DNSOP
Internet-Draft
Intended status: Standards Track
Expires: 23 September 2026

A. Ferro
ApertoID

22 March 2026

ApertoID: DNS-Based Agent Identity Declaration Protocol
draft-ferro-dnsop-apertoid-00

Abstract

This document defines ApertoID, a DNS-based protocol that enables domain owners to declare authorized AI agents acting on their behalf, publish cryptographic keys for agent identity verification, and specify enforcement policies for unauthorized agents. ApertoID uses existing DNS TXT records under the "_apertoid" underscore-scoped domain name to provide a decentralized, standards-based mechanism for AI agent identity declaration and verification.

ApertoID defines two record types: a Policy Record analogous to DMARC that specifies domain-level enforcement behavior, and Agent Declaration Records analogous to DKIM key records that bind agent endpoints to Ed25519 public keys with mandatory expiration. A companion document [APERTOID-SIG] defines the HTTP request signing mechanism that enables agents to cryptographically prove their identity on each request.

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 23 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The rapid proliferation of AI agents acting autonomously on the internet has created a fundamental identity gap. When an AI agent claims to act on behalf of a domain (e.g., "I represent example.com"), no standard protocol exists to verify this claim. The current internet infrastructure — designed for human users operating browsers — provides no mechanism to distinguish legitimate AI agents from impersonators, to verify which domain authorized an agent, or to enforce policies when verification fails.

This gap is causing measurable harm. In the Amazon v. Perplexity litigation (2025-2026), an AI agent disguised itself as a standard web browser to access services under false pretenses. The Salesloft-Drift OAuth breach (2025) exploited over-permissioned machine tokens to compromise over 700 companies. Research indicates that non-human identities outnumber human identities by a factor of 144:1 in enterprise environments, yet 88% of organizations lack identity controls for AI systems.

Email faced an analogous identity problem two decades ago: any server could send email claiming any sender address. The solution was a layered DNS-based authentication framework: SPF [RFC7208] declared authorized sending IPs, DKIM [RFC6376] provided cryptographic signatures, and DMARC [RFC7489] unified them with enforcement policies. ApertoID applies the same proven architectural pattern to AI agent identity.

ApertoID is designed to complement, not replace, existing agent discovery mechanisms such as DNS-AID [I-D.mozleywilliams-dnsop-dnsaid] and agent authentication frameworks such as [I-D.klrc-aiagent-auth]. It also complements application-layer agent protocols such as the Model Context Protocol (MCP) and Agent-to-Agent (A2A) protocol. DNS-AID provides discovery ("find the agents for this domain"); [I-D.klrc-aiagent-auth] provides a composable authentication framework; ApertoID provides DNS-based authorization and identity declaration ("verify that this agent is genuinely authorized by this domain").

The ApertoID protocol builds on the same DNS infrastructure philosophy as the ApertoDNS Protocol [I-D.ferro-dnsop-apertodns-protocol], which modernized Dynamic DNS updates using well-known URIs and RESTful patterns. Both protocols demonstrate that DNS infrastructure can be extended through lightweight, deployable mechanisms without requiring changes to DNS servers or resolvers.

Additionally, the EU AI Act (Regulation 2024/1689), Article 50, requires AI systems interacting with natural persons to identify themselves as AI in a machine-readable format, effective August 2, 2026. ApertoID’s optional "type" field provides a DNS-based mechanism to satisfy this requirement.

1.1. 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. Comparison with Existing Approaches

Several related efforts address aspects of AI agent identity and discovery. ApertoID is designed to complement these efforts, not replace them:

DANE (RFC 6698) DANE authenticates TLS certificates for servers via TLSA DNS records. However, DANE binds identity to a TLS endpoint (server), not to an agent. Multiple agents may share a server, and a single agent may operate across multiple servers. ApertoID binds identity to the agent itself, independent of the underlying TLS infrastructure.

DNS-AID (draft-mozleywilliams-dnsop-dnsaid) DNS-AID provides DNS-

based discovery of agents associated with a domain using SVCB records. It answers "what agents does this domain have?" but does not provide cryptographic identity binding or enforcement policies. ApertoID answers "is this specific agent genuinely authorized by this domain?" The two are complementary.

AIMS (draft-klrc-aiagent-auth) AIMS proposes an agent authentication framework composing SPIFFE, WIMSE, and OAuth 2.0. It is designed for intra-organization authentication but does not address cross-domain trust bootstrapping via DNS. ApertoID provides the DNS-based trust anchor that frameworks like AIMS can build upon.

