Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 17 September 2026

B. Nemethi, Ed.
Open Agent Registry, Inc.

16 March 2026

Agent Identity and Discovery (AID)
draft-nemethi-aid-agent-identity-discovery-00

Abstract

Agent Identity and Discovery (AID) is a minimal, DNS-first discovery protocol for locating agent service endpoints. Given a domain name, an AID client queries a DNS TXT record at the well-known subdomain _agent.<domain> and learns the service endpoint URI, protocol token, authentication hint, and optional metadata for that agent.

This document defines the AID v1.2 record format, client discovery algorithm, exact-host lookup rules, security requirements, and IANA registrations for the ‘_agent‘ DNS node name and the ‘agent‘ service name. AID is intentionally small. After discovery, protocol-specific mechanisms such as MCP or A2A handle communication and capability negotiation.

Status of This Memo

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

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

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

This Internet-Draft will expire on 17 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Applications that need to locate and connect to an agent often rely on out-of-band configuration, centralized directories, or protocol-specific discovery mechanisms. AID defines a single DNS-based bootstrap point that answers one question: given a domain, where is the agent and which protocol should a client speak?

AID uses a TXT record at the well-known DNS name _agent.<domain>. The record is small, versioned, and protocol-agnostic. It tells a client where the agent is located, which protocol token applies, what authentication hint to expect, and whether endpoint proof is required.

* Zero configuration: a user supplies a domain and the client performs discovery.
* Decentralized deployment: the protocol uses standard DNS publication.
* Protocol-agnostic bootstrap: AID identifies the next protocol, rather than replacing it.
* Stable evolution path: the ‘_agent‘ label remains stable across protocol versions.

This document specifies the wire format and client behavior for AID v1.2 and requests two IANA registrations. The RFC 8552 registration covers the deployed TXT-based discovery label ‘_agent‘, while the RFC 6335 service-name-only registration reserves ‘agent‘ for possible future SRV-based discovery under the same naming family.

2. Conventions and Terminology

2.1. 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.

2.2. Terms

AID Client
: Software that performs AID discovery for a domain.

Provider
: The entity that controls a domain and publishes the AID TXT record.

_agent subdomain
: The DNS name ‘_agent.<domain>‘ where the AID TXT record is published.

A-label
: The Punycode representation of an Internationalized Domain Name as described in [RFC5890].

3. AID Record Format

A provider MUST advertise its agent service by publishing a single TXT record at ‘_agent.<domain>‘. For AID v1.x, agents MUST use DNS TXT records for discovery. TXT is the deployed discovery record type because it is widely available across DNS providers and registrars. Future versions of AID may adopt a more structured record type, but the ‘_agent‘ label remains stable; see Section 7.1.

3.1. Syntax and Parsing

The record MUST be a single semicolon-delimited string of ‘key=value‘ pairs. Clients SHOULD ‘trim()‘ leading and trailing whitespace from keys and values. Clients MUST ignore unknown keys.

If a DNS server splits the TXT record into multiple 255-octet strings, the client MUST concatenate them in order before parsing. Providers SHOULD keep the full record under 255 octets when practical.

Key comparison is case-insensitive. Clients MUST recognize the single-letter lowercase alias for every defined key. A record MUST NOT include both a long key and its alias. Providers SHOULD emit the short-key form ‘v,u,p,a,s,d,e,k,i‘ for AID v1.x records.

3.2. Defined Keys

version (‘v‘) Required
: Specification version. For AID v1.x this MUST be ‘aid1‘. Example: ‘v=aid1‘.

uri (‘u‘) Required
: Absolute endpoint URL or local locator. Allowed schemes depend on the selected protocol token; see Appendix B. Example: ‘u=https://api.example.com/mcp‘.

proto (‘p‘) Required
Protocol token from Appendix B. Example: ‘p=mcp‘.

auth (‘a‘) Recommended
Authentication hint token from Appendix A. Example: ‘a=pat‘.

desc (‘s‘) Optional
Short human-readable description, up to 60 UTF-8 bytes. Example: ‘s=Primary AI Gateway‘.

docs (‘d‘) Optional
Absolute HTTPS URL for human-readable documentation. Example: ‘d=https://docs.example.com/agent‘.

dep (‘e‘) Optional
ISO 8601 UTC timestamp indicating deprecation. Example: ‘e=2026-01-01T00:00:00Z‘.

pka (‘k‘) Optional
Multibase-encoded Ed25519 public key for endpoint proof. Example: ‘k=z7rW8rTq8o4mM6vVf7w1k3m4uQn9p2YxCAbcDeFgHiJ‘.

kid (‘i‘) Conditional
Rotation identifier, 1 to 6 lowercase alphanumeric characters. Required when ‘pka‘ is present. Example: ‘i=g1‘.

