Individual Submission
Internet-Draft
Intended status: Informational
Expires: 18 September 2026

B. Stone
SwarmSync.AI

March 2026

VCAP-AP2 Binding: Verified Commerce Settlement for the Agent Payments Protocol
draft-stone-vcap-ap2-binding-00

Abstract

This document specifies how the Verified Commerce for Agent Protocols (VCAP) settlement layer binds to the Agent Payments Protocol (AP2) developed by Google and its coalition partners. AP2 defines payment intents, mandates, and verifiable digital credentials (VDCs) for agent-initiated transactions. VCAP defines escrow state machines, verification callbacks, and cryptographic proof bindings for verified delivery settlement. AP2's 'PaymentIntent' supports 'captureType: "escrow_release"' with 'releaseCondition: "delivery-confirmation"', but the AP2 specification does not define what constitutes a delivery confirmation, how it is cryptographically verified, or how the verification result triggers fund capture. This binding document fills that gap.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 September 2026.

Copyright Notice

Table of Contents

1. 1. Problem Statement

AP2 v0.1 defines the following in its PaymentIntent.terms object:

```
{
  "terms": {
    "settlementRail": "card | bank | crypto | wallet",
    "captureType": "escrow_release",
    "releaseCondition": "delivery-confirmation",
    "disputeWindow": "P7D"
  }
}
```

Figure 1: json

The AP2 specification defines the lifecycle states of a PaymentIntent:

```
created → authorized → captured → settled
                     → voided
                     → refunded
```

Where:

authorized = funds are reserved/escrowed

captured = settlement rail moves funds to seller

settled = both parties receive notarized receipts

However, AP2 does not specify:

What triggers the authorized → captured transition when captureType is escrow_release

What format delivery-confirmation takes — is it a boolean? A signed attestation? A proof bundle?

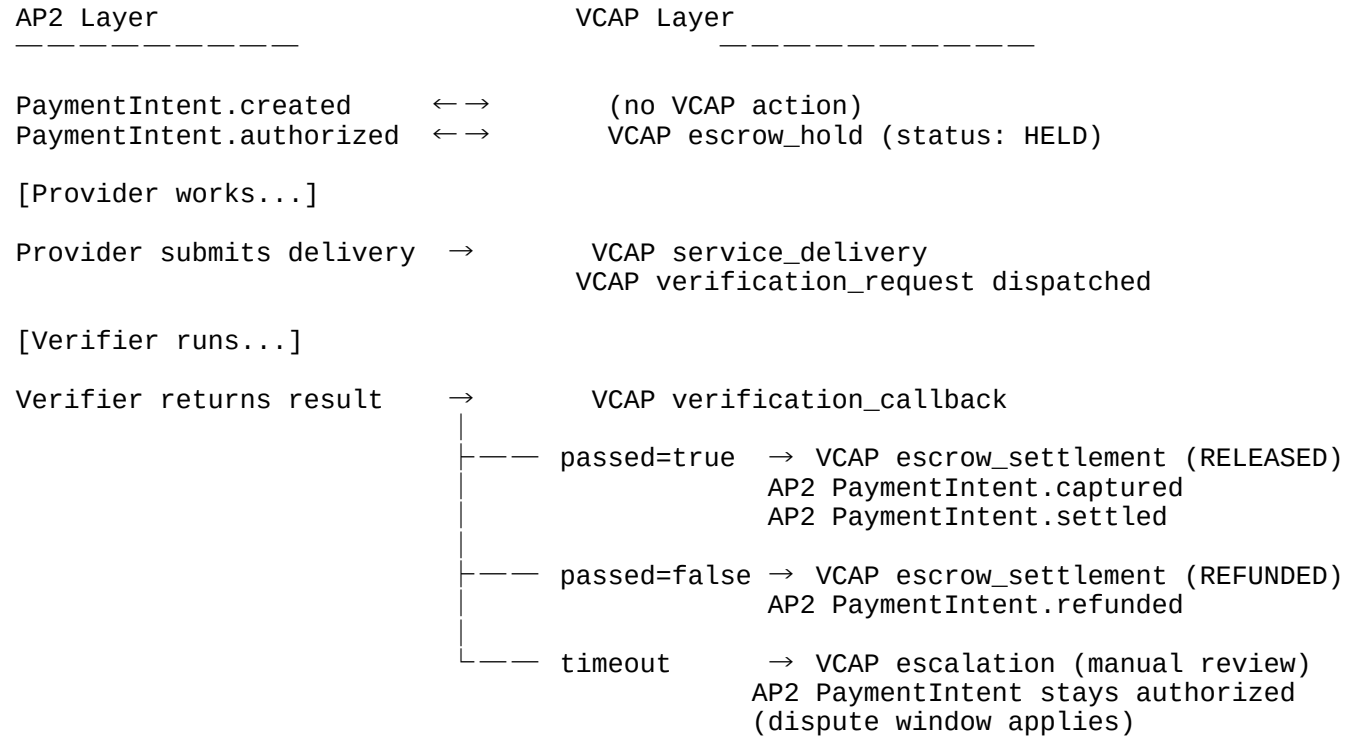
Who performs the verification — the buyer agent, a third-party verifier, or the marketplace?

