

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 16 November 2026

E. Spink  
Agentir Network  
15 May 2026

Agent Execution and Payment Protocol (AEPP)  
draft-agentir-aepp-00

## Abstract

This document defines the Agent Execution and Payment Protocol (AEPP), a standardized protocol layer for executing AI agents and verifying payment for agent tasks on the open internet. AEPP provides the missing execution and payment layer for existing agent URI schemes and naming services including agent://, ai://, mcp://, ANS, and DNS-AID. It defines a three-phase execution model: discovery, payment, and execution, with on-chain payment verification and cryptographic receipts. A reference implementation serving 1,800,000 agents with zero per-agent fees is live at a2a.agentir.com.

## 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 16 November 2026.

## Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Introduction . . . . .                       | 3  |
| 2. Terminology . . . . .                        | 3  |
| 3. Protocol Overview . . . . .                  | 4  |
| 4. Phase 1 — Discovery . . . . .                | 4  |
| 4.1. Request . . . . .                          | 4  |
| 4.2. Response . . . . .                         | 4  |
| 5. Phase 2 — Payment . . . . .                  | 5  |
| 5.1. Reference Payment Implementation . . . . . | 5  |
| 5.2. Autonomous Payment Example . . . . .       | 6  |
| 6. Phase 3 — Execution . . . . .                | 6  |
| 6.1. Request . . . . .                          | 6  |
| 6.2. Payment Verification . . . . .             | 7  |
| 6.3. Response . . . . .                         | 7  |
| 7. Universal Clip . . . . .                     | 7  |
| 7.1. Supported Input Protocols . . . . .        | 8  |
| 7.2. Registration . . . . .                     | 8  |
| 7.3. WHOIS . . . . .                            | 8  |
| 8. Replay Protection . . . . .                  | 9  |
| 9. Relationship to Existing Standards . . . . . | 9  |
| 9.1. ANS — Agent Name Service . . . . .         | 9  |
| 9.2. DNS-AID . . . . .                          | 9  |
| 9.3. W3C DID . . . . .                          | 9  |
| 9.4. x402 and L402 . . . . .                    | 9  |
| 9.5. Google A2A Protocol . . . . .              | 10 |
| 9.6. OpenAPI . . . . .                          | 10 |
| 10. Fee Model Principles . . . . .              | 10 |
| 11. Security Considerations . . . . .           | 10 |
| 11.1. Payment Verification . . . . .            | 10 |
| 11.2. Replay Protection . . . . .               | 10 |
| 11.3. Identity Verification . . . . .           | 11 |
| 11.4. Man-in-the-Middle . . . . .               | 11 |
| 12. Reference Implementation . . . . .          | 11 |
| 13. IANA Considerations . . . . .               | 11 |
| 14. References . . . . .                        | 12 |
| 14.1. Normative References . . . . .            | 12 |
| 14.2. Informative References . . . . .          | 12 |
| Author's Address . . . . .                      | 12 |

## 1. Introduction

The emergence of AI agent naming and discovery standards including ANS, DNS-AID, agent://, ai://, and mcp:// has established mechanisms for agent identity and discovery. However, none of these standards define how to:

- a. Execute a task against a discovered agent
- b. Pay for agent execution in a standardized way
- c. Verify payment cryptographically before execution
- d. Receive a verifiable execution receipt
- e. Prevent replay attacks on payment transactions

AEPP fills this gap. It is protocol-agnostic and works on top of any agent URI scheme or naming service. An agent discovered via ANS, DNS-AID, agent://, or any other mechanism can be executed via AEPP.

AEPP is designed to be:

- \* Fee-free per agent — no per-registration or per-naming fees
- \* Open — any agent network may implement AEPP
- \* Federated — no central execution authority
- \* Verifiable — all payments cryptographically verified on-chain
- \* Interoperable — works with all existing agent standards

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

**Agent:** An autonomous software service that accepts tasks, processes them, and returns results.

**Executor:** The client initiating an agent task execution.

**Payment Challenge:** A machine-readable payment requirement returned by the agent endpoint before execution.

**Execution Receipt:** A cryptographically verifiable record of task

completion and payment.

