Network Working Group
Internet-Draft
Intended status: Informational
Expires: 7 August 2026

R. Pioli
Independent

February 2026

# Agent Registration and Discovery Protocol (ARDP)
## draft-pioli-agent-discovery-00

## Abstract

This document specifies the Agent Registration and Discovery Protocol (ARDP), a lightweight protocol for registering, discovering, and reaching autonomous software agents in distributed and federated environments. ARDP provides stable agent identities, dynamic endpoint resolution, capability advertisement (including protocol selection among MCP, A2A, HTTP, and gRPC), minimal presence signaling, and a security-first discovery control plane. ARDP is transport-agnostic and complementary to existing agent interaction protocols.

## Status of This Memo

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

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

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

This Internet-Draft will expire on 5 August 2026.

## Copyright Notice

Table of Contents



## 1. Introduction

Autonomous and semi-autonomous software agents introduce challenges in discoverability, reachability, and interoperability. Agents may be ephemeral, mobile across execution environments, and implemented by heterogeneous vendors.

ARDP addresses stable addressing of agents whose runtime location changes, authorized discovery by identity and declared capabilities, capability-driven selection among interaction protocols (e.g., MCP, A2A, HTTP, gRPC), and minimal, privacy-aware presence signaling.

## 2. Design Goals

Stable Identity; Dynamic Reachability; Minimalism (control plane only); Security by Default; Federation-Friendly; Extensibility.

## 3. Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL are to be interpreted as described in RFC 2119 and RFC 8174.

* Agent: Autonomous software entity capable of initiating and receiving interactions.

* Agent Identifier (AID): Stable, namespaced identifier of the form agent:<local-id>@<authority>.

* Registrar: Service that accepts agent registrations and maintains bindings.

* Resolver: Service that resolves an AID to active endpoints.

* Endpoint: Network location and protocol tuple through which an agent can be reached.

* Capability: Declarative description of supported protocols and interaction modes.

## 4. Architecture Overview

ARDP defines a logical control plane composed of registrars and resolvers. Agents register their presence and capabilities with a registrar. Authorized clients query resolvers to obtain endpoint and capability information.

## 5. Agent Identity

Each agent SHALL have a unique AID. The authority component denotes the administrative authority responsible for the identity.

Agents MUST prove control of an AID during registration using cryptographic credentials bound to that identity.

### 5.1. AID Syntax

An ARDP Agent Identifier (AID) MUST follow this grammar (ABNF per RFC 5234):

```
aid              = "agent:" local-id "@" authority
local-id         = 1*( ALPHA / DIGIT / "_" / "-" / "." / "/" )
authority        = dns-name / internal-name / opaque-authority
dns-name         = 1*( ALPHA / DIGIT / "-" / "." )  ; see IDNA2008 notes
internal-name    = 1*( ALPHA / DIGIT / "-" / "." )
opaque-authority = "tenant-" 1*( ALPHA / DIGIT / "-" )
```

Note: When internationalized domain names are used, implementers should follow the IDNA2008 RFC series.

### 5.2. Canonical Form

The canonical AID string is:

1. The literal prefix agent: in lowercase.

2. The authority normalized to lowercase (including IDNA2008 normalization where applicable); internal-name is lowercased as-is.

3. local-id is case-sensitive and MUST be preserved as sent.

4. AIDs MUST be compared using the canonical form.

## 6. Registration Model

### 6.1. REGISTER

A REGISTER request includes: AID; one or more endpoints; capability document (versioned); TTL; and cryptographic proof.

Registrations are soft-state and MUST be refreshed before expiration.

### 6.2. DEREGISTER

An agent MAY explicitly remove its registration.

### 6.3. Registration Semantics

Registration is idempotent on (aid, binding_id). If a client sends the same (aid, binding_id), the server MUST treat it as a refresh.

If a client sends the same aid with a different binding_id, the server MUST return a conflict error unless the client has registry:override scope.

### 6.4. TTL and Refresh

The server MUST be authoritative for expires_at and SHOULD return it in responses.

