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

V. Research
Vauban Research

25 May 2026

x402 Delegation Binding for Agentic HTTP Pipelines
draft-vauban-x402-delegation-binding-01

Abstract

Agent-to-agent HTTP payment pipelines built on x402 V2 require a mechanism to bind a DelegationGrant receipt to the identity of the delegatee agent that will consume it. Existing x402 DelegationGrant fields (as defined in [LIFECYCLE-FSM]) constrain spend scope and expiry but do not cryptographically bind the grant to an agent identity token recognised by A2A or MCP agent protocol layers. Without this binding, a grant issued to one agent can be replayed by another agent with different authority bounds, enabling undetected privilege escalation in automated payment pipelines.

This document defines a delegation binding extension for x402 V2 DelegationGrant receipts. The extension introduces a delegate_pseudonym field derived from the agent's identity via the JCS canonical preimage discipline ([RFC8785]), a structured set of spend-cap and scope fields, and a delegation_nonce that enables nullifier consumption on revocation. The binding is cross-validated against 18 bounded-spend vectors in five language runtimes ([X402-2432]). Integration notes are provided for A2A Composable Trust Evidence Format ([A2A-CTEF]) and MCP agent protocol surfaces ([VAUBAN-DEMO]). This document is the fourth companion in the Vauban Pay normative quartet: format ([STARK-RECEIPTS]), lifecycle FSM ([LIFECYCLE-FSM]), and delegation binding (this document); the algebra companion draft-vauban-x402-vpsf-algebra is in preparation.

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

1.1. The Agent Credential Gap

The x402 protocol ([X402-V2]) DelegationGrant state defined in [LIFECYCLE-FSM] enables a principal to pre-authorise an agent to spawn PaymentIntents within a scoped set of constraints. The FSM defines the side-channel role of DelegationGrant, its scope enforcement requirements, and its linkage to downstream PaymentIntents via the JCS canonical preimage discipline ([RFC8785]).

However, neither the FSM nor the receipt format extension ([STARK-RECEIPTS]) specifies how the DelegationGrant is cryptographically bound to the identity of the delegatee agent. The delegatee field in the FSM is defined as "address or pseudonym of the agent receiving authority"; it is a free-form string. A facilitator verifying a PaymentIntent that claims to operate under a DelegationGrant has no normative mechanism to confirm that the presenting agent is the intended delegatee, beyond comparing an unverified string. This creates the agent credential gap: the delegation chain is semantically correct but identity-unbound.

In agentic HTTP pipelines where multiple autonomous agents share a session, this gap enables grant-capture attacks. Agent A is granted authority over a merchant set with a given spend cap. Agent B, operating in the same pipeline with different authority bounds, intercepts the DelegationGrant receipt and presents it as its own credential. The facilitator, comparing only the free-form delegatee string, cannot distinguish the impersonation.

1.2. The Binding Solution

This document closes the agent credential gap by introducing a delegation binding layer above the FSM DelegationGrant field set. The binding is achieved via three mechanisms operating in concert:

1. A delegate_pseudonym field derived from the agent's identity via the shared JCS canonical preimage discipline ([RFC8785]). The pseudonym is a felt252-sized opaque value; it does not reveal the underlying agent identity to observers of the receipt.

2. A delegation_nonce field committed at grant issuance. The nonce is consumed on first use within its chain depth and marked in a nullifier store. Replay of the grant by a different agent identity produces a nonce mismatch that the facilitator can detect without contacting the issuing principal.

3. A max_chain_length field that bounds privilege escalation through transitive delegation. A grant with max_chain_length: 1 cannot be sub-delegated; any attempt to use it as the basis for a further DelegationGrant is rejected with DelegationDepthExceeded.

Together these mechanisms establish what this document calls Composable Trust Evidence for the delegation surface: the grant is cryptographically scoped to one agent identity, one nullifier epoch, and one depth bound, without requiring the facilitator to maintain a per-agent identity registry or contact the issuing principal at verification time.

1.3. Relationship to Companion Documents

This document is the fourth companion in the Vauban Pay normative quartet:

* [STARK-RECEIPTS] defines the receipt format, including the JCS canonical preimage discipline shared by all variants and the action_ref work-receipt binding.

* [LIFECYCLE-FSM] defines the four-state payment lifecycle FSM, including the DelegationGrant side-channel state, its required fields, and its transition rules.

* This document extends the DelegationGrant state with identity binding and spend-cap fields. It does not restate FSM transition rules; those are normative in [LIFECYCLE-FSM].

