Internet-Draft
Intended status: Informational
Expires: 15 July 2026

Ioannis Vandoulas
January 2026

# Agent Interaction & Delegation Protocol (AIDP)
draft-vandoulas-aidp-02

## Abstract

This document specifies the Agent Interaction & Delegation Protocol (AIDP), a control-plane protocol for secure, auditable, and interoperable software agents. AIDP defines standardized mechanisms for expressing intent, enforcing authority, delegating capabilities, executing actions, and binding execution results to agent reasoning across heterogeneous systems and administrative domains.

## 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."

## License Notice



## Table of Contents

# 1. Introduction

## 1.1 Motivation

Recent advances in Large Language Models (LLMs) have enabled the construction of software agents capable of complex reasoning, planning, and adaptive behavior. While such agents increasingly participate in real-world computational processes— interacting with servers, services, and other agents— there exists no standardized protocol governing how agents express intent, exercise authority, delegate capabilities, or bind execution outcomes to their internal reasoning.

Current implementations rely on ad-hoc mechanisms that lack formal identity, revocation semantics, delegation safety, and cross-domain trust guarantees. This absence of standardization creates systemic risks, including privilege escalation, confused-deputy vulnerabilities, audit failures, and uncontrolled delegation chains.

The Agent Interaction & Delegation Protocol (AIDP) defines a formal, interoperable framework for agent-based interaction, enabling secure, auditable, and revocable agency across heterogeneous systems and administrative domains.

## 1.2 Goals

AIDP defines a protocol that:

- Enables agents to express verifiable intent toward external systems.
- Provides formal identity and authority models for agents.
- Supports constrained, revocable delegation across trust boundaries.
- Binds execution results to originating intent.
- Enables deterministic, auditable agent control loops.
- Remains transport-agnostic and model-agnostic.

## 1.3 Non-Goals

AIDP does not specify:

- Specific cryptographic algorithms.
- Concrete serialization formats.
- Transport protocols.
- LLM architectures or prompting strategies.

These concerns are explicitly layered beneath the AIDP control plane.

# 2. Terminology and Conventions

## 2.1 Definitions

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

Agent:
An autonomous software entity capable of producing Intent Envelopes and processing Observations.

Intent Envelope:
A protocol message expressing a requested action, including its governance and security context.

Authority:
A verifiable capability granting an Agent permission to perform specific actions under defined constraints.

Delegation:
The controlled transfer of a subset of authority from one Agent to another.
Execution Boundary:
The component responsible for enforcing authorization, executing actions, and producing O
bservations.
Observation:
A protocol message binding the outcome of execution to the originating Intent Envelope.
Trust Boundary:
A boundary across which identity, authority, and delegation MUST be revalidated.

## 3. Architecture Overview

### 3.1 System Components

An AIDP-compliant system consists of the following logical components:

- Reasoning Engine: Produces Intent Envelopes and consumes Observations.
- Agent Runtime: Hosts the Reasoning Engine and manages agent state.
- Execution Boundary: Enforces authority, executes actions, and emits Observations.

- Authority Service: Issues, validates, and revokes authority objects.
- Observation Channel: Delivers bound execution outcomes to agents.

These components MAY be co-located or distributed.

### 3.2 Layered Model

AIDP defines a layered architecture:

| Layer | Description |
| --- | --- |
| Agency Control | Intent, Delegation, Authority |
| Trust | Identity, Validation, Revocation |
| Execution | Action dispatch and enforcement |
| Transport | Delivery mechanism (out of scope) |

Each layer MUST operate independently of specific transport or serialization technologies
.

### 3.3 Control Loop

The normative agent control loop is:
A.IntentEnvelope B.Authority Validation C.Execution D.Observation E.Reasoning F.Next Inte
ntEnvelope
An Agent MAY have multiple outstanding Intent Envelopes at any given
time.

An Agent MUST NOT generate a new Intent Envelope whose semantics
causally depend on the outcome of a prior outstanding Intent Envelope
unless the corresponding Observation has been received and validated.

This constraint applies only to causally dependent intents and does not
prohibit parallel or independent intent dispatch.

## 4. Relation to Existing Work

This section describes how AIDP relates to existing work in the areas
of authorization, identity management, and capability-based security.
The intent is to clarify scope and positioning, and to avoid
ambiguity regarding overlap with prior protocols.

### 4.1 Grant Negotiation and Authorization Protocol (GNAP)

AIDP shares architectural similarities with GNAP in that both define
a control-plane interaction model based on structured requests and
policy evaluation, rather than static, scope-based authorization.

GNAP focuses on negotiating authorization grants for access to
protected resources among clients, authorization servers, and
resource servers. AIDP, by contrast, addresses delegated agentic
action, where autonomous or semi-autonomous agents express intent,

exercise constrained authority, and receive execution-bound observations.

Execution semantics, post-execution observation, reasoning failure handling, and runtime accountability are explicitly within scope for AIDP and are outside the scope of GNAP. GNAP may therefore be used as an authorization mechanism within an AIDP-compliant system, but it does not define the full interaction lifecycle specified by AIDP.

### 4.2 OAuth 2.0 Rich Authorization Requests (RAR)

OAuth 2.0 Rich Authorization Requests define a mechanism for expressing fine-grained authorization requirements beyond simple scope strings.

AIDP adopts a similar principle by supporting structured expressions of authority, but applies it in a broader context. In AIDP, such expressions are embedded within Intent Envelopes and are evaluated in conjunction with agent identity, delegation chains, and execution constraints.

Unlike OAuth RAR, AIDP does not assume a client-resource server architecture and is not limited to access control semantics.

### 4.3 Capability-Based Authorization (ZCAPs and UCAN)

AIDP is conceptually aligned with capability-based authorization systems such as ZCAPs and UCAN, particularly with respect to explicit capabilities and delegation chains.

As in these systems, authority in AIDP is represented as a transferable object that can be delegated across multiple actors without reliance on a centralized authorization server.

AIDP differs in that capability delegation is integrated into an agent interaction and execution lifecycle, where capabilities are evaluated as part of intent processing and execution rather than solely as access credentials.

### 4.4 Identity Management and SCIM

AIDP assumes the existence of stable and verifiable agent identities and is compatible with identity provisioning and lifecycle management systems such as the System for Cross-domain Identity Management (SCIM).

SCIM defines mechanisms for identity representation, provisioning, and synchronization across administrative domains. AIDP builds upon such identity systems to specify how authority is delegated and exercised during agent interaction and execution.

### 4.5 Summary

AIDP does not replace existing authorization or identity protocols. Instead, it composes concepts from prior work into a unified control-plane protocol for agent interaction, delegation, execution, and accountability.

Existing mechanisms such as GNAP, OAuth 2.0 Rich Authorization Requests, capability-based authorization systems, and SCIM may be used as complementary components within an AIDP-compliant architecture.

## 5. Agent Identity Model

5.1 Identity Requirements
Every Agent participating in AIDP MUST possess a verifiable identity.
An Agent Identity MUST provide the following properties:
- Uniqueness: A globally unique identifier.
- Issuance: Identification of the issuing authority.
- Lifetime: Explicit validity interval.
- Revocability: Capability to be invalidated independently of expiration.
- Referencability: Stable reference suitable for inclusion in protocol messages.
An Agent Identity MUST NOT be implicitly inferred from transport, host, or network contex
t.

5.2 Identity Resolution and Validation
Identity references MUST be resolved and validated at every Trust
Boundary crossing.

For cross-domain interoperability, an identity reference MUST conform
to one of the following profiles:

(A) Dereferenceable Identity Profile:
    The identity_ref MUST be resolvable by the verifying domain via a
    deterministic and reachable resolution mechanism defined by the
    issuer.

(B) Self-Contained Identity Profile:
    The identity_ref MUST be accompanied by cryptographically verifiable
    identity assertions sufficient to establish authenticity without
    issuer reachability.

If neither profile can be satisfied, the receiving domain MUST reject
the request or apply fail-closed behavior according to its risk policy.

6. Authority & Capability Model
6.1 Authority Objects
Authority is represented as a first-class object, hereafter termed a Capability.
A Capability MUST contain:
- Subject: Agent Identity to which authority is granted.
- Action: Permitted operation(s).
- Resource: Target resource(s).
- Constraints: Scope, quantity, temporal, and contextual limits.
- Issuer: Authority that issued the capability.
- Revocation Pointer: Mechanism for revocation verification.
Capabilities MUST be evaluated independently of any agent claims.

6.2 Capability Delegation
Agents MAY delegate authority only by issuing new Capabilities with strictly reduced scop
e.
Delegation MUST satisfy:
- Subsetting Rule: Delegated authority MUST NOT exceed original authority.
- Traceability: Each delegated capability MUST reference its parent.
- Revocability: Revocation of a parent capability MUST invalidate all derived capab
ilities.
Delegation chains MUST be preserved and transmitted with each Intent Envelope.