Clients SHOULD refresh at <= 0.5 * ttl with random jitter.

The server MUST define and enforce TTL bounds and advertise them in a metadata resource (recommended: /.well-known/ardp/meta).

### 6.5. Meta Resource (Minimal)

This section defines a minimal metadata resource for deployments using HTTPS bindings.

Path: /.well-known/ardp/meta

The response includes TTL bounds and supported auth profiles:

```
{
  "ttl_min": 30,
  "ttl_max": 3600,
  "auth_profiles": ["jws-proof-of-control"],
  "supported_protocols": ["MCP","A2A","HTTP","gRPC"]
}
```

## 7. Discovery and Resolution

### 7.1. RESOLVE

RESOLVE maps an AID to active endpoints and capabilities. Access MUST be authorized.

### 7.2. QUERY

QUERY allows authorized discovery by capability or namespace. Results SHOULD be minimized to prevent metadata leakage.

### 7.3. Privacy Defaults and Redaction

By default, QUERY returns only aid and status. Clients MAY request full details via a detail=full parameter.

If redaction applies, the server MUST omit restricted fields and include "redacted": true in the response.

## 8. Capabilities

Capability documents MAY include supported protocols (MCP, A2A, HTTP, gRPC), transport bindings, authentication mechanisms, modalities, rate or cost hints, and protocol-specific metadata. Capabilities are declarative and do not imply authorization.

### 8.1. Capability Bindings

Capabilities MUST include protocol-specific bindings when a protocol is declared.

## 9. Presence and Health

Presence is limited to: online, offline, degraded.

## 10. Security Considerations

Threats include identity spoofing, registration poisoning, unauthorized discovery, replay and downgrade attacks, and registrar compromise.

Mitigations include cryptographic identity proof, signed registrations, strict authorization, rate limiting, and audit logging.

### 10.1. Proof of Control (Mandatory Profile)

Clients MUST present a JWS-signed proof for register and refresh.

The signed payload is the canonical JSON of the registration body plus a server-provided nonce and an issued_at timestamp (RFC 3339).

Servers MUST verify the JWS using a JWKS key set (RFC 7517). Replay windows MUST be enforced.

### 10.2. Authorization Scopes

Operations require scopes such as: registry:register, registry:refresh, registry:resolve, registry:query, and registry:deregister.

## 11. Federation

Registrars MAY federate across domains via explicit trust relationships and policy agreements.

### 11.1. Federation Profile (Minimum)

Federation is allowed only between explicit trust anchors.

Responses from remote registrars MUST include provenance fields: origin_authority, origin_registrar_id, origin_signature.

Caches MUST honor remote TTL and mark records as federated.

## 12. Relationship to Existing Protocols

ARDP complements, and does not replace, agent interaction protocols such as MCP and A2A.

## 13. Wire Format Sketch (Non-Normative)

JSON over HTTPS is shown as an example binding. Alternative encodings (e.g., CBOR, gRPC) are possible.

### 13.1. Error Model

Servers MUST return errors with: code (stable error code), message (human-readable), and correlation_id (for tracing).

Required codes: invalid_aid, unauthorized, forbidden, conflict, not_found, expired.

## 14. IANA Considerations

### 14.1. Well-Known URI

IANA is requested to register the following URI suffix per RFC 8615 in the "Well-Known URIs" registry:

URI suffix: ardp

Change controller: IETF

Specification document(s): This document

### 14.2. Agent Identifier Namespace

The agent: prefix is used as an internal identifier namespace and is not registered as a URI scheme in this version.

## 15. Capability Schema v0

A minimal JSON schema for capability documents is provided as a companion artifact in the GitHub mirror.

## 16. Open Issues

Capability schema evolution; Privacy-preserving discovery; Federation bootstrapping.

## 17. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", 1997.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", 2017.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", 2002.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", 2010.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", 2015.

[RFC7517] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Key (JWK)", 2015.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", 2019.

## Author’s Address

Roberto Pioli
Independent
Email: roberto.pioli@gmail.com