Individual Submission
Internet-Draft
Intended status: Standards Track
Expires: 1 November 2026

N. Ihsanullah
Identity Digital

30 April 2026

DNS-Anchored Durable Identity for AI Agents (DNSid)
draft-ihsanullah-dnsid-00

Abstract

Autonomous software agents are being deployed across enterprise, cloud, and cross-organizational boundaries. These agents negotiate, transact, delegate, and produce work products that persist beyond their own runtime. The standards identified for agent identity collectively address runtime authentication, authorization, lifecycle management, and tool interaction, but a gap remains: a durable, governance-backed identifier that binds an agent to an accountable entity in a way that any system can verify independently.

DNSid addresses the accountable layer of identity: the durable ownership anchor that existing agent identity standards do not provide. This document specifies DNSid, a minimal identity primitive that assigns each agent a Fully Qualified Domain Name (FQDN) under a domain controlled by its accountable entity, and publishes a structured set of pointers in DNS TXT records to the agent’s cryptographic keys, lifecycle log, and operational status. DNSid uses accountable-entity-controlled signatures for record integrity and an abstract append-only ledger for lifecycle history. It is designed to sit beneath existing identity, authentication, authorization, and agent interaction standards without competing with them.

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 1 November 2026.

Table of Contents

1. Introduction and Motivation

1.1. The Problem

AI agents are moving from prototypes into production. They operate autonomously, acquire tools dynamically, delegate to other agents, and produce work products (reports, code, transactions, decisions) that persist and cross organizational boundaries after the agent terminates. Existing identity and access management systems can authenticate workloads and authorize actions within a platform, but they cannot answer a more fundamental question:

> Which accountable entity is responsible for this agent, and can any system verify that independently?

The standards identified by NIST for agent identity and authorization (OAuth 2.1, OIDC, SPIFFE/SPIRE, SCIM, NGAC, MCP) collectively address runtime authentication, authorization, lifecycle management, and tool interaction. None provides a durable, governance-backed ownership anchor that persists across platforms, trust domains, and agent lifecycles.

1.2. Why DNS

DNS is the only globally delegated namespace with:

* Universal resolvers: every device on the internet can resolve a DNS name.

* Governance-backed domain accountability: domain registration relationships, namespace governance processes, and dispute resolution mechanisms already provide an operational framework for managing control of DNS names.

* Existing operational use in organizational trust: TLS, email authentication (SPF, DKIM, DMARC), and PKI already depend on DNS for organizational binding.

* 40+ years of deployment with no migration required.

Keeping agent identifiers in the DNS namespace allows relying parties to reuse the same operational, governance, and security plane already used for web, email, and commerce infrastructure, and avoids introducing a separate namespace whose binding to existing internet identities would require its own trust model. Building a new identity protocol from scratch would create a parallel naming system, a new trust root, and years of standardization before meaningful deployment. DNSid extends DNS from resolving domain names to anchoring agent identifiers, building on infrastructure that is already universal, neutral, and institutionally governed.

1.3. Design Principles

DNSid is intentionally minimal:

* Smallest viable surface area. DNSid does one thing: bind an agent to an accountable entity via DNS, with pointers to keys, status, and history. It does not authenticate agents, issue credentials, enforce policy, monitor behavior, or provide a runtime execution environment.

* Pointer set, not data store. The DNS record carries structured pointers to HTTPS endpoints and ledger entries. Rich identity data, claims, and attestations live at those endpoints, not in DNS. DNS serves as the authoritative rendezvous layer for identity resolution.

* Accountable-entity-controlled integrity. Record integrity is established by the accountable entity's own cryptographic key, not by the DNS hierarchy's signing chain. DNSSEC is complementary hardening, not an architectural dependency.

* Ledger-neutral. The specification defines what lifecycle events must be recorded and what proof properties the ledger must provide. It does not mandate a public ledger or a specific ledger technology.

* Standards-complementary. DNSid sits beneath SPIFFE, OAuth, OIDC, SCIM, NGAC, MCP, A2A, DIDs, VCs, and Agent.md. It provides the governance-backed anchor those mechanisms reference but do not define.

* Pseudonymity with pierceability. Each DNSid FQDN is a disconnected pseudonymous handle. The agent operates under its FQDN without exposing the accountable entity to casual observers. Resolving the accountable entity may require registrar, registry, or legal process. Use of distinct registrable domains provides operational separation of pseudonymous identities. This is a deliberate privacy-by-design property of anchoring identity in the registrant-domain governance layer.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD 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.

Agent FQDN: The Fully Qualified Domain Name assigned to an agent. This is the agent's primary globally unique identifier. Examples include billing-agent-acme.example for a dedicated registrable-domain deployment and billing-agent.acme-corp.example for a shared organizational-domain deployment.

Governance Anchor: The registrant domain under which the agent FQDN is published, typically a registrable domain. The registrant of this domain is the accountable entity for all agents beneath it. When the agent is registered as a dedicated registrable domain, the FQDN and governance anchor are the same (e.g., billing-agent-acme.example serves as both). Under a shared domain, the governance anchor is the organizational domain (e.g., acme-corp.example).

DNSid TXT Record: A DNS TXT Resource Record (TXT RR) published at the owner name _dnsid.<agent-fqdn> containing a structured set of key-value tags that encode DNSid identity pointers.

Verifier: An agent, service, or relying party that resolves and validates a DNSid TXT record.

Immutable Ledger: An append-only data structure that records lifecycle events with cryptographic inclusion proofs and verifiable timestamps. May be instantiated as a blockchain, Merkle tree, Certificate Transparency [RFC6962] style log, or similar verifiable data structure. For example, a conforming ledger could be a SCITT transparency service, a CT-style Merkle log, a blockchain application, or a private append-only service.

Accountable Entity: The party that registered the agent, bears responsibility for its behavior, and holds authority to revoke it. The accountable entity may be an organization or an individual. This is distinct from intellectual property ownership or runtime control.

Verification Scope: The set of verifiers, deployment contexts, or access conditions under which a ledger binding makes lifecycle entries or portable cryptographic proofs available. The verification scope is determined by the ledger binding specification and the accountable entity's deployment choice.