ANS (OWASP Agent Name Service) ANS is a centralized registry with PKI-based identity. ApertoID is fully decentralized, using existing DNS infrastructure with no central registry, making it deployable by any domain owner without third-party registration.

HTTP Message Signatures (RFC 9421) RFC 9421 provides a general-purpose HTTP signing framework. ApertoID-Signature (defined in the companion document [APERTOID-SIG]) is a deliberately simplified, single-purpose mechanism optimized for AI agent identity verification with Ed25519 only. It is designed to be implementable in minimal code without requiring the full complexity of RFC 9421's component identifiers and algorithm negotiation.

3. Terminology

Agent A software system that acts autonomously on behalf of a domain, typically an AI-powered service that interacts with external systems via HTTP-based protocols (e.g., MCP, A2A, REST APIs).

Declaring Domain The domain that publishes ApertoID DNS records to declare its authorized agents. Analogous to the "domain owner" in SPF/DKIM/DMARC.

Agent Selector A label that uniquely identifies an agent within a declaring domain. Used as the leftmost label in the agent declaration record name (e.g., "leadhunter" in "leadhunter._apertoid.example.com"). Analogous to the DKIM selector.

Verifier An entity that queries ApertoID DNS records to determine whether an agent is authorized by the claimed domain and, optionally, to verify the agent's cryptographic identity.

Policy Record A DNS TXT record at "_apertoid.<domain>" that

specifies the domain's ApertoID enforcement policy. Analogous to a DMARC record.

Agent Declaration Record A DNS TXT record at "<selector>._apertoid.<domain>" that declares an authorized agent with its endpoint URL, public key, and metadata. Analogous to a DKIM key record.

4. Protocol Overview

ApertoID operates in two layers, both implemented as DNS TXT records under the "_apertoid" underscore-scoped name:

1. *Policy Layer:* A single record at "_apertoid.<domain>" declares that the domain participates in ApertoID and specifies the enforcement policy (reject, warn, or none) for agents that fail verification. This is analogous to DMARC.

2. *Identity Layer:* One record per authorized agent at "<selector>._apertoid.<domain>" declares the agent's endpoint URL, Ed25519 public key, type classification, and expiration. This is analogous to DKIM key records.

A companion document [APERTOID-SIG] defines a third layer: the ApertoID-Signature HTTP header that agents attach to outgoing requests to prove their identity cryptographically. This document focuses exclusively on the DNS records and their semantics.

The verification flow proceeds as follows:

1. An agent makes a request claiming to represent a domain (e.g., via the ApertoID-Signature header or an out-of-band claim).

2. The verifier queries "_apertoid.<claimed_domain>" to obtain the policy record.

3. The verifier queries "<selector>._apertoid.<claimed_domain>" to obtain the agent declaration record.

4. The verifier checks that the agent's endpoint URL matches the declared URL, that the public key has not expired, and optionally that the agent's presented key matches the declared key.

5. If verification fails, the verifier applies the enforcement policy specified in the policy record.

Note: Without the cryptographic signature mechanism defined in [APERTOID-SIG], ApertoID provides authorization-level assurance only — the domain has declared which agents are authorized, but agents cannot actively prove their identity on each request. This is analogous to how SPF provides authorization (which IPs may send) without DKIM's cryptographic proof. Full identity verification requires both this document and [APERTOID-SIG].

5. Record Syntax

Both ApertoID record types use the same syntax: a DNS TXT record containing semicolon-separated tag-value pairs. This section defines the formal grammar using ABNF [RFC5234].