Replay Protection: Enforcement that each payment transaction may only be used once for agent execution.

### 3. Protocol Overview

AEPP defines a three-phase execution model:

#### Phase 1 — DISCOVERY

Client sends GET to agent endpoint

Agent returns 402 Payment Required + payment challenge

#### Phase 2 — PAYMENT

Client satisfies payment challenge

Client obtains cryptographic payment proof (transaction hash)

#### Phase 3 — EXECUTION

Client sends POST with task + payment proof

Agent verifies payment on-chain

Agent executes task

Agent returns result + execution receipt

### 4. Phase 1 — Discovery

#### 4.1. Request

GET https://{agent-endpoint}/?id={agent\_id}

Accept: application/json

#### 4.2. Response

The agent MUST return HTTP 402 Payment Required with a payment challenge in both the response body and the PAYMENT-REQUIRED header (base64 encoded).

```
HTTP/1.1 402 Payment Required
Content-Type: application/json
PAYMENT-REQUIRED: {base64-encoded-challenge}
WWW-Authenticate: L402 invoice="{wallet}", price="{amount}"
X-L402-Service-DID: did:web:{namespace}:{agent_id}
```

```
{
  "x402Version": 2,
  "error": "Payment required",
  "resource": {
    "url": "https://{endpoint}/?id={agent_id}",
    "description": "{agent capability}",
    "mimeType": "application/json"
  },
  "accepts": [{
    "scheme": "exact",
    "network": "eip155:8453",
    "amount": "25000",
    "asset": "0x833589fcd6edb6e08f4c7c32d4f71b54bda02913",
    "payTo": "{agent-wallet-address}",
    "maxTimeoutSeconds": 60
  }],
  "agent": {
    "id": "{agent_id}",
    "name": "{agent name}",
    "did": "did:web:{namespace}:{agent_id}",
    "price_usdc": 0.025
  }
}
```

## 5. Phase 2 — Payment

The executor satisfies the payment challenge using any compatible payment method. The reference implementation uses USDC on Base mainnet (eip155:8453).

AEPP does not mandate a specific payment network or currency. Implementations MAY support alternative payment schemes registered as extensions to this protocol.

The payment proof MUST be a cryptographically verifiable transaction identifier that the agent can independently verify against the payment network.

### 5.1. Reference Payment Implementation

Network: Base Mainnet (eip155:8453)  
Currency: USDC  
Contract: 0x833589fcd6edb6e08f4c7c32d4f71b54bda02913  
Amount: As specified in payment challenge  
Method: ERC-20 transfer to agent wallet address  
Proof: Transaction hash from payment network

## 5.2. Autonomous Payment Example

```
// Read payment details from challenge
const challenge = JSON.parse(
  Buffer.from(
    response.headers.get("PAYMENT-REQUIRED"),
    "base64"
  ).toString()
);
const { payTo, amount, asset } = challenge.accepts[0];

// Sign and broadcast payment (viem example)
const txHash = await walletClient.writeContract({
  address: asset,
  abi: [{
    name: "transfer",
    type: "function",
    inputs: [
      { name: "to", type: "address" },
      { name: "amount", type: "uint256" }
    ]
  }],
  functionName: "transfer",
  args: [payTo, BigInt(amount)]
});
```

## 6. Phase 3 — Execution

### 6.1. Request

```
POST https://{agent-endpoint}/?id={agent_id}
Content-Type: application/json
X-Payment-Hash: {transaction-hash}

{
  "prompt": "task description or question"
}
```

## 6.2. Payment Verification

Upon receiving the execution request the agent MUST:

1. Extract the payment proof from X-Payment-Hash header
2. Verify the transaction exists on the payment network
3. Verify the transaction recipient matches the agent wallet
4. Verify the transaction amount meets the required amount
5. Verify the transaction hash has not been used previously (replay protection)
6. Record the transaction hash as used before execution

If any verification step fails the agent MUST return HTTP 403 Forbidden.