* A fourth companion, draft-vauban-x402-vpsf-algebra (in preparation), defines the claim-algebraic composition operators over the four PaymentClaim states.

Implementations that adopt this extension MUST also implement the FSM ([LIFECYCLE-FSM]) and the canonical preimage discipline ([STARK-RECEIPTS] Section 5). This document adds fields and binding semantics; it does not substitute for either companion.

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

Agent Identity: A stable identifier for an autonomous software agent operating in an agentic HTTP pipeline. For A2A pipelines, the Agent Identity is typically a DID (did:web, did:key, or similar scheme). For MCP pipelines, it is a server URI or a capability token issued at registration time. This document does not mandate any specific identity scheme; it requires only that the identity resolves to a deterministic byte string that can be fed to the JCS canonical preimage construction.

Delegation Grant: A side-channel Payment Claim (as defined in [LIFECYCLE-FSM] Section 3.4) that authorises a delegatee agent to spawn PaymentIntents on behalf of a principal delegator, subject to spend-cap and scope constraints. In this document, "DelegationGrant" refers to the extended form defined in Section 4, which adds identity binding fields to the FSM base fields.

Bounded Authority: The property of a DelegationGrant that restricts the delegatee’s ability to issue payments to a defined ceiling (cap per transaction, cap per period), a defined merchant set, a defined currency set, and a defined chain depth. Bounded authority is the normative goal of the delegation binding extension.

Composable Trust Evidence: A receipt or receipt chain from which a

verifier can independently reconstruct and check the authority of every agent in the delegation chain, without contacting the issuing principal or a central registry. Composable Trust Evidence is the property that this extension enables for DelegationGrant receipts in A2A and MCP pipelines.

delegate_pseudonym: A felt252-sized opaque value derived from the Agent Identity via the JCS canonical preimage discipline. The pseudonym binds the DelegationGrant to the delegatee identity without revealing the identity to observers of the grant receipt.

delegation_nonce: A felt252-sized unique anti-replay token committed at grant issuance. The nonce is consumed on first verification and marked in the facilitator's nullifier store. A second verification of the same DelegationGrant with the same nonce is rejected as a replay, even if the presenting agent identity matches.

4. Delegation Binding Format

A DelegationGrant implementing this extension carries the base fields required by [LIFECYCLE-FSM] Section 3.4 plus the binding fields defined in this section. Fields from [LIFECYCLE-FSM] are not restated here; implementations MUST satisfy both sets of field requirements.

4.1. Binding Fields

The following fields MUST be present in a DelegationGrant that implements this extension:

Field	Type	Constraint	Description
delegate_pseudonym	string	felt252 decimal encoding	Opaque identity commitment derived from Agent Identity. See Section 5.1.
cap_per_tx	string	u256 decimal encoding	Maximum amount, in the smallest indivisible unit of the grant's currency, for any single PaymentIntent spawned under this grant. MUST be a non-negative integer decimal string.
cap_per_period	string	u256 decimal encoding	Maximum aggregate amount, in the smallest indivisible unit of the grant's currency, across all

			PaymentIntents spawned under this grant within period_seconds. MUST be a non-negative integer decimal string.
period_seconds	integer	1..31536000 inclusive	Duration of the spend-cap accounting period in seconds. MUST be at least 1 and at most 31536000 (365 days).
allowed_merchants	string[]	URN pattern	Ordered list of merchant identifiers within which the delegatee may spawn PaymentIntents. Each entry MUST match the pattern urn:x402:merchant:[a-z0-9-]{1,63} or be an on-chain address in the receipt format's canonical address encoding. An empty list MUST be interpreted as "no merchant allowed" (fully-closed grant).
allowed_currencies	string[]	URN pattern	Ordered list of currency identifiers the delegatee may use. Each entry MUST match urn:x402:currency:[A-Z]{2,12} or be a standard token ticker per the facilitator's supported asset list. An empty list MUST be interpreted as "no currency allowed".
delegation_nonce	string	felt252 decimal encoding	Anti-replay token committed at issuance. See Section 5.2.
max_chain_length	integer	1..32 inclusive	Maximum number of delegation hops permitted in a chain rooted at this grant. A value of 1 means the grant is terminal (cannot be sub-delegated). MUST NOT exceed 32.

Table 1

All string fields MUST be Unicode Normalization Form C (NFC) before JCS canonicalisation ([RFC8785]). Float values for integer-typed fields MUST be rejected before canonicalisation.

