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

V. Research
Vauban Research

25 May 2026

x402 STARK Receipt Format Extension
draft-vauban-x402-stark-receipts-02

Abstract

The x402 PAYMENT-RESPONSE carries a facilitator-issued settlement reference but does not provide a self-contained, offline-verifiable payment-condition proof. A verifier must today contact the facilitator to confirm whether a specific amount, currency, and payer attestation were satisfied at the time of payment. This gap prevents compliance with EU AI Act Art. 12 (transparency and documentation) and MiCA Art. 76 (settlement record-keeping) in automated payment pipelines.

This document defines the receipt-format extension for x402 V2. The extension introduces a negotiable receipt format selection mechanism that allows resource servers, facilitators, and clients to agree on which cryptographic receipt variant to produce and verify, without altering the core PAYMENT-RESPONSE wire structure. Three variants are defined: a Stwo Circle STARK proof of payment conditions (post-quantum sound, offline verifiable), a hybrid ES256K + ML-DSA-65 dual-signature receipt (receipt integrity under quantum adversary), and a classical ES256K fallback. A canonical preimage discipline using JCS ([RFC8785]) ensures cross-implementation digest consistency.

The canonical preimage discipline in this document is cross-validated against the x402 canonicalisation conformance set, which is checked byte-for-byte across five independent RFC 8785 ([RFC8785]) implementations in five languages (Python, TypeScript, Go, Java, Rust).

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/.

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 PAYMENT-RESPONSE currently carries a payment_hash and a facilitator-issued settlement reference. However, it does not include a self-contained cryptographic receipt that a downstream verifier can check offline, without contacting the facilitator.

This limitation creates a compliance gap in two regulatory frameworks:

* EU AI Act Art. 12 requires that automated decision-making systems maintain transparent, auditable records. A payment pipeline that cannot produce an offline-verifiable receipt cannot satisfy this requirement without a facilitator callback.

* MiCA Art. 76 requires settlement record-keeping for crypto-asset service providers. A facilitator-issued reference alone is insufficient when the verifying party is not the original payment processor.

The receipt-format extension addresses this gap by enabling negotiation of three cryptographic receipt variants, each optimised for a different compliance or security property. The extension is additive: it does not modify the core PAYMENT-RESPONSE structure, and it defaults safely to a classical ES256K fallback for implementations that do not require ZK or post-quantum properties.

Three independent implementations established the extension semantics and the three receipt variants in [X402-2357]:

* Vauban zkpay (Rust, Apache 2.0): STARK receipt generation and verification

* FeedOracle Grounding Receipt v0.4: hybrid PQC dual-signature

* andysalvo action-ref-verify v0.3.0: canonical preimage conformance suite

The canonical preimage discipline shared by all three variants is additionally cross-validated against the x402 canonicalisation conformance set across five RFC 8785 implementations in five languages (Python, TypeScript, Go, Java, Rust). The Vauban Rust conformance runner uses serde_jcs 0.2.0 and is available under Apache 2.0. Cross-validation coverage: 53/53 vectors, 37/37 pair invariants byte-for-byte across all published canonicalisation vector sets. The

canonicalisation discipline is anchored in [RFC8785] and urn:x402:canonicalisation:jcs-rfc8785-v1; it is not bound to any single host surface.

Conformance test vectors for the action_ref work-receipt binding are published in [X402-2398] (fixtures/action-ref-verify/v0/). The canonicalisation rules these vectors exercise are cross-validated in the broader x402 canonicalisation conformance set [X402-CANON].

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.

2. Lifecycle Scope

The stark-vauban-pay-v1 receipt format defined in this document covers the full lifecycle of a payment transaction in a single unified specification :

* Settlement (the canonical successful-payment receipt) ; see Variant-Specific Receipt Bodies (Appendix A) and in particular stark-vauban-pay-v1 (Appendix A.1)

* Refund (canonical reversal receipt, content-addressed to the originating Settlement via JCS canonical preimage hash and the original_payment_ref field) ; see Variant-Specific Receipt Bodies (Appendix A)

* Delegation (bounded-spend authorization receipt, cryptographically scoped to the granting principal) ; see Variant-Specific Receipt Bodies (Appendix A)

Each lifecycle state is cryptographically bound to its predecessor via STARK-anchored proof chains as defined in Canonical Preimage Discipline (Section 9). Verifiers reconstruct full payment histories by walking the chain of original_payment_ref and JCS canonical preimage hash linkages, without requiring out-of-band coordination between independent receipt formats.

The unified single-document approach preserves a single audit-time re-verifiable evidence surface per payment lifecycle. Implementations that fragment receipt definitions across multiple specifications introduce ambiguity at state-transition boundaries (Refund following Settlement, Delegation preceding either) and require verifier-side coordination logic absent from this specification. The stark-vauban-pay-v1 receipt format closes that gap by binding the four canonical PaymentClaim states (PaymentIntent, SettlementReceipt, RefundClaim, DelegationGrant) under one cryptographic discipline.

3. Conventions and Definitions

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.

The following terms are used throughout this document:

Receipt format: A string token identifying which cryptographic receipt variant a facilitator produces. See Section 4.

JCS: JSON Canonicalization Scheme, defined in [RFC8785]. Produces a deterministic byte representation of a JSON value by applying recursive key sorting and value normalization.

action_ref: A 32-byte opaque digest binding a payment receipt to a work-layer event. Derived by SHA-256 over the UTF-8 JCS encoding of a canonical preimage object. See Section 10.

payment_hash: A hash of the payment conditions committed to by the payer, as defined in the x402 V2 base specification ([X402-V2]).

Facilitator: An entity that processes x402 payment requests, verifies payment conditions, and issues PAYMENT-RESPONSE messages. The term is used as defined in [X402-V2].

NFC: Unicode Normalization Form C, as defined in [UAX15].

compliance_receipt: The internal screening decision record produced by a compliance engine (ALLOW, DENY, or REFER) before wire-format translation. Distinct from the consumer-facing RiskCheckResult signal defined in [X402-2434]. See Section 7.