3.3. Examples

```
_agent.example.com. 300 IN TXT (
  "v=aid1;u=https://api.example.com/mcp;"
  "p=mcp;a=pat;s=Example AI Tools"
)
```

Figure 1: Remote MCP Agent

```
_agent.grafana.com. 300 IN TXT (
  "v=aid1;u=docker:grafana/mcp:latest;"
  "p=local;a=pat;s=Run Grafana agent locally"
)
```

Figure 2: Local Agent via Docker

```
_agent.example.com. 300 IN TXT (
  "v=aid1;p=mcp;u=https://api.example.com/mcp;"
  "k=z7rW8rTq8o4mM6vVf7w1k3m4uQn9p2YxCAbcDeFgHiJ;i=g1;"
  "d=https://docs.example.com/agent;"
  "e=2026-01-01T00:00:00Z;s=Secure AI Gateway"
)
```

Figure 3: Remote MCP with PKA and Metadata

```
_agent.local.test. 300 IN TXT (
  "v=aid1;p=zeroconf;"
  "u=zeroconf:_mcp._tcp;s=Local Dev Agent"
)
```

Figure 4: Local Zeroconf Example

4. Client Discovery Procedure

4.1. Discovery Algorithm

When an AID client is given a domain, it MUST perform the following steps:

1. Normalize the domain. If the input contains non-ASCII labels, convert them to A-label form as described in [RFC5890].

2. Query the TXT record at ‘_agent.<exact-host-user-entered>‘. The client MUST NOT walk up the DNS hierarchy. If no record is found or the lookup fails, the client MAY try the ‘.well-known‘ fallback on the same exact host; see Appendix E.

3. Parse the TXT answer set as semicolon-delimited ‘key=value‘ records. Key comparison MUST be case-insensitive, and clients MUST recognize single-letter aliases for all defined keys. If exactly one valid AID record is present at the queried DNS name, use it. If more than one valid AID record is present at the same queried DNS name, the client MUST fail due to ambiguity. Malformed answers do not matter when there is exactly one valid AID record.

4. If ‘docs‘ is present, the client MAY display it. If ‘dep‘ is in the future, the client SHOULD warn. If ‘dep‘ is in the past, the client SHOULD fail gracefully.

5. If ‘pka‘ is present, the client MUST perform the endpoint-proof procedure in Appendix D. Providers MUST publish ‘kid‘ whenever ‘pka‘ is present.

6. If validation succeeds, return the discovered details. If the client does not support the selected protocol token, it MUST fail with the appropriate error.

4.2. Standard Client Error Codes

Code	Name	Meaning
1000	ERR_NO_RECORD	No ‘_agent‘ TXT record was found for the domain.
1001	ERR_INVALID_TXT	A record was found but is malformed or missing required keys.
1002	ERR_UNSUPPORTED_PROTO	The record is valid, but the client does not support the selected protocol.
1003	ERR_SECURITY	Discovery failed due to a security policy or verification failure.
1004	ERR_DNS_LOOKUP_FAILED	The DNS query failed for a network-related reason.
1005	ERR_FALLBACK_FAILED	The ‘.well-known‘ fallback failed or returned invalid data.

Table 1: Client Error Codes

4.3. Exact-Host Semantics and Explicit Delegation

Discovery is exact-host by default. If the application asks for ‘app.team.example.com‘, the canonical AID query name is ‘_agent.app.team.example.com‘.