6.3 Revocation Semantics
Revocation MUST be supported for:
- Agent Identity
- Capabilities
- Delegation chains
Revocation MUST take effect immediately upon verification and MUST NOT depend on token ex
piration alone.

7. Intent Envelope

7.1 Envelope Structure
An Intent Envelope is the fundamental protocol message produced by an Agent.
Each Intent Envelope MUST contain:

- envelope_id: Globally unique, non-reusable identifier.
- timestamp: Time of issuance.
- actor_ref: Reference to Agent Identity.
- authority_ref: Reference to Capability.
- intent_body: Declared action, target, and parameters.
- constraints: Execution limitations.
- delegation_chain: Ordered list of delegated Capabilities.
- observability_hooks: Declaration of required observation behavior.

7.2 Envelope Validation Rules
Upon receipt, the Execution Boundary MUST verify:
1. Actor identity validity.
2. Capability authenticity and scope.
3. Delegation chain integrity.
4. Revocation status for all identities and capabilities.
5. Constraint satisfaction.
6. Non-reuse of envelope_id (replay protection).
Failure of any validation step MUST abort execution.

7.3 Authority Separation Principle
Agents MUST NOT modify:
- actor_ref
- authority_ref
- delegation_chain
These elements are managed exclusively by the Authority Service and Execution Boundary.

8. Execution Model
8.1 Execution Boundary
The Execution Boundary is the sole component authorized to perform actions on behalf of a
n Agent.
The Execution Boundary MUST:
- Enforce identity and authority validation.
- Apply constraint verification.
- Execute the requested action.
- Emit a bound Observation.
The Execution Boundary MUST NOT accept any action request outside of a validated Intent E
nvelope.

8.2 Authorization & Enforcement Flow
Normative execution flow:
1. Receive Intent Envelope.
2. Validate identity.
3. Validate authority and delegation chain.
4. Verify revocation status.
5. Enforce constraints.
6. Execute action.
7. Generate Observation.
8. Deliver Observation via Observation Channel.
Any failure in steps 25 MUST result in rejection.

9. Observation & Feedback Binding

9.1 Observation Structure
Each Observation MUST contain:
- envelope_id
- execution_id
- status
- result
- side_effects
- timestamp
- attestation

9.2 Binding Requirements
Reasoning progression within an Agent MUST be gated on validated
Observations for all intents upon which subsequent reasoning steps
depend.

Independent intents MAY be issued and processed concurrently, provided that their resulting Observations are correctly bound to their respective envelope_id values.

9.3 Replay Protection & Determinism
The tuple (envelope_id, execution_id, timestamp) MUST provide replay protection.
Agent reasoning loops MUST be deterministic with respect to received Observations.

10. Multi-Agent Interaction

10.1 Trust Boundaries
At every Trust Boundary crossing, the receiving domain MUST independently validate:
- Agent Identity
- Capability authenticity
- Delegation chain
- Revocation status

No implicit trust MAY cross a Trust Boundary.

10.2 Cross-Domain Delegation
Delegation across domains MUST follow:
- Capability subsetting rules
- Explicit delegation chain construction
- Independent revocation enforcement

Shared secrets or inherited credentials MUST NOT be used as delegation mechanisms.

10.3 Agent-to-Agent Protocol Flow
Normative multi-agent interaction:
Agent A → Intent Envelope → Trust Boundary → Agent B
Agent B → Execution Boundary → Observation → Agent A
Every stage MUST preserve auditability and authority constraints.

11. Security Considerations
11.1 Scope
This section defines the threat model for AIDP systems, including attacker capabilities, assets, trust boundaries, and normative mitigations. Unless explicitly stated otherwise, all requirements in this section are normative.

11.2 Assets
An AIDP deployment MUST treat the following as protected assets:
- AIDP Identity Material: Agent Identity objects and their validation artifacts.
- Capabilities: Authority objects, including parent/derived relationships and constraint sets.
- Delegation Chains: Ordered proof of authority derivation across one or more domains.
- Intent Envelopes: Including envelope_id, actor_ref, authority_ref, constraints, and hooks.
- Observations: Including attestation and bindings to prior intents.
- Audit Trail: Immutable records sufficient to reconstruct intent-execution-observation causality.
- Execution Targets: External resources/services whose state can be modified by actions.

11.3 Trust Boundaries and Security Domains
AIDP recognizes the following boundaries:
- TB-1: Agent Runtime  Execution Boundary

The Agent Runtime is untrusted for enforcement. The Execution Boundary is the policy enforcement point.
- TB-2: Domain A  Domain B

Cross-domain interactions MUST assume zero implicit trust.
- TB-3: Authority Service  Verifiers

Capability issuance and revocation status MUST be independently verifiable.
- TB-4: Observation Channel  Agent

Observation delivery MUST prevent substitution, replay, and confusion of results.
Crossing any Trust Boundary MUST trigger independent identity, authority, chain, and revocation validation.

11.4 Attacker Model
Attackers MAY possess one or more of the following capabilities:
- Network Adversary: Can observe, replay, delay, or reorder messages across a transport.
- Malicious Agent: Can generate arbitrary Intent Envelopes and attempt to induce execution.
- Compromised Agent Runtime: Can manipulate prompts, local state, tool outputs, and message ordering.
- Compromised Execution Target: Can respond with misleading outputs or partial failures.
- Insider: Possesses legitimate credentials or access in one domain.
- Supply Chain/Implementation Bugs: Leverages parser ambiguities, canonicalization issues, and validation gaps.
AIDP MUST remain secure against malicious agents and network adversaries. Compromise of the Execution Boundary is out of scope (if it fails, enforcement fails), but audit and containment SHOULD still limit blast radius.

11.5 Security Goals
An AIDP system MUST provide:
- G-1 Least Privilege: Capabilities constrain what may be executed.
- G-2 Non-Repudiation of Control Flow: Audit can reconstruct who requested what and what executed.
- G-3 Replay Resistance: Intents and observations cannot be reused to re-trigger actions.
- G-4 Delegation Safety: Delegation cannot amplify privileges; revocation propagates.
- G-5 Outcome Integrity: Observations are bound to the originating intent and execution.
- G-6 Compromise Containment: A compromised agent cannot exceed granted authority.
- G-7 Cross-Domain Robustness: No implicit trust crosses domain boundaries.

11.6 Threats and Mitigations

11.6.1 Confused Deputy
Threat: A less-privileged agent induces a more-privileged system/component to perform actions outside the attacker’s authority.
Mitigations (normative):
- The Execution Boundary MUST authorize based solely on validated authority_ref and delegation_chain, not on agent-provided explanations or natural-language intent.
- The Execution Boundary MUST verify that the capability subject matches actor_ref (or matches a valid delegation subject in the chain).
- Capabilities MUST encode the target resource(s) and permitted action(s) explicitly; wildcards SHOULD be avoided or constrained.

11.6.2 Privilege Escalation via Delegation
Threat: An agent delegates authority beyond what it possesses (scope amplification), or a receiver misinterprets delegation semantics.
Mitigations:
- Delegated capabilities MUST be strict subsets of parent capabilities (Subsetting Rule).
- Delegation chains MUST be validated end-to-end at execution time.
- Receivers MUST reject any chain that contains an invalid or unverifiable link.
- Revocation of a parent capability MUST invalidate all derived capabilities.

11.6.3 Replay of Intent Envelopes
Threat: Reuse of an Intent Envelope to trigger repeated execution (financial transfers, destructive actions).
Mitigations:
- envelope_id MUST be globally unique and MUST NOT be accepted more than once per relevant replay window.
- Execution Boundaries MUST maintain replay state appropriate to the risk profile of the action class.
- Constraints SHOULD include explicit idempotency requirements for side-effectful actions.
- When an Execution Boundary (EB) receives an Intent Envelope (IE) whose

envelope_id has already been processed, the EB MUST treat the request as a replay and MUST NOT re-execute the intent.

If a corresponding Observation (OB) has already been produced and is still within the EB's observation retention window, the EB SHOULD return the previously generated Observation or provide a deterministic mechanism for the Agent to retrieve it (e.g., via an observation retrieval endpoint).

A replayed envelope_id MUST NOT result in a new execution or a semantically distinct outcome.

Whether a replayed Intent results in an inline Observation, a retrieval pointer, or a REPLAY_DETECTED Problem Details response is deployment- and transport-binding-specific. In all cases, the EB MUST NOT re-execute the intent.

11.6.4 Replay or Substitution of Observations
Threat: An attacker replays a prior Observation or substitutes a different Observation to steer reasoning.
Mitigations:
- Observations MUST include envelope_id and execution_id.
- Observations MUST carry an attestation from the Execution Boundary.
- Agents MUST reject Observations that do not match an outstanding Intent Envelope.

- Agents MUST NOT progress reasoning for an intent until a valid bound Observation is received.