5.1. ABNF Definition

```
apertoid-record = version-tag *( ";" [WSP] tag-value )
version-tag     = "v" "=" "APERTOID1"
tag-value       = tag "=" value
tag             = 1*ALPHA
value           = 1*VCHAR
WSP             = SP / HTAB

; Policy Record specific tags
policy-tag      = "p" "=" ( "reject" / "warn" / "none" )
rua-tag         = "rua" "=" "mailto:" email-address
email-address   = 1*VCHAR "@" domain-name    ; per RFC 5321

; Agent Declaration Record specific tags
url-tag         = "url" "=" https-uri
keytype-tag     = "k" "=" "ed25519"
pubkey-tag      = "pk" "=" base64-ed25519
exp-tag         = "exp" "=" 1*DIGIT
type-tag        = "type" "=" ( "ai" / "human" / "hybrid" )
include-tag     = "include" "=" domain-name
status-tag      = "status" "=" "revoked"
prev-tag        = "prev" "=" "sig:" base64-ed25519-sig

; Value format definitions
base64-ed25519     = 44*44(BASE64CHAR)   ; 32 bytes = 44 chars
base64-ed25519-sig = 88*88(BASE64CHAR)   ; 64 bytes = 88 chars
BASE64CHAR         = ALPHA / DIGIT / "+" / "/" / "="
https-uri          = "https://" 1*URI-CHAR
URI-CHAR           = ALPHA / DIGIT / "-" / "." / "_" / "~" /
                     ":" / "/" / "?" / "#" / "[" / "]" / "@" /
                     "!" / "$" / "&" / "'" / "(" / ")" / "*" /
                     "+" / "," / ";" / "="
domain-name        = label *("." label)
label              = ALPHA *(ALPHA / DIGIT / "-")
```

Tags are case-insensitive. Values are case-sensitive unless otherwise specified. Whitespace around semicolons is OPTIONAL and MUST be ignored by parsers. Unknown tags MUST be ignored by verifiers to allow forward compatibility. The email-address production follows the addr-spec syntax defined in [RFC5321].

6. Policy Record

6.1. Record Location

The ApertoID Policy Record is a DNS TXT record published at:

_apertoid.<domain>

The underscore-scoped name "_apertoid" MUST be registered per [RFC8552]. A domain MUST NOT publish more than one ApertoID Policy Record. If multiple TXT records exist at this name, the verifier MUST select the record that begins with "v=APERTOID1" and discard others.

6.2. Example

```
_apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; p=reject; rua=mailto:apertoid@example.com"
```

6.3. Policy Tags

v (REQUIRED) Protocol version. MUST be "APERTOID1" for this specification. This tag MUST be the first tag in the record.

p (REQUIRED) Enforcement policy for agents that fail verification. Valid values:

reject The verifier SHOULD reject requests from agents that fail verification.

warn The verifier SHOULD accept but flag requests from agents that fail verification.

none The verifier SHOULD take no specific action on failure. Used for monitoring and gradual deployment.

Domain owners SHOULD deploy with "p=none" initially and progress to "p=reject" after confirming that legitimate agents pass verification.

rua (OPTIONAL) Reporting URI for aggregate verification reports. MUST be a "mailto:" URI. Verifiers that support reporting SHOULD send periodic aggregate reports to this address. The format of aggregate reports is outside the scope of this document.

7. Agent Declaration Record

7.1. Record Location

Each authorized agent is declared in a separate DNS TXT record at:

<selector>._apertoid.<domain>

The <selector> is a label chosen by the domain owner to identify the agent. Selectors MUST conform to the syntax of a DNS label: 1-63 characters, consisting of alphanumeric characters and hyphens, not starting or ending with a hyphen. Selectors are case-insensitive.

7.2. Example

```
leadhunter._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; url=https://agent.example.com/mcp;
   k=ed25519; pk=MCowBQYDK2VwAyEAexamplekeybase64here;
   type=ai; exp=1759276800"
```

Note: The record above is shown on multiple lines for readability. In practice, it is a single TXT record string. If the record exceeds 255 bytes, it MUST be split into multiple character-strings within a single TXT RDATA as specified in [RFC1035] Section 3.3.14.

7.3. Agent Declaration Tags

v (REQUIRED) Protocol version. MUST be "APERTOID1". MUST be the first tag.

url (REQUIRED, mutually exclusive with include) The canonical endpoint URL of the authorized agent. MUST be an HTTPS URI. The verifier compares this URL against the agent's actual endpoint. URL matching is performed as specified in Section 11.4.

k (RECOMMENDED) Key type. MUST be "ed25519" for this specification. Additional key types MAY be registered via the ApertoID Key Type Registry (Section 15.3). If present, the "pk" tag MUST also be present.

pk (RECOMMENDED) The agent's Ed25519 public key, encoded as unpadded Base64 per [RFC4648] Section 4. For Ed25519, this is exactly 44 characters (32 bytes encoded). This key is used by verifiers to authenticate the agent's identity via the ApertoID-Signature mechanism defined in [APERTOID-SIG].

exp (REQUIRED when k is present) Key expiration as a Unix timestamp (seconds since 1970-01-01T00:00:00Z). Verifiers MUST reject keys whose expiration has passed. Domain owners SHOULD set expiration to no more than 90 days in the future and rotate keys before expiration.

type (OPTIONAL) Agent type classification. Valid values are registered in the ApertoID Agent Type Registry (Section 15.2). Initial values:

ai Autonomous AI agent.

human Human-operated tool or interface.

hybrid AI-assisted system with human oversight.