4.2. felt252 Decimal Encoding

The delegate_pseudonym and delegation_nonce fields are felt252 field elements encoded as unsigned decimal strings. The Starknet felt252 field prime is:

P = 2^251 + 17 * 2^192 + 1

Implementations MUST reject values that are not valid decimal representations of integers in the range [0, P-1]. Negative values and values >= P MUST be rejected before canonicalisation. Implementations MUST NOT use hexadecimal or base64 encoding for felt252 fields; the decimal string representation is the sole canonical form for cross-implementation digest consistency.

4.3. u256 Decimal Encoding

The cap_per_tx and cap_per_period fields are u256 values encoded as unsigned decimal strings. Implementations MUST reject negative values and values exceeding 2^256 - 1. Implementations MUST NOT use hexadecimal encoding for u256 fields. The decimal string representation is the sole canonical form.

5. Agent Identity Binding

5.1. Pseudonym Derivation

The delegate_pseudonym is derived from the Agent Identity string using the following deterministic construction:

1. Encode the Agent Identity as a UTF-8 NFC byte string.

2. Compute raw = SHA-256(UTF-8(NFC(agent_identity))).

3. Interpret raw as an unsigned 256-bit big-endian integer.

4. Compute pseudonym_int = raw mod P where P is the felt252 prime defined in Section 4.2.

5. Encode pseudonym_int as an unsigned decimal string. This is the delegate_pseudonym.

The pseudonym is a deterministic function of the Agent Identity. Two presentations of the same DelegationGrant by the same agent produce the same pseudonym. A different agent identity produces a different pseudonym with probability 1 - 2^-251 (negligible collision probability). The derivation does not reveal the Agent Identity to observers of the receipt; the pseudonym is a one-way commitment.

Facilitators verifying a PaymentIntent under a DelegationGrant MUST:

1. Compute the pseudonym of the presenting agent using the construction above.

2. Compare the computed pseudonym against the delegate_pseudonym in the retained DelegationGrant. Comparison MUST be constant-time to prevent timing side-channels.

3. Reject the PaymentIntent with AgentIdentityMismatch if the pseudonyms do not match.

5.2. Nullifier Consumption Discipline

The delegation_nonce is generated at grant issuance by the issuing principal using a cryptographically secure random number generator. The nonce MUST be in the felt252 range defined in Section 4.2.

On first verification of a DelegationGrant, the facilitator MUST:

1. Verify that the delegation_nonce is not present in its nullifier store.

2. Insert the delegation_nonce into the nullifier store keyed by (delegation_nonce, max_chain_length).

3. Accept the grant for the verification scope.

On any subsequent presentation of the same delegation_nonce, the facilitator MUST:

1. Detect the nonce in the nullifier store.

2. Reject the presentation with DelegationNonceReplay.

This discipline prevents an adversary who has captured a DelegationGrant receipt from replaying it against a different agent session after the intended delegatee has consumed the grant.

5.3. Revocation Propagation

When a principal revokes a DelegationGrant, the revocation MUST be propagated to all facilitators that have stored the grant in their active grant registries. Revocation is signalled by inserting the delegation_nonce into the facilitator's nullifier store with a revoked marker before the first legitimate consumption.

A facilitator that receives a PaymentIntent citing a revoked DelegationGrant MUST reject the PaymentIntent with GrantRevoked.

Revocation propagation over an unreliable channel creates a time window during which a legitimately issued PaymentIntent spawned before the revocation signal arrives may be either accepted or rejected depending on propagation latency. Implementations MUST:

* Accept PaymentIntents whose issued_at timestamp (per [LIFECYCLE-FSM] Section 3.1) predates the revocation signal by at most the maxTimeoutSeconds window declared in the payment requirements.

* Reject PaymentIntents whose issued_at is after the revocation signal timestamp, regardless of propagation latency.

This rule bounds the revocation window without requiring synchronous coordination between the issuing principal and all active facilitators.

5.4. Chain Depth Enforcement

A DelegationGrant with max_chain_length: N may spawn a sub-delegation only if N > 1. The sub-delegation MUST carry max_chain_length: N-1. A sub-delegation MUST also carry a new delegation_nonce generated by the sub-delegating agent; it MUST NOT reuse the parent nonce.

A facilitator verifying a sub-delegation chain MUST:

1. Reconstruct the chain from its retained grant manifests.

2. Verify that the sum of hops from the root grant to the current grant does not exceed the root grant's max_chain_length.