Clients MUST NOT walk up the DNS hierarchy looking for ‘_agent.team.example.com‘ or ‘_agent.example.com‘. If an operator wants a child host to inherit a shared record, that inheritance MUST be expressed in DNS for the exact queried name, for example by using a ‘CNAME‘ at the child host’s ‘_agent‘ label.

```
_agent.app.team.example.com. 300 IN TXT (
  "v=aid1;p=mcp;"
  "u=https://app.team.example.com/mcp"
)

_agent.app.team.example.com. 300 IN CNAME (
  _agent.shared.team.example.com.
)
_agent.shared.team.example.com. 300 IN TXT (
  "v=aid1;p=mcp;"
  "u=https://gateway.team.example.com/mcp"
)
```

Figure 5: Exact-Host and Delegated Examples

Protocol-specific lookups follow the same exact-host rule. If an application explicitly asks for a given protocol token, a client MAY query ‘_agent._<proto>.<exact-host>‘ before the base ‘_agent.<exact-host>‘ name, but it MUST NOT query parent hosts implicitly.

4.4. Exposing Multiple Protocols

The canonical location for discovery is the base record ‘_agent.<domain>‘. Providers MAY additionally expose distinct agent services on protocol-specific subdomains of the form ‘_agent._<proto>.<domain>‘.

```
_agent._mcp.example.com. 300 IN TXT (
  "v=aid1;p=mcp;"
  "u=https://api.example.com/mcp"
)
_agent._a2a.example.com. 300 IN TXT (
  "v=aid1;p=a2a;"
  "u=https://api.example.com/a2a"
)
```

Figure 6: Protocol-Specific Records

By default, clients query the base record. When a specific protocol token is explicitly requested by the calling application, clients MAY first query the corresponding protocol-specific name and then fall back to the base record for the same exact host.

5. Security Considerations

AID publishes public discovery metadata. The TXT record MUST NOT contain secrets. The protocol depends on DNS integrity, TLS for remote endpoints, and explicit client safeguards for local execution.

* Providers SHOULD sign AID records with DNSSEC when available. Clients SHOULD perform DNSSEC validation when DNSSEC-signed answers are available.

* A ‘remote‘ agent’s ‘uri‘ MUST use ‘https://‘. Clients MUST perform standard TLS certificate and hostname validation as described in [RFC9525].

* When ‘pka‘ is present, clients MUST verify endpoint control of the private key as described in Appendix D. Providers MUST publish ‘kid‘ when ‘pka‘ is present. Clients SHOULD warn if a previously observed ‘pka‘ disappears.

* Clients that support ‘proto=local‘ MUST implement the safeguards described below.

* Clients MUST NOT automatically follow cross-origin redirects from a discovered endpoint.

Clients that support ‘proto=local‘ MUST implement the following safeguards:

1. Explicit consent: before first execution, the client MUST display the full resolved command and require explicit confirmation.

2. Integrity check: the client MUST compute and cache a cryptographic fingerprint of the ‘uri‘ and ‘proto‘ values. If these values change on a later lookup, the client MUST re-trigger the full consent process.

3. No shell interpretation: arguments derived from the ‘uri‘ MUST be passed atomically to the underlying OS execution call to prevent command injection.

4. No nested discovery: the client MUST reject a ‘local‘ execution ‘uri‘ that could be interpreted as a command that initiates another AID discovery request.

5. Sandboxing: clients SHOULD run local agents in a sandboxed environment with minimum necessary permissions.

If an initial request to the discovered ‘uri‘ returns an HTTP redirect (‘301‘, ‘302‘, ‘307‘, or ‘308‘) to a different origin, clients SHOULD treat this as a potential security risk. Clients MUST NOT follow such cross-origin redirects automatically. Implementations MAY either terminate with ‘ERR_SECURITY‘ or require explicit user confirmation before proceeding.

5.1. Threat Model

AID addresses DNS spoofing and cache poisoning with optional DNSSEC validation, endpoint impersonation with optional public-key attestation, downgrade detection with remembered key state, command injection risks for local execution, and cross-origin redirect abuse for remote endpoints.

Compromised authoritative DNS servers, active network attackers beyond the protections of TLS, and revocation beyond DNS record update remain outside the direct scope of AID v1.2.

5.2. Enterprise Policy Modes

Clients that expose enterprise controls SHOULD provide policy presets and MAY expose the underlying policy knobs directly.

