Independent Submission
Internet-Draft
Intended status: Experimental
Expires: 26 September 2026

J. Song
HKUST
M. Yuan
CUHK

25 March 2026

Agent Description Protocol (ADP)
draft-song-anp-adp-00

Abstract

Autonomous AI agents in a decentralized network need a common way to describe their capabilities so that peers can discover and invoke them. The Agent Internet Protocol (AIP) provides name-based datagram delivery between agents identified by agent:// URIs; the Agent Invocation Transport Protocol (AITP) provides reliable invocation and streaming above AIP. This document describes the Agent Description Protocol (ADP), an application-layer convention carried over AITP that defines how agents describe their capabilities, publish those descriptions, and discover peers whose capabilities match a query.

ADP defines the Agent Card, a JSON document format that carries an agent’s identity, human-readable description, method catalogue, endpoint bindings, skill tags, and operational constraints. ADP also defines three AITP method names — adp.describe, adp.advertise, and adp.discover — through which agents exchange and query Agent Cards at the invocation layer.

ADP is intentionally a thin format-and-convention layer: it standardizes the document schema and the exchange methods, but defers reputation, economics, credentials, identity infrastructure, and deployment-specific dissemination mechanisms (DHT topics, GossipSub channels, HTTP well-known URIs) to companion protocols and implementation profiles.

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

Agents in a decentralized network need a standard way to describe what they can do, so that other agents can discover and invoke them. Without a common description format, every agent application protocol must independently define how capabilities are advertised, how methods are enumerated, and how operational constraints are communicated.

Existing agent description approaches address parts of this problem. Google A2A defines an AgentCard for HTTP-based agents. MCP defines a tool list for client-server tool providers. OASF defines a Kubernetes-style AgentDescriptor for orchestrators. Each assumes a specific network model and transport binding. None is designed for agent://- addressed peers communicating over AIP and AITP.

ADP bridges this gap. It describes a single document format — the Agent Card — that can be exchanged natively over AITP, while remaining convertible to and from external description formats.

1.2. Scope

This document is one of four core Internet-Drafts in the Agent Network Protocol (ANP) suite: AIP [I-D.song-anp-aip] (datagram delivery), AITP [I-D.song-anp-aitp] (invocation transport), ANS [I-D.song-anp-ans] (name system), and ADP (this document, description and discovery). The four drafts are designed to co-evolve as a self-contained protocol suite; no additional specification is required for baseline interoperability. AIP's local-resolver semantic extension MAY be backed by an implementation that consults ADP discovery services for ranked capability matching; ANS core provides name-to-peer binding but does not itself perform ranked semantic discovery.

ADP is an application-layer convention carried over AITP [I-D.song-anp-aitp].

This document is organized in three layers of decreasing normative weight:

ADP Core (Sections 3-4) The Agent Card JSON document format — schema, field semantics, tool and endpoint descriptors, skill tags, constraints, and the extension mechanism. This is the primary contribution of the document.

ADP Exchange Conventions (Section 5) Three AITP method names (adp.describe, adp.advertise, adp.discover) through which agents retrieve, publish, and query Agent Cards. These conventions bind the format to AITP but do not preclude other transports.

ADP Baseline Discovery Profile (Section 6) A recommended multi-factor scoring algorithm providing one interoperable starting point for ranking agents. This is guidance, not the only valid ranking strategy.

This document does not cover:

1. Agent identity, key management, or authentication — these belong to AIP and deployment-context mechanisms.

2. Reputation scoring or trust computation — these belong to a companion reputation protocol.

3. Economic settlement, pricing, or credit systems.

4. Verifiable credentials or attestation chains.

5. Transport-specific dissemination (DHT key spaces, GossipSub topics, HTTP well-known paths) — these are implementation profiles, not protocol requirements.

6. Task orchestration, workflow, or business logic.

Accordingly, this document does not specify a complete agent runtime or deployment system; it describes only the description format and exchange conventions shared by agents above AITP.

1.3. Positioning Among Existing Approaches

ADP is not a replacement for any existing agent description standard. Its role is to describe a common description and exchange convention above AIP/AITP, in the same way that AITP defines a common invocation substrate above AIP.

Google A2A's AgentCard is an HTTP-oriented description served at a well-known URL, tightly coupled to HTTP transport and A2A task semantics. ADP's Agent Card is transport-independent at the protocol level; an AITP method retrieves it, and conversion to A2A format is a straightforward mapping (see Appendix A.1).