11.6.5 Parameter Confusion and Canonicalization Attacks
Threat: Ambiguities in serialization or parsing allow an attacker to smuggle parameters or exploit differences between validators and executors.
Mitigations:
- Implementations MUST define a canonical representation for signing/verifying (even if encoding is out of scope, the "to-be-verified" form MUST be unambiguous).
- Validators and executors MUST share identical parsing and validation logic.
- Unknown fields in security-relevant sections MUST be rejected or handled by strict extension rules.

11.6.6 Time-of-Check / Time-of-Use (TOCTOU) on Revocation
Threat: A capability is valid at validation time but revoked before execution completes; or cache staleness causes use of revoked authority.
Mitigations:
- Execution Boundaries MUST check revocation status at execution time, not solely at receipt time, for high-risk actions.
- Revocation mechanisms MUST support near-immediate effect.
- Deployments SHOULD define risk tiers: low-risk actions MAY tolerate bounded staleness; high-risk actions MUST NOT.

11.6.7 Compromised Agent Runtime / Prompt Injection
Threat: The agent runtime is manipulated so the reasoning engine emits malicious intents or misinterprets observations.
Mitigations:
- Enforcement MUST occur at the Execution Boundary, not in the agent.
- Capabilities MUST be constrained such that unintended intents have limited impact.
- Observation binding MUST prevent fabricated "success" states.
- High-risk capabilities SHOULD require additional policy controls (e.g., approvals) outside agent reasoning.

11.6.8 Cross-Domain Identity and Issuer Spoofing
Threat: An attacker forges identities/capabilities from an untrusted issuer or exploits issuer confusion.
Mitigations:
- Domains MUST maintain explicit trust anchors for issuers.
- All identities and capabilities MUST include issuer identification.

- If issuer validation fails, the intent MUST be rejected.

11.6.9 Audit Trail Gaps and Non-Attribution
Threat: Missing logs prevent reconstruction of actions, enabling undetected abuse.
Mitigations:
- Execution Boundaries MUST log Intent Envelope receipt, authorization decision, execution outcome, and Observation emission.
- Audit records MUST include envelope_id, execution_id, actor_ref, authority_ref, and a digest of intent_body.

11.6.10 Denial of Service
Threat: Flooding the Execution Boundary with envelopes or forcing expensive validation.
Mitigations:
- Implementations SHOULD rate-limit by actor/issuer.
- Validation SHOULD be staged (cheap rejection first).
- Deployments MAY require preflight checks for costly operations.

11.7 Residual Risk and Out-of-Scope Conditions
- If the Execution Boundary is fully compromised, authorization cannot be trusted. Deployments SHOULD still maintain independent audit and anomaly detection.
- If the Execution Target returns misleading results, AIDP can ensure binding/attestation but cannot guarantee target truthfulness beyond attestation scope.
- Human approvals, legal policies, and organizational controls are deployment concerns and are not specified by AIDP, though AIDP enables their enforcement hooks.

11.8 Security Requirements Summary
Implementations claiming AIDP compliance MUST, at minimum:
- Validate identity and issuer trust anchors at each Trust Boundary.
- Enforce capabilities and delegation chain subsetting at execution time.
- Support revocation with near-immediate effect for high-risk actions.
- Provide replay protection for intents and bound observations.
- Produce attestable observations linked to intents and executions.
- Maintain auditable records sufficient for forensic reconstruction.

12. Governance & Compliance Considerations
AIDP deployments SHOULD provide:
- Comprehensive audit trails.
- Policy-driven revocation procedures.
- Capability lifecycle management.
- Regulatory compliance support.
Governance mechanisms MUST remain external to agent reasoning processes.

13. Extensibility Model
AIDP is intentionally extensible.
Extensions MUST:
- Preserve all normative validation requirements.
- Avoid weakening identity, authority, or revocation semantics.
- Declare compatibility with base AIDP semantics.

14. IANA Considerations
This document does not require any IANA actions at this time.

15. Acknowledgements
The authors acknowledge the emerging agent systems community whose practical challenges motivated the creation of this protocol.

16. References

16.1 Normative references

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

16.2 Informative References

[GNAP] Hardt, D., Parecki, A., and T. Lodderstedt,

"Grant Negotiation and Authorization Protocol",
Internet-Draft, work in progress,
draft-ietf-gnap-core.

[RAR] Lodderstedt, T., McGloin, M., and P. Hunt,
"OAuth 2.0 Rich Authorization Requests",
RFC 9396, August 2023.

[ZCAP] Sporny, M., Longley, D., and D. Chadwick,
"Authorization Capabilities for Linked Data",
W3C Recommendation,
https://www.w3.org/TR/vc-data-model/.

[UCAN] Fission Codes,
"User Controlled Authorization Networks (UCAN)",
https://ucan.xyz/.

[SCIM] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E.,
and C. Mortimore,
"System for Cross-domain Identity Management: Core Schema",
RFC 7643, September 2015.

## 17. Wire Format

### 17.1 Serialization Families

AIDP messages MUST be representable in at least one of the following serialization families:

- AIDP-JS (JSON): Human-readable encoding for debugging, documentation, and early adoption.
- AIDP-CB (CBOR): Binary encoding for production deployments with strict canonicalization.

An implementation MAY support additional encodings, provided that:

1. The semantic model is preserved exactly, and
2. A deterministic canonical signing input (Section 16.3) is defined.

### 17.2 Message Types

AIDP defines three primary wire messages:

- Intent Envelope (IE): request to execute a governed action.
- Observation (OB): bound result of an execution.
- Problem Details (PD): structured rejection / error.

Each message MUST carry:

- aidp_version
- msg_type
- payload

### 17.3 Canonical Signing Input

To prevent canonicalization attacks, AIDP requires an unambiguous signing input.
Canonical AIDP-JS encoding requires:

- UTF-8 encoding
- Deterministic lexicographic ordering of object members
- No duplicate object member names
- Deterministic number and string representations

JSON messages containing duplicate object member names MUST be rejected.
This rejection MUST occur during parsing, prior to schema validation or canonicalization.

- For AIDP-CB, the signing input MUST use:
- CBOR canonical encoding (deterministic)

The signing input MUST be computed over the payload object only (not transport metadata).
The exact canonicalization profile MUST be declared as canon.

### 17.4 Attestations and Proofs

AIDP supports two proof-bearing fields:

- proof: signature or MAC over canonical signing input

- proof_chain: optional chain of proofs for delegation links

Intent Envelopes SHOULD be signed by the Agent Runtime when any non-trivial authority is exercised. Observations MUST be attested by the Execution Boundary.

17.5 Common Header

All AIDP wire messages MUST include:

- aidp_version (string, e.g. "1.0-draft")
- msg_type (string: "IE" | "OB" | "PD")
- canon (string: canonicalization profile identifier)
- payload (object)
- proof (object, OPTIONAL unless required by local policy)

18. Wire Schemas

Below are normative field sets. Implementations MUST reject unknown fields in security-critical sub-objects unless explicitly permitted by the Extensibility Model.

18.1 Intent Envelope Payload (IE)

Required:

- envelope_id (string/UUID)
- timestamp (RFC3339 string)
- actor_ref (object)
- authority_ref (object)
- intent_body (object)
- constraints (object)
- delegation_chain (array, MAY be empty)
- observability_hooks (object)

18.1.1 actor_ref

- agent_id (string)
- issuer (string)
- identity_ref (string) — dereferenceable URI/URN-like reference (transport-agnostic)

18.1.2 authority_ref

- cap_id (string)
- issuer (string)
- cap_ref (string) — dereferenceable reference
- rev_ref (string) — revocation pointer reference

18.1.3 intent_body

- action (string)
- target (object)
- parameters (object)

target MUST include at least:

- resource (string)
- domain (string) — security domain identifier

18.1.4 constraints

Constraints SHOULD include (as applicable):

- not_before / not_after (RFC3339)
- max_cost (number)
- max_uses (integer)
- risk_tier (string: "low"|"med"|"high", OPTIONAL but RECOMMENDED)
- idempotency_key (string, OPTIONAL)

The idempotency_key is OPTIONAL and is intended for application-level idempotency across transport retries or envelope re-issuance scenarios.

Protocol-level replay protection and execution idempotency are primarily enforced via the envelope_id. Implementations MUST NOT require the presence of idempotency_key to provide replay safety or result recovery.

18.1.5 delegation_chain

Each element MUST include:
- cap_id
- issuer
- cap_ref
- parent_cap_id (string)
- rev_ref
- link_proof (object) — proof binding child to parent

18.1.6 observability_hooks
The return_to field specifies the destination for Observation delivery.

The specified endpoint MUST be pre-registered, pre-approved, or otherwise explicitly authorized for the identity identified by actor_ref.

Execution Boundaries MUST NOT deliver Observations to arbitrary or unverified endpoints supplied dynamically by an Agent.