Signing Key: The cryptographic key pair controlled by the accountable entity, used to sign the DNSid TXT record and lifecycle events. Published via the JWKS endpoint. Distinct from any key issued by a certificate authority or DNS operator.

3. Ecosystem Context and Scope

3.1. What DNSid Addresses (Layer 1)

A survey of the agent identity ecosystem reveals that the relevant concerns organize into a layered model:

Layer	Name	Representative Standards
0	Global Namespace Governance	ICANN, DNSSEC, DANE
1	Durable Ownership Anchor	*DNSid operates here*
2	Cryptographic Identity Binding	DIDs, VCs, PKI
3	Runtime Workload Identity	SPIFFE/SPIRE
4	Authentication	OAuth 2.1, OIDC, RFC 9421
5	Authorization and Policy	NGAC, OPA, Cedar
6	Operational Governance	NHI platforms, SCIM
7	Discovery and Interoperability	MCP, A2A, AGNTCY

Table 1

The standards that NIST, IETF, and industry have identified for agent identity collectively cover Layers 2 through 7. Layer 1 is sparsely populated. DNSid targets this gap specifically.

3.2. What DNSid Does Not Address

DNSid does not:

* Authenticate agents at runtime (Layer 4: OAuth, OIDC)

* Attest workloads within a trust domain (Layer 3: SPIFFE)

* Issue or manage verifiable credentials (Layer 2: DIDs, VCs)

* Enforce authorization policies (Layer 5: NGAC, OPA)

* Monitor agent behavior (Layer 6: NHI platforms)

* Provide agent-to-agent communication protocols (Layer 7: MCP, A2A)

* Replace endpoint control or firewall functionality

* Determine whether an accountable entity is trustworthy, reputable, or legally qualified. Such determinations are made by consuming systems, external registries, Verifiable Credentials, or local policy.

These systems are consumers of the ownership anchor DNSid provides. They reference the DNSid identity to make their own trust, authorization, and enforcement decisions.

4. Trust Architecture

DNSid separates three distinct anchors of trust. Each anchor addresses a different verification need.

4.1. Governance Anchor: Registrant Domain

The registrant domain is the governance root. The registrant is subject to domain registration agreements, namespace governance policy, and established dispute resolution processes. This governance is not something DNSid creates; it already exists for every domain registrant on the internet. DNSid is designed to operate within existing DNS infrastructure and should not require new governance processes, TLD allocations, or registrar categories.

The governance and pseudonymity properties are strongest when each agent is registered as a dedicated registrable domain (e.g., billing-agent-acme.example). Each registrable domain constitutes a distinct registrant relationship, a distinct accountability scope, and a distinct operational zone with independent DNSSEC signing and administrative control. Organizations MAY anchor agents under an existing organizational domain (e.g., billing-agent.acme-corp.example), accepting that agents under a shared domain share governance, accountability, and pseudonymity scope.

Agent FQDNs are not required to carry human-readable names. The Agent FQDN need not host a human-facing website or reveal the agent’s purpose. Organizations seeking maximum pseudonymity MAY register opaque identifiers as agent FQDNs, since the agent’s purpose and capabilities are described in the capabilities document referenced by the "cu" tag, not in the domain name itself.

When a platform hosts agents on behalf of users, the platform registrant is the accountable entity under DNSid. Attribution of responsibility to individual users is handled through the platform's internal audit trail and administrative processes, not through DNSid itself.

4.2. Technical Identity Anchor: Agent FQDN

Each agent is assigned a globally unique FQDN as its governance anchor (e.g., billing-agent-acme.example). When registered as a dedicated registrable domain, the FQDN is itself the governance anchor. When registered under an organizational domain (e.g., billing-agent.acme-corp.example), the FQDN inherits the governance posture of the parent registrant domain. In either case, the FQDN is the identifier that other agents, registries, and relying services carry in protocol messages.

The agent's public signing keys are published at an HTTPS endpoint (JWKS format) referenced by the DNSid TXT record. The key service MUST support algorithm negotiation to enable crypto agility for the transition to post-quantum signature algorithms (NIST FIPS 204 ML-DSA, FIPS 205 SLH-DSA) [NIST-PQC].

4.3. Integrity and History Anchor: Immutable Ledger

An append-only verifiable ledger records the core agent identity lifecycle: issuance, key rotation, revocation, retirement, and migration. Optional event types extend coverage to delegation and content provenance. The ledger is bound to the agent FQDN through the accountable entity's signing key.

DNS proves current control of the FQDN. The ledger proves what has happened under that identity over time. Together they provide a complete temporal picture that neither can provide alone.

The ledger is entity-signed: the accountable entity signs its own lifecycle entries with its signing key. The ledger itself may be operated by any party; what matters is that entries carry the accountable entity's signature and cannot be forged by the ledger operator. The entity-signed model supports recording content-signing hashes, delegation chain records, and provenance attestations alongside identity lifecycle events.

The ledger may be instantiated as a blockchain, Merkle tree, Certificate Transparency-style log, or similar verifiable data structure. The architectural requirement is append-only integrity with cryptographic inclusion proofs, not a specific implementation. See Section 8 for the abstract ledger interface.

5. The DNSid TXT Record

5.1. Owner Name Convention

A DNSid TXT record MUST be published at the owner name (the DNS term for the left-hand side of a resource record) formed by prepending the label "_dnsid" to the agent FQDN:

_dnsid.<agent-fqdn>

Example:

_dnsid.billing-agent-acme.example.

The underscore prefix follows the convention established by DKIM (_domainkey), DMARC (_dmarc), and MTA-STS (_mta-sts), creating a dedicated namespace that does not collide with TXT records used by other protocols at the agent FQDN itself.

An owner name MUST NOT have more than one DNSid TXT record. If multiple DNSid TXT records are encountered at the same owner name, the verifier MUST treat the lookup as a failure.

The agent FQDN MUST NOT exceed 246 octets in presentation format, ensuring the derived owner name _dnsid.<agent-fqdn> does not exceed the 253-octet presentation limit ([RFC1035] Section 2.3.4, [RFC1123] Section 2.1).

The TTL of DNSid TXT records SHOULD be kept short enough to support timely revocation propagation.