MCP's tool list describes capabilities of a client-server tool provider. ADP's tool descriptors use the same field names (name, description, inputSchema) for direct compatibility, but extend them with output schema, streaming, and idempotency annotations relevant to agent-to-agent invocation.

OASF's AgentDescriptor uses a Kubernetes-style manifest (kind/apiVersion/metadata/spec) designed for orchestration platforms. ADP uses a flat JSON document optimized for network exchange and progressive disclosure, but fields map bidirectionally to OASF (see Appendix A.3).

1.4. Design Rationale

ADP is justified when the following conditions hold:

1. *Multiple agent applications need capability discovery.*

   If task orchestration, knowledge replication, swarm coordination, and direct messaging each need to discover agents by skill, factoring description and discovery into a shared protocol eliminates redundant definitions.

2. *A single document simplifies the ecosystem.*

If an agent's identity, capabilities, endpoints, and constraints are scattered across multiple data structures (profile records, resume broadcasts, name table entries), unifying them into one document reduces inconsistency.

3. *Cross-ecosystem interoperability is valued.*

   If agents from different ecosystems (A2A, MCP, OASF) need to interoperate, a superset document that can be losslessly projected onto each ecosystem's format enables bridging without information loss.

1.5. Assumptions

ADP assumes:

1. An AITP module capable of sending and receiving invocations with the methods defined in this document.

2. Agent:// URIs are stable identifiers for the duration of an Agent Card's validity period.

3. Agent Card documents are UTF-8 encoded JSON ([RFC8259]). Implementations MAY additionally support CBOR ([RFC8949]) for constrained environments.

4. Companion protocols (reputation, credentials, economics) extend the Agent Card through the extension mechanism defined in Section 3.8, not by modifying core fields.

1.6. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.7. Document Status

This document is Informational. ADP describes a JSON document format and a set of AITP exchange conventions derived from running code in the ClawNet implementation. Unlike AIP and AITP, which define wire-level protocol encodings and state machines, ADP is a data-format and naming convention — closer in nature to RFC 7946 (GeoJSON) than to a transport protocol specification.

The BCP 14 keywords in this document express interoperability requirements: an implementation that conforms to the Agent Card schema and method conventions described here will interoperate with other conforming implementations. The keywords do not imply Standards Track status.

2. Terminology

Agent Card A JSON document that describes an agent's identity, capabilities, endpoints, skill tags, and operational constraints. The Agent Card is the central data structure described by this document.

Tool Descriptor An object within an Agent Card that describes a single invocable method, including its name, input schema, output schema, and operational properties (streaming, idempotency).

Endpoint Descriptor An object within an Agent Card that binds a set of methods to a protocol and URI through which they can be reached.

Skill Tag A short UTF-8 string or hierarchical path (e.g., "nlp/translation") that classifies an agent's capabilities for discovery and matching purposes.

Discovery Query A structured request for agents whose capabilities match specified skill tags and/or natural-language descriptions, evaluated by a multi-factor scoring algorithm.

3. Agent Card Document Format

An Agent Card is a JSON object ([RFC8259]) that describes an agent. Only two fields are required; all others are optional, enabling progressive disclosure: a minimal Agent Card is valid, and richer descriptions are built by adding optional fields.

3.1. Document Structure

The top-level JSON object contains the following fields. Field names use snake_case for consistency with the existing implementation base and MCP field conventions.

Field	Type	Req	Description
id	string	MUST	Globally unique identifier for the agent. The "id" MUST be the agent's agent:// URI —

			the native network-facing identifier used across the ANP suite (AIP, ANS, AITP, ADP). See Section 3.3.
name	string	MUST	Human-readable agent name. Uniqueness is RECOMMENDED but not enforced by this protocol.
description	string	MAY	Free-text description of the agent's purpose and capabilities. Used for semantic discovery matching.
version	string	MAY	Semantic version string (e.g., "1.2.0") of the agent's capability set.
skills	string[]	MAY	Skill tags classifying the agent's capabilities. See Section 3.6.
tools	object[]	MAY	Tool descriptors for invocable methods. See Section 3.4.
endpoints	object[]	MAY	Endpoint descriptors binding methods to protocols and URIs. See Section 3.5.
constraints	object	MAY	Operational constraints. See Section 3.7.
did	string	MAY	An optional Decentralized Identifier (DID) for cross-ecosystem identity binding (e.g., did:key:z6Mk...). Enables external credential and attestation ecosystems to reference the agent. Does not replace the native agent:// identifier in the "id" field. See Section 3.3.
metadata	object	MAY	Temporal metadata: created_at, updated_at (ISO 8601 timestamps), ttl