This field supports compliance with regulations that require AI systems to identify themselves, such as EU AI Act Article 50(2).

include (OPTIONAL, mutually exclusive with url) Delegation to a third-party agent declaration. The value is a fully qualified domain name pointing to an Agent Declaration Record published by the third party. See Section 8.

status (OPTIONAL) Agent status. If set to "revoked", verifiers MUST treat this agent as unauthorized regardless of other fields. See Section 10.

prev (OPTIONAL) Key rotation continuity proof. Contains a signature of the new public key made with the old private key, prefixed by "sig:". See Section 10.2.

7.4. Record Size Considerations

A typical Agent Declaration Record with all recommended fields is approximately 170 bytes, fitting comfortably within a single 255-byte TXT character-string and well within the practical UDP DNS response size limit. Each agent has its own record at a distinct DNS name, so the number of agents per domain is not constrained by record size.

8. Delegation

When a domain uses third-party AI agent services, the domain owner delegates agent authorization by publishing an Agent Declaration Record with an "include" tag pointing to the third party's own ApertoID record.

```
; Domain delegates to a Salesforce agent
salesforce._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; include=agent1._apertoid.salesforce.com"

; Salesforce publishes the actual agent declaration
agent1._apertoid.salesforce.com. 3600 IN TXT
  "v=APERTOID1; url=https://agents.salesforce.com/crm;
   k=ed25519; pk=<base64-pubkey>; type=ai; exp=1759276800"
```

Verifiers MUST follow "include" references to resolve the final Agent Declaration Record. To prevent abuse and excessive DNS lookups:

* The maximum delegation depth is 2 (i.e., the original record plus one level of "include").

* A verifier MUST NOT follow more than 10 "include" references in total per verification attempt.

* Circular "include" references MUST be detected and treated as a verification failure.

The "include" tag is mutually exclusive with the "url" tag. A record MUST contain either "url" or "include", but not both.

9. Key Publication

9.1. Ed25519 Key Format

ApertoID uses Ed25519 [RFC8032] as its mandatory-to-implement key type. Ed25519 was chosen for compact key size (32 bytes / 44 Base64 characters), compact signature size (64 bytes), fast verification (approximately 50 microseconds), and wide implementation support across all major cryptographic libraries.

The public key is published as the raw 32-byte Ed25519 public key encoded in unpadded Base64. Implementations MUST support Ed25519. Future specifications MAY define additional key types via the ApertoID Key Type Registry (Section 15.3).

9.2. Key Lifecycle

Keys published in Agent Declaration Records MUST have an expiration timestamp in the "exp" field. Domain owners SHOULD set key expiration to no more than 90 days in the future, rotate keys at least 7 days before expiration, and use TTL values of 3600 seconds (1 hour) for Agent Declaration Records to balance caching efficiency with timely key rotation.

10. Revocation

ApertoID provides two revocation mechanisms: immediate revocation via status record and key rotation with continuity proof.

10.1. Immediate Revocation

To immediately revoke an agent, the domain owner replaces the Agent Declaration Record with a revocation record:

```
leadhunter._apertoid.example.com. 300 IN TXT
  "v=APERTOID1; status=revoked"
```

Verifiers MUST check for "status=revoked" BEFORE performing any other verification steps. If "status=revoked" is present, the verifier MUST treat the agent as unauthorized.

The TTL for revocation records SHOULD be 300 seconds (5 minutes) to minimize the window during which cached records may allow a revoked agent to pass verification.

10.2. Key Rotation

When rotating keys, the domain owner publishes a new Agent Declaration Record with the new key and a "prev" tag containing a signature proving continuity:

```
leadhunter._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; url=https://agent.example.com/mcp;
   k=ed25519; pk=<new-public-key>;
   prev=sig:<base64-signature-of-new-key-by-old-key>;
   type=ai; exp=1762000000"
```

The "prev" tag value is constructed as follows: (1) Let "new_pubkey" be the raw 32-byte new Ed25519 public key. (2) Sign "new_pubkey" using the old Ed25519 private key, producing a 64-byte signature. (3) Encode the signature as unpadded Base64. (4) Prepend "sig:" to form the "prev" tag value.