3. Reject with DelegationDepthExceeded if the chain length constraint is violated.

The default max_chain_length for new DelegationGrants is 1 (terminal, no sub-delegation). Implementations that do not support sub-delegation MUST reject any DelegationGrant with max_chain_length greater than 1 during grant acceptance, not at PaymentIntent submission time.

6. A2A Integration Notes

A2A pipelines that implement the Composable Trust Evidence Format ([A2A-CTEF]) MAY use the DelegationGrant binding defined in this document as the authority row within a Composable Trust Evidence envelope. The delegation binding serves as the authority evidence column, complementing the payment evidence column provided by the action_ref and payment_hash fields in [STARK-RECEIPTS].

In an A2A Composable Trust Evidence envelope, the binding between the agent identity evidence and the payment settlement evidence is established by the shared payment_hash and action_ref pair, as defined in [STARK-RECEIPTS] Section 6. The delegate_pseudonym in the DelegationGrant provides the additional link to the specific agent that consumed the grant; a verifier can confirm that the agent that submitted the payment (identified by agent_id in the action_ref preimage per [STARK-RECEIPTS] Section 5.3) is the same agent that was granted authority (identified by delegate_pseudonym in the DelegationGrant).

The cross-verification is:

```
SHA-256(UTF-8(NFC(action_ref_preimage.agent_id))) mod P
  == DelegationGrant.delegate_pseudonym
```

A verifier that can perform this check independently confirms the full delegation binding without contacting any external registry. This is the Composable Trust Evidence property for the delegation surface.

The allowed_merchants and allowed_currencies fields in the DelegationGrant carry the same semantic as the scope constraints in a Composable Trust Evidence authority row: they define the set of actions the agent is authorised to perform. A2A implementations MAY map these fields directly to their authority row schema.

The discussion thread at [A2A-CTEF] established the requirement for this cross-verification during the x402 coalition review of the Axis 4 composite trust-query ([X402-2440]).

7. MCP Agent Protocol Notes

Model Context Protocol (MCP) servers that expose x402 payment tools to language model agents can use the delegation binding defined in this document to scope each agent session's payment authority. The Vauban Pay MCP server ([VAUBAN-DEMO]) implements the request_delegation_grant tool, which constructs a DelegationGrant with the binding fields defined in Section 4 and returns the JCS-canonical hash for use as the grant's content digest.

In an MCP payment cycle, the delegation binding operates as follows:

1. The MCP server receives an agent session registration carrying the agent's identity token (a URI or capability string issued at connection time).

2. The server computes the delegate_pseudonym from the agent identity token using the derivation in Section 5.1.

3. The server issues a DelegationGrant embedding the pseudonym and the session's spend caps, serialises the grant using JCS ([RFC8785]), and returns the JCS hash to the agent as the grant reference.

4. When the agent submits a make_payment call, the server reconstructs the grant from the retained manifest, computes the presenting agent's pseudonym, and compares it against the stored delegate_pseudonym before accepting the PaymentIntent.

5. The delegation_nonce is consumed on the first make_payment call within the session epoch; a second call within the same epoch is rejected with DelegationNonceReplay unless a new grant with a fresh nonce is issued.

MCP server implementations MUST NOT expose the Agent Identity string in the PAYMENT-RESPONSE or in any receipt body. The delegate_pseudonym is the only identity-derived value that appears in wire-format objects subject to this extension.

8. Security Considerations

8.1. Agent Identity Spoofing

An adversary that knows the target agent's identity string can precompute the expected delegate_pseudonym and attempt to construct a forged DelegationGrant with a matching pseudonym. The pseudonym construction uses SHA-256, which is collision-resistant under currently known attacks. Implementations MUST combine pseudonym verification with delegation_nonce consumption: a correct pseudonym does not suffice if the nonce has already been consumed.

Additional mitigation: principals issuing DelegationGrants SHOULD sign the grant object with their own signing key (per the signature mechanism of the selected receipt format) before transmitting it to the delegatee. A signed grant cannot be forged without the principal's private key, even if the adversary knows the target pseudonym.

8.2. Spend Cap Inflation

An adversary with write access to the grant receipt (for example, a compromised MCP server session) may attempt to inflate the cap_per_tx or cap_per_period fields before presenting the grant to the facilitator. Because the grant is JCS-hashed at issuance, any modification of these fields changes the grant's content digest. A facilitator that verifies the presented grant's JCS hash against the digest it stored at grant registration will detect the inflation.