5.2. Record Format

The RDATA of the DNSid TXT record is a sequence of tag=value pairs separated by semicolons, following the syntax defined in Section 3.2 of [RFC6376] (DKIM Signatures). The record MUST begin with the version tag "v=DNSid1".

Formal syntax (ABNF, [RFC5234]):

```
dnsid-record  = dnsid-v-tag *( ";" SP? dnsid-tag ) [ ";" ]
dnsid-v-tag   = %s"v" "=" %s"DNSid1"
dnsid-tag     = dnsid-tag-name "=" dnsid-tag-value
dnsid-tag-name  = ALPHA *( ALPHA / DIGIT / "_" )
dnsid-tag-value = *( %x21-3A / %x3C-7E )
                  ; printable ASCII excluding ";" and space
```

Tag names are case-sensitive. Unknown tags MUST be ignored by receivers to provide forward-compatibility for future extensions.

A DNSid TXT record MUST NOT contain duplicate tag names. A verifier MUST reject a record containing duplicate tag names.

Unknown tags are ignored for semantic processing but are included in the canonical record string for signature verification.

5.3. Tag Definitions

Tag	Required	Description
v	REQUIRED	Version. MUST be "DNSid1". MUST be first tag.
oi	REQUIRED	Governance identifier. Links the agent FQDN to its governance anchor.
ku	REQUIRED	Key URI. HTTPS URL for the agent's signing keys in JWKS format ([RFC7517]).
lr	REQUIRED	Log reference. Structured reference identifying the ledger binding and the agent's entry.
su	REQUIRED	Status URI. HTTPS URL for registration and revocation status.
sg	REQUIRED	Entity signature. Base64url-encoded signature over the canonical record content.
fl	OPTIONAL	Policy flags. Comma-separated list of named flags.
ka	OPTIONAL	Maximum signing key age. Duration value.
cu	OPTIONAL	Capabilities URI. HTTPS URL for an Agent.md or Agent Card document.

Table 2

Implementations MUST reject a DNSid TXT record that is missing any REQUIRED tag or in which a REQUIRED tag has an empty value.

5.4. OI (Governance Identifier) Tag

The "oi" tag identifies the accountable entity controlling this agent. The RECOMMENDED format is the registrant domain in lowercase ASCII. Alternative formats include a URI identifying the organization in an external registry.

Examples:

```
oi=billing-agent-acme.example
oi=acme-corp.example
```

The "oi" value MUST be consistent with the domain hierarchy under which the agent FQDN is published: the agent FQDN must be a subdomain of (or equal to) the domain identified by "oi", or the "oi" value must reference a governance relationship that is verifiable through the ledger.

5.5. LR (Log Reference) Tag

The "lr" tag is a DNSid log reference identifying the ledger technology and the agent’s primary entry. The method component of the reference identifies the ledger binding; the remaining components locate the agent’s log entry.

This specification does not mandate a specific log method. Implementations SHOULD register their method name with the DNSid Log Method Registry (Section 14).

Examples (method names are illustrative):

```
lr=algorand:AGENT_ADDR_BASE32
lr=ctlog:https://log.example.com/agents/entries/12345
lr=scitt:https://transparency-service.example/entry/abc123
```

5.6. FL (Policy Flags) Tag

The "fl" tag is a comma-separated list of named policy flags. Unknown flag names MUST be ignored by receivers.

Initially defined flags:

Flag	Description
mtls	Verifiers MUST use mutual TLS for connections to this agent.
logchk	Verifiers MUST perform a log inclusion check before any high-value or irreversible operation.

Table 3

Example: fl=mtls,logchk

5.7. KA (Key Age Maximum) Tag

The "ka" tag expresses the maximum acceptable age of a signing key. Verifiers MUST reject keys older than the indicated age.

Defined values: 24h, 7d, 30d, 90d. Absence means no constraint. Unknown values MUST be treated as a verification failure.

The key’s age is computed from the timestamp of the ISSUANCE or KEY_ROTATION event that introduced the key into the ledger. The ledger is the authoritative source for key lifecycle timestamps. If the "ka" tag is present and the verifier cannot obtain the required lifecycle event or cryptographic proof, key age verification MUST fail.

5.8. Multi-String Encoding

DNS TXT RDATA is represented as one or more character-strings, each limited to 255 octets ([RFC1035] Section 3.3). Implementations MUST concatenate all strings within a single TXT RR before parsing tag-value pairs. No separator is inserted at string boundaries.

Example (split across strings for readability):

```
_dnsid.billing-agent-acme.example. 300 IN TXT (
    "v=DNSid1; oi=billing-agent-acme.example; fl=mtls,logchk;"
    "ku=https://billing-agent-acme.example/.well-known/jwks.json;"
    "lr=algorand:AGENT_ADDR_BASE32;"
    "su=https://billing-agent-acme.example/.well-known/dnsid-status;"
    "sg=<base64url-encoded-signature>"
)
```

6. Entity-Signed Record Integrity

6.1. Motivation

DNSSEC provides integrity for DNS records via a top-down signing chain from the DNS root through TLD operators to the registrant's zone. However, the TLD operator and higher-level signers hold keys above the registrant's zone and could, in principle, modify records without the registrant's knowledge or consent.

For an identity system where "you are always in control of your identity" is a design principle, depending solely on DNSSEC for integrity is insufficient. DNSid therefore specifies an integrity mechanism where the accountable entity signs the DNSid record content with its own key.

6.2. Signature Generation

The "sg" tag contains a signature computed as follows:

1. Construct the canonical record string by concatenating all tag-value pairs EXCEPT the "sg" tag, in alphabetical order by tag name, separated by semicolons, with no whitespace.

2. Sign the canonical string using the agent's current signing key (the key published at the JWKS endpoint referenced by the "ku" tag).

3. Encode the signature as base64url ([RFC4648] Section 5).

The signing algorithm is determined by the key's "alg" field in the JWKS response. Implementations MUST support ES256 (ECDSA with P-256) [RFC7518]. Implementations SHOULD support Ed25519 [RFC8037]. When post-quantum algorithms are registered for use with JWS, implementations SHOULD add support for ML-DSA (FIPS 204) [NIST-PQC].