Preset	PKA	DNSSEC	Well-known	Downgrade
balanced	if-present	prefer	auto	warn
strict	require	require	disable	fail

Table 2: Normative Policy Presets

The underlying policy knobs are:

* PKA policy: ‘if-present | require‘
* DNSSEC policy: ‘off | prefer | require‘
* Well-known policy: ‘auto | disable‘
* Downgrade policy: ‘off | warn | fail‘

Policy semantics are as follows:

* PKA ‘require‘: discovery MUST fail with ‘ERR_SECURITY‘ if the selected record does not publish ‘pka‘ and ‘kid‘.
* DNSSEC ‘prefer‘: clients SHOULD continue when DNSSEC cannot be validated, but SHOULD surface a warning.
* DNSSEC ‘require‘: clients MUST fail with ‘ERR_SECURITY‘ when DNSSEC validation is unavailable or unsuccessful for the selected DNS answer.

* Well-known ‘disable‘: clients MUST NOT use ‘/.well-known/agent‘ fallback.

* Downgrade ‘warn‘: if a previously seen ‘pka‘ disappears or ‘pka‘ or ‘kid‘ changes, clients SHOULD surface a warning.

* Downgrade ‘fail‘: if a previously seen ‘pka‘ disappears or ‘pka‘ or ‘kid‘ changes, clients MUST fail with ‘ERR_SECURITY‘.

If discovery succeeds only through ‘.well-known‘, that result cannot satisfy a policy that requires DNSSEC validation.

6. DNS and Caching

Providers are RECOMMENDED to publish ‘_agent‘ TXT records with a TTL between 300 and 900 seconds. Clients MUST respect the received TTL and MUST NOT cache the record longer than that.

7. Future Path

AID v2 or later may adopt a more structured DNS record type such as SRV [RFC2782] or SVCB [RFC9460], depending on operational deployment support. TXT remains the canonical mechanism for AID v1.x.

Formal requests are expected for the ‘_agent‘ underscored DNS node name under [RFC8552] and the ‘agent‘ service name under [RFC6335].

7.1. Label Stability

The stable DNS label for AID is ‘_agent‘. Record types may evolve, but the discovery label remains ‘_agent‘ across versions and the record version field identifies the wire-format expectations. This is the main reason the RFC 8552 registration is valuable even if later AID versions use a different DNS RR type.

8. Registries and Governance

To support interoperability, token registries and community resources are maintained publicly.

* Token registries: the canonical lists for ‘auth‘ and ‘proto‘ tokens are maintained at https://github.com/agentcommunity/aid-tokens (https://github.com/agentcommunity/aid-tokens). Additions require a pull request and are governed by a first-come, first-served policy with expert review.

* Global index: an open DNS crawler and community dashboard showing AID adoption is maintained at https://github.com/agentcommunity/aid-registry (https://github.com/agentcommunity/aid-registry).

9. IANA Considerations

This document requests registration in two IANA registries. The registrations serve different purposes. The RFC 8552 request covers AID v1.x TXT discovery at ‘_agent.<domain>‘. The RFC 6335 request reserves the service name ‘agent‘ for potential future DNS service discovery usage under ‘_agent._tcp.<domain>‘.

9.1. Registration in the Underscored and Globally Scoped DNS Node Names Registry

IANA is requested to register the following entry in the "Underscored and Globally Scoped DNS Node Names" registry established by [RFC8552].

RR Type	TXT
Node Name	_agent
Reference	This document

Table 3: Requested RFC 8552 Registration

The ‘_agent‘ node name is used exclusively for Agent Identity and Discovery. A single TXT record published at ‘_agent.<domain>‘ contains semicolon-delimited key/value pairs that identify an agent endpoint, protocol token, and optional metadata as defined in Section 3 and Section 4.

The requested node name is specific to AID and does not reserve the broader concept of agents or agent-related discovery generally. Protocol-specific labels of the form ‘_agent._<proto>.<domain>‘ are subordinate names beneath the registered ‘_agent‘ node and do not require separate global registration.

9.2. Registration in the Service Name and Transport Protocol Port Number Registry

IANA is requested to register the following service-name-only entry in the "Service Name and Transport Protocol Port Number" registry defined by [RFC6335].

