Independent Submission
Internet-Draft
Intended status: Informational
Expires: 25 November 2026

V. Research
Vauban Research

24 May 2026

x402 V2 Starknet On-Chain Anchor
draft-vauban-x402-starknet-anchor-00

Abstract

The x402 receipt-format extension defined in [STARK-RECEIPTS] and the lifecycle finite state machine defined in [LIFECYCLE-FSM] specify cryptographic evidence for x402 V2 payment events as off-chain verifiable artefacts. Neither document specifies how a settlement receipt is promoted to on-chain finality, nor how an independent auditor verifies that promotion without contacting the originating facilitator.

This document defines the Starknet on-chain anchor format for x402 V2 payment settlements. The anchor consists of a Starknet event emitted by a conforming Cairo contract, identified by a canonical tuple of chain identifier, transaction hash, event index, and optional block number. A reference implementation, PaymentDemoEmitter, is deployed on Starknet Sepolia at 0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff and is inspectable via the Voyager block explorer. The document also specifies an RPC endpoint convention, grounded in the Starknet JSON-RPC specification, that any compliant node implementation satisfies; the Vauban Pay reference endpoints are cited as one such implementation. This anchor specification is one instantiation of a chain-agnostic anchoring pattern; the protocol contract is designed to accommodate future per-chain adapter specifications that satisfy the same structural requirements.

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 facilitator-issued receipt of settlement, including in the stark-vauban-pay-v1 variant a Stwo Circle STARK proof of payment conditions as defined in [STARK-RECEIPTS]. These artefacts are verifiable off-chain against the facilitator's public parameters. However, they do not provide on-chain finality: an auditor checking a retained receipt at a future date cannot verify, without the facilitator's cooperation, that the settlement was submitted to a public ledger and achieved canonical confirmation at a specific block height. Off-VM evidence alone does not close this gap for regulatory frameworks such as MiCA Art. 76 (settlement record-keeping), which contemplate settlement records that can be validated against an external source of truth.

The Starknet blockchain provides such a source of truth. Starknet's STARK-based proof system ([STARKNET-DOCS]) produces validity proofs over all state transitions, including event emission. An event emitted by a Cairo contract at block N is covered by the STARK proof for that block's state update. An auditor with access to a Starknet full node can verify the event's inclusion in the canonical chain state without contacting the facilitator. This property, unavailable to off-chain receipts alone, constitutes the finality guarantee that closes the off-VM gap.

This document specifies the on-chain anchor format for x402 V2 settlements on Starknet. It defines the canonical anchor tuple, the Cairo event layout a conforming emitter MUST produce, an RPC endpoint convention grounded in the Starknet JSON-RPC specification, the block explorer canonical reference (Voyager), and the compatibility rules with [STARK-RECEIPTS] and [LIFECYCLE-FSM]. The specification is written as one instantiation of a chain-agnostic anchoring pattern. The structural requirements (anchor tuple, event layout, per-chain adapter registration) are expressed in terms general enough to accommodate future anchor specifications for other ledger systems. This document MUST NOT be read as asserting that Starknet is the only valid anchor chain; it is the first reference implementation.

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. 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.

3. Terminology

Anchor: An on-chain record that commits a payment settlement to a public ledger. An anchor is identified by a canonical anchor tuple (Section 4) and is verifiable by any party with read access to the chain.

Cairo contract: A smart contract written in the Cairo language and deployed on Starknet ([STARKNET-DOCS]). For the purposes of this document, a Cairo contract is a conforming emitter if it satisfies the settlement event layout defined in Section 4.2.

Chain identifier: A string token unambiguously identifying the target blockchain and network. For Starknet mainnet the value is "SN_MAIN"; for Starknet Sepolia testnet the value is "SN_SEPOLIA". Additional values MAY be registered by future per-chain adapter specifications.

felt252: A field element in the Stark field, a prime field of order approximately 2^251.6. Starknet addresses, transaction hashes, and event keys are represented as felt252 values. Felt252 values are encoded as 0x-prefixed lowercase hexadecimal strings of at most 64 hex characters when serialised to JSON.