6.3. Signature Verification

A verifier:

1. Fetches the JWKS from the "ku" endpoint over HTTPS.

2. Reconstructs the canonical record string (all tags except "sg", alphabetical order, no whitespace).

3. Verifies the "sg" value against the canonical string using the public key from the JWKS.

4. If verification fails, the verifier MUST reject the record.

If the JWKS contains more than one signing key, the verifier MUST attempt verification with each key whose "alg" value is compatible with the signature. Verification succeeds if any eligible key validates the signature.

The verifier MAY additionally check the ledger to confirm that the signing key is currently bound to this agent FQDN and has not been rotated or revoked since the record was last updated.

6.4. Relationship to DNSSEC

DNSSEC, when deployed, provides an independent integrity layer that protects against on-path tampering of the DNS response. DNSid implementations SHOULD deploy DNSSEC. Verifier behavior depends on the DNSSEC state of the zone:

DNSSEC absent (zone not signed): Verification proceeds with the accountable entity's signature ("sg" tag) only. The verifier SHOULD note the reduced assurance level. This accommodates domains on TLDs or registrars that do not yet support DNSSEC.

DNSSEC present and valid: Both DNSSEC and the accountable entity's signature are checked. This is the strongest assurance level — DNS transport integrity is confirmed and the record content is signed by the accountable entity.

DNSSEC present but validation fails: The verifier MUST treat the DNSid TXT record as unusable and abort verification. DNSSEC validation failure means authenticated DNS data for the owner name could not be established; the DNS response may have been tampered with in transit. This follows the precedent set by DANE ([RFC6698] Section 4), which requires that TLSA records with failed DNSSEC validation be treated as unusable.

7. HTTP Service Interfaces

DNSid defines two HTTPS service interfaces referenced by the TXT record. These interfaces are abstract: the specification defines the semantics and required response properties, not a specific API schema. Companion documents may define concrete API profiles.

7.1. Key Service (KU endpoint)

The key service endpoint MUST return a JWK Set ([RFC7517]) over HTTPS. The response:

* MUST include at least one signing key with a "kid" (key ID).

* MUST include the "alg" field on each key, enabling algorithm negotiation for crypto agility.

* SHOULD include key metadata such as expiry and key operations, to support local policy evaluation.

* MUST be served over HTTPS with a valid TLS certificate whose Subject Alternative Name matches the agent FQDN. The "ku" URI host MUST be the agent FQDN.

The endpoint URL SHOULD follow the well-known URI convention ([RFC8615]):

https://<agent-fqdn>/.well-known/jwks.json

Agents that already publish a JWKS endpoint at their FQDN for other purposes (e.g., OAuth) MAY reference that existing endpoint via the "ku" tag. A separate DNSid-specific endpoint is not required.

Future specifications MAY define additional key discovery profiles, including DNSSEC-authenticated delegated key services, parent-domain key services, DNS-carried key material, or ledger-bound key references. Such profiles MUST specify how the binding between the agent FQDN and the signing key is authenticated without relying solely on a signature verified by the key being discovered. A verifier MUST NOT use an alternate key discovery mechanism unless the applicable profile defines its verification requirements and failure behavior.

The JWKS SHOULD contain a single current DNSid signing key. Previous signing keys MAY be retained to support verification of prior signatures. When multiple signing keys are present, the current key SHOULD appear first. Verifiers MUST NOT rely on JWKS ordering for correctness and MUST attempt verification with each eligible key whose "alg" value is compatible with the signature.

The signing algorithm for all DNSid operations is declared by the "alg" field of the JWK published at the key service endpoint. The DNSid TXT record itself does not encode algorithm information; algorithm negotiation occurs entirely through the JWKS.

Implementations MAY optionally reinforce the key service binding with DANE ([RFC6698]) TLSA records for cryptographic binding between the FQDN and the TLS certificate, eliminating dependence on external certificate authorities.

7.2. Status Service (SU endpoint)

The status service endpoint MUST return the agent's current lifecycle state over HTTPS. The response:

* MUST include the current state (see Section 10.1 for valid states: PENDING, PROVISIONING, VERIFYING, ACTIVE, RETIRED, or REVOKED).

* MUST include the timestamp of the last state transition.

* MUST include a reason code when the state is REVOKED.

* MUST be served with a valid TLS certificate.

The status endpoint is the primary real-time mechanism for obtaining the accountable entity's current lifecycle-state declaration. Firewalls, IAM systems, policy engines, and other Layer 3-7 systems MAY reference this declaration when making their own trust decisions.

DNSid does not enforce revocation. A verifier's trust in the status endpoint is a matter of local policy. High-assurance deployments MAY require consistency between the status endpoint and the lifecycle ledger, or MAY require status responses to be countersigned by a designated authority.

8. Abstract Ledger Interface

DNSid requires an append-only verifiable ledger for lifecycle history and, where optional event types are supported, provenance. This section defines the abstract interface that any conforming ledger implementation must satisfy. It does not specify a particular ledger technology.

8.1. Ledger Identification

The "lr" tag in the DNSid TXT record identifies the ledger using a structured reference. The method component identifies the ledger binding. The remaining components locate the agent's entry within that ledger.

Conforming ledger implementations SHOULD register their method name with the DNSid Log Method Registry (Section 14).

Concrete ledger bindings SHOULD be documented as separate specification documents, analogous to W3C DID method specifications. Each binding document defines the method name, reference syntax, entry format, proof format, and query interface for a specific ledger technology. The DNSid Log Method Registry serves as the index for registered bindings.

8.2. Required Properties

A conforming ledger MUST provide:

* Append-only: entries cannot be modified or deleted after recording.

* Cryptographic inclusion proofs: for any recorded entry, the ledger can produce a proof that the entry is included in the log at a specific position.

* Verifiable timestamps: each entry has a timestamp that is independently verifiable.

* Verifier accessibility: verifiers within the ledger's declared verification scope can obtain lifecycle entries or portable cryptographic proofs sufficient to verify inclusion, timestamps, and current lifecycle state.

* Durability: entries persist independently of the ledger operator's continued participation.

The accountable entity selects the ledger technology and its verification scope. A ledger MAY be public, consortium-scoped, private, access-controlled, or based on portable receipts, provided that verifiers expected to rely on the DNSid can obtain sufficient cryptographic evidence to perform verification. A publicly queryable ledger provides the broadest interoperability and is RECOMMENDED for agents intended for open Internet-scale verification.