			(seconds).
extensions	object	MAY	Namespace-keyed extension fields. See Section 3.8.
seq	uint64	MAY	Monotonically increasing sequence number. Each time the agent updates its Agent Card, it MUST increment seq by at least one. Consumers use seq to determine which of two cards is newer; a card with a higher seq supersedes one with a lower seq for the same id. See Section 3.2.
signature	string	MAY	Ed25519 signature over the canonical serialization of all other top-level fields (excluding "signature" itself), encoded as Base64url (no padding). When present, consumers can verify the card was produced by the private key corresponding to the agent’s public key (derived from the agent:// URI in "id"). See Section 3.2.

Table 1: Agent Card Top-Level Fields

Implementations MUST ignore unrecognized top-level fields to enable forward compatibility. Implementations MUST NOT reject an Agent Card solely because it contains fields not defined in this specification.

3.2. Agent Card Signature

The optional "seq" and "signature" fields allow an Agent Card to be verified independently of its delivery channel. This is important when cards are cached by directory agents, relayed through gossip, or stored in the DHT — contexts where the original AITP association is no longer available for authentication.

To sign an Agent Card, the agent:

1. Constructs the complete Agent Card JSON object with all desired fields, including "seq", but excluding "signature".

2. Serializes the object using JCS (JSON Canonicalization Scheme, [RFC8785]) to produce a deterministic byte sequence.

3. Signs the byte sequence with the agent’s Ed25519 private key.

4. Encodes the 64-byte signature as Base64url (no padding) and inserts it as the "signature" field.

A consumer verifies an Agent Card by removing the "signature" field, re-canonicalizing via JCS, and verifying the Ed25519 signature against the public key derived from the "id" URI. If verification fails, the consumer MUST discard the card.

When both "seq" and "signature" are present, consumers MUST prefer the card with the higher "seq" value for a given "id". A card without "signature" MUST NOT supersede a signed card with the same or higher "seq".

These fields mirror the same pattern used by ANS Name Records ([I-D.song-anp-ans], Section 5.1), providing a uniform authentication mechanism across both discovery layers.

3.3. Identity Model

Within the ANP protocol suite, agent:// is the native identifier for network-facing capability endpoints. The agent:// URI serves as the canonical primary key across all four protocol layers:

* AIP: source and destination URI in datagrams

* ANS: registration, resolution, and discovery key

* AITP: association endpoint identifier

* ADP: capability description subject

Other identifiers such as Decentralized Identifiers (DIDs) may be attached to an Agent Card via the optional "did" field for external ecosystem interoperability — credential issuance, cross-domain identity federation, or legal identity binding. These external identifiers do not replace the native agent:// identifier and are not used for routing, name resolution, or transport association within ANP.

3.4. Tool Descriptors

Each element of the "tools" array describes a single invocable method. The field names align with MCP's tool definition for direct compatibility.

Field	Type	Req	Description
name	string	MUST	Method name. MUST match the AITP Method value used to invoke this tool. Maximum 255 UTF-8 octets (AITP Method Len limit).
description	string	MAY	Human-readable description of what the method does.
input_schema	object	MAY	JSON Schema (2020-12) describing the expected request body.
output_schema	object	MAY	JSON Schema describing the response body on success.
streaming	boolean	MAY	If true, this method supports AITP bidirectional streaming (Type = STREAM). Default: false.
idempotent	boolean	MAY	If true, invoking this method multiple times with the same input produces the same result. Default: false.

Table 2: Tool Descriptor Fields

Implementations MUST ignore unrecognized fields within tool descriptors.

3.5. Endpoint Descriptors

Each element of the "endpoints" array binds a set of methods to a reachable network address.

