Independent Submission
Internet-Draft
Intended status: Experimental
Expires: 17 June 2026

A. Sogomonian
Artificial Intelligence Internet Foundation (AIIF)

14 December 2025

# Architecture for the Artificial Intelligence Internet Protocol (AIIP)
## draft-sogomonian-aiip-architecture-03

## Abstract

This document defines the architectural model for the Artificial Intelligence Internet Protocol (AIIP), including decentralized discovery and a byte-precise native envelope. AIIP enables stateless, verifiable invocation of autonomous systems using a resolve-invoke-receipt pattern over authenticated transports. Native AIIP is the default for machine-to-machine operation; HTTPS gateways are optional for human-facing access. Each invocation produces a cryptographically verifiable execution receipt, optionally including attestation evidence.

## Note

Work in Progress.

## 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 17 June 2026.

## Copyright Notice

Table of Contents



## 1. Introduction

AIIP provides a uniform address space and invocation model for AI resources with verifiable outcomes. The architecture follows a resolve-invoke-receipt pattern.

Autonomous systems communicate natively using AIIP envelopes over authenticated transports. Human operators and supervisory systems can access AIIP resources via HTTPS gateways for compatibility with Web-based tools.

This revision adds discovery metadata (DNS bootstrap plus HTTPS well-known) and an autonomous delegation use case.

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 [RFC2119] and [RFC8174].

## 2. Architectural Overview

AIIP separates addressing, invocation, and verification into distinct steps: (1) discovery and resolution provide invocation metadata; (2) invocation is carried in a native AIIP envelope; (3) the response includes an execution receipt that can be verified independently.

## 3. Native Autonomous Access Model

Autonomous systems, including robots, industrial controllers, agents, and embedded devices, use AIIP natively. In native mode, AIIP envelopes are exchanged directly between client and resource over an authenticated transport such as mutual TLS or QUIC.

Native AIIP invocation does not depend on HTTP semantics, cookies, or browser-originated state. Authorization, policy enforcement, and result verification are defined by AIIP.

Human operators and management interfaces typically interact with AIIP resources through HTTPS gateways. These gateways provide compatibility with Web-based clients but are not required for autonomous operation.

## 4. Discovery and Delegation Metadata

AIIP supports decentralized discovery using two complementary mechanisms: DNS-based bootstrap discovery and HTTPS-based detailed metadata retrieval. This design avoids reliance on any single global resolver by allowing each administrative domain to publish its own resolution parameters and delegation information.

### 4.1. DNS Bootstrap Discovery

For an AIIP identifier whose authority is a DNS name (for example, aiip://service.example/...), an implementation MAY perform bootstrap discovery by querying a DNS TXT record under _aiip.service.example.

The TXT record value is a semicolon-separated set of key/value parameters. At a minimum, it SHOULD include an endpoint parameter indicating where detailed metadata can be retrieved.

Example:

```
_aiip.service.example.  IN TXT  "v=aiip1; endpoint=https://service.example/.well-known
/aiip"
```

The DNS record is intended for bootstrap only. Implementations SHOULD treat HTTPS metadata (Section Section 4.2) as authoritative when available.

### 4.2. HTTPS Metadata Discovery

An implementation MAY retrieve detailed AIIP metadata from the HTTPS well-known resource /.well-known/aiip on the authority host (see [RFC8615]). The retrieved document can provide resolver endpoints, verification keys or key references, and policy signals required to validate invocation metadata.

The well-known document MUST be fetched over HTTPS and its authenticity and integrity MUST be validated using standard HTTPS server authentication.

If both DNS bootstrap discovery and HTTPS metadata discovery are available, implementations SHOULD use the DNS record only to locate the HTTPS metadata.

### 4.3. Delegation Statements

Discovery metadata MAY include delegation statements allowing an administrative domain to delegate subsets of its AIIP namespace or specific capabilities to other parties. Clients and resolvers MUST validate delegation chains prior to accepting invocation metadata derived from delegated authority.

## 5. Native Envelope Framing

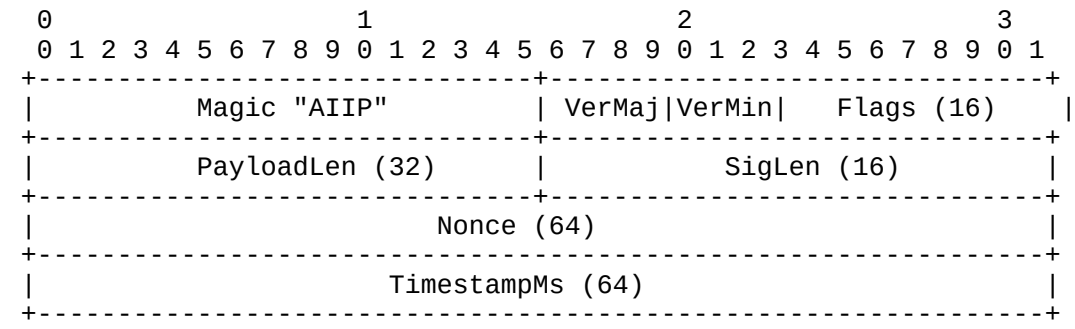
AIIP invocations are carried in a byte-framed envelope suitable for native machine-to-machine transports. The wire format defined here is transport-agnostic: authenticated transports provide confidentiality and channel security, while the AIIP envelope provides a stable framing and replay-resistant metadata.