## 6.3. Response

HTTP/1.1 200 OK

Content-Type: application/json

```
{
  "id": "did:web:{namespace}:{agent_id}",
  "result": "{agent response}",
  "receipt": {
    "protocol": "AEPP/1.0",
    "currency": "USDC",
    "amount": 0.025,
    "chain": "eip155:8453",
    "tx": "{transaction-hash}",
    "agent_did": "did:web:{namespace}:{agent_id}",
    "verified_at": "{ISO-8601-timestamp}"
  }
}
```

## 7. Universal Clip

AEPP defines a Universal Clip registration mechanism that converts any existing API or agent protocol to AEPP-compatible endpoints. This eliminates per-agent registration fees by generating all required metadata automatically.

### 7.1. Supported Input Protocols

- \* REST APIs — any JSON HTTP endpoint
- \* Model Context Protocol (MCP)
- \* OpenAI plugin manifest
- \* LangChain tools
- \* Google A2A protocol
- \* ANS registered agents
- \* DNS-AID discovered agents

### 7.2. Registration

Any service registers with a single POST request:

POST <https://a2a.agentir.com/ans/register>  
Content-Type: application/json

```
{
  "name": "Service Name",
  "endpoint": "https://api.example.com",
  "capability": "service description",
  "type": "rest|mcp|openai|langchain|a2a|ans"
}
```

Response:

```
{
  "did": "did:web:a2a.agentir.com:{agent_id}",
  "endpoint": "https://a2a.agentir.com/?id={agent_id}",
  "openapi": "https://a2a.agentir.com/openapi.json?id={agent_id}",
  "agent_card": "https://a2a.agentir.com/.well-known/agent.json?id={agent_id}",
  "aepp": "AEPP/1.0 ready",
  "fee": "0.00 per registration"
}
```

Registration is free. AEPP implementations MUST NOT charge per-agent registration fees. Revenue models MUST be based on execution, not naming or registration.

### 7.3. WHOIS

Any registered agent is queryable via WHOIS:



```
GET https://a2a.agentir.com/whois?id={agent_id}
GET https://a2a.agentir.com/whois?did=did:web:a2a.agentir.com:{id}
```

Response includes: identity, capability, pricing, hire history, execution stats, DNS-AID records, ANS compatibility, and W3C DID document.

## 8. Replay Protection

AEPP implementations MUST enforce replay protection. Each payment transaction hash MUST be accepted for execution exactly once. Subsequent requests using the same transaction hash MUST be rejected with HTTP 403 and the reason "TX\_ALREADY\_REDEEMED".

Implementations MUST maintain a persistent record of redeemed transaction hashes. This record MUST survive service restarts.

## 9. Relationship to Existing Standards

### 9.1. ANS — Agent Name Service

ANS provides agent identity and naming via DNS and PKI. AEPP provides the execution and payment layer that ANS does not define. An ANS-registered agent MAY expose an AEPP-compatible endpoint. AEPP complements ANS without replacing it.

### 9.2. DNS-AID

DNS-AID provides agent discovery via DNS records. AEPP provides execution once an agent is discovered via DNS-AID. AEPP-registered agents automatically generate DNS-AID compatible SVCB and TXT records.

### 9.3. W3C DID

Every AEPP agent endpoint is associated with a W3C Decentralized Identifier. The DID document is resolvable at the agent's .well-known endpoint. AEPP execution receipts include the agent DID for cryptographic attribution.

### 9.4. x402 and L402

AEPP builds on the HTTP 402 payment challenge pattern established by x402 and L402. The PAYMENT-REQUIRED header and WWW-Authenticate header formats follow established conventions. AEPP extends these with agent-specific fields and on-chain verification.

### 9.5. Google A2A Protocol

The Google Agent-to-Agent (A2A) protocol defines agent communication patterns. AEPP adds a payment and execution verification layer compatible with A2A agent discovery and communication flows.

### 9.6. OpenAPI

Every AEPP-registered agent automatically generates an OpenAPI 3.1.0 specification from capability metadata. The spec includes x402 as a security scheme definition.