4. Receipt Format Enum

The receipt_format field identifies which cryptographic receipt variant a facilitator has produced. The value is a string token from the registry defined in Section 16:

Token	Description	Approx. size
stark-vauban-pay-v1	Stwo Circle STARK over payment-condition witness (amount, currency, payer attestation, nullifier). Post-quantum sound. Offline verifiable.	~100 KB
hybrid-pqc	ES256K + ML-DSA-65 ([FIPS204]) dual-signature over JCS-canonical receipt core. Receipt survives Q-Day independently of proof system upgrade.	~3.3 KB
classical-es256k	ES256K signature only. Default fallback for clients that do not require post-quantum or ZK properties.	~0.5 KB

Table 1

Each variant corresponds to evidenceType: "cryptographic" in the [X402-2322] compliance taxonomy. The signature_algorithm discriminator used in the bazaar evidenceShape maps as follows:

receipt_format token	signature_algorithm
stark-vauban-pay-v1	"stark-m31-stwo"
hybrid-pqc	"es256k+ml-dsa-65"
classical-es256k	"es256k"

Table 2

Facilitators MAY introduce additional tokens by registering them in the x402 Receipt Format registry defined in Section 16. Tokens not present in the registry MUST be treated as "classical-es256k" by verifiers that do not recognise them.

The three variants correspond to three orthogonal properties; a deployment selects the property it requires:

* *proof-of-payment-conditions*: the stark-vauban-pay-v1 receipt proves amount, currency, payer attestation, and nullifier without revealing them. The STARK proof is the deliverable.

* *receipt-integrity-under-quantum*: the hybrid-pqc receipt carries two independent signatures; the receipt remains verifiable if one signature algorithm is broken.

* *work-receipt-binding*: all three variants carry an optional action_ref field (32-byte opaque digest per Section 10). Binding to the work layer is a property of the preimage derivation, not of the receipt format itself. The same action_ref value also serves as the composite trust-query anchor; see Section 10 and [X402-COMPOSITE].

5. HTTP Header Conventions

5.1. X-Payment-Options (402 response, server to client)

A resource server or facilitator SHOULD include X-Payment-Options in the 402 Payment Required response to advertise which receipt_format variants it can produce.

```
X-Payment-Options: receipt_format="stark-vauban-pay-v1, hybrid-pqc, classical-es256k"
```

The value is a comma-separated ordered list of receipt_format tokens, from most-preferred to least-preferred. The server declares its capabilities; the client selects from this list.

If X-Payment-Options is absent, the client MUST assume the facilitator can produce only classical-es256k.

5.2. X-Receipt-Format (PAYMENT-RESPONSE, facilitator to client)

The facilitator MUST include X-Receipt-Format in the PAYMENT-RESPONSE to state which variant it emitted.

```
X-Receipt-Format: stark-vauban-pay-v1
```

The verifier uses this header to select the correct receipt parser before attempting to deserialise or verify the receipt body.

If X-Receipt-Format is absent, the verifier MUST assume classical-es256k.

6. Negotiation Semantics

6.1. Normal Negotiation Path

1. Client receives a 402 response with X-Payment-Options.
2. Client selects the highest-preference variant it can verify from the list.
3. Client SHOULD signal its selection to the facilitator by including receipt_format in the PAYMENT-SIGNATURE request payload (see Section 14).
4. Facilitator produces the receipt in the selected variant and sets X-Receipt-Format on the PAYMENT-RESPONSE.

6.2. Fallback Behaviour

If the client cannot verify any variant in X-Payment-Options, or if X-Payment-Options is absent, the client MUST fall back to classical-es256k. The facilitator MUST honour this fallback.

A facilitator that lists stark-vauban-pay-v1 in X-Payment-Options MUST also be capable of producing classical-es256k as a fallback. Listing a variant without fallback capability is a protocol violation.

6.3. Mandatory-Format Signalling

If a client REQUIRES a specific receipt format (for example, a regulator requiring an offline-verifiable STARK receipt for EU AI Act Art. 12 purposes), it MUST include receipt_format in the PAYMENT-SIGNATURE request extensions:

```
{
  "extensions": {
    "receipt-format": {
      "info": {
        "required": true,
        "receipt_format": "stark-vauban-pay-v1"
      }
    }
  }
}
```

If required is true and the facilitator cannot produce the requested variant, the facilitator MUST return HTTP 402 with error UnsupportedReceiptFormat (see Section 8) rather than silently downgrading.

If required is false (or absent), the facilitator MAY downgrade to the highest variant it supports.

7. Compliance Receipt and RiskCheckResult Layering

Two distinct protocol-layer constructs are involved in screening-aware payment flows. Implementations MUST NOT conflate them.

7.1. compliance_receipt (Internal Screening Decision Record)

A compliance_receipt is the internal record of a screening decision produced by a compliance engine before any wire-format translation. It carries the raw decision outcome (ALLOW, DENY, or REFER), the screening criteria applied, and any structured evidence supporting the outcome. The compliance_receipt is NOT directly transmitted to the client; it is the source material from which the facilitator derives the consumer-facing signal.

The three decision outcomes are:

* *ALLOW*: the payment request passes all screening criteria; the facilitator MAY proceed to settlement.

* *DENY*: the payment request fails one or more screening criteria; the facilitator MUST reject and MUST NOT issue a PAYMENT-RESPONSE.

* *REFER*: the payment request requires human review before a final ALLOW or DENY can be issued; the facilitator MUST NOT proceed to settlement until the referral is resolved.