Verifiers that have cached the previous public key SHOULD verify the "prev" signature to confirm that the key rotation was authorized by the holder of the previous key. If the "prev" signature is invalid, verifiers SHOULD treat this as a potential key compromise and apply the domain’s enforcement policy.

The "prev" tag is OPTIONAL. When absent, verifiers accept the new key based solely on DNS authority (i.e., the domain owner’s control of the DNS zone).

11. Verification Procedure

This section defines the procedure for verifying an agent’s authorization via DNS records. Cryptographic signature verification on individual HTTP requests is defined in the companion document [APERTOID-SIG].

11.1. Verification Inputs

The verifier receives: "claimed_domain" (the domain the agent claims to represent), "selector" (the agent selector), "agent_url" (the URL from which the request originates), and optionally "agent_pubkey" (the agent's presented public key). If no selector is available (e.g., no ApertoID-Signature header is present), the verifier can only confirm that the domain publishes an ApertoID policy by querying "_apertoid.<claimed_domain>", but cannot verify a specific agent. Full per-agent verification requires a selector, which is typically obtained from the ApertoID-Signature header defined in [APERTOID-SIG].

11.2. Verification Algorithm

VERIFY_APERTOID(claimed_domain, selector, agent_url, agent_pubkey):

1. Query TXT record at "_apertoid.<claimed_domain>"
2. If no record found:
   Return result="none" (domain does not publish ApertoID)
3. Parse policy record; extract p= value
4. Query TXT record at "<selector>._apertoid.<claimed_domain>"
5. If no record found:
   Return result="permerror", apply policy p=
6. Parse agent declaration record
7. If status=revoked:
   Return result="revoked", apply policy p=
8. If record contains include= tag:
   Follow delegation (max depth 2, max 10 total lookups)
   If delegation fails: Return result="temperror"
   Continue verification with resolved record
9. Check exp= timestamp:
   If current_time > exp: Return result="expired",
     apply policy p=
10. Compare agent_url with record url= value:
    Match rules per Section 11.4.
    If no match: Return result="url_mismatch",
      apply policy p=
11. If record contains k= and pk= tags AND agent_pubkey
    is provided:
    Compare agent_pubkey with record pk= value
    If mismatch: Return result="key_mismatch",
      apply policy p=
12. Return result="pass"

11.3. Result Values

pass The agent is authorized by the domain and, if applicable, the

cryptographic key matches.

none The domain does not publish ApertoID records.

revoked The agent has been explicitly revoked.

expired The agent’s key has expired.

url_mismatch The agent’s URL does not match the declared URL.

key_mismatch The agent’s presented key does not match the declared key.

permerror A permanent error occurred (e.g., no agent record found, malformed syntax).

temperror A temporary error occurred (e.g., DNS timeout, delegation failure).

11.4. URL Matching Rules

When comparing the agent’s URL against the declared URL: the scheme MUST be "https" (HTTP MUST NOT be accepted); host comparison is case-insensitive per [RFC3986]; path comparison is case-sensitive; trailing slashes are normalized; query strings and fragments are ignored; port numbers, if present, MUST match (default HTTPS port 443 is assumed when absent).

12. Operational Considerations

12.1. Deployment Guidance

Domain owners deploying ApertoID SHOULD follow this progression: (1) Publish a Policy Record with "p=none" and a "rua=" address to begin collecting verification data without affecting agent operations. (2) Publish Agent Declaration Records for all known authorized agents, initially without cryptographic keys (url= only). (3) Monitor aggregate reports to identify unauthorized agents. (4) Add Ed25519 keys (k= and pk= tags) to agent records. (5) Progress to "p=warn" and then "p=reject" as confidence grows.

12.2. CDN and Reverse Proxy Interaction

When agents operate behind CDNs or reverse proxies, the agent’s apparent URL may differ from the URL in the Agent Declaration Record. Domain owners MUST ensure that the "url=" value matches the URL as seen by verifiers, not the internal origin URL.