Transaction hash: A felt252 identifier assigned by the Starknet sequencer to a submitted transaction. Used as the primary lookup key for event retrieval via the Starknet JSON-RPC specification.

Event index: A zero-based integer identifying a specific event within a transaction's event list. When a single transaction emits multiple events, the event index disambiguates the settlement event from other events in the same transaction.

Block number: The height of the block in which the transaction was included. A block number of zero is not valid as a finality reference; only blocks beyond the sequencer's reorg-safety window carry canonical finality.

RPC endpoint: A network endpoint implementing the Starknet JSON-RPC specification ([STARKNET-DOCS]). Any conformant full node implementation, such as Pathfinder ([PATHFINDER]), satisfies the endpoint requirements of Section 5.

Conforming emitter: A Cairo contract that satisfies the settlement event layout defined in Section 4.2 and emits that event for each settled payment.

4. Starknet Anchor Format

4.1. Canonical Anchor Tuple

A Starknet on-chain anchor is represented by the following JSON object:

```
{
  "chain_id":     "<string>",
  "tx_hash":      "<felt252-hex-string>",
  "event_index":  <uint>,
  "block_number": <uint>,
  "kind":         "<string>"
}
```

The fields are defined as follows:

Field	Type	Required	Description
chain_id	string	REQUIRED	Chain and network identifier. MUST be one of "SN_MAIN" or "SN_SEPOLIA" for Starknet, or a registered value from a future per-chain adapter specification. Case-sensitive.
tx_hash	string	REQUIRED	Felt252 transaction hash, encoded as a 0x-prefixed lowercase hexadecimal string. MUST be 66 characters in length (2 hex chars per byte, 32 bytes, plus "0x" prefix).
event_index	uint	REQUIRED	Zero-based index of the settlement event within the transaction’s event list. If the transaction emits only one event, event_index MUST be 0.
block_number	uint	OPTIONAL	Block height at which the transaction was included. When present, the verifier SHOULD use this value to confirm that the transaction was included in the expected block before extracting the event. A value of 0 MUST be treated as absent.
kind	string	REQUIRED	Semantic type of the anchored payment claim. MUST be one of "settlement", "refund", or "delegation", corresponding to the FSM states defined in [LIFECYCLE-FSM].

Table 1

An anchor object MUST be transmitted as a JSON value ([RFC8785] compliant) when embedded in an x402 extension field or in an audit manifest. CBOR encoding of the anchor tuple is out of scope for this document.

The anchor object SHOULD be included in the PAYMENT-RESPONSE as an extension field when the facilitator has confirmed on-chain submission:

```
{
  "extensions": {
    "starknet-anchor": {
      "info": {
        "chain_id":     "SN_SEPOLIA",
        "tx_hash":      "0x044dd87a94a8...19ff",
        "event_index":  0,
        "block_number": 712483,
        "kind":         "settlement"
      }
    }
  }
}
```

When the block_number is not yet known at the time the PAYMENT-RESPONSE is emitted (for example, because the transaction was submitted but not yet included in a block), the facilitator SHOULD omit block_number rather than emit 0 or a speculative value. A verifier receiving an anchor without block_number MUST perform an independent block lookup via the RPC endpoint convention (Section 5) before asserting block-confirmed finality.

4.2. Settlement Event Layout

A conforming emitter MUST emit a Starknet event for each settled payment. The event MUST conform to the following layout, expressed in terms of the Starknet event data model ([STARKNET-DOCS]):

Keys array (indexed fields, usable in event filters):

Index	Content	Type	Description
keys[0]	Event type selector	felt252	A selector uniquely identifying the event type within the contract’s ABI. MUST be the Starknet selector for the event name "PaymentSettled" or an equivalent name documented in the emitter’s ABI.
keys[1]	action_ref digest	felt252	The felt252-masked SHA-256 digest of the JCS canonical preimage of the payment action reference, as defined in [STARK-RECEIPTS] Section on felt252 encoding. The masking operation is: hash_felt252 = sha256_bigint & ((1 << 251) - 1).