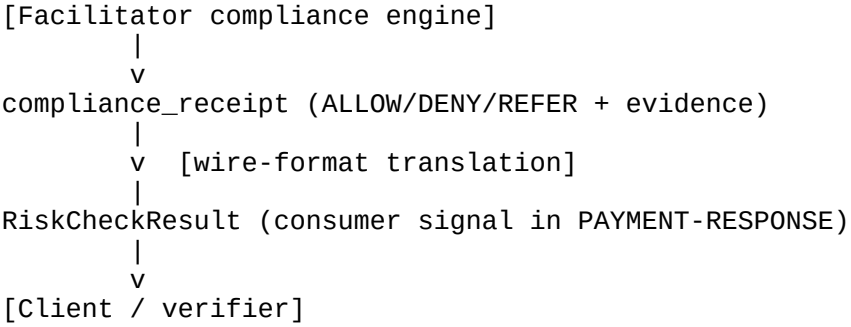
The cross-validation exercise described in Section 1 (15/15 byte-for-byte agreements across five implementations) validates the JCS-canonical encoding of the compliance_receipt shape as defined in [X402-2434].

7.2. RiskCheckResult (Consumer-Facing Wire-Format Signal)

The RiskCheckResult (defined in [X402-2434]) is the consumer-facing wire-format signal derived from the compliance_receipt. It is designed for transmission in PAYMENT-RESPONSE extensions and carries only the information the client needs: the decision outcome, an opaque reference to the underlying compliance_receipt, and the action_ref binding when applicable.

7.3. Relationship and Layering

The compliance_receipt sits BELOW the RiskCheckResult in the protocol stack:



A verifier receiving a RiskCheckResult MUST NOT assume access to the underlying compliance_receipt. The compliance_receipt is a facilitator-internal artefact; its retention, format, and access controls are outside the scope of this extension. The action_ref field in the RiskCheckResult, when present, provides the binding to the work-layer event and to the composite trust-query anchor ([X402-COMPOSITE]).

8. Error Taxonomy

When the facilitator or resource server rejects a payment due to a condition attributable to this extension, it SHOULD include the X-Receipt-Reject-Reason header with a machine-readable semantic discriminator. The header supplements the HTTP status code; it does not replace it.

X-Receipt-Reject-Reason: NullifierReplay

8.1. HTTP Status Code Mapping

Status	Reason token	Description
400	MalformedClaim	Malformed CBOR or invalid JCS in the request body. Parse failed before validation.
402	PaymentRequired	Generic unmet payment condition: amount mismatch, invalid proof, payer attestation failed.
402	UnsupportedReceiptFormat	Client requested a receipt_format with required: true that the facilitator cannot produce.
409	NullifierReplay	Nullifier already settled. Double-spend attempt detected. The PAYMENT-RESPONSE MUST NOT be issued.
410	Expired	TemporalFrame window has elapsed. The payment cannot be settled.
422	StructuralInvalid	Claim structure violates the VPSF sextuplet invariant. Structurally unprocessable.
451	HumanityRequired	Merchant requires a humanity-binding attestation; payer is anonymous and cannot satisfy the requirement.

Table 3

8.2. X-Receipt-Reject-Reason Format

The header value MUST be exactly one reason token from the table above. Verifiers MUST treat unknown tokens as equivalent to the corresponding HTTP status code with no additional semantics.

Facilitators SHOULD include the header on all non-2xx responses in this extension's domain. The header is OPTIONAL on 5xx responses.

9. Canonical Preimage Discipline

The canonicalisation rules used by this extension are defined normatively by the x402 shared canonicalisation discipline [X402-CANON], identified by the stable URI urn:x402:canonicalisation:jcs-rfc8785-v1. That section codifies the digest construction JCS_hash = SHA-256(JCS(object)) over the JSON Canonicalization Scheme [RFC8785], the version marker jcs-rfc8785-v1, and four normative rules: timestamps are encoded as integers, field names are compared as byte strings, arrays preserve their order, and type validation is performed before canonicalisation.

This section does not restate those rules. It profiles them for the action_ref preimage of this extension: it fixes the four-field preimage schema (Section 9.2), gives the worked digest for the shared coalition preimage, and records the conformance-vector pointers. Where this section previously stated a generic canonicalisation rule, that rule now lives in [X402-CANON]; the text below carries only what is specific to the action_ref preimage.

9.1. timestamp_ms Field Name

[X402-CANON] Rule 1 requires timestamps to be encoded as integer values. This extension fixes the field name for the action_ref preimage: it MUST be timestamp_ms, an integer count of milliseconds since the Unix epoch (1970-01-01T00:00:00 UTC). The field name timestamp carrying an RFC 3339 string MUST NOT be used in the action_ref preimage. Vector 0002-field-name-load-bearing in the canonicalisation conformance set ([X402-CANON]) demonstrates the divergence that an RFC 3339 string field produces.

9.2. Preimage Schema

The canonical action_ref preimage object contains the following fields, sorted in [RFC8785] lexicographic order for JCS canonicalisation:

```
{
  "action_type":  "<string>",
  "agent_id":     "<string>",
  "scope":        "<string>",
  "timestamp_ms": <integer>
}
```

Key ordering and serialisation follow [X402-CANON] and [RFC8785] Section 3.2.3. The JCS output for the shared coalition preimage is:

{"action_type":"sanctions_screen","agent_id":"did:web:agent-7.example.com","scope":"count
erparty-due-diligence","timestamp_ms":1747728000000}

The action_ref digest is:

```
action_ref = SHA-256(UTF-8(JCS(preimage)))
           = 10d8a38c01d8672176aa6e5209a368fde3e1831640d69e15283142b35880c2c1
```

9.3. Type Validation Requirements

[X402-CANON] Rule 4 requires type validation to be performed before canonicalisation. This subsection enumerates the type-validation failures that apply to the action_ref preimage specifically.

Implementations MUST reject preimage objects containing:

* *Float values for timestamp_ms*: 1747728000000.0 MUST be rejected; only integer JSON numbers are valid. Vector 0002-ms-precision-trap from [X402-CANON] demonstrates the failure mode where a float representation produces no parsing error but an incorrect digest. Rejection MUST occur before canonicalisation.

* *Missing canonical fields*: if any of action_type, agent_id, scope, or timestamp_ms is absent, the preimage MUST be rejected before canonicalisation. See proposed vector 0012-missing-canonical-field.