12.3. TTL Recommendations

The following TTL values are RECOMMENDED: Policy Records: 3600 seconds (1 hour), as these change infrequently. Agent Declaration Records: 3600 seconds (1 hour), balancing caching with timely key rotation. Revocation Records: 300 seconds (5 minutes), minimizing the vulnerability window after revocation.

13. Security Considerations

13.1. Authorization vs. Identity Verification

This document alone provides authorization-level assurance: the domain declares which agents are authorized. Without the cryptographic signature mechanism defined in [APERTOID-SIG], a verifier can confirm that an agent's URL appears in the domain's DNS records but cannot confirm that the requester is genuinely that agent. An attacker with network access could potentially proxy requests through an authorized URL.

Full identity verification — where the agent proves on each request that it possesses the private key corresponding to the public key published in DNS — requires implementation of both this document and [APERTOID-SIG].

13.2. DNSSEC

ApertoID records SHOULD be protected by DNSSEC [RFC4033] [RFC4034] [RFC4035].

Without DNSSEC, an attacker capable of DNS cache poisoning can inject fraudulent ApertoID records, including false public keys. ApertoID provides two levels of assurance: without DNSSEC, it provides authorization-level assurance comparable to SPF and DKIM records (the common case today); with DNSSEC, it provides cryptographic-level assurance where the public key in DNS is authenticated by the DNSSEC chain of trust.

Verifiers SHOULD validate DNSSEC when available and MAY treat DNSSEC-validated results with higher confidence. DNSSEC is REQUIRED for full cryptographic verification of agent identity.

13.3. Key Compromise

If an agent's Ed25519 private key is compromised, the domain owner MUST immediately publish a revocation record (Section 10.1) with a low TTL (300 seconds). The window of vulnerability equals the TTL of the previously cached Agent Declaration Record. Domain owners SHOULD use TTL values no higher than 3600 seconds to limit the maximum vulnerability window to one hour.

13.4. Delegation Abuse

Attackers may attempt to abuse the delegation mechanism to create excessively long resolution chains or circular references. The limits defined in Section 8 (maximum depth of 2, maximum 10 lookups) mitigate this risk. Verifiers MUST enforce these limits and treat violations as permanent errors.

13.5. Agent Enumeration

An adversary may attempt to enumerate a domain's agents by querying common selector names. Domain owners who wish to limit enumeration SHOULD use non-predictable selector names. DNSSEC-signed zones using NSEC3 [RFC5155] provide resistance against zone walking.

13.6. DNS Amplification

ApertoID records are TXT records of moderate size (typically 170-250 bytes), comparable to DKIM key records. Standard DNS amplification mitigations (response rate limiting, BCP 38 source address validation) apply.

14. Privacy Considerations

ApertoID records are published in DNS and are therefore publicly queryable. Domain owners should be aware that publishing ApertoID records reveals: the existence and endpoint URLs of AI agents (which may reveal internal infrastructure); the type classification ("type=ai") indicating the domain uses AI agents; and agent selectors that may reveal organizational structure. Domain owners who wish to limit exposure SHOULD use generic endpoint URLs and non-descriptive selectors.

Verifiers querying ApertoID records may reveal interest in specific domains to DNS operators. Standard DNS privacy mechanisms (DNS over TLS [RFC7858], DNS over HTTPS [RFC8484]) mitigate this concern.

15. IANA Considerations

15.1. Underscore-Scoped Domain Name Registration

This document requests registration of the following entry in the "Underscored and Globally Scoped DNS Node Names" registry per [RFC8552]:

RR Type	_NODE NAME	Reference
TXT	_apertoid	[this document]

Table 1

15.2. ApertoID Agent Type Registry

This document requests IANA to create the "ApertoID Agent Type" registry with the following initial values. New values are registered via the "Specification Required" policy per [RFC8126].

Value	Description	Reference
ai	Autonomous AI agent	[this document]
human	Human-operated tool	[this document]
hybrid	AI-assisted with human oversight	[this document]

Table 2

15.3. ApertoID Key Type Registry

This document requests IANA to create the "ApertoID Key Type" registry with the following initial value. New values are registered via the "Specification Required" policy per [RFC8126].