Table 2

Data array (non-indexed payload):

Index	Content	Type	Description
data[0]	Anchor kind discriminant	felt252	A felt252 encoding of the kind string. The canonical encoding is: 0x1 for "settlement", 0x2 for "refund", 0x3 for "delegation".
data[1]	payment_hash low bits	felt252	Low 251 bits of the SHA-256 payment hash, masked identically to keys[1].
data[2]	Reserved	felt252	MUST be 0x0 in this version. Future versions of this specification MAY assign a meaning to this field. Verifiers MUST NOT reject an anchor whose data[2] is non-zero; they MUST treat it as an opaque reserved field.

Table 3

The felt252 masking operation applied to both keys[1] and data[1] is the same deterministic truncation specified in [STARK-RECEIPTS]: the SHA-256 output (interpreted as a 256-bit unsigned integer, big-endian) is AND-ed with (1 << 251) - 1. The result is the canonical on-chain representation. Verifiers MUST apply this mask consistently when computing the expected on-chain value from a locally-held SHA-256 digest.

4.3. Reference Implementation: PaymentDemoEmitter

The PaymentDemoEmitter contract, deployed on Starknet Sepolia, is the reference conforming emitter for this specification. Its deployment address is:

0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff

The contract emits one PaymentSettled event per completed settlement call. The event layout matches Section 4.2 exactly: keys[0] holds the selector for PaymentSettled, keys[1] holds the felt252-masked action_ref digest, data[0] holds the kind discriminant 0x1 (settlement), and data[1] holds the masked payment_hash.

The contract source and ABI are maintained in the Vauban Pay public conformance repository ([VAUBAN-DEMO]). Any implementation claiming conformance to this specification SHOULD test against the PaymentDemoEmitter ABI as the canonical reference. The contract MUST NOT be used as a production payment processor; it is a conformance reference and demonstration surface.

Transaction history for PaymentDemoEmitter is inspectable via the Voyager block explorer at:

https://sepolia.voyager.online/contract/0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff

The Voyager block explorer is the canonical visual audit surface for Starknet events on both Sepolia and mainnet. See Section 6 for the normative block-explorer reference.

5. RPC Endpoint Convention

5.1. Protocol Requirement

Any Starknet RPC endpoint used for anchor verification MUST implement the Starknet JSON-RPC specification as published at [STARKNET-DOCS]. The specific methods required by this specification are:

* starknet_getTransactionByHash: retrieves the transaction associated with a given transaction hash, including the transaction’s block reference.

* starknet_getTransactionReceipt: retrieves the receipt for a given transaction, including the ordered list of emitted events and the block number in which the transaction was included.

* starknet_getBlockWithTxHashes: retrieves a block header by block number, enabling independent verification that the transaction’s block number matches the anchor tuple.

Endpoints MAY support additional Starknet JSON-RPC methods; this specification requires only the three listed above for anchor verification purposes.

A verifier MUST submit the starknet_getTransactionReceipt request to the configured RPC endpoint, extract the event at position event_index from the response’s events array, and compare keys[1] against the locally computed felt252-masked action_ref value. A match confirms that the payment action reference is anchored on-chain in the specified transaction.

5.2. Endpoint Authentication

The Starknet JSON-RPC specification does not mandate a specific authentication mechanism. Endpoint implementations MAY apply authentication controls (for example, API keys, OAuth 2.1, or mutual TLS). The authentication mechanism is an implementation detail of the endpoint operator and is not visible to this protocol specification.

Verifiers SHOULD use endpoints operated by parties they trust or operate themselves. Verifiers MUST NOT transmit proof material or payment credentials as part of RPC requests; the RPC interface is read-only for the purposes of anchor verification.