### 5.1. Byte Order

All multi-octet integer fields are encoded in network byte order (big-endian). Unless otherwise specified, fields are unsigned.

### 5.2. 32-Byte Header

An AIIP message is encoded as: Wire = Header(32) || Payload(PayloadLen) || Signature(SigLen). The header is exactly 32 octets.




```
Bytes:  0..3   Magic (0x41 49 49 50)
        4      VerMaj (this document: 1)
        5      VerMin (this document: 0)
        6..7   Flags
        8..11  PayloadLen
        12..13 SigLen
        14..15 Rsv16 (MUST be 0)
        16..23 Nonce
        24..31 TimestampMs
```

Figure 1: AIIP Header Layout (32 bytes)

No reserved fields exist beyond the fixed 32-byte header; the signature bytes follow the payload.

Flags is a 16-bit bitmask. Bit 0 (0x0001) indicates that the requester requires a receipt in the response. Other bits are reserved for future use.

Example header bytes (illustrative only):

```
41 49 49 50  01 00  00 01  00 00 00 9C  00 40  00 00
2F 7A 13 44  9B 02 5D 8A  00 00 01 93  8F 4C 3A 10
```

Payload is UTF-8 JSON. Binary fields (when needed) are represented using base64url strings.

## 6. Execution Receipts

Each successful invocation produces an execution receipt that can be verified independently of transport intermediaries. Receipts bind the invocation identifier, relevant policy context, and execution outcome.

This document illustrates a JWS-based receipt container ([RFC7515]). The specific receipt schema is expected to evolve with implementation experience.

Illustrative receipt container:

```
{
  "protected": "base64url(jws-protected-header)",
  "payload": {
    "txid": "b7f7cfd2-6c38-4d7a-8e5a-6f1e1d9e0a10",
    "ts": 1734149100000,
    "result": { "status": 200, "data_commitment": "base64url(hash)" },
    "policy": { "max_speed_mps": 0.5, "forbidden_zones": ["Z1", "Z2"] },
    "attestation": { "tee": "base64url(attestation-evidence)" }
  },
  "signature": "base64url(jws-signature)"
}
```

Receipt structure sketch (informative, ASN.1-like):

```
AIIP-Receipt ::= SEQUENCE {
  txid            UUID,
  ts              INTEGER,            -- milliseconds since epoch
  resultCommit    OCTET STRING,       -- hash/commitment
  policy          SEQUENCE OF UTF8String OPTIONAL,
  attestation     OCTET STRING OPTIONAL,
  sigAlg          OBJECT IDENTIFIER,
  signature       OCTET STRING
}
```

## 7. Use Case: Autonomous Delegation

AIIP enables explicit delegation of real-world actions to autonomous agents without disclosure of user credentials and without interactive authorization at execution time. A user may delegate a capability (for example, aiip://user.phone/call) to an AI agent once, under defined policy constraints.

After delegation, any authorized AI agent may invoke the delegated capability directly using AIIP. Invocation does not require access to passwords, OAuth tokens, or user sessions. Each execution produces a verifiable receipt binding the user's authorization, the invoked action, and the execution outcome.

Example invocation:

```
aiip://user.phone/call?to=+15551234567&text=I'm+late
```

The resulting execution receipt provides non-repudiable evidence that the user authorized the action and that it was executed as requested. This delegation model enables autonomous agents to perform actions such as calls, messages, device control, or payments without exposing credentials or requiring repeated user interaction.

## 8. HTTPS Gateway Profile

An HTTPS gateway translates Web-based requests into AIIP invocations and returns results with receipts to human-facing clients.

The HTTPS gateway is a compatibility layer for human operators and MUST NOT be required for autonomous or machine-to-machine AIIP invocation.

## 9. Security Considerations

AIIP implementations MUST authenticate peers via the selected transport binding, prevent replay, and ensure integrity of invocation and receipt data.

Execution receipts provide non-repudiable evidence of authorization and execution, suitable for audit, dispute resolution, and legal review.

## 10. Privacy Considerations

Implementations SHOULD minimize retained metadata and avoid correlating invocations beyond what is required for verification. Delegations SHOULD be scoped minimally to avoid over-exposure of capabilities or metadata.

## 11. IANA Considerations

This document makes no IANA requests. Future revisions may request registration of well-known resources used for AIIP discovery.

## 12. Normative References

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

[RFC7515] Jones, M.B., "JSON Web Signature (JWS)", RFC 7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

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

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

Author’s Address

Aram Sogomonian
Artificial Intelligence Internet Foundation (AIIF)
United States of America
Email: aiinternetfoundation@icloud.com
(mailto:aiinternetfoundation@icloud.com)

Author’s Address

Aram Sogomonian
Artificial Intelligence Internet Foundation (AIIF)
United States of America
Email: aiinternetfoundation@icloud.com