18.2 Observation Payload (OB)
Required:
- envelope_id
- execution_id (string)
- timestamp (RFC3339)
- status (string: accepted|rejected|executed|failed|partially_executed)
- result (object, MAY be empty)
- side_effects (array, MAY be empty)
- attestation (object)

18.2.1 attestation
- boundary_id (string)
- issuer (string)
- attest_profile (string)
- decision (string: authorized|not_authorized|constraint_violation|invalid_chain|revoked|replay)
- policy_digest (string) — hash/reference to policy version used
- evidence (object, OPTIONAL) — minimal additional evidence (non-sensitive by default)

Observations MUST be signed/attested by the Execution Boundary (i.e., proof REQUIRED for OB by default).

18.2.x execution_advice (OPTIONAL)

The execution_advice object provides non-binding guidance to the Agent Runtime regarding retry, backoff, or revision expectations following an Observation that does not represent successful execution.

The execution_advice object MAY include:
- retriable (boolean):
  Indicates whether retrying the same intent MAY succeed.
- retry_after_sec (integer):
  Suggested minimum delay before attempting a retry.
- reason_code (string):
  Machine-readable classification of the failure condition
  (e.g., CONSTRAINT_VIOLATION, BUDGET_EXCEEDED, RATE_LIMITED,
  POLICY_REFUSED, NOT_AUTHORIZED).

18.3 Problem Details Payload (PD)
Required:
- envelope_id (if applicable)
- timestamp
- error_code (string)
- error_message (string)
- details (object, OPTIONAL)

Recommended error_code values:
- INVALID_IDENTITY
- UNTRUSTED_ISSUER

```
-       REVOKED
-       INVALID_CAPABILITY
-       INVALID_DELEGATION_CHAIN
-       CONSTRAINT_VIOLATION
-       REPLAY_DETECTED
-       MALFORMED_MESSAGE
-       UNSUPPORTED_VERSION

Problem Details messages MAY include an execution_advice object
with the same semantics as defined for Observation messages.

When present, execution_advice applies to the failed Intent Envelope
identified by envelope_id.

19. Example Wire Messages (AIDP-JS)

19.1 Intent Envelope (single-agent → server)
{
  "aidp_version": "1.0-draft",
  "msg_type": "IE",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:00+02:00",
    "actor_ref": {
      "agent_id": "agent:alpha",
      "issuer": "did:example:issuerA",
      "identity_ref": "urn:aidp:id:issuerA:agent-alpha"
    },
    "authority_ref": {
      "cap_id": "cap:alpha:pay-v1",
      "issuer": "did:example:authA",
      "cap_ref": "urn:aidp:cap:authA:cap-alpha-pay-v1",
      "rev_ref": "urn:aidp:rev:authA:list-01"
    },
    "intent_body": {
      "action": "payment.create",
      "target": { "domain": "svc:payments", "resource": "acct:merchant-123" },
      "parameters": { "amount": 50, "currency": "EUR", "memo": "invoice-8841" }
    },
    "constraints": {
      "not_before": "2026-01-13T09:14:00+02:00",
      "not_after": "2026-01-13T09:19:00+02:00",
      "max_uses": 1,
      "risk_tier": "high",
      "idempotency_key": "idem-3d7f0d"
    },
    "delegation_chain": [],
    "observability_hooks": {
      "delivery_mode": "push",
      "return_to": { "type": "inbox", "ref": "urn:aidp:inbox:agent:alpha" },
      "required_fields": ["envelope_id", "execution_id", "status", "attestation"],
      "timeout_sec": 30
    }
  },
  "proof": {
    "alg": "ed25519",
    "kid": "key:agent-alpha-1",
    "sig": "BASE64URL(...)"
  }
}

19.2 Observation (Execution Boundary → agent)
{
  "aidp_version": "1.0-draft",
  "msg_type": "OB",
```

```
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "execution_id": "exec-9c2d1a",
    "timestamp": "2026-01-13T09:14:02+02:00",
    "status": "executed",
    "result": { "payment_id": "pay_7712", "state": "captured" },
    "side_effects": [
      { "resource": "acct:merchant-123", "delta": { "captured_eur": 50 } }
    ],
    "attestation": {
      "boundary_id": "boundary:payments-gw-1",
      "issuer": "did:example:paymentsDomain",
      "attest_profile": "AIDP-OB-Attest1",
      "decision": "authorized",
      "policy_digest": "sha256:5d3a...e91"
    }
  },
  "proof": {
    "alg": "ed25519",
    "kid": "key:boundary-payments-1",
    "sig": "BASE64URL(...)"
  }
}
```


19.3 Problem Details (rejection example)


```
{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:01+02:00",
    "error_code": "REVOKED",
    "error_message": "Capability or identity has been revoked.",
    "details": { "rev_ref": "urn:aidp:rev:authA:list-01", "cap_id": "cap:alpha:pay-v1" }
  },
  "proof": {
    "alg": "ed25519",
    "kid": "key:boundary-payments-1",
    "sig": "BASE64URL(...)"
  }
}
```


20. Example Flows

20.1 Flow 1 — Single-Agent Action (Agent  Server)

Goal: Agent requests a state-changing action on an external service with bounded authority.

Steps (normative):

1. Agent Runtime obtains/holds a capability authority_ref.
2. Agent produces an Intent Envelope (IE) referencing actor_ref and authority_ref.
3. Execution Boundary validates: identity, issuer trust anchors, capability scope, revocation, constraints, replay.
4. Execution Boundary executes the action on the target service.
5. Execution Boundary emits an Observation (OB) bound to envelope_id and execution_id.
6. Agent MUST NOT proceed for that intent until OB is received and validated.

Key security properties:

- Enforcement at Execution Boundary, not the LLM
- Replay resistance via envelope_id
- Outcome integrity via OB attestation

20.2 Flow 2 — Delegation (Agent A → Agent B → Execution)

Goal: Agent A delegates limited authority to Agent B to perform a specific action.

Steps:

1. Agent A possesses parent capability cap:A0.
2. Authority Service issues delegated capability cap:B1 with:
- parent_cap_id = cap:A0
- scope subset (e.g., only resource=X, only max_uses=1, only within 2 minutes)
3. Agent A sends IE to Agent B with delegation_chain = [cap:B1 link proof ...]
4. Agent B either:
- accepts responsibility and forwards IE to its Execution Boundary, or
- rejects (policy/local risk)
5. Execution Boundary validates entire chain end-to-end (including parent/child link age proofs).
6. Execution produces OB, returned to the return_to hook (Agent A and/or B depending on hook policy).

Key security properties:
- No credential sharing
- Subsetting rule enforced at execution
- Revocation propagates along chain

20.3 Flow 3 — Revocation Mid-Flight (TOCTOU)

Goal: Capability revoked after IE issuance but before execution finalization.

Steps:
1. IE is received and preliminarily validated.
2. Before executing high-risk action, Execution Boundary MUST re-check revocation status (per risk tier/policy).
3. If revoked, execution MUST abort and PD returned with REVOKED.
4. Audit record MUST note decision and revocation evidence used.

Key security properties:
- Near-immediate revocation effect
- Clear failure semantics and auditability

20.4 Flow 4 — Observation Substitution Attempt

Goal: Network attacker tries to replay an old OB to trick the agent.

Steps:
1. Attacker replays OB with mismatched or already-settled envelope_id.
2. Agent validates OB proof and checks if envelope_id is outstanding.
3. Agent MUST reject OB if:
o envelope_id not outstanding, or
o OB proof invalid, or
o attestation issuer untrusted
4. Agent continues waiting or initiates recovery (out of scope).

Key security properties:
- OB binding prevents confusion
- Agent loop does not advance without valid OB

20.5 Flow 5 — Multi-Domain Boundary Crossing

Goal: Agent in Domain A requests execution in Domain B.

Steps:
1. Domain B applies TB rules: zero implicit trust.
2. Domain B validates issuer trust anchors for identities and capabilities.
3. Domain B validates revocation pointers accessible/acceptable under its policy.
4. If any issuer untrusted, reject with UNTRUSTED_ISSUER.

## 21. Reference Architecture

21.1 Overview

This section defines a reference architecture for AIDP deployments. The architecture is model-agnostic (LLM vendor-independent) and transport-agnostic. It separates reasoning from enforcement and binds execution outcomes to originating intents.

AIDP systems MUST implement an Execution Boundary as the enforcement point. Agent reasoning components MUST NOT be relied upon for authorization, revocation, replay prevention, or audit integrity.

21.2 Logical Components

An AIDP deployment consists of the following logical components. Components MAY be co-located or distributed.