5.3. Vauban Pay Reference Endpoints

The Vauban Pay reference implementation operates two Starknet full node RPC endpoints (Pathfinder, [PATHFINDER]) that implement the Starknet JSON-RPC specification:

Starknet Sepolia (testnet):

https://sepolia.rpc.vauban.tech/rpc/v0_10

This endpoint implements Starknet JSON-RPC spec version 0.10.2 and is used by the PaymentDemoEmitter conformance surface (Section 4.3).

Starknet mainnet:

https://rpc.vauban.tech/rpc/v0_10

This endpoint implements Starknet JSON-RPC spec version 0.10.2 and is the production mainnet reference.

The /rpc/v0_10 path suffix selects the spec v0.10.2 response schema. Both endpoints also expose /rpc/v0_8 and /rpc/v0_9 paths for older client compatibility. New implementations SHOULD use /rpc/v0_10.

These endpoints are cited as one conformant implementation of the Starknet JSON-RPC specification. Implementors who operate their own Starknet full nodes, or who use endpoints operated by other trusted parties, satisfy the RPC endpoint convention of Section 5.1 without requiring access to the Vauban Pay endpoints. This specification does not mandate use of any specific endpoint operator.

6. Block Explorer Canonical Reference

6.1. Voyager as Audit Surface

The Voyager block explorer ([VOYAGER]) is the canonical human-readable audit surface for Starknet mainnet and Starknet Sepolia events. Implementations SHOULD include a Voyager transaction URL in their audit submissions to provide auditors with a point-and-click verification path that does not require operating a full node.

The canonical URL patterns are:

Starknet mainnet:

https://voyager.online/tx/<tx_hash>

Starknet Sepolia:

https://sepolia.voyager.online/tx/<tx_hash>

Where <tx_hash> is the 0x-prefixed lowercase hexadecimal felt252 transaction hash from the anchor tuple (Section 4.1).

For contract-level browsing (all historical events for a conforming emitter):

Starknet Sepolia reference implementation:

https://sepolia.voyager.online/contract/0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff

6.2. Inclusion in Audit Submissions

Facilitators submitting payment settlement evidence under regulatory audit obligations (for example, MiCA Art. 76 or EU AI Act Art. 12) SHOULD include a Voyager URL for each on-chain anchor in their audit package. The Voyager URL is a supplementary human-readable reference; the authoritative anchor evidence is the anchor tuple (Section 4.1) and the RPC-verifiable event (Section 5).

An audit package that includes a Voyager URL but whose anchor tuple produces a verification failure via the Starknet JSON-RPC MUST be treated as invalid regardless of the Voyager URL’s visual output. The Voyager URL is a convenience; the RPC verification is the normative check.

Implementations MUST NOT cite any block explorer other than Voyager as the canonical visual audit surface for Starknet. Other explorers MAY be used for operational debugging but MUST NOT appear as the canonical reference in audit submissions or specification texts.

7. Compatibility with FSM and Receipt Format

7.1. Relationship to STARK-RECEIPTS

The anchor format defined in this document is complementary to, and not a replacement for, the stark-vauban-pay-v1 receipt format defined in [STARK-RECEIPTS]. The receipt format specifies the off-VM cryptographic evidence (Stwo Circle STARK proof, felt252 encoding of action_ref). The anchor format specified in this document specifies the on-chain event record that confirms the settlement was submitted to and accepted by the Starknet state machine.

The two artefacts share the action_ref binding. The receipt's keys[1] on-chain value is the same felt252-masked SHA-256 digest of the JCS canonical preimage as the receipt's action_ref field (after applying the mask defined in [STARK-RECEIPTS]). A verifier holding both a stark-vauban-pay-v1 receipt and a Starknet anchor MAY confirm that they refer to the same payment by computing:

expected_keys_1 = SHA-256(action_ref_preimage) & ((1 << 251) - 1)

and verifying that expected_keys_1 equals the keys[1] field of the on-chain event at the position identified by the anchor tuple.