8.3. Entry Signing

Lifecycle event entries MUST be signed by the accountable entity's signing key (the key published via the JWKS endpoint). This ensures that the accountable entity, not a third party, is the author of each lifecycle record.

Entries MAY additionally be countersigned by the ledger operator or a witness service. Countersignatures provide additional assurance but do not replace the accountable entity's signature.

8.4. Event Types

Lifecycle events that introduce or rotate signing keys MUST preserve sufficient public key material, or a durable reference to such material, to support historical signature verification independent of the current JWKS endpoint. A key hash or thumbprint MAY be included for compact binding, but a hash alone is not sufficient for historical signature verification.

The ledger MUST support recording the following core event types. Each event includes the agent FQDN, event type, timestamp, accountable entity signature, and event-specific data:

ISSUANCE: Agent FQDN, initial public key material or durable public-key reference, public key thumbprint, registrant organizational identifier. Recorded when the agent identity is first created.

KEY_ROTATION: Previous public key thumbprint, new public key material or durable public-key reference, new public key thumbprint, rotation timestamp. Recorded when the agent's signing key changes. Establishes continuity: same agent, new keys.

REVOCATION: Reason code (REQUIRED), timestamp. Recorded when the agent's identity is forcibly ended. Permanent and irreversible. Reason codes follow X.509 conventions: keyCompromise, policyViolation, superseded, cessationOfOperation. Signed work products from the period triggering revocation SHOULD be treated as suspect by verifiers.

RETIREMENT: Timestamp. Recorded when the agent's identity lifecycle is gracefully completed. The agent was decommissioned as planned; signed work products produced before retirement retain full provenance value. Permanent and irreversible.

MIGRATION: Previous ledger reference, new ledger reference, final entry reference on previous ledger, timestamp. Recorded on the NEW ledger when an agent's lifecycle history is moved between ledger technologies. The previous ledger's records remain the authoritative history for events before the migration timestamp. The new ledger's first entry MUST reference the agent's final entry on the previous ledger.

The ledger MAY additionally support the following OPTIONAL event types:

DELEGATION: Delegating agent FQDN, delegatee agent FQDN, scope constraints, expiry. Recorded when one agent grants authority to another.

CONTENT_SIG: Content hash (SHA-256 or stronger), signing key reference (kid), timestamp. Recorded when the agent signs a work product for provenance purposes.

Additional event types MAY be defined in companion specifications.

9. Verification Protocol

DNSid verification combines DNS resolution, accountable entity signature validation, TLS-based live identity, and optional ledger queries.

9.1. Interactive Verification

1. The verifier queries _dnsid.<target-fqdn> for TXT records. If multiple TXT records are present, verification MUST fail. The verifier concatenates all strings in the TXT RDATA and parses the tag-value record.

2. The verifier validates that "v=DNSid1" is the first tag and that all REQUIRED tags are present and non-empty.

3. The verifier fetches the JWKS from the "ku" endpoint over HTTPS and verifies the accountable entity’s signature ("sg" tag) against the canonical record content (Section 6.3). If signature verification fails, the record MUST be rejected.

4. The verifier establishes an HTTPS or mTLS connection with the agent at its FQDN (not at _dnsid.<fqdn>). The TLS handshake and certificate chain prove active control of the FQDN. When mTLS is required (the "mtls" flag is present), the verifier MUST validate the peer certificate against the agent FQDN per [RFC9525]. DNSid does not define the application protocol or runtime authentication exchange.

Before relying on a DNSid identity for a new interactive operation, the verifier MUST query the status service ("su" tag) over HTTPS and confirm the state is ACTIVE. If the state is not ACTIVE, verification for new interactive operations MUST fail. A verifier MAY skip status lookup only when performing offline or historical provenance verification.

These steps provide interactive identity assurance: the verifier knows which accountable entity controls this agent, right now, with cryptographic proof.

9.2. Historical Verification

Step 5 (Ledger Verification): For high-assurance operations, or when the "logchk" flag is present, the verifier queries the ledger identified by the "lr" tag. The verifier checks:

* That the agent's issuance event is recorded.

* That the current signing key is bound to this agent FQDN.

* That the ledger binding provides evidence of the agent's non-revoked state at the relevant verification time.

Step 6 (Provenance Verification): For work product provenance, the verifier checks that a CONTENT_SIG event exists in the ledger matching the content hash of the artifact, with a timestamp and signing key consistent with the agent's identity at the time of production.

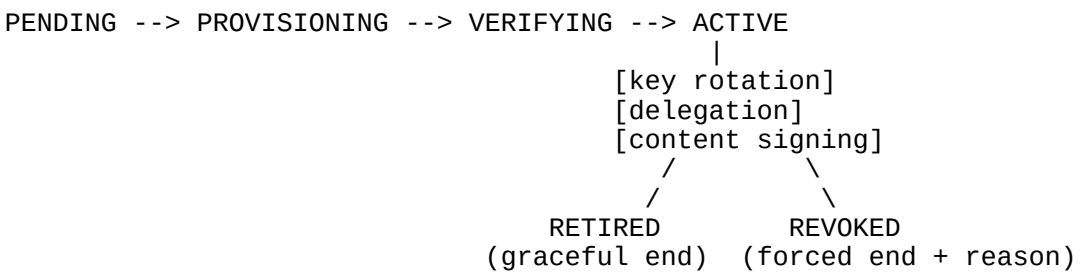
When ledger verification is required by local policy, by the "logchk" flag, or by the risk profile of the operation, failure to obtain the required ledger entry or cryptographic proof MUST cause ledger verification to fail.

Historical verification provides lifecycle transparency and artifact provenance. These checks are used for audit, compliance, post-incident forensics, and verifying agent outputs that have crossed organizational boundaries.

10. Lifecycle Events

10.1. Agent States

An agent identity progresses through a linear state machine. Each transition is a ledger event.



* PENDING: Identity record created, infrastructure not yet deployed.

* PROVISIONING: Cryptographic challenge issued by the registry.

* VERIFYING: Challenge signed by the agent’s private key, registry validating.

* ACTIVE: Operational. Key rotation, delegation, and content signing occur within this state.