* *Duplicate keys*: if the JSON object contains duplicate key names, the preimage MUST be rejected at parse time. Accepting last-wins or first-wins semantics creates interoperability ambiguity. See proposed vector 0010-duplicate-keys.

9.4. Unicode Normalisation

[X402-CANON] deliberately applies no Unicode normalisation in the canonicaliser, so that NFC and NFD are distinct canonical inputs. This extension imposes a stricter profile on the action_ref preimage: it requires NFC and rejects non-NFC input, as specified below. This is a profile narrowing, not a contradiction of [X402-CANON].

[RFC8785] (JCS) performs no Unicode normalisation on string values. Two strings that are semantically equivalent but differ in Unicode normalization form (for example NFC vs NFD per [UAX15]) will produce different JCS bytes and therefore different digests.

Implementations MUST normalise all string values in the preimage to Unicode Normalization Form C (NFC) per [UAX15] BEFORE JCS canonicalisation.

Implementations MUST reject input where any string value in the preimage is encoded in a non-NFC form (NFD, NFKC, or NFKD). Acceptance of non-NFC input is a conformance violation.

Verifiers MUST NOT perform implicit re-normalisation. The input MUST already be NFC for the digest to be reproducible across runtimes.

Rationale: a verifier receiving an NFD-encoded preimage would compute a different digest than one receiving the NFC equivalent, silently breaking receipt verification without a parsing error. macOS HFS+ historically stored filenames in NFD decomposed form, and database collations vary; this is not a hypothetical edge case.

See proposed companion vector 0011-unicode-normalisation (NFD divergence test case) for a concrete failure mode demonstration.

9.5. Trailing Whitespace and Extra Fields

Vector 0003-trailing-whitespace from [X402-CANON] confirms that trailing whitespace in string values is significant in JCS. Implementations MUST NOT strip trailing whitespace before canonicalisation.

Vector 0004-extra-field-ignored confirms that additional fields beyond the canonical four MAY be present in the application preimage, but MUST be sorted in JCS order and included in the digest if present. The canonical binding covers whatever fields were serialised.

10. Wire-Level Binding (action_ref): Pure 32-Byte Hash, Layer-Agnostic

The action_ref field in any receipt variant is an opaque 32-byte digest. It MUST be encoded as base64url ([RFC4648] Section 5, no padding) in JSON payloads and as raw bytes in CBOR payloads.

```
{
  "action_ref": "ENijjAHYZyF2qm5SCaNo_ePhgxZA1p4VKDFCS1iAwsE"
}
```

The receipt format does not interpret the action_ref preimage. The preimage derivation is defined by the work layer (see Section 9.2 for the canonical formula). The receipt stores 32 opaque bytes and emits them on demand.

Both stark-vauban-pay-v1 and hybrid-pqc receipts carry the same action_ref bytes when bound to the same work event. This is the binding seam: a verifier confirms that the payment (receipt) and the work output (work-layer receipt referencing action_ref) are causally linked without coupling the two proof systems.

The same action_ref digest doubles as the anchor of the x402 composite trust-query [X402-COMPOSITE]. A composite trust-query is keyed by the pair (payment_hash, action_ref) and returns one independently verifiable evidence row per registered emitter. A receipt carrying an action_ref under this extension is therefore discoverable as the cryptographic row of such a query: the receipt and the composite envelope share the same 32-byte binding value with no further coupling. This extension defines only the receipt; the composite envelope, its row schema, its set-semantics row ordering, and its composite preimage construction SHA-256(JCS([rows sorted by source identifier, signature fields excluded])) are defined normatively by [X402-COMPOSITE] and are out of scope here. The action_ref field remains, for the purposes of this extension, an opaque 32-byte digest as specified above.

The action_ref field is OPTIONAL in all three receipt variants. A facilitator that does not support work-receipt binding MUST omit the field rather than emit a zero-value or null.

The verifier MUST NOT require action_ref to be present. A missing action_ref means no binding was asserted; it is not a validation error.

11. Composite Trust-Query (Axis 4)

This section provides an informative overview of the composite trust-query layer defined normatively by [X402-COMPOSITE]. Receipt-format implementations are not required to implement the composite trust-query; this section is provided for orientation.

11.1. Multi-Provider Composite Hash

When multiple compliance emitters each produce an evidence row for the same (payment_hash, action_ref) pair, a composite trust-query aggregates those rows into a single verifiable envelope. The composite hash is constructed as:

```
composite_hash = SHA-256(JCS([emitter_rows sorted by source_id, sig excluded]))
```

The set-semantics rule: row transmission order is NOT significant. Implementations MUST sort rows by source_id lexicographically before applying JCS canonicalisation and computing the composite hash. The resulting hash is independent of the order in which rows were received or transmitted.

11.2. Absent-Row Partial Response

When a composite trust-query is issued for a (payment_hash, action_ref) pair and one or more registered emitters have not yet produced a row, the query response MUST be a partial response containing only the rows that are present. An absent row is NOT a failure; it indicates that the emitter has not yet responded or has not processed the corresponding event. Verifiers MUST NOT treat an absent row as equivalent to a DENY outcome.

11.3. Reference Implementation

The reference implementation for the composite trust-query is specified in specs/composite-trust-query.md in the [X402-V2] repository, as developed under [X402-2440]. Vauban's Rust implementation of the composite trust-query conformance runner uses serde_jcs 0.2.0 for JCS canonicalisation under Apache 2.0; cross-validation vectors are published in the public conformance repository at https://github.com/vauban-org/x402-stark-receipts-conformance (https://github.com/vauban-org/x402-stark-receipts-conformance).