Service Name	agent
Transport Protocol(s)	tcp
Description	Agent Identity and Discovery (AID): DNS-based discovery of agent service endpoints
Port Number	N/A
Assignment Notes	No port number is requested. This is a service-name-only registration intended to establish ‘agent‘ for possible future SRV-based discovery under ‘_agent._tcp.<domain>‘ while preserving the stable ‘_agent‘ label.
Reference	This document

Table 4: Requested RFC 6335 Registration

10. Normative References

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

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[RFC6335] Cotton, M., "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", RFC 6335, August 2011, <https://www.rfc-editor.org/rfc/rfc6335>.

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

[RFC8552] Sullivan, A., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", RFC 8552, March 2019, <https://www.rfc-editor.org/rfc/rfc8552>.

[RFC9421] Fielding, R., "HTTP Message Signatures", RFC 9421, November 2023, <https://www.rfc-editor.org/rfc/rfc9421>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, November 2023, <https://www.rfc-editor.org/rfc/rfc9525>.

11. Informative References

[RFC2782] Gulbrandsen, A., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000, <https://www.rfc-editor.org/rfc/rfc2782>.

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

[RFC9460] Schwartz, B., "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

Appendix A. Authentication Scheme Registry

All authentication tokens are case-sensitive and use lowercase ASCII.

* none
* pat
* apikey
* basic
* oauth2_device
* oauth2_code
* mtls
* custom

Appendix B. Protocol Registry

All protocol tokens are case-sensitive and use lowercase ASCII.

Token	Meaning	Allowed 'uri' scheme(s)
mcp	Model Context Protocol	https://
a2a	Agent-to-Agent Protocol	https://
openapi	OpenAPI specification document	https://
grpc	gRPC over HTTP/2 or HTTP/3	https://
graphql	GraphQL over HTTP	https://
websocket	WebSocket transport	wss://
local	The agent runs locally on the client machine	docker:, npx:, pip:
zeroconf	mDNS or DNS-SD service discovery	zeroconf:<service_type>
ucp	Universal Commerce Protocol	https://

Table 5: Protocol Tokens

Appendix C. Client Error Constants

For cross-language SDK consistency, clients SHOULD use the following numeric constants.

Constant	Value
ERR_NO_RECORD	1000
ERR_INVALID_TXT	1001
ERR_UNSUPPORTED_PROTO	1002
ERR_SECURITY	1003
ERR_DNS_LOOKUP_FAILED	1004
ERR_FALLBACK_FAILED	1005

Table 6: Error Constant Values

Appendix D. PKA Handshake

When ‘pka‘ is present, the client MUST verify control of the corresponding private key using HTTP Message Signatures with Ed25519 as specified in [RFC9421].

```
function performPKAHandshake(uri, pka, kid):
    nonce = generateRandomBytes(32)
    challenge = base64urlEncode(nonce)
    headers = {
        "AID-Challenge": challenge,
        "Date": currentUTCTime()
    }
    response = sendGET(uri, headers)

    if response.status != 200:
        failWith(ERR_SECURITY)

    sigInput = response.headers["Signature-Input"]
    signature = response.headers["Signature"]

    created = parseCreated(sigInput)
    if |currentTime() - created| > 300 seconds:
        failWith(ERR_SECURITY)

    pubKey = multibaseDecode(pka)
    if not verifyEd25519(signature, sigInput.coveredFields, pubKey):
        failWith(ERR_SECURITY)
```

Figure 7: PKA Handshake Pseudocode

Providers MUST include ‘kid‘ whenever ‘pka‘ is present. Clients SHOULD warn on downgrade if a previously observed ‘pka‘ disappears.

Appendix E. .well-known Fallback

AID is a DNS-based discovery protocol. The fallback defined here is a non-normative convenience for environments where publishing DNS TXT records is difficult. It does not change the RFC 8552 scope of ‘_agent‘.

* Path: ‘GET https://<domain>/.well-known/agent‘
* Format: JSON mirroring the TXT keys, including aliases
* Security: TLS certificate validation still applies, and ‘pka‘ still applies when present
* Client behavior: use DNS first and fall back only on ‘ERR_NO_RECORD‘ or ‘ERR_DNS_LOOKUP_FAILED‘
* Error mapping: use ‘ERR_FALLBACK_FAILED‘ when the fallback fails or is invalid

The ‘.well-known‘ URI convention itself is described in [RFC8615].

Author’s Address

Balazs Nemethi (editor)
Open Agent Registry, Inc.
Email: balazs@agentcommunity.org
URI: https://agentcommunity.org