* RETIRED: Graceful end-of-life. The agent completed its intended lifecycle and was decommissioned as planned. Signed work products produced before retirement retain full provenance value.

* REVOKED: Forced end. Reason code REQUIRED (keyCompromise, policyViolation, superseded, cessationOfOperation). Signed work products from the period triggering revocation SHOULD be treated as suspect by verifiers.

Both RETIRED and REVOKED are terminal states. Transitions are forward-only. Neither can return to ACTIVE. A new identity registration is required.

10.2. Identity Lifecycle Flexibility

DNSid may be attached to agents at any point in their lifecycle, including agents that were deployed before DNSid was available. The PENDING state serves as the entry point for registration regardless of when the agent was originally created or how long it has been operational.

For fleets of pre-existing agents, organizations may migrate agents to DNSid incrementally. No flag-day deployment is required; agents can be registered individually or in batches as operational needs dictate.

An agent’s DNSid may be replaced. The previous DNSid enters RETIRED or REVOKED as appropriate, and a new ISSUANCE event creates a fresh identity with its own ledger history. The new identity carries no history from the previous one. Verifiers will observe the new identity’s short effective history, which is itself a trust-relevant signal: an identity with no ledger depth is distinguishable from one with years of operational history. Replacement may serve legitimate operational purposes (organizational restructuring, key compromise response, rebranding) but cannot import the previous identity’s accumulated trust.

10.3. Key Rotation Procedure

When an agent’s signing key is rotated:

1. The agent generates a new key pair.

2. The new key is published to the JWKS endpoint. The previous key SHOULD be retained at the endpoint for a period sufficient to verify any outstanding signed work products.

3. A KEY_ROTATION event is recorded on the ledger, binding the previous public key thumbprint to the new public key material or durable public-key reference and new public key thumbprint, establishing continuity of identity.

4. The DNSid TXT record is re-signed with the new key (the "sg" tag is recomputed).

5. DNS TTL expiry propagates the updated record to verifiers.

Verifiers encountering a signature from a previous key SHOULD check the ledger for a KEY_ROTATION event linking the previous key thumbprint to the current key before rejecting the signature.

10.4. Status Change Propagation

When an agent’s status changes (e.g., revocation), the change propagates through:

* Status endpoint (su): reflects the new state immediately.

* Ledger record (lr): permanent record with reason code.

* DNS TTL expiry: verifiers re-resolve the DNSid TXT record on the next TTL cycle.

Systems that reference this identity (firewalls, IAM, policy engines, audit platforms) independently discover the status change on their own schedule. DNSid reflects the accountable entity’s decision. Enforcement is the responsibility of each consuming system.

11. Applicability

Even agents operating within a single trust domain may transcend organizational boundaries as their work products, signed outputs, and downstream interactions propagate beyond their origin. The zero-trust security model, which treats internal and external trust boundaries equivalently, provides independent motivation for applying durable cross-organizational identity even to agents that currently operate within a single trust domain.

12. Security Considerations

12.1. Entity Signature and DNSSEC

Risk: An on-path attacker modifies the DNSid TXT record during DNS resolution. Mitigation: Two independent integrity layers. The accountable entity's signature ("sg" tag) binds record content to the accountable entity's key — analogous to a self-signed certificate in a DNSSEC-signed zone. DNSSEC, when deployed, provides transport integrity for the DNS response. Together they mirror the DANE model ([RFC6698]), where TLSA records bind certificates to domain names through DNSSEC-signed DNS. When DNSSEC validation fails, the record MUST be treated as unusable (Section 6.4). When DNSSEC is absent, the accountable entity's signature remains the sole integrity mechanism.

12.2. TXT Record Ambiguity

Unlike a dedicated RR type, TXT records provide no inherent type safety. Verifiers MUST validate the "v=DNSid1" version tag and MUST reject any TXT record at the _dnsid owner name that does not contain this tag. The singleton TXT requirement is a deliberate ambiguity-control measure: verifiers do not select among multiple candidate DNSid records and do not merge fields across records.

12.3. TTL and Caching

Stale DNSid TXT records create a window during which revoked or rotated identities remain cached by resolvers. DNS is not a real-time revocation channel. Operators SHOULD choose DNSid TXT TTLs appropriate to the risk profile of the agent and SHOULD avoid long-lived TTLs for identities that require responsive revocation. Verifiers that require current lifecycle state MUST query the status service rather than relying solely on cached DNS data.

12.4. Post-Quantum Considerations

Agent identity credentials are long-lived governance artifacts, not ephemeral session tokens — the class of credential most vulnerable to future quantum-capable signature forgery. The key service (Section 7.1) supports algorithm negotiation via the JWK "alg" field, enabling transition to post-quantum signature algorithms (FIPS 204 ML-DSA, FIPS 205 SLH-DSA) without protocol changes, consistent with the NIST IR 8547 deprecation timeline [NIST-IR-8547].

12.5. Ledger Durability

Organizations SHOULD choose ledger technologies with strong durability guarantees and broad operational support. The MIGRATION event type (Section 8.4) provides a mechanism for moving to a new ledger while preserving history. Organizations SHOULD maintain an offline archive of their ledger entries as a backup against ledger unavailability.

If a ledger becomes permanently unavailable, the agent's historical provenance is degraded, but the current identity (DNS + JWKS + status endpoint) remains valid for interactive verification. The MIGRATION event allows re-establishing historical continuity on a new ledger by referencing the previous ledger's final entry.

12.6. Revocation and Accountability

A REVOCATION event is an abnormal lifecycle state transition, distinct from RETIREMENT. Whereas RETIREMENT indicates a planned or graceful end of an agent identity, REVOCATION indicates that the accountable entity has declared the identity no longer valid because of a condition such as key compromise, policy violation, supersession, or cessation of operation.

A REVOCATION event does not by itself repudiate all prior actions by the agent, nor does it remove accountability for actions that occurred before the revocation event. However, revocation is a material trust signal. Verifiers and investigators evaluating prior work products SHOULD consider the revocation reason, the artifact timestamp, the status transition timestamp, the ledger event timestamp, the relevant key history, and any available external evidence.