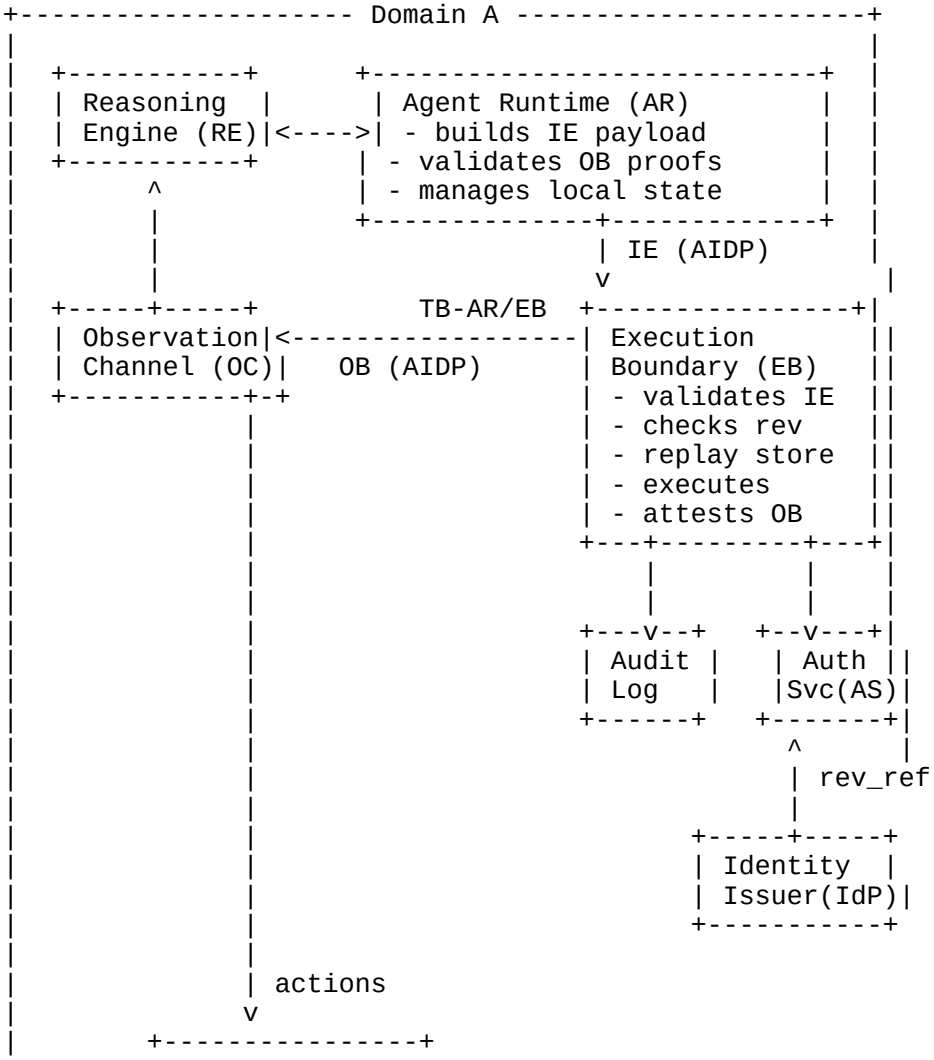
1. Reasoning Engine (RE)

Produces Intent Envelopes and consumes Observations. (Often an LLM, but not required.)

2. Agent Runtime (AR)
Hosts the RE; manages local state, memory, and tool mediation. The AR is not a policy enf
orcement point.
3. Execution Boundary (EB)
Policy enforcement and action execution component. Validates identity/capabilities/delega
tion, checks revocation, prevents replay, executes actions, and attests Observations.
4. Authority Service (AS)
Issues Capabilities, manages lifecycle, and provides revocation status via rev_ref.
5. Identity Provider / Issuer (IdP)
Issues Agent Identity artifacts referenced by actor_ref.
6. Observation Channel (OC)
Delivers Observations to agents via push and/or pull. (e.g., inbox, queue, webhook)
7. Audit Log (AL)
Append-only event record of intent receipt, authorization decisions, execution results, a
nd observation emission.
8. Execution Targets (ET)
External systems acted upon (APIs, services, websites via automation, databases, etc.).

21.3 Trust Boundaries
AIDP defines explicit trust boundaries. Crossing a trust boundary MUST trigger independen
t validation.
- TB-AR/EB: Agent Runtime to Execution Boundary
AR is untrusted for enforcement; EB MUST validate all incoming IEs regardless of origin.
- TB-Domain: Cross-domain boundary
Identity, capabilities, issuer trust, and revocation MUST be revalidated in the receiving
 domain.
- TB-OC: Observation Channel boundary
Observations MUST be attested by EB and verified by the Agent Runtime/RE.

21.4 Reference Architecture Diagram



```
|        | Execution      |
|        | Targets (ET)   |
|        +----------------+
+-------------------------------------------------+
```

21.5 Normative Interfaces
This section defines the conceptual interfaces between components.

21.5.1 IE Submission Interface (AR → EB)
- Input: AIDP Intent Envelope (IE)
- Output: Immediate PD or async OB (or both if policy allows)

EB MUST implement:
- Identity validation for actor_ref
- Capability validation for authority_ref
- Delegation chain validation
- Revocation checks via rev_ref
- Replay detection for envelope_id
- Constraint enforcement

AR MAY implement preflight validation, but EB validation is authoritative.

21.5.2 Authority Resolution Interface (EB  AS)
- EB dereferences cap_ref and rev_ref
- EB verifies:
- capability authenticity
- scope/constraints
- revocation status (including parents in chain)
- issuer trust anchor

AS MUST support:
- capability retrieval (or verification artifacts)
- revocation status lookup with bounded latency for high-risk actions

21.5.3 Identity Validation Interface (EB  IdP)
- EB dereferences identity_ref and validates issuer trust
- EB checks identity revocation status

IdP MUST support:
- identity verification artifacts
- revocation status

21.5.4 Observation Delivery Interface (EB → OC → AR)
- EB emits Observation (OB) with attestation proof
- OC delivers via:
- push (webhook/queue) and/or
- pull (inbox retrieval)

AR MUST verify:
- OB proof (attestation)
- envelope_id matches outstanding intent
- issuer trust anchor for the EB attestation

21.5.5 Audit Interface (EB → AL)
EB MUST record:
- receipt of IE (digest)
- authorization decision and reasons
- execution_id
- status/outcome
- emitted OB digest

AL SHOULD be append-only. Tamper-evidence is RECOMMENDED.

21.6 Data Flow Specifications

21.6.1 Standard Single-Domain Flow
RE -> AR: propose action
AR -> EB: IE(envelope_id, actor_ref, authority_ref, constraints, hooks)
EB -> AS/IdP: validate capability + identity + revocation
EB -> ET: execute action
EB -> AL: write audit events
EB -> OC: emit OB(attested)

OC -> AR: deliver OB
AR -> RE: provide validated OB for next-step reasoning

21.6.2 Cross-Domain Flow (Domain A → Domain B)
Domain A: AR_A -> EB_B (IE)
(crosses TB-Domain)
Domain B: EB_B validates issuers/trust anchors independently
EB_B executes, attests OB
OB delivered back per hooks (possibly to OC_A and/or OC_B)
Receiving domains MUST NOT assume trust in issuer sets unless explicitly configured.

21.7 Policy Enforcement Points
EB is the required enforcement point. The following controls MUST be enforced at EB:
- Revocation checks (risk-tier dependent strictness)
- Replay protection
- Constraint verification (time, uses, scope)
- Delegation chain integrity
- Audit emission
AR/RE controls are advisory only.

21.8 Minimal Deployment Profiles

21.8.1 Minimal Profile (Single Domain)
Required: RE, AR, EB, AS (or embedded capability verifier), OC, AL
Optional: external IdP (can be embedded issuer)

21.8.2 Multi-Domain Profile
Required: EB and trust anchor policy per domain, cross-domain issuer validation, revocation accessibility policy, OC bridging rules.

21.9 Failure Modes and Required Behaviors
- If AS/IdP is unreachable for a high-risk action, EB MUST fail closed (reject) unless local policy explicitly permits bounded staleness.
- If OB delivery fails, EB SHOULD support pull-based retrieval via OC to avoid “lost observations”.
- If replay is detected, EB MUST reject with REPLAY_DETECTED and log the attempt.

22. Concrete Transport Binding: HTTP (AIDP-HTTP)

22.1 Overview
This section specifies a concrete transport binding for AIDP using HTTP. It defines:
- submission of Intent Envelopes (IE) to an Execution Boundary (EB),
- delivery/retrieval of Observations (OB),
- structured error reporting (PD),
- minimal requirements for replay safety and correlation.
This binding is compatible with AIDP-JS (JSON) payloads as specified in Sections 1619. AIDP-CB (CBOR) MAY be supported via content negotiation.

22.2 Resources and Roles
- Client: Agent Runtime (AR) or Agent-facing proxy.
- Server: Execution Boundary (EB).
- Observation Channel: either:
- push: EB calls an AR webhook, or
- pull: AR polls an EB-hosted inbox.