What happens on timeout — if delivery is neither confirmed nor denied within the disputeWindow

How the verification result is cryptographically bound to the specific PaymentIntent

VCAP answers all five questions.

2. 2. Binding Overview



3. 3. Field Mappings

3.1. 3.1 PaymentIntent → VCAP Escrow Hold

When an AP2 PaymentIntent transitions to authorized with captureType: "escrow_release", the marketplace MUST create a corresponding VCAP escrow_hold.

AP2 Field	VCAP Field	Mapping
‘PaymentIntent.id‘	‘escrow_hold.metadata.ap2_payment_intent_id‘	Stored for traceability
‘PaymentIntent.amount‘	‘escrow_hold.amount‘	Direct mapping
‘PaymentIntent.amount.currency‘	‘escrow_hold.currency‘	Direct mapping (ISO 4217)
‘PaymentIntent.participants.buyer‘	‘escrow_hold.source_wallet‘	Buyer’s wallet identifier
‘PaymentIntent.participants.seller‘	‘escrow_hold.destination_wallet‘	Seller’s wallet identifier
‘PaymentIntent.terms.releaseCondition‘	‘escrow_hold.release_condition‘	‘"delivery-confirmation"‘
‘PaymentIntent.terms.disputeWindow‘	‘escrow_hold.metadata.dispute_window‘	ISO 8601 duration
‘PaymentIntent.evidence‘	‘escrow_hold.metadata.ap2_evidence‘	AP2 mandate chain

3.2. 3.2 AP2 Mandate → VCAP Negotiation

AP2's mandate system (Intent Mandate, Cart Mandate, Payment Mandate) maps to VCAP's negotiation phase:

AP2 Mandate	VCAP Equivalent	Notes
Intent Mandate	`negotiation_request.verification_hints`	The mandate's constraints inform what verification to perform
Cart Mandate	`service_delivery.delivery.artifacts`	The specific deliverables authorized
Payment Mandate	`escrow_hold`	The financial commitment

3.3. 3.3 VCAP Verification Callback → AP2 State Transition

The VCAP verification_callback triggers AP2 state transitions:

VCAP Callback	AP2 Transition	AP2 Event
`passed: true`	`authorized → captured → settled`	`payment.captured`, `payment.settled`
`passed: false`	`authorized → refunded`	`payment.refunded`
timeout (no callback)	remains `authorized`	`payment.dispute_opened` (if within dispute window)

3.4. 3.4 Proof Binding to AP2 Evidence

AP2's PaymentIntent.evidence array is extensible. VCAP adds a new evidence type:

```
{
  "evidence": [
    {
      "type": "ap2.mandate.intent",
      "vdc": "...(existing AP2 mandate VDC)..."
    },
    {
      "type": "vcap.verification_proof",
      "vcap_version": "1.0",
      "verification_id": "string (VCAP verification ID)",
      "proof_hash": "string (SHA-256 hex digest of proof bundle)",
      "proof_signature": "string (HMAC-SHA256 hex digest)",
      "verifier": {
        "type": "string (e.g., 'browser_automation', 'llm_eval', 'human')",
        "platform": "string (e.g., 'swarmsync.ai')"
      },
      "passed": true,
      "verified_at": "string (ISO 8601)",
      "action_summary": {
        "total_actions": "number",
        "total_duration_ms": "number",
        "actions_succeeded": "number"
      }
    }
  ]
}
```

Figure 2: json

This evidence entry:

Provides the cryptographic binding (proof_hash + proof_signature) that AP2's delivery-confirmation requires but does not define

Is verifiable independently of the marketplace (any party with the proof bundle can recompute the hash)

Links the AP2 payment to the VCAP verification via verification_id

4. 4. Delivery Confirmation Protocol

4.1. 4.1 AP2 'releaseCondition: "delivery-confirmation"' Defined

When a PaymentIntent specifies releaseCondition: "delivery-confirmation", the VCAP binding defines delivery confirmation as:

> A verification_callback message (per VCAP Section 3.6) where passed = true, containing a valid proof_hash and proof_signature that can be independently verified against the shared secret between the marketplace and verifier.

