SettlementReceipt or RefundClaim for audit purposes.

* Reject mixed-format chains unless schema compatibility is confirmed.

* Reject all transitions listed as forbidden in Appendix B.

* Emit the error tokens defined in Section 3.4 when rejecting illegal transitions.

Known Adopters and Reference Implementations

This appendix documents reference implementations and adopters of this specification confirmed at the time of publication. The list is informational and will be updated in subsequent revisions as additional implementations are reported.

Primary Maintainer

Vauban Pay (https://pay.vauban.tech) maintains the reference specification, the published conformance vectors (https://github.com/vauban-org/x402-stark-receipts-conformance), and the reference implementations listed below. The Vauban Pay live demonstration emits compliant payloads at https://demo.pay.vauban.tech. The on-chain anchor PaymentDemoEmitter is deployed on Starknet Sepolia at contract 0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff.

Reference Implementation Matrix

The conformance vector suite maintains an 8-implementation reference matrix : Python (rfc8785@0.1.4 by Trail of Bits), JavaScript (canonicalize@3.0.0 by Erdtman and Rundgren), Go (gowebpki/jcs v1.0.1), Java (cyberphone/json-canonicalization RFC 8785 reference), Rust (serde_jcs 0.2.0 via vauban-x402-jcs-conformance), PHP/Ruby/C# (pure-stdlib reference runners). The first five are validated byte-for-byte ; the last three are published as reference runners pending CI execution. Detailed validation status is documented in _attestations/2026-05-25-vauban-8-impl-extended.md in the conformance vectors repository.

The published Vauban Pay packages across 3 ecosystems : vauban-x402-jcs-conformance@0.1.0, vauban-x402-canonical@0.1.0, vauban-x402-wire@0.1.0 on crates.io ; vauban-x402-stark-receipt@0.1.0 on PyPI ; @vauban-pay/substrate@0.1.0 on npm.

Adoption Process

Implementers SHOULD notify the IETF contact at research@vauban.tech when adopting this specification in production. Adoption notifications include the production endpoint or product identifier, the lifecycle FSM revision implemented, the canonicalisation discipline emitted, and the contact for follow-on coordination. Reported adopters will be listed in the next revision of this appendix following a verification step against the conformance vector matrix.

Author’s Address

Vauban Research
Vauban Research
Email: research@vauban.tech
URI: https://pay.vauban.tech