22.3 Media Types
Implementations MUST support JSON.
- IE: application/aidp+json; msg=IE
- OB: application/aidp+json; msg=OB
- PD: application/aidp+json; msg=PD
Servers MAY additionally support CBOR:
- application/aidp+cbor; msg=IE|OB|PD

22.4 Common HTTP Headers
Clients and servers MUST use:
- Content-Type: as above

- Accept: one or more supported AIDP media types
The following headers are RECOMMENDED:
- Idempotency-Key: value of constraints.idempotency_key (when present)
- X-AIDP-Envelope-ID: mirrors payload.envelope_id (for tracing; not authoritative)

22.5 Authorization at HTTP Layer
AIDP authorization is enforced via authority_ref/capabilities at the AIDP layer. However, the HTTP channel itself MUST be authenticated to prevent anonymous flooding and to enable policy decisions.
EB deployments MUST require at least one of:
- mutual TLS (mTLS), or
- OAuth 2.0 Bearer token, or
- another mutually authenticated mechanism.
Important: HTTP-layer auth MUST NOT be treated as sufficient authorization for actions. EB MUST still validate AIDP identity/capability/delegation/revocation per the core spec.

22.6 Endpoint: Submit Intent Envelope
POST /v1/aidp/intents
Request body: an IE message.
Success responses:
- 202 Accepted: intent accepted for processing; OB will be delivered via hooks (push) and/or made available for pull.
- 200 OK: intent executed synchronously and OB is returned in the response body (allowed for low latency operations).
Failure responses:
- 400 Bad Request: malformed AIDP message (PD returned)
- 401 Unauthorized: HTTP-layer auth failure (PD OPTIONAL; may be plain HTTP)
- 403 Forbidden: AIDP authorization failure (PD returned)
- 409 Conflict: replay detected / envelope_id reuse (PD returned)
- 415 Unsupported Media Type: unsupported encoding
- 429 Too Many Requests: rate limiting (PD OPTIONAL but RECOMMENDED)
- 500/503: server failure / dependency failure (PD RECOMMENDED)
Response body:
- For 200: an OB
- For 202: either empty or a minimal acknowledgment object (OPTIONAL). If included, it MUST NOT claim execution success.
- For failures: a PD SHOULD be returned.

22.6.1 Correlation Rules
- EB MUST treat payload.envelope_id as authoritative.
- If X-AIDP-Envelope-ID is present and does not match the payload, EB MUST reject with MALFORMED_MESSAGE.

22.6.2 Synchronous vs Asynchronous Execution
EB MAY execute synchronously (200) when:
- action is low-risk and low-latency, and
- revocation checks can be performed inline.
Otherwise EB SHOULD default to 202 + asynchronous OB.

22.7 Endpoint: Retrieve Observation (Pull Mode)
GET /v1/aidp/observations/{envelope_id}
Returns:
- 200 OK + OB if available
- 404 Not Found if unknown (or not yet available)
- 410 Gone if envelope_id is known but observation expired/garbage-collected (PD RECOMMENDED)
- 409 Conflict if multiple observations exist and client must page (see 21.7.1)
EB SHOULD retain observations for a configurable retention window.

Execution Boundaries MUST retain Observations for a deployment-defined
retention period sufficient to allow Agents to recover results in the
presence of delivery failure, Agent restarts, or network partition.

If an Observation cannot be returned inline due to replay detection,
the EB MUST expose a deterministic retrieval mechanism bound to the
envelope_id.

22.7.1 Multiple Observations per Envelope
In advanced deployments, an envelope may yield multiple OBs (e.g., accepted then executed ). If multiple are retained, EB MUST support:
GET /v1/aidp/observations?envelope_id=...&after=...&limit=...
Returning either:
- a single latest OB (default), or
- a list of OBs when mode=all.

22.8 Endpoint: Observation Inbox (Pull Mode, Queue Semantics)
GET /v1/aidp/inbox
Query params:
- cursor (opaque)
- limit (1100)
Returns 200 OK with:
- items: array of OB messages
- next_cursor: cursor for pagination
EB MUST only deliver observations intended for the authenticated caller per observability_hooks.return_to.

22.9 Observation Delivery (Push Mode)
If observability_hooks.delivery_mode = "push", return_to MUST include a webhook URL reference or a resolvable endpoint reference.
EB will deliver via:
POST <agent_webhook_url>
Body: an OB message.
Expected responses by AR:
- 200 OK: received and accepted
- 409 Conflict: duplicate delivery (idempotent accept)
- 410 Gone: endpoint deprecated (EB SHOULD fall back to pull if configured)
EB MUST implement retries with backoff. Duplicate deliveries MUST be possible; AR MUST handle idempotently using (envelope_id, execution_id).

Execution Boundaries MUST enforce SSRF protections when performing
Observation delivery, including but not limited to:

- Rejection of loopback, link-local, private, or non-globally routable addresses
- Domain allowlisting or identity-bound endpoint registration
- Protection against DNS rebinding

If an endpoint cannot be validated or authorized, the EB MUST fall back
to inbox-based or pull-based Observation retrieval.

22.10 Problem Details over HTTP
PD is carried as the response body for AIDP-layer errors.
Mapping guidance:
- INVALID_IDENTITY, UNTRUSTED_ISSUER, REVOKED, INVALID_DELEGATION_CHAIN → 403
- CONSTRAINT_VIOLATION → 403 (or 409 if semantic conflict)
- REPLAY_DETECTED → 409
- MALFORMED_MESSAGE, UNSUPPORTED_VERSION → 400
- UNSUPPORTED_MEDIA_TYPE → 415

22.11 HTTP Caching
Responses containing OB or PD MUST include headers preventing intermediary caching:
- Cache-Control: no-store

22.12 Transport-Level Integrity and Confidentiality
AIDP-HTTP MUST be carried over TLS. mTLS is RECOMMENDED for high-risk tiers.

22.13 Minimal HTTP Sequence Examples

22.13.1 Submit IE (Async)
1. AR → EB
POST /v1/aidp/intents with IE

EB → AR
202 Accepted
2. EB → AR webhook
POST https://agent.example/aidp/ob with OB
AR → EB
200 OK

22.13.2 Submit IE (Sync)
AR → EB
POST /v1/aidp/intents with IE
EB → AR
200 OK with OB

22.13.3 Pull OB
AR → EB
GET /v1/aidp/observations/{envelope_id}
EB → AR
200 OK with OB (or 404 Not Found until ready)

22.14 Full HTTP Examples
All of the above assume:
- TLS active
- Content-Type: application/aidp+json; msg=...
- canon: AIDP-JS-Canon1
- sig are placeholders

22.14.1 Async Success (202 + webhook OB)
Request (AR → EB)
POST /v1/aidp/intents HTTP/1.1
Host: eb.payments.example
Content-Type: application/aidp+json; msg=IE
Accept: application/aidp+json; msg=OB, application/aidp+json; msg=PD
Idempotency-Key: idem-3d7f0d
X-AIDP-Envelope-ID: 0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10
Authorization: Bearer eyJ... (transport auth; not sufficient for AIDP auth)
Cache-Control: no-store


```
{
  "aidp_version": "1.0-draft",
  "msg_type": "IE",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:00+02:00",
    "actor_ref": { "agent_id": "agent:alpha", "issuer": "did:example:issuerA", "identity_
ref": "urn:aidp:id:issuerA:agent-alpha" },
    "authority_ref": { "cap_id": "cap:alpha:pay-v1", "issuer": "did:example:authA", "cap_
ref": "urn:aidp:cap:authA:cap-alpha-pay-v1", "rev_ref": "urn:aidp:rev:authA:list-01" },
    "intent_body": {
      "action": "payment.create",
      "target": { "domain": "svc:payments", "resource": "acct:merchant-123" },
      "parameters": { "amount": 50, "currency": "EUR", "memo": "invoice-8841" }
    },
    "constraints": {
      "not_before": "2026-01-13T09:14:00+02:00",
      "not_after": "2026-01-13T09:19:00+02:00",
      "max_uses": 1,
      "risk_tier": "high",
      "idempotency_key": "idem-3d7f0d"
    },
    "delegation_chain": [],
    "observability_hooks": {
      "delivery_mode": "push",
      "return_to": { "type": "webhook", "ref": "https://agent.example/aidp/ob" },
      "required_fields": ["envelope_id", "execution_id", "status", "attestation"],
      "timeout_sec": 30
```

```
    }
  },
  "proof": { "alg": "ed25519", "kid": "key:agent-alpha-1", "sig": "BASE64URL(...)" }
}
Response (EB → AR)
HTTP/1.1 202 Accepted
Cache-Control: no-store
Content-Type: application/aidp+json; msg=PD

{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:01+02:00",
    "error_code": "ACCEPTED_ASYNC",
    "error_message": "Accepted for processing; observation will be delivered via hooks.",

    "details": { "estimated": "not_provided" }
  },
  "proof": { "alg": "ed25519", "kid": "key:boundary-payments-1", "sig": "BASE64URL(...)"
}
}
Note: use of PD as “ack” for 202 to have a structured message. Alternatively it can be
blank.
Webhook Delivery (EB → AR endpoint)
POST /aidp/ob HTTP/1.1
Host: agent.example
Content-Type: application/aidp+json; msg=OB
Accept: application/aidp+json; msg=PD
Cache-Control: no-store
X-AIDP-Delivery-ID: deliv-77a1
X-AIDP-Envelope-ID: 0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10

{
  "aidp_version": "1.0-draft",
  "msg_type": "OB",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "execution_id": "exec-9c2d1a",
    "timestamp": "2026-01-13T09:14:02+02:00",
    "status": "executed",
    "result": { "payment_id": "pay_7712", "state": "captured" },
    "side_effects": [{ "resource": "acct:merchant-123", "delta": { "captured_eur": 50 } }
],
    "attestation": {
      "boundary_id": "boundary:payments-gw-1",
      "issuer": "did:example:paymentsDomain",
      "attest_profile": "AIDP-OB-Attest1",
      "decision": "authorized",
      "policy_digest": "sha256:5d3a...e91"
    }
  },
  "proof": { "alg": "ed25519", "kid": "key:boundary-payments-1", "sig": "BASE64URL(...)"
}
}
Webhook Response (AR → EB)
HTTP/1.1 200 OK
Cache-Control: no-store
Content-Type: application/aidp+json; msg=PD

{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
```

```
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:03+02:00",
    "error_code": "DELIVERY_OK",
    "error_message": "Observation received."
  }
}
```