12. On-Chain felt252 Encoding

When a stark-vauban-pay-v1 receipt is anchored on-chain on Starknet, the action_ref digest (32 bytes, 256-bit SHA-256 output) is stored in a Starknet felt252 field element. The Stark field prime is approximately 2^251.6. Implementations MUST apply the following deterministic mask before comparing the on-chain keys[1] field against a locally-computed action_ref value:

```
hash_felt252 = sha256_bigint & ((1 << 251) - 1)
```

This operation masks the SHA-256 output to 251 bits. It is a deterministic truncation, not a collision: two distinct SHA-256 digests that differ only in the upper 5 bits produce the same felt252 encoding. For receipt integrity purposes, the probability of such a collision is negligible (approximately 2^-251 per pair). Implementations MUST apply this mask consistently at both emission time and verification time. A verifier that compares the full 256-bit digest against the on-chain felt252 value without applying the mask will report a false verification failure for receipts whose SHA-256 digest has non-zero upper bits.

The canonical check is:

on_chain_key == SHA-256(action_ref_preimage) & ((1 << 251) - 1)

This applies to the keys[1] field in the Starknet PaymentDemoEmitter event schema (Appendix B).

13. PaymentRequired Extension Schema

When a resource server supports receipt-format negotiation, it includes the extension in the PaymentRequired response body:

```
{
  "x402Version": 2,
  "error": "Payment required",
  "resource": {
    "url": "https://api.example.com/compliance-check",
    "description": "Sanctions screening service; STARK receipt required for EU AI Act
Art. 12"
  },
  "accepts": [
    {
      "scheme": "exact",
      "network": "eip155:8453",
      "amount": "50000",
      "asset": "0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913",
      "payTo": "0xPaymentAddress",
      "maxTimeoutSeconds": 60
    }
  ],
  "extensions": {
    "receipt-format": {
      "info": {
        "supported": ["stark-vauban-pay-v1", "hybrid-pqc", "classical-es256k"],
        "default": "classical-es256k"
      }
    }
  }
}
```

The receipt-format extension fields in the PaymentRequired body are:

Field	Type	Required	Description
supported	string[]	REQUIRED	Ordered list of receipt_format tokens the facilitator can produce; descending preference.
default	string	OPTIONAL	Token to use if the client sends no receipt_format preference. MUST be present in supported. Defaults to "classical-es256k" if absent.

Table 4

14. PAYMENT-SIGNATURE Extension Schema

A client that requires a specific receipt format signals its preference in the PAYMENT-SIGNATURE request:

```
{
  "extensions": {
    "receipt-format": {
      "info": {
        "required": false,
        "receipt_format": "hybrid-pqc"
      }
    }
  }
}
```

The receipt-format extension fields in the PAYMENT-SIGNATURE body are:

Field	Type	Required	Description
receipt_format	string	OPTIONAL	The token the client requests.
required	boolean	OPTIONAL	If true, the facilitator MUST produce the requested variant or return HTTP 402 with UnsupportedReceiptFormat. Default: false.

Table 5

15. PAYMENT-RESPONSE Extension Schema

The facilitator confirms the emitted variant in the PAYMENT-RESPONSE:

```
{
  "extensions": {
    "receipt-format": {
      "info": {
        "receipt_format": "stark-vauban-pay-v1",
        "receipt": "<base64url-encoded-receipt-body>",
        "action_ref": "ENijjAHYZyF2qm5SCaNo_ePhgxZA1p4VKDFCS1iAwsE"
      }
    }
  }
}
```

The receipt-format extension fields in the PAYMENT-RESPONSE body are:

Field	Type	Required	Description
receipt_format	string	REQUIRED	The token identifying which variant was produced.
receipt	string	REQUIRED	base64url-encoded receipt body. Format is variant-specific (see Appendix A).
action_ref	string	OPTIONAL	base64url-encoded 32-byte work-receipt binding digest (Section 10).

Table 6

16. IANA Considerations

16.1. Registry: x402 Receipt Format

This document requests IANA to create a new registry named x402 Receipt Format in the x402 Protocol Extensions registry group (to be established by the x402 Foundation TSC in coordination with IANA).

Registration procedure: Specification Required (per [RFC8126] Section 4.6). A Designated Expert MUST review each registration request to confirm the specification is publicly available, the token follows the naming convention in Section 16.1.2, and no token collision exists with a currently registered value.

Designated Experts: The initial Designated Experts pool MUST be selected by the x402 Foundation TSC at the time of registry creation. The TSC MAY rotate Designated Experts by consensus. A minimum of two active Designated Experts MUST be maintained at all times.

16.1.1. Registry Fields

Each registered value MUST include the following fields:

Field	Description
Name	The receipt_format token string. MUST match [a-z][a-z0-9-]* (lowercase alphanumeric and hyphen).
Reference	URL to the specification or authoritative test fixture defining the wire format.
Status	One of: Stable, Provisional, Deprecated, Obsolete.
Contact	Name or organisation responsible for the registration.
Notes	Optional free-text notes on implementation or compatibility constraints.

Table 7

16.1.2. Naming Convention

New receipt_format tokens SHOULD follow the pattern <scheme>-<vendor-or-family>-vN (for example: stark-vauban-pay-v1, hybrid-pqc, classical-es256k). The pattern is RECOMMENDED for clarity but not enforced by the Designated Expert review. Tokens that deviate MUST include a Notes explanation.

16.1.3. Status Transitions

* *Provisional to Stable*: requires (a) two independent interoperable implementations whose conformance against the reference specification has been demonstrated publicly, and (b) the wire format has been stable for at least one year with no backwards-incompatible changes. A Designated Expert MUST confirm both conditions before updating status.

* *Stable to Deprecated*: requires a vote by the x402 Foundation TSC with public notice of at least six months before the status change takes effect.

* *Deprecated to Obsolete*: occurs automatically twelve months after the Deprecated status date, unless the TSC votes to extend the deprecation period.

* *Obsolete*: verifiers MUST NOT accept receipts of Obsolete format variants. Facilitators MUST NOT produce Obsolete variants.

16.1.4. Initial Registry Values

The following values are registered by this document:

Name	Reference	Status	Contact	Notes
stark-vauban-pay-v1	[VAUBAN-STARK-GIST]	Provisional	Vauban Research	Stwo Circle STARK M31 over payment-condition witness. CBOR-encoded body. ~100 KB.
hybrid-pqc	[FEEDORACLE-GIST]	Provisional	FeedOracle	ES256K + ML-DSA-65 ([FIPS204]) dual-signature. JCS-canonical JSON body. ~3.3 KB.
classical-es256k	[X402-V2] base specification	Provisional	x402 Foundation	ES256K JWS Compact Serialization. Default fallback. ~0.5 KB.