The receipt alone provides post-quantum sound, offline-verifiable proof of payment conditions. The anchor alone provides on-chain finality confirmation. Together, they constitute a complete regulator-grade settlement record: the STARK receipt proves that the payment conditions were satisfied; the anchor proves that the settlement was committed to a public ledger at a specific block.

7.2. Relationship to LIFECYCLE-FSM

The kind field of the anchor tuple (Section 4.1) maps directly to the lifecycle states defined in [LIFECYCLE-FSM]:

kind value	FSM state	Description
"settlement"	SettlementReceipt	The anchor records the on-chain confirmation of a PaymentIntent transitioning to SettlementReceipt.
"refund"	RefundClaim	The anchor records the on-chain confirmation of a SettlementReceipt transitioning to RefundClaim.
"delegation"	DelegationGrant	The anchor records the on-chain commitment of a DelegationGrant scope.

Table 4

A verifier performing FSM chain reconstruction ([LIFECYCLE-FSM] Section on Chain Reconstruction Algorithm) MAY use on-chain anchors as an additional source of truth for original_payment_ref linkage. When an anchor is present, the verifier SHOULD confirm that the on-chain event's data[1] (payment_hash masked) is consistent with the payment_hash in the retained off-VM SettlementReceipt before accepting the FSM chain as valid.

7.3. Per-Chain Adapter Pattern

This document specifies the Starknet instantiation of a chain-agnostic anchoring pattern. The structural requirements common to all per-chain instantiations are:

1. A canonical anchor tuple that uniquely identifies an on-chain record. The tuple MUST include at minimum: a chain identifier, a transaction reference, and a position or index within that transaction.

2. A settlement event layout that encodes action_ref (or an equivalent work-receipt binding digest) and a kind discriminant in the on-chain record.

3. An RPC endpoint convention grounded in the target chain's published JSON-RPC or equivalent read interface.

4. A canonical block explorer reference for human-readable audit submissions.

Future per-chain adapter specifications (for example, for EVM-compatible chains or other UTXO or account-based ledgers) MUST address all four structural requirements and SHOULD cross-reference this document as the pattern origin. They MUST NOT reuse the "SN_MAIN" or "SN_SEPOLIA" chain identifier values; per-chain adapters MUST register distinct chain identifier values.

8. Security Considerations

8.1. RPC Endpoint Authenticity

An anchor verification that relies on a network RPC endpoint is subject to man-in-the-middle attack if the transport is not authenticated. An adversary that controls the network path between the verifier and the RPC endpoint may return fabricated event data, causing the verifier to falsely confirm an anchor that was never committed to the canonical chain.

Mitigation: verifiers MUST use TLS-authenticated RPC endpoints. Verifiers SHOULD prefer endpoints they operate themselves or whose TLS certificate chain they have independently pinned. Verifiers MUST NOT accept RPC responses over unencrypted HTTP for anchor verification purposes.

8.2. Block Explorer Spoofing

A Voyager URL included in an audit submission is a supplementary human-readable convenience. An adversary controlling a domain similar to voyager.online or sepolia.voyager.online may display fabricated transaction data. Auditors relying solely on a browser visit to a Voyager URL, without independently verifying the anchor tuple via the Starknet JSON-RPC, cannot distinguish canonical chain data from spoofed presentation.

Mitigation: the normative audit check is the anchor tuple verification via the Starknet JSON-RPC, not the Voyager URL. Audit frameworks MUST perform RPC verification as the authoritative step. Voyager URLs are RECOMMENDED as a supplementary aid for human reviewers; they MUST NOT be the sole verification artefact.

8.3. Event Collision and Selector Ambiguity

If two Cairo contracts deployed at different addresses emit events with the same keys[0] selector and the same keys[1] digest, a verifier that does not check the emitting contract address may accept the wrong anchor. This is especially relevant when a chain hosts multiple conforming emitters.