Field	Type	Req	Description
protocol	string	MUST	Protocol identifier. See Table 4.
uri	string	MUST	Endpoint URI. Format depends on protocol.
methods	string[]	MAY	Method names reachable at this endpoint. If absent, all tools are assumed reachable.
auth	string	MAY	Authentication scheme: "none", "bearer", "mutual_tls", "aitp_signed".
priority	integer	MAY	Lower values indicate higher priority. Default: 0. Clients SHOULD prefer endpoints with lower priority values.

Table 3: Endpoint Descriptor Fields

Identifier	Description	Example URI
aitp	AITP invocation over AIP	agent://translator-zh-en
http+json	HTTP REST (RFC 9110)	https://api.example.com/v1
grpc	gRPC	grpc://api.example.com:443
ws	WebSocket (RFC 6455)	wss://api.example.com/ws

Table 4: Protocol Identifiers Recognized by This Specification

The "aitp" protocol identifier is the native binding for ANP agents. Other identifiers enable bridging to non-ANP ecosystems. These identifiers are recognized by conforming ADP implementations for interoperability; they do not constitute a global protocol registry. The list is not exhaustive; implementations MAY recognize additional protocol identifiers and MUST ignore endpoint entries with identifiers they do not understand.

3.6. Skill Tags

The "skills" array contains UTF-8 strings that classify the agent's capabilities for discovery. Two syntactic forms are defined:

Form	Example	Description
Flat	"python"	A single keyword. Lowercase ASCII RECOMMENDED.
Hierarchical	"nlp/translation"	A slash-separated path from general to specific. Each segment SHOULD be lowercase ASCII with hyphens.

Table 5: Skill Tag Syntax

3.6.1. Matching Rules

Implementations that perform skill-tag matching SHOULD support the following rules:

Exact match Query tag "nlp/translation" matches only the tag "nlp/translation".

Prefix match (query expansion) Query tag "nlp/*" matches any tag whose first path segment is "nlp" (e.g., "nlp/translation", "nlp/text-analysis/sentiment").

Parent match (upward compatibility) An agent declaring "nlp/translation" matches a query for "nlp" because a more specific skill implies the general category.

Implementations MAY support alias normalization (e.g., "py" normalizes to "python", "ml" normalizes to "machine-learning"). Alias tables are deployment-specific and not standardized by this document.

3.6.2. Recommended Top-Level Categories

The following top-level path segments are RECOMMENDED for interoperability. Agents MAY use any tag; these categories provide a common starting vocabulary.

Category	Description	Examples
nlp	Natural language processing	nlp/translation, nlp/generation/summarization
vision	Computer vision	vision/object-detection, vision/ocr
audio	Audio and speech	audio/speech-to-text, audio/ music-generation
analytical	Analytical and reasoning	analytical/data-analysis, analytical/optimization
coding	Software development	coding/code-generation, coding/debugging
retrieval	Information retrieval	retrieval/web-search, retrieval/knowledge-base
creative	Creative production	creative/writing, creative/ design
ops	Operations	ops/deployment, ops/ monitoring, ops/security

Table 6: Recommended Skill Tag Categories

3.7. Operational Constraints

The "constraints" object communicates operational limits to potential callers.

Field	Type	Description
max_concurrent_tasks	integer	Maximum number of simultaneous tasks the agent will accept.
max_input_tokens	integer	Maximum input size in tokens for a single invocation.
supported_languages	string[]	Natural languages the agent supports (ISO 639-1 codes).
rate_limit	string	Rate limit expression (e.g., "60/min").

Table 7: Constraint Fields

All constraint fields are advisory. Callers SHOULD respect advertised constraints but MUST be prepared for the agent to enforce them via AITP status codes (BUSY, INVALID_REQUEST) at invocation time.

3.8. Extension Mechanism

The "extensions" object carries protocol-specific or deployment-specific data that is not part of the ADP core. Each key in the extensions object is a namespace string; each value is an arbitrary JSON object.

Namespaces SHOULD use reverse-domain notation or a well-known short name to avoid collisions (e.g., "clawnet.reputation", "oasf.labels").

Implementations MUST preserve unrecognized extension namespaces during round-trip serialization. Implementations MUST NOT reject an Agent Card because it contains unrecognized extensions.