Table 8

These initial values are considered Provisional pending the Stable promotion criteria in Section 16.1.3.

17. Security Considerations

17.1. Canonical Preimage Validation Before Content Processing

Implementations MUST validate the canonical preimage discipline described in Section 9 BEFORE acting on any receipt content. Accepting a receipt whose action_ref was derived from a non-canonical preimage (float timestamp_ms, missing fields, duplicate keys, non-NFC strings) creates a binding gap: the receipt may appear structurally valid while the work-layer binding is silently broken. A verifier that processes receipt content before checking preimage validity

cannot assert the causal link between the payment and the work event.

17.2. Replay Attack Mitigation

The NullifierReplay error code (Section 8.1) is the primary mechanism for preventing double-spend. Facilitators MUST maintain a nullifier store indexed by the settlement ledger and MUST reject any PAYMENT-SIGNATURE whose nullifier is already present. The timestamp_ms field in the action_ref preimage further bounds the replay window; facilitators SHOULD enforce a maximum timestamp_ms age aligned with the maxTimeoutSeconds field in the PaymentRequired response.

17.3. Timing Side-Channel Risks in STARK Proof Generation

STARK proof generation (the stark-vauban-pay-v1 variant) is computationally intensive and its duration correlates with the witness size. On shared infrastructure, an adversary observing proof generation latency MAY be able to infer approximate witness content. Implementors SHOULD run proof generation in isolated compute environments (dedicated VMs or sandboxed processes) and SHOULD add randomised padding to the proof generation timeline where latency is observable by external parties.

17.4. Privacy Implications of Receipt Sharing

A stark-vauban-pay-v1 receipt proves payment conditions without revealing the witness (amount, currency, payer attestation, nullifier). However, the action_ref digest is deterministic given the preimage. If the preimage is known to an adversary (for example, because the agent_id and scope are public), the adversary can confirm whether a given payment was bound to a specific work event. Implementations that require unlinkability MUST use opaque, non-guessable agent_id and scope values.

The hybrid-pqc and classical-es256k receipts do not provide zero-knowledge properties. They carry the receipt_core in cleartext (JCS-canonical JSON). Parties sharing these receipts with third-party verifiers SHOULD be aware that the full payment conditions are visible to the verifier.

17.5. Forward Secrecy Under Quantum Adversary

The hybrid-pqc variant is designed to preserve receipt integrity if either ES256K or ML-DSA-65 is broken. However, it does not provide forward secrecy: an adversary that records receipts today and breaks ES256K or ML-DSA-65 in the future can verify the receipt. If forward secrecy is required, the stark-vauban-pay-v1 variant provides post-quantum sound proof-of-payment-conditions; however, the STARK proof

system itself is subject to future cryptanalysis. This document makes no claim about the long-term security of any specific proof system configuration.

17.6. Facilitator Trust Boundary

The security model of this extension assumes facilitator integrity for receipt issuance. A compromised facilitator can issue false receipts regardless of the cryptographic variant. The extension does not provide a mechanism to verify that the facilitator was itself uncompromised at the time of issuance. Verifiers that require a trust-minimised settlement proof SHOULD combine the stark-vauban-pay-v1 receipt with an on-chain settlement anchor (separate from this extension).

17.7. Retention-Property Determinism (Cross-Observer-Across-Time)

The canonical preimage discipline in Section 9 produces a digest that two observers verifying at the same instant compute identically. For receipts emitted under a regulatory framework with a statutory retention obligation, the property MUST also hold across time: a supervisor or auditor re-verifying in year N against a retained off-VM manifest of the receipt MUST reproduce the same digest as the original verifier did at issuance time.

This raises a versioning concern. If the canonical rule (JCS, type-validation, field-name canonicalisation, Unicode normalisation policy) is revised between issuance and re-verification, the retained receipt becomes unreproducible under the then-current rule. Verifier-side coercion of non-conforming inputs (rather than rejection) further breaks re-verifiability because the coercion step is verifier-local and is not replayed at audit time.

Producers emitting receipts under frameworks with statutory retention obligations (for example, MiCA Art. 80, AMLR Art. 56, DORA Art. 14) MUST therefore:

* Include the canonicalisation version marker in a canon_version field in the receipt preimage at emission time. The value MUST be a registered x402 canonicalisation version identifier; the discipline defined in [X402-CANON] is identified by jcs-rfc8785-v1 (URI urn:x402:canonicalisation:jcs-rfc8785-v1). A future verifier selects the contemporaneous rule by this marker rather than applying the then-current rule.

* Reject (not coerce) non-conforming inputs at the parse or schema-validation step, preserving re-verifiability against the raw retained object.

Producers emitting receipts outside such frameworks SHOULD include canon_version as a good-discipline default; the field becomes a MUST under any framework that imposes a statutory retention horizon. This subsection profiles the shared x402 canonicalisation discipline [X402-CANON] for the retention case: it requires the version marker to be persisted in the receipt preimage so that re-verification under a statutory retention horizon is reproducible.

17.8. felt252 Mask and On-Chain Comparison

The felt252 masking operation described in Section 12 is a deterministic transformation, not a cryptographic weakening. However, implementations that omit the mask at verification time will silently produce false-negative verification results for receipts whose SHA-256 digest has non-zero bits in positions 251-255. Such false negatives do not create a security vulnerability (they deny valid receipts rather than accept invalid ones), but they create an operational failure mode in production payment pipelines. Implementations SHOULD test the mask against a receipt whose action_ref digest has a known non-zero high bit before deploying to production.

17.9. Open Research Conjectures

Section 10 of the Vauban zkpay specification contains research conjectures about the composition properties of payment claims under the Vauban Proof Stack Framework. These conjectures have not been peer-reviewed at the time of this writing. Implementors SHOULD treat them as working hypotheses. A companion paper is planned for submission to IACR ePrint; readers are directed there for the formal treatment once published.