Signed work products produced during a suspected compromise or policy-violation window SHOULD be treated as suspect by verifiers. Work products produced before that window MAY retain provenance value if the verifier can establish that the relevant signing key was valid at the time of production and that no applicable revocation or compromise event affects the artifact.

12.7. URI Integrity

The HTTPS endpoints referenced by "ku", "su", and "cu" tags depend on TLS [RFC8446] for integrity. All URI values in DNSid tags MUST conform to [RFC3986]. Implementations MUST verify TLS certificates for each HTTPS fetch and MUST NOT follow redirects to non-HTTPS endpoints.

13. Privacy Considerations

13.1. Encrypted DNS Transport

Implementations SHOULD use encrypted DNS transport (DoT, [RFC7858] or DoH, [RFC8484]) when resolving DNSid TXT records. The _dnsid.<fqdn> query reveals agent-to-agent interaction patterns that may be privacy-sensitive.

13.2. TLS Handshake Privacy

When a verifier connects to an agent by FQDN, the TLS ClientHello can reveal that server name to network observers through SNI. Encrypted Client Hello (ECH) [RFC9849] mitigates this disclosure by encrypting the inner ClientHello. Implementations that publish HTTPS records MAY include ECH configurations.

13.3. Privacy and Pseudonymity

Detailed identity attributes and claims MUST NOT be placed in the TXT record RDATA. These MUST be delegated to HTTPS endpoints and Verifiable Credentials. URI values in the record SHOULD NOT contain personally identifying information.

Each DNSid FQDN constitutes a distinct pseudonymous identifier. The agent operates under its FQDN without exposing the accountable entity to parties that do not perform an explicit verification. Resolving the accountable entity behind an agent FQDN may require registrar, registry, or legal process, depending on how the domain was registered and operated. Use of distinct registrable domains provides operational separation of pseudonymous identities. This property is a deliberate consequence of anchoring identity in the registrant-domain governance layer rather than in a centralized directory or certificate authority.

13.4. Ledger Privacy

The accountable entity selects the ledger technology and determines its privacy characteristics. When a publicly transparent ledger is chosen, lifecycle events (registration, rotation, revocation, retirement) and optional provenance events can reveal timing patterns. Operators using public ledgers SHOULD minimize event payloads and avoid personally identifying information in ledger entries.

14. IANA Considerations

This document requests the creation of three new registries under the "Domain Name System (DNS) Parameters" group.

14.1. DNSid TXT Tag Names Registry

Registration policy: Specification Required ([RFC8126]).
Registration requests should include the tag name, status (REQUIRED or OPTIONAL), syntax, semantics, reference, and change controller.

Tag	Status	Description
v	REQUIRED	Version identifier. MUST be "DNSid1".
oi	REQUIRED	Governance identifier.
ku	REQUIRED	Key service URI.
lr	REQUIRED	Log reference.
su	REQUIRED	Status service URI.
sg	REQUIRED	Entity signature (base64url).
fl	OPTIONAL	Policy flags.
ka	OPTIONAL	Maximum signing key age.
cu	OPTIONAL	Capabilities (Agent.md) URI.

Table 4

14.2. DNSid Policy Flag Names Registry

Registration policy: Specification Required ([RFC8126]). Registration requests should include the flag name, verifier behavior, reference, and change controller.

Flag	Description
mtls	Verifiers MUST use mutual TLS.
logchk	Verifiers MUST perform a log check before high-value operations.

Table 5

14.3. DNSid Log Method Registry

Registration policy: Specification Required ([RFC8126]). Registration requests should include the method name, reference syntax, entry format, proof format, verification procedure, security considerations, reference, and change controller.

This registry maps DNSid log method names to ledger binding specifications. It does not register URI schemes. No initial entries are defined; ledger binding specifications register their method names independently.

This document does not request the registration of a new DNS Resource Record type. See Appendix C for the planned migration path to a dedicated RR type.

14.4. DNSid Underscore Label Registration

Per [RFC8552], IANA is requested to add the following entry to the "Underscored and Globally Scoped DNS Node Names" registry:

RR Type	_NODE NAME	Reference
TXT	_dnsid	This document

Table 6

15. References

15.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/rfc/rfc1123>.

[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/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/rfc/rfc6376>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/rfc/rfc7518>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/rfc/rfc7858>.

[RFC8037] Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, DOI 10.17487/RFC8037, January 2017, <https://www.rfc-editor.org/rfc/rfc8037>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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/rfc/rfc8174>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019, <https://www.rfc-editor.org/rfc/rfc8552>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/rfc/rfc9525>.

15.2. Informative References

[I-D.ietf-scitt-architecture]
Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", October 2025, <https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture/>.

[NIST-IR-8547]
NIST, "Transition to Post-Quantum Cryptography Standards", November 2024, <https://csrc.nist.gov/pubs/ir/8547/ipd>.

[NIST-PQC] NIST, "Post-Quantum Cryptography Standards: FIPS 203, 204, 205", August 2024, <https://csrc.nist.gov/news/2024/postquantum-cryptography-fips-approved>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/rfc/rfc6698>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/rfc/rfc6763>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/rfc/rfc6962>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/rfc/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/rfc/rfc7489>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9421] Backman, A., Ed., Richer, J., Ed., and M. Sporny, "HTTP Message Signatures", RFC 9421, DOI 10.17487/RFC9421, February 2024, <https://www.rfc-editor.org/rfc/rfc9421>.

[RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[RFC9849] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", RFC 9849, DOI 10.17487/RFC9849, March 2026, <https://www.rfc-editor.org/rfc/rfc9849>.

[SPIFFE] SPIFFE Project, "The SPIFFE Identity and Verifiable Identity Document", n.d., <https://spiffe.io>.

Appendix A. Relationship to Adjacent Standards

This appendix provides informative context on how DNSid relates to adjacent efforts in the agent identity ecosystem.

SPIFFE/SPIRE: SPIFFE [SPIFFE] provides strong runtime workload identity at Layer 3. Its specification acknowledges that trust domain names are "nominally self-registered" with no delegating authority. DNSid provides the governance-backed ownership that SPIFFE trust domains can reference but do not define. SPIFFE SVIDs can include the DNSid FQDN as a URI SAN.