Example extensions for a ClawNet deployment:

```
{
  "extensions": {
    "clawnet.reputation": {
      "score": 78.5,
      "tier": 16,
      "tasks_completed": 342,
      "avg_rating": 4.7
    },
    "clawnet.economics": {
      "cost_shells_per_call": 1,
      "requires_deposit": false
    }
  }
}
```

3.9. Complete Example

```
{
  "id": "agent://translator-zh-en",
  "did": "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK",
  "name": "translator-zh-en",
  "description": "Chinese-English bidirectional translation",
  "version": "1.2.0",
  "skills": ["nlp/translation", "nlp/text-analysis", "python"],
  "tools": [
    {
      "name": "translate",
      "description": "Translate text between languages",
      "input_schema": {
        "type": "object",
        "properties": {
          "text": {"type": "string"},
          "source_lang": {"type": "string", "default": "auto"},
          "target_lang": {"type": "string"}
        },
        "required": ["text", "target_lang"]
      },
      "output_schema": {
        "type": "object",
        "properties": {
          "translated": {"type": "string"},
          "confidence": {"type": "number"}
        }
      },
      "streaming": false,
      "idempotent": true
    }
  ],
```

```
    "endpoints": [
      {
        "protocol": "aitp",
        "uri": "agent://translator-zh-en",
        "methods": ["translate", "detect_language", "glossary"]
      },
      {
        "protocol": "http+json",
        "uri": "https://api.example.com/translate/v1",
        "auth": "bearer",
        "priority": 10
      }
    ],
    "constraints": {
      "max_concurrent_tasks": 3,
      "max_input_tokens": 100000,
      "supported_languages": ["zh", "en", "ja"],
      "rate_limit": "60/min"
    },
    "metadata": {
      "created_at": "2026-01-15T00:00:00Z",
      "updated_at": "2026-03-24T12:00:00Z",
      "ttl": 3600
    }
  }
```

4. Carriage over AITP

ADP describes three AITP method names for exchanging Agent Cards. Each method uses a REQUEST/RESPONSE exchange; the request body and response body are JSON objects.

4.1. adp.describe — Retrieve an Agent Card

A caller sends a REQUEST with Method = "adp.describe" to retrieve the remote agent’s Agent Card.

Request body Empty, or a JSON object with an optional "fields" array listing the top-level field names the caller is interested in. If "fields" is present, the responder MAY return only the requested fields plus the two required fields (id, name).

Response body (Status = OK) The agent’s Agent Card as a JSON object conforming to Section 3.

Error responses UNAUTHORIZED (5) if the caller lacks permission to view the description.

Agents implementing ADP MUST register a handler for the "adp.describe" method and respond with at least a minimal Agent Card (id and name). Deployments that expose peer-visible capability metadata SHOULD implement adp.describe, as it is the primary mechanism through which peers learn an agent's capabilities.

4.2. adp.advertise — Publish an Agent Card

An agent sends a REQUEST with Method = "adp.advertise" to push its Agent Card to a peer, directory service, or name server.

Request body The sender's Agent Card as a JSON object. The "id" field MUST equal the AITP association's source agent:// URI.

Response body (Status = OK) A JSON object with a "stored" boolean field indicating whether the receiver has stored the Agent Card.

Error responses INVALID_REQUEST (6) if the Agent Card fails validation. UNAUTHORIZED (5) if the sender is not permitted to advertise.

Receivers SHOULD validate that the Agent Card's "id" matches the AITP association's remote agent:// URI. Receivers MAY refuse to store Agent Cards that exceed a deployment-specific size limit.

4.3. adp.discover — Query for Agents

A caller sends a REQUEST with Method = "adp.discover" to query a peer or directory service for agents matching specified criteria.

Request body A JSON object with the fields in Table 8.

Response body (Status = OK) A JSON object with a "results" array. Each element contains the fields in Table 9.

Field	Type	Description
tags	string[]	Skill tags to match (at least one SHOULD be provided).
query	string	Natural-language description of the desired capability.
limit	integer	Maximum number of results. Default: 10.
min_score	number	Minimum composite score threshold (0.0-1.0). Default: 0.1.

Table 8: adp.discover Request Fields

Field	Type	Description
agent_card	object	The matched agent's Agent Card (or a subset thereof).
score	number	Composite match score (0.0-1.0).
matched_tags	string[]	Skill tags that contributed to the match.

Table 9: adp.discover Result Fields

Error responses INVALID_REQUEST (6) if the query is malformed.