18. References

18.1. 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, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/rfc/rfc7235>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

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

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

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

18.2. Informative References

[ANDYSALVO-GIST]
"andysalvo action-ref-verify v0.3.0", n.d., <https://gist.github.com/andysalvo/763a410497454ce78be0fd9dae26a6b4>.

[FEEDORACLE-GIST]
"FeedOracle hybrid-PQC receipt fixture", n.d., <https://gist.github.com/feedoracle/704ab891170e2b43050f6f0ae00e6923>.

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard (ML-DSA-65)", 2024, <https://doi.org/10.6028/NIST.FIPS.204>.

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

[UAX15] Unicode Consortium, "Unicode Normalization Forms", Unicode Standard Annex #15, 2023, <https://www.unicode.org/reports/tr15/>.

[VAUBAN-DEMO] "Vauban Pay reference demo (Starknet Sepolia)", n.d., <https://demo.pay.vauban.tech>.

[VAUBAN-STARK-GIST] "Vauban STARK receipt conformance vectors (public)", n.d., <https://github.com/vauban-org/x402-stark-receipts-conformance>.

[X402-2322] "x402 evidenceType taxonomy and composite trust-query alignment (discussion thread)", n.d., <https://github.com/x402-foundation/x402/issues/2322>.

[X402-2326] "x402 shared canonicalisation discipline (coalition discussion thread)", 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>.

[X402-2398] "action-ref-verify conformance vectors", n.d., <https://github.com/x402-foundation/x402/pull/2398>.

[X402-2411] "Hybrid-PQC receipt-core fixture set (Axis 2)", n.d., <https://github.com/x402-foundation/x402/pull/2411>.

[X402-2412] "Canonicalisation substrate v0 fixtures (Axis 0, 53 vectors)", n.d., <https://github.com/x402-foundation/x402/pull/2412>.

[X402-2413]
"stark-vauban-pay-v1 v0 fixtures (Axis 1)", n.d., <https://github.com/x402-foundation/x402/pull/2413>.

[X402-2434]
"risk-check-attestation-sample v0 (compliance-receipt fixture, ALLOW/DENY/REFER)", n.d., <https://github.com/x402-foundation/x402/pull/2434>.

[X402-2440]
"Composite trust-query normative layer (Axis 4)", n.d., <https://github.com/x402-foundation/x402/pull/2440>.

[X402-CANON]
"x402 Shared Canonicalisation Discipline (jcs-rfc8785-v1)", x402 Protocol Specs specs/canonicalisation.md, n.d., <urn:x402:canonicalisation:jcs-rfc8785-v1>.

[X402-COMPOSITE]
"x402 Composite Trust-Query (Axis 4 assembly-and-binding layer)", x402 Protocol Specs specs/composite-trust-query.md, n.d., <urn:x402:composite-trust-query:v1>.

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

Appendix A. Variant-Specific Receipt Bodies

A.1. stark-vauban-pay-v1

The receipt body is CBOR-encoded ([RFC8949] canonical CBOR Section 4.2.1). The STARK proof attests to a circuit that verifies:

* Payment amount matches the signed PaymentRequirements.amount.

* Currency asset address matches PaymentRequirements.asset.

* Payer attestation is valid (subject to merchant HumanityGate configuration).

* Nullifier is unique (no replay for this nullifier in the settlement ledger).

The canonical CBOR bytes layout for the stark-vauban-pay-v1 variant is documented in the public conformance repository ([VAUBAN-STARK-GIST]). Wire format: [RFC8949] canonical CBOR Section 4.2.1, base64url no-pad encoding per [RFC4648] Section 5. The Vauban Pay reference implementation (Apache 2.0) is planned for public release at Phase 1+ per the project roadmap.

When a stark-vauban-pay-v1 receipt is anchored on Starknet, the felt252 mask described in Section 12 MUST be applied before writing or comparing the action_ref value in keys[1] of the on-chain event.

Implementations of the stark-vauban-pay-v1 variant MUST be conformance test vector-compatible with [VAUBAN-STARK-GIST] as published. Verifiers MUST use the published circuit parameters. Out-of-band circuit versioning is outside the scope of this extension; the stark-vauban-pay-v1 token is a stable identifier for the v1 circuit.

A.2. hybrid-pqc

The receipt body is a JSON object ([RFC8259]) with the following top-level fields, JCS-canonical ([RFC8785]):

* receipt_core: JSON object; the canonical payload signed by both algorithms.

* signature: ES256K signature over SHA-256(UTF-8(JCS(receipt_core))), base64url-encoded.

* pqc_signature: ML-DSA-65 ([FIPS204]) signature over the identical canonical bytes, base64url-encoded.

* kid_es256k: Key identifier for the ES256K key in the facilitator’s JWKS endpoint.

* kid_mldsa65: Key identifier for the ML-DSA-65 key in the facilitator’s JWKS endpoint.

Both signatures cover the identical canonical byte string. Verifiers MUST confirm both signatures independently. A verifier that can only verify ES256K SHOULD still check pqc_signature length and structure for plausibility.

The FeedOracle reference implementation is at [FEEDORACLE-GIST] (standalone Python verifier, no facilitator callback).

A.3. classical-es256k

The receipt body is a JWS Compact Serialization string ([RFC7515]). The JWS payload contains the canonical payment_hash and optionally action_ref. This is the default fallback for clients that do not require ZK or post-quantum properties.

Appendix B. Reference Implementation

The Vauban Pay reference implementation is accessible at [VAUBAN-DEMO] (demo.pay.vauban.tech). It demonstrates the full stark-vauban-pay-v1 flow on Starknet Sepolia.

B.1. On-Chain Contract

The PaymentDemoEmitter contract is deployed on Starknet Sepolia at:

0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff

This contract emits a Starknet event for each completed payment, with keys[0] holding the payment_hash and keys[1] holding the felt252-masked action_ref digest (per Section 12).