Value	Description	Key Size	Reference
ed25519	Ed25519 (EdDSA on Curve25519)	32 bytes	[RFC8032]

Table 3

16. References

16.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", RFC 1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

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

[RFC3986] Berners-Lee, T., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4033] Arends, R., "DNS Security Introduction and Requirements", RFC 4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., "Resource Records for the DNS Security Extensions", RFC 4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5155] Laurie, B., "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC5234] Crocker, D., "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC8032] Josefsson, S., "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8126] Cotton, M., "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

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

16.2. Informative References

[RFC6376] Crocker, D., "DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF)", RFC 7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC7489] Kucherawy, M., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC7858] Hu, Z., "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8484] Hoffman, P., "DNS Queries over HTTPS (DoH)", RFC 8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[I-D.mozleywilliams-dnsop-dnsaid] Mozley-Williams, J., "DNS Agent Identity Discovery (DNS-AID)", Work in Progress, Internet-Draft, draft-mozleywilliams-dnsop-dnsaid-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-mozleywilliams-dnsop-dnsaid-01>.

[I-D.klrc-aiagent-auth] Lehman, K. and R. Chen, "Agent Identity Management System (AIMS)", Work in Progress, Internet-Draft, draft-klrc-aiagent-auth-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-klrc-aiagent-auth-00>.

[I-D.ferro-dnsop-apertodns-protocol]
Ferro, A., "ApertoDNS Protocol: A Modern Dynamic DNS Update Protocol", Work in Progress, Internet-Draft, draft-ferro-dnsop-apertodns-protocol-02, January 2026, <https://datatracker.ietf.org/doc/html/draft-ferro-dnsop-apertodns-protocol-02>.

[APERTOID-SIG]
Ferro, A., "ApertoID-Signature: HTTP Request Signing for AI Agent Identity", Work in Progress, Internet-Draft, draft-ferro-httpbis-apertoid-sig-00, 2026, <https://datatracker.ietf.org/doc/html/draft-ferro-httpbis-apertoid-sig-00>. Work in progress.

Appendix A. Complete Examples

A.1. Basic Deployment

```
; Policy: warn on failures, receive reports
_apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; p=warn; rua=mailto:apertoid@example.com"

; Agent: MCP server with Ed25519 key
assistant._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; url=https://mcp.example.com/agent;
   k=ed25519; pk=MCowBQYDK2VwAyEAb5VxRcGh1biKLfQ5YD4mFkq;
   type=ai; exp=1761955200"
```

A.2. Third-Party Delegation

```
; Policy: reject unauthorized agents
_apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; p=reject; rua=mailto:sec@example.com"

; Own agent
leadhunter._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; url=https://agents.example.com/leadhunter;
   k=ed25519; pk=MCowBQYDK2VwAyEAxyz123base64keyhere456;
   type=ai; exp=1761955200"

; Delegated third-party agent
crm._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; include=client42._apertoid.salesforce.com"
```

A.3. Emergency Revocation and Key Rotation

```
; Step 1: Immediately revoke compromised agent
leadhunter._apertoid.example.com. 300 IN TXT
  "v=APERTOID1; status=revoked"

; Step 2: Publish new key with rotation proof
leadhunter._apertoid.example.com. 3600 IN TXT
  "v=APERTOID1; url=https://agents.example.com/leadhunter;
   k=ed25519; pk=<new-public-key-base64>;
   prev=sig:<signature-of-new-key-by-old-key>;
   type=ai; exp=1764547200"
```

Appendix B. Integration with Application-Layer Protocols

This appendix is non-normative.

ApertoID is designed to be protocol-agnostic at the application layer. MCP Server Cards at "/.well-known/mcp/server-card.json" provide capability discovery; ApertoID complements MCP by providing DNS-based verification that a given MCP server is authorized by the claimed domain. A2A Agent Cards at "/.well-known/agent.json" provide agent discovery; ApertoID provides a DNS-based trust anchor independent of the agent's self-declared card. DNS-AID [I-D.mozleywilliams-dnsop-dnsaid] provides DNS-based agent discovery; ApertoID provides authorization and identity verification. The two are complementary: DNS-AID answers "what agents does this domain have?" while ApertoID answers "is this agent genuinely authorized by this domain?"

Acknowledgements

The design of ApertoID was informed by the architectural patterns established by SPF, DKIM, and DMARC for email authentication, and by DANE for DNS-based authentication of named entities. The author acknowledges the DNSOP working group participants whose feedback on draft-ferro-dnsop-apertodns-protocol shaped the approach taken in this document.

Author's Address

Andrea Ferro
ApertoID
Verona
Italy
Email: irn@irn3.com
URI: https://github.com/ApertoID