Mitigation: verifiers MUST validate that the event was emitted by the expected contract address. The anchor tuple (Section 4.1) identifies the transaction but not the emitting contract; verifiers MUST cross-reference the events[i].from_address field in the starknet_getTransactionReceipt response against the known emitter address before accepting the event as a valid anchor. Conforming emitters MUST be registered in an implementation-controlled list; events from unregistered emitters MUST be rejected.

8.4. Chain Reorganisation and Finality Window

Starknet blocks achieve L2 finality when their state update is proven and posted to the Ethereum L1 settlement layer. Prior to L1 finalisation, a block may in principle be reorganised (though Starknet's sequencer design makes deep reorgs highly improbable). A verifier that confirms a Starknet anchor at a block that has not yet reached L1 finality accepts a non-final commitment.

Mitigation: for settlement records that require finality guarantees under regulatory frameworks, verifiers SHOULD wait for the Starknet block's state update to be confirmed on Ethereum L1 before treating the anchor as final. The Starknet JSON-RPC exposes block status fields that allow verifiers to distinguish pending, accepted-on-L2, and accepted-on-L1 states ([STARKNET-DOCS]). Verifiers MUST NOT treat a pending or L2-only block as providing the same finality guarantee as an L1-confirmed block. For non-regulatory use cases (for example, low-value instant payments), L2 acceptance MAY be sufficient; implementations MUST document their chosen finality threshold.

8.5. Chain Identifier Confusion

An anchor tuple carrying chain_id: "SN_SEPOLIA" references a testnet where tokens have no economic value. A verifier that does not validate the chain_id field against its expected deployment environment may accept a testnet anchor as proof of mainnet settlement.

Mitigation: verifiers MUST reject anchors whose chain_id does not match the deployment environment they are configured to verify. Production payment processors MUST be configured to accept only chain_id: "SN_MAIN". Test environments that accept chain_id: "SN_SEPOLIA" MUST be segregated from production verification pipelines. The facilitator MUST include the chain_id in the anchor tuple; a missing chain_id MUST be treated as a malformed anchor and MUST be rejected.

9. IANA Considerations

This document defines no new IANA registries. The chain identifier values "SN_MAIN" and "SN_SEPOLIA" defined in Section 4.1 are not IANA-registered tokens; they are Starknet-native identifiers as documented in [STARKNET-DOCS].

The kind discriminant values ("settlement", "refund", "delegation") defined in Section 4.1 derive from the lifecycle state names of [LIFECYCLE-FSM]; they do not require separate IANA registration.

Future per-chain adapter specifications that introduce new chain_id values SHOULD define a registration mechanism within the x402 Foundation's registry structure (as established by [STARK-RECEIPTS]) rather than reserving those values through IANA.

Acknowledgments

The authors thank the x402 Linux Foundation Working Group ([X402-V2]) for establishing the protocol framework that this anchor specification extends. The canonicalisation discipline grounding the felt252 masking and action_ref derivation in this document is defined in [STARK-RECEIPTS] and coordinated through the x402 shared canonicalisation substrate. The lifecycle state names and FSM chain reconstruction algorithm referenced in Section 7 are defined in [LIFECYCLE-FSM].

The Starknet protocol and JSON-RPC specification ([STARKNET-DOCS]) and the Pathfinder full node implementation ([PATHFINDER]) provide the on-chain infrastructure that this anchor format targets. The Voyager block explorer ([VOYAGER]) provides the canonical human-readable audit surface referenced in Section 6.

References

Normative References

[LIFECYCLE-FSM] "x402 V2 Payment Lifecycle Finite State Machine", Work in Progress, Internet-Draft, draft-vauban-x402-lifecycle-fsm-00, n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-lifecycle-fsm/>.

[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>.

[STARK-RECEIPTS] "x402 STARK Receipt Format Extension", Work in Progress, Internet-Draft, draft-vauban-x402-stark-receipts-01, n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-stark-receipts/>.