22.14.2 Revoked Capability (403 + PD)

```
HTTP/1.1 403 Forbidden
Cache-Control: no-store
Content-Type: application/aidp+json; msg=PD

{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:01+02:00",
    "error_code": "REVOKED",
    "error_message": "Capability or identity has been revoked.",
    "details": {
      "rev_ref": "urn:aidp:rev:authA:list-01",
      "cap_id": "cap:alpha:pay-v1",
      "decision": "revoked"
    }
  },
  "proof": { "alg": "ed25519", "kid": "key:boundary-payments-1", "sig": "BASE64URL(...)"
}
}
```

22.14.3 Replay Detected (409 + PD)

```
HTTP/1.1 409 Conflict
Cache-Control: no-store
Content-Type: application/aidp+json; msg=PD

{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:14:01+02:00",
    "error_code": "REPLAY_DETECTED",
    "error_message": "Envelope ID has already been processed.",
    "details": { "replay_window": "configured", "first_seen": "2026-01-13T09:14:00+02:00"
 }
  },
  "proof": { "alg": "ed25519", "kid": "key:boundary-payments-1", "sig": "BASE64URL(...)"
}
}
```

22.14.4 Constraint Violation (403 + PD)

```
HTTP/1.1 403 Forbidden
Cache-Control: no-store
Content-Type: application/aidp+json; msg=PD

{
  "aidp_version": "1.0-draft",
  "msg_type": "PD",
  "canon": "AIDP-JS-Canon1",
  "payload": {
    "envelope_id": "0f2e3c1a-9b9a-4a8c-8c2b-2f3b9f3c5a10",
    "timestamp": "2026-01-13T09:20:10+02:00",
```

```
    "error_code": "CONSTRAINT_VIOLATION",
    "error_message": "Constraint check failed.",
    "details": {
      "violations": [
        { "field": "constraints.not_after", "reason": "expired" },
        { "field": "constraints.max_uses", "reason": "already_consumed" }
      ]
    }
  },
  "proof": { "alg": "ed25519", "kid": "key:boundary-payments-1", "sig": "BASE64URL(...)"
}
}
```

22.15 Webhook Verification Model (Push Observations)

22.15.1 Trust Anchors
Each Agent Runtime (AR) MUST maintain a set of trust anchors for validating incoming Obse
rvation deliveries. These trust anchors MUST include:
- Approved issuers of Execution Boundary (EB) attestations (e.g., did:example:payme
ntsDomain)
- (OPTIONAL) Explicit allowlists of specific EB instances identified by boundary_id

The AR MUST reject any webhook-delivered Observation if:
- attestation.issuer is not included in the trust anchors, or
- the public key referenced by proof.kid cannot be resolved for the declared issuer
, or
- the boundary_id is disallowed by local policy.

22.15.2 Required Verification Steps (AR-side)
Upon receipt of an Observation via webhook, the Agent Runtime MUST perform the following
steps, in order:
1. Parse and schema-validate the Observation message using strict validation rules.
2. Verify the attestation proof (proof) over the canonical signing input.
3. Validate issuer trust, ensuring attestation.issuer is trusted.
4. Bind the Observation to an outstanding Intent:
o payload.envelope_id MUST correspond to a currently pending Intent Envelope, other
wise the Observation MUST be rejected or quarantined according to policy.
5. Enforce replay protection:
o The tuple (envelope_id, execution_id) MUST be accepted at most once.
6. OPTIONAL but RECOMMENDED: Verify attestation.policy_digest against a locally know
n policy registry when available.
If any verification step fails, the AR MUST NOT accept the Observation and MUST respond w
ith a non-2xx HTTP status code.

22.15.3 Delivery Idempotency
Execution Boundaries MUST treat webhook delivery as retry-safe.
Agent Runtimes MUST process webhook Observations idempotently, keyed by:
(envelope_id, execution_id)
If a duplicate delivery is received, the AR MUST NOT re-apply any state transition and SH
OULD return either:
- 409 Conflict (duplicate), or
- 200 OK with a Problem Details payload indicating duplicate acceptance.

22.15.4 Webhook Channel Authentication
In addition to cryptographic attestation inside the Observation itself, the webhook trans
port channel SHOULD be protected using:
- Mutual TLS (mTLS), or
- An HTTP authentication mechanism that cryptographically binds the EB identity to
the TLS session.
Channel-level authentication mitigates denial-of-service attacks and prevents unauthentic
ated injection of traffic toward the AR.

22.15.5 Anti-Substitution Rules
The Agent Runtime MUST enforce the following anti-substitution rules:
- The payload.envelope_id in the Observation MUST correspond exactly to the pending
 Intent Envelope.

- If the AR maintains a local digest of the submitted Intent Envelope, it SHOULD ensure the received Observation corresponds to that digest entry.
- All fields declared in observability_hooks.required_fields MUST be present before the Observation is accepted.

Failure of any rule MUST cause the Observation to be rejected.

22.15.6 Error Responses from Webhook Endpoint

The AR webhook endpoint MUST use the following status codes:

| HTTP Code | Meaning |
|---|---|
| 200 | Observation accepted |
| 409 | Duplicate delivery |
| 400 | Malformed Observation |
| 403 | Untrusted issuer or invalid proof |
| 410 | Endpoint deprecated |
| 429 | Rate limited |

22.15.7 Fallback to Pull Mode

If repeated webhook deliveries fail, the Execution Boundary SHOULD:

- Retain the Observation for retrieval via GET /v1/aidp/observations/{envelope_id}, and
- Optionally emit a Problem Details message indicating push delivery failure.

22.16 Reference Validation Pipeline

This section defines the normative validation and execution pipeline that every AIDP-compliant Execution Boundary (EB) and Agent Runtime (AR) MUST implement.

Any deviation from this sequence constitutes a protocol violation.

22.16.1 Execution Boundary Validation Pipeline

Upon receipt of an Intent Envelope over HTTP, the EB MUST execute the following steps in order:

1. Transport Authentication
2. Message Parsing
3. Canonicalization
4. Proof Verification
5. Schema Validation
6. Identity Validation
7. Capability Resolution
8. Delegation Chain Validation
9. Revocation Checks
10. Constraint Enforcement
11. Replay Protection
12. Authorization Decision
13. Execution
14. Observation Construction
15. Attestation & Emission
16. Audit Recording

22.16.2 Detailed Step Semantics

1. Transport Authentication

The EB MUST verify HTTP-layer authentication (mTLS, OAuth, etc.) and reject unauthenticated clients.

Transport authentication does not grant execution authority.

2. Message Parsing

The EB MUST parse the incoming message strictly.

Malformed messages MUST be rejected with MALFORMED_MESSAGE.

During message parsing, Execution Boundaries and Agent Runtimes MUST use a JSON parser configuration that detects and rejects duplicate object member names.

Messages that contain duplicate keys MUST be rejected before any validation, canonicalization, or cryptographic verification is performed.

3. Canonicalization
The EB MUST canonicalize the payload using the declared canon profile.
Failure MUST abort processing.

4. Proof Verification
If present, proof MUST be verified against the canonical payload.
Failure MUST abort processing.

5. Schema Validation
The EB MUST validate the payload against the AIDP schema for the declared message type.
Unknown fields in security-critical objects MUST cause rejection.