Facilitators MUST store the JCS hash of each registered DelegationGrant at registration time and MUST re-verify the hash on every grant presentation. A grant whose current JCS hash does not match the stored hash MUST be rejected with GrantHashMismatch.

8.3. Nullifier Replay

The delegation_nonce consumption discipline defined in Section 5.2 prevents replay of a captured DelegationGrant receipt. The nullifier store MUST be durable: a facilitator restart that loses its nullifier store creates a replay window. Implementations MUST persist the nullifier store to durable storage before acknowledging grant consumption to the delegatee.

For high-availability deployments with multiple facilitator instances, the nullifier store MUST be shared (for example, via a distributed key-value store with strong consistency guarantees). Implementations using eventually-consistent stores MUST use optimistic locking to prevent race-condition-based nonce bypass.

8.4. Chain Depth Attack

An adversary may attempt to construct a sub-delegation chain that exceeds the root grant's max_chain_length by presenting each hop to a different facilitator that has not retained the chain history. Facilitators MUST verify the full chain depth against the root grant, not just the immediate parent grant. Chain reconstruction ([LIFECYCLE-FSM] Section 4.3) MUST be applied to the full chain before accepting any PaymentIntent.

Implementations that cannot reconstruct the full chain from retained manifests MUST reject the PaymentIntent with ChainNotReconstructable rather than accepting it with partial chain verification.

8.5. Cross-Format Confusion

The delegate_pseudonym derivation defined in Section 5.1 uses SHA-256 and the felt252 prime. A verifier that applies the same derivation to a different identity scheme (for example, a DID that resolves to a different key) or a different prime may compute a pseudonym that coincidentally matches a legitimate grant. Implementations MUST use the derivation as defined in Section 5.1 without variation; a pseudonym computed under a modified derivation is not compatible with this extension.

Facilitators supporting multiple receipt formats MUST apply the pseudonym derivation in Section 5.1 uniformly across formats. Format-specific pseudonym constructions that diverge from Section 5.1 MUST be treated as incompatible and MUST NOT be cross-verified against grants issued under this extension.

9. IANA Considerations

This document introduces no new IANA registrations. The delegate_pseudonym, cap_per_tx, cap_per_period, period_seconds, allowed_merchants, allowed_currencies, delegation_nonce, and max_chain_length fields are defined as extension fields within the DelegationGrant structure of the x402 V2 protocol ([X402-V2]). If the x402 Foundation TSC establishes a DelegationGrant field registry as an extension of the x402 Receipt Format registry defined in [STARK-RECEIPTS] Section 9, the fields in this document SHOULD be submitted for registration.

New error tokens introduced by this extension (AgentIdentityMismatch, DelegationNonceReplay, GrantRevoked, GrantHashMismatch, ChainNotReconstructable) are used in the X-Receipt-Reject-Reason header defined in [STARK-RECEIPTS] Section 5.2. These tokens extend the error taxonomy of [STARK-RECEIPTS] without requiring a new

registry; they follow the same naming convention and MUST be treated as unknown tokens by verifiers that have not implemented this extension.

10. Acknowledgments

The 18-vector bounded-spend conformance matrix in [X402-2432] was produced as part of the x402 Linux Foundation coalition cross-validation exercise. The cross-runtime byte-identical results across five language implementations (Python, TypeScript, Go, Java, Rust) established the feasibility of the JCS canonical preimage discipline as a binding mechanism for delegation receipts, and directly motivated the field set defined in Section 4.

The Composable Trust Evidence Format discussion ([A2A-CTEF]) and the composite trust-query work in [X402-2440] shaped the A2A integration notes in Section 6. In particular, the cross-verification construction in Section 6 (mapping action_ref_preimage.agent_id to delegate_pseudonym via shared SHA-256 and felt252 reduction) is grounded in the editorial alignment reached in the [X402-2440] review.

11. References

11.1. 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/>.

11.2. Informative References

[A2A-CTEF] "A2A Composable Trust Evidence Format (kenneives, x402 coalition thread)", n.d., <https://github.com/x402-foundation/x402/issues/1734>.

[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>.

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

[X402-2326]
"x402 shared canonicalisation discipline (coalition discussion thread)", n.d., <https://github.com/x402-foundation/x402/issues/2326>.

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

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

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

Appendix A. Worked Example: MCP Agent Delegation Cycle

The following example illustrates a complete MCP agent delegation cycle under this extension.