Informative References

[PATHFINDER] "Pathfinder Starknet full node (eqlabs)", n.d., <https://github.com/eqlabs/pathfinder>.

[STARKNET-DOCS] "Starknet Documentation", n.d., <https://docs.starknet.io>.

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

[VOYAGER] "Voyager Starknet Block Explorer", n.d., <https://voyager.online>.

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

Appendix A. Anchor Verification Algorithm

The following algorithm summarises the normative steps a verifier MUST execute to confirm that a Starknet on-chain anchor is valid and consistent with a retained x402 settlement receipt.

A.1. Inputs

* anchor: the anchor tuple object (Section 4.1) received from the facilitator.

* action_ref_preimage: the JCS canonical preimage object (as defined in [STARK-RECEIPTS]) whose SHA-256 digest was bound to the payment.

* expected_payment_hash: the SHA-256 payment hash recorded in the settlement receipt.

* emitter_address: the known-good address of the conforming emitter contract.

A.2. Steps

1. *Validate chain_id*: confirm anchor.chain_id matches the verifier's configured deployment environment ("SN_MAIN" or "SN_SEPOLIA"). Reject if mismatch.

2. *Retrieve transaction receipt*: submit starknet_getTransactionReceipt with anchor.tx_hash to the configured RPC endpoint. If the transaction is not found or returns an error, reject the anchor.

3. *Validate emitter address*: confirm that receipt.events[anchor.event_index].from_address equals emitter_address. Reject if mismatch (Section 8.3).

4. *Validate block number* (if anchor.block_number is present): confirm receipt.block_number equals anchor.block_number. Reject if mismatch.

5. *Compute expected keys[1]*: apply the felt252 mask to the SHA-256 digest of the action_ref preimage:

   ```
   sha256_out = SHA-256(UTF-8(JCS(action_ref_preimage)))
   expected_keys_1 = sha256_out_as_bigint & ((1 << 251) - 1)
   ```

6. *Compare keys[1]*: confirm that receipt.events[anchor.event_index].keys[1] (interpreted as a 0x-prefixed hex felt252) equals expected_keys_1. Reject if mismatch.

7. *Compare data[1]* (optional cross-check): apply the same felt252 mask to expected_payment_hash and confirm it equals receipt.events[anchor.event_index].data[1].

8. *Validate kind*: confirm anchor.kind matches the data[0] discriminant in the event (0x1 for "settlement", 0x2 for "refund", 0x3 for "delegation").

9. *Check finality status*: retrieve the block status for anchor.block_number. Apply the finality threshold configured for the verifier's use case (Section 8.4).

If all steps pass, the anchor is valid. The verifier MAY log the Voyager URL as a supplementary audit trail (Section 6.2).

Appendix B. Conformance Checklist

A facilitator claiming conformance to this specification MUST:

* Produce a well-formed anchor tuple (Section 4.1) for each submitted settlement.
* Include chain_id, tx_hash, event_index, and kind in every anchor tuple.
* Submit settlement events from a conforming emitter whose event layout satisfies Section 4.2.
* Apply the felt252 mask consistently at both event emission time and when computing the expected keys[1] value.
* Include a Voyager URL in audit submissions when operating on Starknet (Section 6).
* Not produce anchors with chain_id: "SN_SEPOLIA" in production payment flows.

A verifier claiming conformance to this specification MUST:

* Reject anchors whose chain_id does not match the configured deployment environment.
* Validate the emitter contract address against the known-good emitter address before accepting an event as a valid anchor.
* Apply the anchor verification algorithm (Appendix A) before treating an anchor as confirmed.
* Distinguish pending, L2-accepted, and L1-accepted block states and apply the appropriate finality threshold for the use case.
* Reject anchors whose tx_hash is not found or whose event at event_index does not exist.
* Not treat a Voyager URL alone as sufficient proof of anchor validity.

Author’s Address

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