6. Identity Validation
The EB MUST resolve and validate actor_ref.identity_ref and issuer trust.
Failures MUST result in INVALID_IDENTITY or UNTRUSTED_ISSUER.

7. Capability Resolution
The EB MUST resolve authority_ref.cap_ref, validate issuer trust, and load associated con
straints.

8. Delegation Chain Validation
If delegation_chain is present, the EB MUST:
- Validate every link’s proof
- Enforce subsetting rules
- Verify parent-child relationships
- Reject any amplification of authority
Failures MUST result in INVALID_DELEGATION_CHAIN.

9. Revocation Checks
The EB MUST consult all rev_ref references (identity, capability, delegation parents).
For high-risk actions, revocation checks MUST be performed immediately prior to execution
.

10. Constraint Enforcement
All constraints MUST be enforced:
- time windows
- usage limits
- resource scope
- risk tier policies
Failures MUST result in CONSTRAINT_VIOLATION.

11. Replay Protection
The EB MUST verify that envelope_id has not been previously accepted.
Replay MUST result in REPLAY_DETECTED.

12. Authorization Decision
If and only if all prior steps succeed, the EB MUST authorize the action.

13. Execution
The EB executes the action on the target system.
Execution failures MUST still produce an Observation with status = failed.

14. Observation Construction
The EB MUST construct an Observation bound to:
- envelope_id
- execution_id
- decision
- result
- side effects
- applied policy

15. Attestation & Emission
The EB MUST:
- attach an attestation
- sign the Observation
- deliver it via the configured Observation Channel

16. Audit Recording
The EB MUST record:
- envelope digest
- decision
- execution_id
- final status
- emitted Observation digest

22.16.3 Agent Runtime Validation Pipeline (OB/PD)
Upon receiving an Observation or Problem Details message, the AR MUST perform:
1. Parse & Schema Validate
2. Canonicalize
3. Verify Attestation Proof
4. Verify Issuer Trust
5. Bind to Outstanding Intent
6. Enforce Replay Protection
7. Accept and Release Intent
8. Failure Classification and Retry Suppression

If an Observation or Problem Details message indicates a deterministic failure (including but not limited to CONSTRAINT_VIOLATION, NOT_AUTHORIZED, POLICY_REFUSED, or any non-retriable execution_advice), the Agent Runtime MUST NOT re-issue an Intent Envelope whose canonical payload is identical to the failed intent.

The Agent Runtime MUST either:
- revise the intent semantics (e.g., constraints or intent_body), or
- abort the reasoning path, or
- require explicit operator or policy override.
To enforce retry suppression deterministically, the Agent Runtime SHOULD maintain a cache of recently failed intent digests, computed as a hash over the canonical signing input of the Intent Envelope payload.

Re-issuance of an Intent with an identical digest following a deterministic failure SHOULD be suppressed for a deployment-defined cooldown period.

If any step fails, the Observation MUST be rejected and MUST NOT advance agent reasoning.

Deterministic failures are failures for which re-execution of an identical intent is not expected to yield a different outcome, absent a change in constraints, authority, or policy state.

Examples include constraint violations, authorization failures, and explicit policy refusals.

22.16.4 Deterministic Agent Loop Guarantee
An agent loop is compliant if and only if:
No agent reasoning step that causally depends on an Intent Envelope occurs without a validated Observation for that Intent.

22.16.5 Minimal Conformance Checklist
An implementation claiming AIDP-HTTP compliance MUST implement:
- full EB pipeline
- full AR pipeline
- replay store
- revocation resolver
- attestation verification
- audit logging

23. AIDP Conformance Test Suite — Outline

23.1 Purpose
This document defines the test framework for validating whether an implementation is comp

liant with the AIDP specification.
It enables:
- certification of implementations,
- interoperability between vendors,
- objective security verification.

23.2. Certification Levels
Level 1 — AIDP-Core
Minimal functional compliance.
Level 2 — AIDP-Secure
Includes full security & revocation semantics.
Level 3 — AIDP-Enterprise
Includes auditability, cross-domain, governance & recovery.

23.3 Test Domains

| Domain | Focus |
|---|---|
| Identity | Agent identity validation |
| Authority | Capability enforcement |
| Delegation | Subsetting & chain validation |
| Revocation | Immediate invalidation |
| Intent | Schema & canonicalization |
| Execution | Enforcement correctness |
| Observation | Binding & integrity |
| Replay | Duplicate detection |
| Audit | Traceability & non-repudiation |
| Cross-Domain | Trust boundary handling |

23.4 Core Test Categories

23.4.1 Identity & Trust
- T-ID-01: Reject invalid issuer
- T-ID-02: Reject expired identity
- T-ID-03: Enforce trust anchors
- T-ID-04: Revalidate across trust boundary

23.4.2 Capability & Authority
- T-AUTH-01: Enforce action scope
- T-AUTH-02: Enforce resource scope
- T-AUTH-03: Enforce constraints
- T-AUTH-04: Reject capability amplification

23.4.3 Delegation
- T-DEL-01: Validate delegation chain
- T-DEL-02: Reject broken chain
- T-DEL-03: Enforce subsetting rule
-T-DEL-04: Revoke parent invalidates children

23.4.4 Revocation
- T-REV-01: Immediate revocation enforcement
- T-REV-02: Revocation after acceptance but before execution
- T-REV-03: Revocation during execution (high risk path)

23.4.5 Intent Envelope
- T-IE-01: Canonicalization correctness
- T-IE-02: Proof verification
- T-IE-03: Reject malformed envelope
- T-IE-04: Enforce required fields

23.4.6 Execution & Enforcement
- T-EX-01: Reject without valid authority
- T-EX-02: Enforce constraints before execution
- T-EX-03: Execute only after full validation
- T-EX-04: Reject on replay

23.4.7 Observation & Feedback Binding
- T-OB-01: Bind OB to correct envelope_id
- T-OB-02: Reject mismatched OB
- T-OB-03: Enforce deterministic agent loop
- T-OB-04: Reject unsigned OB

23.4.8 Replay & Ordering
- T-RPL-01: Detect duplicate envelope_id
- T-RPL-02: Reject reused execution_id
- T-RPL-03: Maintain replay window

23.4.9 Audit & Compliance
- T-AUD-01: Record full action trace
- T-AUD-02: Reconstruct causal chain
- T-AUD-03: Verify non-repudiation

23.4.10 Cross-Domain & Federation
- T-XD-01: Enforce trust boundary
- T-XD-02: Reject untrusted issuer
- T-XD-03: Preserve delegation semantics across domains

23.5 Failure Injection Tests
- Network failure mid-execution
- Revocation during processing
- Observation delivery failure
- Duplicate webhook delivery
- Corrupted attestation
- Malicious agent input

23.6 Required Artifacts for Certification
An implementation seeking certification MUST provide:
- Test harness access
- Public verification endpoints
- Audit logs
- Policy configuration
- Revocation interface
- Capability issuance interface

23.7 Compliance Declaration
An implementation MAY claim:
“AIDP-Compliant: Level X”
only after passing all tests for that certification level.

## Appendix A. Changes Since draft-vandoulas-aidp-00

This section summarizes substantive changes introduced since draft-vandoulas-aidp-00.

- Clarified replay semantics to explicitly distinguish replay detection from re-execution, and to require deterministic result recovery without side effects (Sections 10.6.3, 21.7).

- Introduced explicit Observation retention and pull-based recovery requirements to prevent loss of execution outcomes due to delivery failure (Sections 21.7, 20.9).

- Refined agent control loop semantics to permit parallel, causally-independent intents while preserving deterministic reasoning for dependent actions (Sections 3.3, 8.2).

- Hardened canonicalization and parsing requirements by mandating rejection of duplicate JSON object member names during parsing, prior to validation and cryptographic verification (Sections 16.3, 21.16.2).

- Added explicit identity resolution profiles to support interoperable cross-domain validation, including dereferenceable and self-contained identity models (Section 4.2).

- Strengthened observability hook security by restricting Observation delivery endpoints to pre-authorized destinations and mandating SSRF protections with deterministic fallback to pull-based retrieval (Sections 17.1.6, 21.9).

- Introduced execution_advice as an optional, non-binding mechanism for communicating retry, backoff, and failure classification guidance in both Observation and Problem Details messages (Sections 17.2, 17.3).

- Added deterministic retry suppression and anti-loop requirements to the Agent Runtime validation pipeline to prevent infinite re-issuance of semantically identical intents following deterministic failures (Section 21.16.3).

- Clarified deterministic agent loop guarantees to ensure reasoning progression is gated only on Observations for causally dependent intents, rather than enforcing global serialization (Section 21.16.4).

Authors' Addresses

Ioannis Vandoulas
Email: jvandoul@gmail.com