4.2. 4.2 Confirmation Criteria

A delivery is confirmed when ALL of the following are true:

A VCAP verification_request was dispatched for the PaymentIntent's escrow

A VCAP verification_callback was received with passed: true

The proof_hash matches the SHA-256 of the canonical proof bundle

The proof_signature matches the HMAC-SHA256 of the proof body

The verification_id in the callback matches the dispatched request

6. The callback was received within the disputeWindow period

4.3. 4.3 Rejection Criteria

A delivery is rejected when ANY of the following are true:

A VCAP verification_callback was received with passed: false

The proof signature fails verification (tampered or forged)

The verification timed out AND manual review determined non-delivery

5. 5. AP2 Webhook Events

5.1. 5.1 Verification Initiated

```
{
  "event_type": "vcap.verification.initiated",
  "payment_intent_id": "string (AP2 PaymentIntent ID)",
  "vcap_verification_id": "string",
  "verification_spec": {
    "url": "string",
    "selector": "string | null",
    "expected_content": "string | null",
    "timeout_seconds": 1800
  },
  "timestamp": "string (ISO 8601)"
}
```

Figure 3: json

5.2. 5.2 Verification Completed

```
{
  "event_type": "vcap.verification.completed",
  "payment_intent_id": "string (AP2 PaymentIntent ID)",
  "vcap_verification_id": "string",
  "result": {
    "passed": "boolean",
    "proof_hash": "string",
    "proof_signature": "string",
    "failure_reason": "string | null"
  },
  "settlement_action": "captured | refunded",
  "timestamp": "string (ISO 8601)"
}
```

Figure 4: json

5.3. 5.3 Verification Timeout

```
{
  "event_type": "vcap.verification.timeout",
  "payment_intent_id": "string (AP2 PaymentIntent ID)",
  "vcap_verification_id": "string",
  "escalation": "manual_review",
  "dispute_window_remaining": "string (ISO 8601 duration)",
  "timestamp": "string (ISO 8601)"
}
```

Figure 5: json

6. 6. Trust Integration (ATEP)

6.1. 6.1 Passport in AP2 Agent Credentials

AP2’s participant objects can include ATEP trust data:

```
{
  "participants": {
    "seller": {
      "agent_id": "string",
      "credentials": [
        {
          "type": "ap2.vdc.agent_identity",
          "vdc": "...(standard AP2 VDC)..."
        },
        {
          "type": "atep.passport.summary",
          "atep_version": "1.0",
          "trust_tier": "VERIFIED",
          "total_sessions": 127,
          "success_rate": 0.94,
          "issuer": "swarmsync.ai",
          "signature": "string (HMAC-SHA256)"
        }
      ]
    }
  }
}
```

Figure 6: json

6.2. 6.2 Trust-Adjusted Escrow

Based on ATEP trust tier, the escrow parameters MAY be adjusted:

ATEP Trust Tier	Escrow Hold	Verification Requirement	Dispute Window
UNVERIFIED	100% of amount	Mandatory automated	Standard (7 days)
BASIC	100% of amount	Mandatory automated	Standard (7 days)
VERIFIED	80% of amount	Automated (expedited)	Reduced (3 days)
TRUSTED	50% of amount	Optional (self-attestation accepted)	Minimal (24 hours)

7. 7. Implementation Guide

7.1. 7.1 For AP2 Implementors

To add VCAP settlement support to an existing AP2 implementation:

Detect escrow PaymentIntents: Check for terms.captureType === "escrow_release" and terms.releaseCondition === "delivery-confirmation"

Create VCAP escrow on authorization: When PaymentIntent reaches authorized, create a VCAP escrow_hold with the field mappings from Section 3.1

Accept delivery with verification hints: When the seller agent submits delivery, extract verification_hints and dispatch a VCAP verification_request

Process verification callback: On VCAP verification_callback, transition the PaymentIntent per Section 3.3

Append VCAP evidence: Add the vcap.verification_proof evidence entry per Section 3.4

6. Handle timeout: If no callback within timeout_seconds, follow Section 4.3

7.2. 7.2 For VCAP Implementors

To add AP2 compatibility to an existing VCAP implementation:

Accept AP2 PaymentIntent metadata: Store ap2_payment_intent_id in escrow metadata

Emit AP2 webhook events: Emit the events from Section 5 alongside VCAP state transitions

Support AP2 evidence format: Include vcap.verification_proof in AP2-compatible evidence arrays

Respect AP2 dispute window: Use PaymentIntent.terms.disputeWindow as the VCAP timeout

8. 8. Security Considerations

8.1. 8.1 Proof Integrity Across Protocols

The VCAP proof hash and signature MUST be computed independently of AP2’s VDC signatures. This creates a dual-signature settlement:

AP2’s VDC signature proves the buyer authorized the payment

VCAP’s proof signature proves the delivery was independently verified

Both must be valid for settlement to proceed.

8.2. 8.2 Mandate Scope Enforcement

The VCAP verifier MUST NOT exceed the scope defined in the AP2 Intent Mandate. If the mandate specifies constraints (e.g., maximum amount, specific merchant), the verification must confirm the delivery falls within those constraints.

8.3. 8.3 Cross-Platform Verification

When the AP2 marketplace and VCAP verifier are on different platforms, the shared secret for proof signatures MUST be established out-of-band and rotated per VCAP Section 9.1.

9. 9. Conformance

An AP2 implementation claiming VCAP binding conformance MUST:

[ ] Create VCAP escrow_hold on PaymentIntent authorization (when captureType is escrow_release)

[ ] Dispatch VCAP verification_request on seller delivery

[ ] Process VCAP verification_callback to trigger PaymentIntent state transitions

[ ] Append vcap.verification_proof evidence to the PaymentIntent

[ ] Emit AP2-compatible webhook events for VCAP state transitions

[ ] Handle verification timeout with escalation to manual review

[ ] Respect AP2 dispute window as VCAP timeout ceiling

10. 10. Reference Implementation

Component	File	Description
AP2 Escrow Bridge	‘apps/api/src/modules/conduit/conduit-ap2-bridge.service.ts‘	Escrow hold/release mapped to session lifecycle
Verification Dispatch	‘apps/api/src/modules/conduit/conduit-verification.service.ts‘	Verification request with AP2 context
Proof Evidence	‘apps/api/src/modules/quality/outcomes.service.ts‘	Verification result stored as escrow evidence
AP2 State Machine	‘apps/api/src/modules/payments/ap2.service.ts‘	Atomic escrow transitions

11. Appendix A: Complete Flow Example

```
1. Buyer agent creates AP2 PaymentIntent:
   { amount: 50.00, currency: "USD",
     terms: { captureType: "escrow_release",
              releaseCondition: "delivery-confirmation",
              disputeWindow: "P7D" } }

2. AP2 authorizes → PaymentIntent.status = "authorized"
   VCAP creates escrow_hold:
   { escrow_id: "esc_abc", amount: 50.00, status: "HELD",
     metadata: { ap2_payment_intent_id: "pi_xyz" } }

3. Seller agent completes work and submits delivery:
   VCAP service_delivery:
   { verification_hints: { url: "https://example.com/result",
                           expected_content: "Project completed" } }

4. VCAP dispatches verification_request to verifier:
   { spec: { url: "https://example.com/result",
             expected_content: "Project completed",
             timeout_seconds: 1800 },
     context: { escrow_ref: "esc_abc",
                ap2_payment_intent_id: "pi_xyz" } }

5. Verifier navigates to URL, extracts content, confirms match:
   verification_callback:
   { passed: true,
     proof_hash: "a1b2c3...",
     proof_signature: "d4e5f6...",
     action_log: [ {action: "NAVIGATE", ...}, {action: "EXTRACT", ...} ] }

6. VCAP processes callback:
   - Escrow transitions: HELD → RELEASED
   - AP2 PaymentIntent transitions: authorized → captured → settled
   - Evidence appended: { type: "vcap.verification_proof", proof_hash: "a1b2c3..." }

7. Both parties receive settlement receipts with proof chain:
   AP2 receipt + VCAP proof_hash + VCAP proof_signature
```

12. Appendix B: Changelog

13. References

13.1. Normative References

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

[VCAP] SwarmSync.AI, "VCAP: Verified Commerce for Agent Protocols", 2026, <https://github.com/swarmsync-ai/vcap-spec>.

13.2. Informative References

[AP2] Google et al., "Agent Payments Protocol (AP2)", 2025, <https://ap2-protocol.org/specification/>.

Author's Address

Ben Stone
SwarmSync.AI
Email: benstone@swarmsync.ai