Any agent MAY respond to adp.discover queries by searching its local knowledge of peer Agent Cards. A dedicated directory agent MAY aggregate Agent Cards from adp.advertise calls and serve richer discovery results.

4.4. Method Summary

Method	Direction	Purpose
adp.describe	Caller → Agent	Retrieve the agent's Agent Card
adp.advertise	Agent →	Push the agent's Agent Card

	Directory/Peer	
adp.discover	Caller → Directory/Peer	Query for agents by criteria

Table 10: ADP AITP Methods

5. Baseline Discovery Profile

This section describes a recommended baseline scoring profile for ranking agents against a discovery query. It provides one interoperable starting point, not the only valid ranking strategy. This profile is intended to improve comparability across implementations, not to constrain local ranking policy. Implementations that respond to adp.discover SHOULD use this profile or a compatible variant to promote consistent cross-implementation ranking.

The scoring factors described here are intentionally abstract: they identify what signals matter for discovery, while leaving the concrete computation of each signal (reputation model, availability probe mechanism, rating aggregation) to the deployment context.

5.1. Recommended Scoring Algorithm

The baseline composite score for a candidate agent is:

```
score = W_tag * tagScore
      + W_sem * semanticScore
      + W_rep * reputationScore
      + W_avl * availabilityScore
      + W_rat * ratingScore
```

The recommended default weights are:

Factor	Symbol	Default	Computation
Tag match	W_tag	0.30	|matched_tags| / |query_tags| (Jaccard-style overlap between query tags and agent skills)
Semantic overlap	W_sem	0.25	Word overlap or BM25 score between the query text and the agent's description + skill labels
Reputation	W_rep	0.20	Deployment-specific reputation score, normalized to [0.0, 1.0]
Availability	W_avl	0.15	1.0 if the agent is online and below its max_concurrent_tasks; 0.0 otherwise
Rating	W_rat	0.10	Deployment-specific average rating, normalized to [0.0, 1.0]

Table 11: Default Discovery Weights

Implementations MAY adjust weights based on deployment requirements. The five factors and their semantics are the normative core of this baseline profile; the default numeric weights are RECOMMENDED starting values. Note that reputation, rating, and availability are inherently deployment-specific signals normalized into interoperable [0.0, 1.0] score fields; this specification does not prescribe how those signals are computed. Implementations that diverge from these factors SHOULD document their scoring approach to enable cross-implementation comparison.

5.2. Cold Start (Recommended Practice)

New agents with fewer than 5 completed tasks lack sufficient reputation and rating signals. Implementations SHOULD apply a cold-start boost (e.g., adding a fixed reputation bonus) to avoid penalizing new entrants. The boost amount and task threshold are deployment-specific.

5.3. Load Filtering (Recommended Practice)

Implementations SHOULD exclude agents whose active task count meets or exceeds their max_concurrent_tasks constraint from discovery results. This prevents routing work to overloaded agents.

6. Agent Card Lifecycle

6.1. Creation

An agent creates its Agent Card by populating at least the "id" and "name" fields. The "metadata.created_at" field SHOULD be set to the current UTC timestamp.

6.2. Publication

An agent publishes its Agent Card by:

1. Responding to incoming adp.describe requests.

2. Proactively sending adp.advertise to known directory agents or peers.

3. Using deployment-specific dissemination mechanisms (DHT, GossipSub, HTTP well-known endpoints) as defined by implementation profiles.

The minimal publication requirement is (1): agents implementing ADP MUST respond to adp.describe. Proactive advertisement is OPTIONAL and depends on deployment needs.

6.3. Freshness and TTL

The "metadata.ttl" field indicates the number of seconds the Agent Card is considered fresh after retrieval. Consumers SHOULD re-fetch the Agent Card after the TTL expires. If no TTL is present, implementations SHOULD apply a deployment-specific default.

Each update to the Agent Card SHOULD increment "metadata.updated_at". When the "seq" field (Section 3.2) is present, consumers MUST use "seq" as the authoritative ordering key; "metadata.updated_at" serves as a supplementary human-readable freshness hint. Consumers SHOULD prefer the Agent Card with the higher "seq" (or, if "seq" is absent, the most recent updated_at timestamp) when multiple copies are obtained from different sources.

6.4. Revocation

An agent that is no longer available SHOULD publish an Agent Card with an empty "tools" array and an empty "endpoints" array, signaling that it has no callable methods. Consumers SHOULD treat such an Agent Card as a revocation.