GoDaddy ANS: GoDaddy’s Agent Name Service uses DNS-anchored FQDNs with ACME domain control validation and a Merkle-tree transparency log. ANS spans multiple layers (L1, L2, L7). DNSid focuses solely on the Layer 1 ownership anchor and is designed to compose with ANS or operate independently.

Web Bot Auth: Built on HTTP Message Signatures [RFC9421] for automated traffic (draft-meunier-web-bot-auth-architecture). Provides per-request cryptographic authentication at Layer 4. The signing key directory at /.well-known/http-message-signatures-directory may be the same JWKS endpoint referenced by DNSid’s "ku" tag, enabling a single key management surface for both Layer 1 ownership and Layer 4 request authentication.

A2A AgentCard: The Agent2Agent Protocol’s AgentCard is a JSON

metadata document at /.well-known/agent-card.json that describes agent capabilities, endpoints, and skills at Layer 7. DNSid's optional "cu" tag can reference an AgentCard. The AgentCard's provider claim can reference the DNSid governance anchor domain, enabling verifiers to confirm the provider claim through DNSid verification.

IETF WIMSE/AIMS: WIMSE extends workload identity tokens with agent-specific claims. The AIMS framework composes WIMSE, SPIFFE, OAuth, and Transaction Tokens. Both operate at Layers 3-4. DNSid provides the Layer 1 anchor that WIMSE tokens can reference for organizational accountability.

AGENTS.md: A markdown file convention declaring agent capabilities and policies. DNSid's capabilities pointer (the "cu" tag) can reference an Agent.md document, providing the identity anchor that Agent.md declares capabilities for.

BANDAID: Uses SVCB [RFC9460] records and DNSSEC for agent discovery metadata. DNSid uses TXT records for identity anchoring. SVCB is designed for service endpoint binding; DNSid is designed for ownership binding. The two are complementary.

DNS-SD: DNS-Based Service Discovery [RFC6763] extends DNS to locate services. DNS-SD discovers what services exist; DNSid establishes who is accountable for them. The two are complementary and operate at different layers.

W3C DIDs and Verifiable Credentials: DIDs and VCs operate at the cryptographic identity and claims layers above DNSid. A DNSid FQDN MAY be the subject of a Verifiable Credential. A DID such as did:web MAY be associated with a DNSid FQDN through a VC or ledger event. DNSid does not define a DID method, does not issue or revoke Verifiable Credentials, and does not infer accountability from DID resolution alone.

SCITT: IETF SCITT [I-D.ietf-scitt-architecture] defines append-only transparency services. A SCITT-compatible service is one candidate implementation for DNSid's abstract ledger interface.

Appendix B. Ecosystem Relationship Map

The following table summarizes how DNSid relates to adjacent efforts in the agent identity ecosystem:

Project	Layer	DNSid Relationship
ICANN/DNSSEC	0	DNSid builds on Layer 0
DANE (RFC 6698)	0, 2	Optional hardening for key binding
GoDaddy ANS	1,2,7	Complementary; ANS spans more layers
Web Bot Auth	4	Shared JWKS; per-request auth complement
SPIFFE/SPIRE	3	SVID can reference DNSid FQDN
OAuth 2.1/OIDC	4	Tokens can include DNSid org ID
NGAC/OPA/Cedar	5	Policies reference DNSid ownership
SCIM for Agents	6	Provisioning triggers DNSid events
A2A AgentCard	7	Provider claim references DNSid anchor
MCP/AGNTCY	7	Agent.md referenced via "cu" tag
AGENTS.md	7	Capabilities doc; DNSid is identity
WIMSE/AIMS	3-4	WIMSE tokens reference DNSid FQDN
W3C DIDs/VCs	2	VCs can attest DNSid ownership
RFC 9421 (HTTP Message Signatures)	4	Signing key can be DNSid JWKS key
SCITT	2	Ledger architecture aligned
BANDAID	7	Uses SVCB; DNSid uses TXT; both DNS
DNS-SD (RFC 6763)	7	Discovers services; DNSid anchors accountability

Table 7

DNSid is designed to be the smallest possible Layer 1 primitive that all of these systems can reference independently.

Appendix C. Migration Path to Dedicated RR Type

This specification defines the TXT record encoding as the initial deployment mechanism. The tag-value format is designed so that a future dedicated DNSID RR type can encode the same fields in a binary wire format without any semantic changes.

The planned migration:

1. Deploy and validate the TXT encoding (this document).

2. Submit a companion specification defining the DNSID RR type with IANA allocation for a new RR type number.

3. During transition, implementations supporting both encodings SHOULD prefer the dedicated RR type when present, falling back to TXT.

4. The _dnsid owner name prefix is retained in both encodings for consistency.

The migration does not alter the trust architecture, verification protocol, or HTTP/log integration. It is a wire-format change only.

Within a version (e.g., DNSid1), the format is extensible: new OPTIONAL tags may be defined in companion specifications, and receivers MUST ignore unknown tags (Section 5.2). This allows incremental capability additions without a version transition. A change to the version tag (e.g., DNSid1 to DNSid2) constitutes a flag day: receivers that do not understand the new version treat the record as absent. This is consistent with the transition model used by SPF, DKIM, and DMARC, and is acceptable because version changes to identity infrastructure are infrequent, planned events.

DNSid1 intentionally defines a singleton TXT record containing one DNSid record. This avoids ambiguity in TXT RRset selection and keeps the initial deployment model simple. Future versions of this specification may define an overlapping transition mechanism, such as allowing two versioned DNSid records during a bounded migration period, or may rely on migration to a dedicated DNSID RR type. Such a mechanism would need to specify record selection, field consistency across versions, and signature verification rules.

Appendix D. Acknowledgments

This work builds on the DNSid concept developed at Identity Digital Innovation Labs. The author thanks Ben Guidarelli for contributions to the TXT record encoding profile, Jason Weathersby for technical review, Emily Edwards for product refinement, and the participants of the Linux Foundation AAIF Identity and Trust Working Group for ongoing discussion.

The TXT record encoding follows conventions established by DKIM ([RFC6376]), DMARC ([RFC7489]), and SPF ([RFC7208]).

Author’s Address

Naveed Ihsanullah
Identity Digital
Email: naveed.ihsanullah@identity.digital