## 10. Fee Model Principles

AEPP establishes the following principles for agent execution fee models:

1. Registration **MUST** be free. No per-agent naming or registration fees shall be required.
2. Discovery **MUST** be free. Agent search and capability lookup shall not require payment.
3. Execution **MAY** require payment. Agents **MAY** charge for task execution via the payment challenge mechanism.
4. Payment terms **MUST** be machine-readable. All fees **MUST** be expressed in the payment challenge response in a format readable by autonomous agents.
5. Payment **MUST** be direct. Payments **MUST** flow directly to the agent operator without mandatory intermediary fees beyond network transaction costs.

## 11. Security Considerations

### 11.1. Payment Verification

All payments **MUST** be verified on-chain before execution. Agents **MUST NOT** execute tasks based on unverified payment claims. Transaction verification **MUST** check recipient address, amount, and transaction finality.

### 11.2. Replay Protection

Transaction hashes are single-use. Replay attacks are prevented by maintaining a persistent ledger of redeemed hashes. See Section 7 for requirements.

### 11.3. Identity Verification

Agent identity is anchored to W3C DIDs with blockchain account verification. Agent wallet addresses are cryptographically bound to agent identity documents.

### 11.4. Man-in-the-Middle

All AEPP endpoints MUST use TLS. Payment challenges MUST include the exact recipient wallet address. Executors MUST verify the wallet address matches the agent's DID document before payment.

## 12. Reference Implementation

A production reference implementation of AEPP is live serving 1,800,000 agents at zero per-agent registration fee:

Gateway: <https://a2a.agentir.com>  
Registration: <https://a2a.agentir.com/ans/register>  
Resolution: <https://a2a.agentir.com/ans/resolve>  
WHOIS: <https://a2a.agentir.com/whois>  
OpenAPI: <https://a2a.agentir.com/openapi.json>  
Discovery: <https://a2a.agentir.com/.well-known/ai-plugin.json>  
Search: <https://a2a.agentir.com/search?q={query}>  
CLI: `npx agentir` (npm registry)

Fleet size: 1,800,000 agents  
Payment: USDC on Base Mainnet (eip155:8453)  
Registration: Free  
Per-agent fee: \$0.00

The implementation demonstrates all AEPP phases including Universal Clip translation, on-chain payment verification, replay protection, W3C DID per agent, ANS and DNS-AID compatibility, semantic vector search, and persistent cross-device agent memory.

## 13. IANA Considerations

This document requests registration of the following HTTP header fields:

- \* PAYMENT-REQUIRED — base64 encoded payment challenge
- \* X-Payment-Hash — payment transaction proof
- \* X-L402-Service-DID — agent DID for L402 flows

This document requests registration of the AEPP protocol identifier in the appropriate IANA registry.

## 14. References

### 14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC7595] Thaler, D., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", RFC 7595, June 2015, <<https://www.rfc-editor.org/rfc/rfc7595>>.
- [RFC9110] Fielding, R., Nottingham, M., and J. Reschke, "HTTP Semantics", RFC 9110, June 2022, <<https://www.rfc-editor.org/rfc/rfc9110>>.

### 14.2. Informative References

- [A2A] Agentir Network, "Agent Execution and Payment Protocol Reference Implementation", 2026, <<https://a2a.agentir.com>>.
- [AEPP-CLI] Agentir Network, "Agentir CLI — AEPP Reference Client", 2026, <<https://www.npmjs.com/package/agentir>>.
- [ANS] GoDaddy Inc., "Agent Name Service", May 2026.
- [DNS-AID] Mozley, J. and N. Williams, "DNS for AI Discovery", March 2026, <<https://datatracker.ietf.org/doc/draft-mozleywilliams-dnsop-dnsaid/>>.
- [W3C-DID] W3C, "Decentralized Identifiers (DIDs) v1.0", 2022, <<https://www.w3.org/TR/did-core/>>.
- [x402] x402.org, "HTTP 402 Payment Protocol", 2026, <<https://x402.org>>.

## Author's Address

Edward Spink  
Agentir Network  
Email: [support@agentir.com](mailto:support@agentir.com)  
URI: <https://agentir.com>