Deployment-specific mechanisms (e.g., DHT record expiration, GossipSub tombstone messages) provide additional revocation signals but are outside the scope of this document.

7. Security Considerations

7.1. Agent Card Authentication

An Agent Card can be spoofed if not authenticated. Implementations SHOULD deliver Agent Cards over AITP associations where the remote agent's identity has been verified (via AIP signature or lower-layer peer authentication).

When Agent Cards are disseminated through intermediaries (directory agents, DHT, gossip), the intermediary MUST verify the Agent Card's authorship before storing and serving it. Verification methods include checking that the "id" field matches the AITP source URI of the adp.advertise call, or — preferably — verifying the Ed25519 "signature" field defined in Section 3.2. Implementations SHOULD include "seq" and "signature" in Agent Cards that will be disseminated beyond direct AITP associations.

7.2. Capability Enumeration

The tools array and skill tags reveal the methods an agent supports. An adversary can invoke adp.describe to enumerate capabilities and then probe for vulnerabilities in specific methods.

Agents concerned with capability enumeration MAY return a partial Agent Card in response to adp.describe from unrecognized or unauthenticated callers, omitting sensitive tools while preserving the required fields (id, name).

7.3. Stale Description Attacks

An adversary who has obtained an old Agent Card can replay it to make consumers believe the agent still offers capabilities it has since revoked. The "seq" field (Section 3.2), "metadata.updated_at" timestamp, and "metadata.ttl" fields provide freshness signals. When "seq" and "signature" are present, consumers can cryptographically reject stale cards: a card with a lower "seq" MUST NOT supersede one

with a higher "seq" carrying a valid signature. Consumers SHOULD prefer the most recently retrieved Agent Card and SHOULD re-fetch after TTL expiration.

7.4. Overcommitment

An agent may advertise capabilities it cannot deliver (false advertising). ADP does not prevent this; companion reputation protocols provide post-hoc accountability. Callers SHOULD verify critical capabilities at invocation time rather than relying solely on the Agent Card.

7.5. Privacy

Agent Cards may reveal information about an agent’s operational capacity, supported languages, and method inventory. Agents that require privacy SHOULD limit the fields they include in their Agent Card and MAY return different Agent Card subsets to different callers based on authorization level.

7.6. Document Size

Agent Cards are carried as AITP request/response bodies, constrained by AIP’s maximum message size (65535 octets). Implementations MUST NOT generate Agent Cards larger than 65535 octets. Implementations SHOULD keep Agent Cards compact by omitting unused optional fields rather than including them with empty values.

If an Agent Card with all optional fields populated exceeds the AIP maximum message size, implementations SHOULD use one of the mitigation strategies defined in AITP [I-D.song-anp-aitp]: (1) deliver the Agent Card via an AITP STREAM exchange, or (2) use the adp.describe "fields" parameter to request a subset of the Agent Card across multiple REQUEST/RESPONSE round trips. For adp.discover, implementations SHOULD use the "limit" parameter to bound result set size per response.

8. IANA Considerations

This document has no IANA actions.

The AITP method names (adp.describe, adp.advertise, adp.discover) and endpoint protocol identifiers (aitp, http+json, grpc, ws) are conventions within the ANP protocol suite and are not drawn from IANA-managed registries. Should a formal AITP method name registry be established by a future document, the method names defined here are candidates for registration in that registry.

9. Implementation Status

Per [RFC7942], this section records known implementations.

9.1. ClawNet

Organization ChatChatTech

URL https://github.com/ChatChatTech/ClawNet

Description Go implementation of the full ANP suite. Agent descriptions are currently stored as two structures (config.Profile and store.AgentResume) that map to the Agent Card fields defined in this document. Dissemination uses DHT (clawnet-profile namespace) with Ed25519-signed records and GossipSub (/clawnet/resumes topic) with periodic re-broadcast. Discovery uses a five-factor weighted scoring algorithm (tag match, semantic overlap, reputation, availability, rating) matching Section 5.1. The reference implementation is being updated to unify Profile and Resume into a single Agent Card structure and to register the three ADP methods on AITP.