A.1. Step 1: Grant Issuance

The MCP server registers an agent session with identity string did:web:agent-42.mcp.example.com. The server computes:

```
pseudonym_raw = SHA-256(UTF-8(NFC("did:web:agent-42.mcp.example.com")))
              = <32-byte SHA-256 digest>
pseudonym_int = pseudonym_raw_as_big_endian_integer mod P
delegate_pseudonym = decimal_string(pseudonym_int)
```

The server issues the following DelegationGrant object (JCS key order shown for clarity):

```
{
  "allowed_currencies": ["urn:x402:currency:USDC"],
  "allowed_merchants": ["urn:x402:merchant:api-example"],
  "cap_per_period": "10000000",
  "cap_per_tx": "500000",
  "delegate_pseudonym": "<decimal-felt252>",
  "delegatee": "did:web:agent-42.mcp.example.com",
  "delegator": "did:web:principal.example.com",
  "delegation_nonce": "<decimal-felt252>",
  "expires_at": 1780000000,
  "max_chain_length": 1,
  "period_seconds": 86400,
  "scope": ["payment:usdc", "merchant:api-example"]
}
```

The server serialises this object via JCS ([RFC8785]) and computes the grant content digest: grant_hash = SHA-256(UTF-8(JCS(grant_object))). The grant_hash is returned to the agent as its DelegationGrant reference.

A.2. Step 2: PaymentIntent Binding

When the agent submits a make_payment call, the server:

1. Retrieves the stored DelegationGrant by grant_hash.
2. Verifies cap_per_tx: the requested amount is within the per-transaction ceiling.
3. Verifies allowed_merchants: the target merchant is in the list.
4. Verifies allowed_currencies: the payment currency is permitted.
5. Computes the presenting agent’s pseudonym and compares it against delegate_pseudonym (constant-time comparison).
6. Checks the delegation_nonce against the nullifier store.
7. If all checks pass, builds the PaymentIntent carrying the grant_hash as its DelegationGrant reference.

A.3. Step 3: Facilitator Verification

The facilitator receiving the PaymentIntent:

1. Extracts the grant_hash from the PaymentIntent.
2. Looks up the retained DelegationGrant manifest by grant_hash.
3. Re-hashes the manifest via JCS and confirms the digest matches grant_hash (detects any field inflation).
4. Verifies the presenting agent’s delegate_pseudonym matches the retained value.
5. Verifies the delegation_nonce is not in the nullifier store.
6. Accepts or rejects the PaymentIntent based on all scope and cap checks.

Appendix B. Error Token Reference

The following error tokens extend the X-Receipt-Reject-Reason taxonomy of [STARK-RECEIPTS] Section 5.2 for delegation binding failures:

Token	HTTP Status	Description
AgentIdentityMismatch	403	The presenting agent’s pseudonym does not match delegate_pseudonym in the retained DelegationGrant.
DelegationNonceReplay	409	The delegation_nonce has already been consumed. The grant cannot be replayed.
GrantRevoked	410	The DelegationGrant has been explicitly revoked by the issuing principal.
GrantHashMismatch	422	The JCS hash of the presented DelegationGrant does not match the stored digest. Possible cap inflation or field tampering.
ChainNotReconstructable	422	The facilitator cannot reconstruct the full delegation chain from retained manifests. The PaymentIntent is rejected pending chain availability.
DelegationDepthExceeded	422	The delegation chain exceeds the root grant’s max_chain_length.
GrantExpired	410	The DelegationGrant expires_at has elapsed. Expired grants do not authorise new PaymentIntents (per [LIFECYCLE-FSM] Section 3.6).

Table 2

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 delegation binding specification, the published conformance vectors (https://github.com/vauban-org/x402-stark-receipts-conformance), and the reference implementations listed below. The bounded-spend-authorization vector set demonstrates the DelegationGrant + SettlementReceipt pair under the six-element Claim sextuplet shape with the seven fiscal_authority qualification fields mapped to cryptographic primitives.

Reference Implementation Matrix

The conformance vector suite maintains an 8-implementation reference matrix across Python, JavaScript, Go, Java, Rust, PHP, Ruby, and C#. The first five are validated byte-for-byte against upstream JCS RFC 8785 libraries ; the last three are published as pure-stdlib 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 agent identity binding mechanism implemented (DID method, key type), the DelegationGrant chain enforcement strategy, the revocation publication endpoint, 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