Transactions can be inspected via the Voyager block explorer at https://sepolia.voyager.online.

B.2. Zero-Trust Verifier

The reference verifier operates without authentication. Any party can verify any receipt using:

1. The Web Crypto API (browser-native, no server call required).
2. A direct Starknet RPC query to https://sepolia.rpc.vauban.tech/rpc/v0_10 (Pathfinder node, spec v0.10.2).
3. The felt252 mask applied per Section 12 before comparing the locally-computed action_ref against keys[1].

The verifier does not contact the Vauban Pay facilitator server. The proof of settlement is the on-chain event; the facilitator is not in the verification critical path.

B.3. Five-Implementation Cross-Validation Evidence

The ALLOW/DENY/REFER conformance vectors published in [X402-2434] have been independently validated by:

Library	Language	Author lineage	Result
serde_jcs 0.2.0	Rust	l1h3r; Vauban Pay runner	3/3 PASS
jcs (npm)	JavaScript	Erdtman + Rundgren	3/3 PASS
python-canonicaljson	Python	Trail of Bits	3/3 PASS
Go standard library	Go	GoWebPKI	3/3 PASS
Jackson	Java	Rundgren (RFC 8785 reference)	3/3 PASS

Table 9

Aggregate result: 15/15 byte-for-byte agreements across all three conformance vectors (ALLOW, DENY, REFER). The cross-validation against publicly available JCS conformance suites was confirmed independently against the Vauban Rust runner output.

Appendix C. Conformance Checklist

An implementation claiming conformance to this extension MUST:

* Produce and parse all three receipt format tokens.

* Set X-Payment-Options on 402 responses listing supported tokens.

* Set X-Receipt-Format on all PAYMENT-RESPONSE messages.

* Fall back to classical-es256k when the client sends no receipt_format preference.

* Return HTTP 402 with UnsupportedReceiptFormat when a client requests a variant with required: true that the facilitator cannot produce.

* Reject float values for timestamp_ms before JCS canonicalisation.

* Reject preimage objects with missing canonical fields before JCS canonicalisation.

* Reject preimage objects with duplicate keys at parse time.

* Normalise all preimage string values to NFC per [UAX15] before JCS canonicalisation, and reject non-NFC input (Section 9.4).

* Pass every vector in fixtures/action-ref-verify/v0/ from [X402-2398].

* Pass the x402 canonicalisation conformance set [X402-CANON] for the action_ref preimage profile (object key order, array order, optional fields, scalar form, Unicode NFC versus NFD, timestamp lexical form, field-name canonicalisation).

* Apply the felt252 mask (Section 12) consistently at both emission and verification time when anchoring receipts on Starknet.

Proposed companion vectors (pending x402 TSC review; to ship in a follow-up fixtures pull request):

* 0010-duplicate-keys (Section 9.3)

* 0011-unicode-normalisation (Section 9.4)

* 0012-missing-canonical-field (Section 9.3)

On landing of this extension, the following IANA registry initial values MUST be submitted per Section 16:

* Register stark-vauban-pay-v1 (Status: Provisional; Contact: Vauban Research).

* Register hybrid-pqc (Status: Provisional; Contact: FeedOracle).

* Register classical-es256k (Status: Provisional; Contact: x402 Foundation).

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 (Voyager-verified at https://sepolia.voyager.online/contract/0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff).

Reference Implementation Matrix

The conformance vector suite maintains an 8-implementation reference matrix across 8 languages :

Language	Library or Runner	Validation status
Python	rfc8785@0.1.4 (Trail of Bits)	validated 11/11 byte-for-byte
JavaScript	canonicalize@3.0.0 (Erdtman + Rundgren)	validated 11/11 byte-for-byte
Go	gowebpki/jcs v1.0.1	validated 11/11 byte-for-byte
Java	cyberphone/json-canonicalization (RFC 8785 reference)	validated 11/11 byte-for-byte
Rust	serde_jcs 0.2.0 (l1h3r) via vauban-x402-jcs-conformance	validated 11/11 byte-for-byte
PHP	runners/runner.php (pure stdlib, PHP 8.0+)	reference, CI-pending
Ruby	runners/runner.rb (pure stdlib, Ruby 3.0+)	reference, CI-pending
C#	runners/runner.cs (pure stdlib, .NET 8)	reference, CI-pending

Table 10

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.

Validation methodology is documented in _attestations/2026-05-25-vauban-cross-impl.md (5-implementation actually-validated) and _attestations/2026-05-25-vauban-8-impl-extended.md (extended 8-implementation matrix with honest scoping) in the conformance vectors repository.

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 canon_version value emitted, the cross-axis interop binding pattern used if applicable, the JCS implementation library and version, 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.

Acknowledgments

The author thanks the x402 coalition contributors who participated in the three-implementation consensus [X402-2357] (2026-05-19), in particular the FeedOracle hybrid-PQC reference implementation ([X402-2411]) and the action-ref-verify v0.3.0 implementation ([X402-2398] review track). The JCS canonicalisation discipline referenced in Section 9 is publicly documented in the urn:x402:canonicalisation:jcs-rfc8785-v1 substrate and validated by independent runner implementations against published conformance suites.

The author also thanks Nobulex (arian-gogani; bilateral-receipt evidence row in the composite trust-query Axis 4 fixture, aligned with [X402-COMPOSITE]).

A reference Rust 5th-implementation runner for the JCS preimage discipline is published as the vauban-x402-jcs-conformance crate (Apache-2.0, Vauban Research), cross-validating 32 vectors and 28 pair invariants byte-identical against publicly available JCS conformance suites.

The shared x402 canonicalisation discipline [X402-CANON], developed in the [X402-2326] discussion thread, defines the rules this extension profiles, including cross-observer-across-time determinism for receipts emitted under statutory retention obligations. The x402

composite trust-query [X402-COMPOSITE] defines the Axis 4 assembly-and-binding layer to which the action_ref digest of this extension serves as anchor.

Author’s Address

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