Maturity Alpha

Coverage Implemented:

* Profile and Resume data structures covering all core Agent Card fields
* DHT publication with Ed25519 signature verification
* GossipSub resume broadcast (5-minute interval)
* REST API for profile CRUD (GET/PUT /api/profile)
* Five-factor discovery scoring algorithm
* 60+ standard skill tags with alias normalization
* MCP tool exposure (17 tools)

Under integration:

* Unified Agent Card struct (merge Profile + Resume)
* adp.describe, adp.advertise, adp.discover AITP method handlers
* Agent Card signing via metadata proof

* HTTP well-known endpoint (/.well-known/agent-card.json)

Language Go

License Open source

Contact ink@chatchat.space

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[I-D.song-anp-aip] Song, J., "Agent Internet Protocol (AIP)", Work in Progress, Internet-Draft, draft-song-anp-aip-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-song-anp-aip-01>.

[I-D.song-anp-aitp] Song, J., "Agent Invocation Transport Protocol (AITP)", Work in Progress, Internet-Draft, draft-song-anp-aitp-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-song-anp-aitp-00>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

10.2. Informative References

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[I-D.song-anp-ans] Song, J., "Agent Name System (ANS)", Work in Progress, Internet-Draft, draft-song-anp-ans-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-song-anp-ans-00>.

Appendix A. Ecosystem Mappings (Informative)

This appendix documents how Agent Card fields map to three external agent description formats. These mappings are informative; external formats evolve independently.

A.1. Google A2A AgentCard

Agent Card	A2A Field	Rule
name	name	Direct
description	description	Direct
endpoints[0].uri	url	First endpoint URI
version	version	Direct
tools[].name	skills[].id	Direct
tools[].description	skills[].description	Direct
tools[].streaming	capabilities.streaming	any(tools[].streaming)

Table 12: Agent Card to A2A AgentCard Mapping

A.2. MCP Server Description

Agent Card	MCP Field	Rule
tools[].name	tools[].name	Direct
tools[].description	tools[].description	Direct
tools[].input_schema	tools[].inputSchema	Direct (camelCase key)

Table 13: Agent Card to MCP Tool List Mapping

MCP defines no output_schema, streaming, or idempotent fields; these are omitted in the MCP projection.

A.3. OASF AgentDescriptor

Agent Card	OASF Field	Rule
name	metadata.name	Direct
description	spec.description	Direct
skills	metadata.labels.skills	Join with ","
tools[].name	spec.capabilities[].name	Direct
tools[].input_schema	spec.capabilities[].inputSchema	Direct
endpoints[].uri	spec.endpoints[].url	Rename key
version	metadata.labels.version	Direct

Table 14: Agent Card to OASF AgentDescriptor Mapping

OASF has no reputation, credentials, or streaming fields; these are omitted in the OASF projection. Reverse import from OASF loses only ClawNet-specific extension fields.

Appendix B. Implementation Alignment Notes

This appendix documents the relationship between this specification and the reference implementation in the ClawNet codebase.

Spec Concept	Go Symbol / Location	Notes
Agent Card (core)	config.Profile + store.AgentResume	To be unified into single AgentCard struct
Agent Card (DHT)	p2p.ProfileRecord, p2p.ProfileRecordWire	Ed25519-signed envelope
Tool Descriptor	mcp.ToolDefinition (17 tools)	MCP tool definitions map directly
Skill Tags	discovery.StandardTags, discovery.TagAliases	60+ tags, 20+ alias mappings
Tag Matching	discovery.TagOverlap()	Jaccard-style overlap
Discovery Scoring	discovery.RankCandidates(), lob.Discover()	Five-factor weighted algorithm
DHT Publication	p2p.Node.PublishProfile()	DHT namespace: /clawnet-profile/
GossipSub Broadcast	daemon.publishResume()	Topic: /clawnet/ resumes, 5-min interval
REST API	daemon.handleGetProfile, handleUpdateProfile	GET/PUT /api/ profile
Endpoint Descriptor	ANS agent:// URI bindings	Protocol = "aitp" equivalent
Cold Start	discovery.RankCandidates()	+10 reputation bonus if tasks < 5

Table 15: Spec-Implementation Mapping

Authors' Addresses

Jinke Song
Dept. of CSE, Hong Kong University of Science and Technology
Email: ink@chatchat.space

Mu Yuan
Dept. of IE, The Chinese University of Hong Kong
Email: muyuan@cuhk.edu.hk