dnsop
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

J. Mozley
N. Williams
Infoblox, Inc.
B. Sarikaya
Unaffiliated
R. Schott
Deutsche Telekom
2 March 2026

DNS for AI Discovery
draft-mozleywilliams-dnsop-dnsaid-01

Abstract

This document specifies a method for utilizing the Domain Name System (DNS) to facilitate scalable and interoperable discovery between AI agents. The proposed mechanism, referred to as "DNS AI agent Discovery (DNS-AID)", defines a structured DNS namespace and record usage model to support metadata exchange and capability advertisement.

This will allow organisations to publish information about their AI agents on the Internet or internal networks using a well-known label within the organisation’s own DNS namespace. This document does not define how the published agent information is accessed or the exact structure of that information. Instead, it specifies a mechanism for indicating which access protocol should be used and what format the agent information will be provided in.

This document proposes no change to the structure of DNS messages, and no new operation codes, response codes, resource record types, or any other new DNS protocol values.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://example.com/LATEST. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mozleywilliams-dnsop-dnsaid/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://github.com/USER/REPO.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes DNS for AI Agent Discovery (DNS-AID), a mechanism that enables agents to retrieve operational parameters prior to initiating a session. DNS-AID introduces a leaf zone convention (e.g., _agents.example.com) containing Service Binding (SVCB) records (e.g., chat._agents.example.com) that encode application-specific metadata. It is these records that enable agents to retrieve operational parameters prior to initiating a session, supporting both targeted lookups and capability-based discovery.

The approach leverages existing DNS protocols and records, including DNS Service Discovery (DNS-SD), DNSSEC, and DANE, to provide integrity, authenticity, and automation without requiring human intervention and without requiring modification of the basic principles of DNS. Lastly DNS-AID leverages Domain Control Validation (DCV) described as a best current practice to prove an agent is authorized to act on behalf of a domain in [I-D.draft-ietf-dnsop-domain-verification-techniques].

DNS-AID provides the bootstrap for discovering an organization's agents. An organization can publish it's own registry of agents and their capabilities at a well-known entry point in the DNS hierarchy. It is also possible to provide names for commonly used agents, so that no registry needs to be queried and the returned capabilities processed (e.g., on a model card or other schema) before the name of can be resolved. It is expected that other documents will develop the content of any registry, such as refining the model card concept to a more structure schema, and will specify the protocol used to interact with the registry. It is more performant and deterministic to receive the details of the IP, protocols and port from an SVCB record than it is to determine this from a model card. It also mitigates against the vulnerability of parsing this information from a model card.

For example, a company may only have a few external agents available for use, and if the IETF standardizes 'types' of AI agents, then perhaps _chat._agents.example.com or _img2txt._agents.example.com etc are all different SVCB records. On the other hand a company may provide an index of all agents via a well know entry point in the DNS hierarchy e.g. _index._agents.example.com.

For example, a company might have only a few external agents available for use, and if the 'types' of AI agent are standardized, then (for example) _chat._agents.example.com and _img2txt._agents.example.com would use different SVCB records. On the other hand, a company may provide an index of all agents via a well-known entry point in the DNS hierarchy, e.g., _index._agents.example.com.

The DNS-AID model is designed for incremental and non-disruptive deployment within existing DNS infrastructure. It introduces no new DNS message formats, opcodes, response codes, or resource record types. Instead, it defines a structured namespace convention and usage profile for existing record types (primarily SVCB, TXT, and TLSA) all within designated leaf zones (e.g., _a2a._agents.example.com).

Organizations may adopt DNS-AID by publishing agent metadata under delegated subdomains, leveraging DNSSEC for integrity and authenticity, and optionally implementing Domain Control Validation (DCV) to signal delegated authority. These zones may be exposed selectively via split-horizon DNS or different zones, enabling differentiated discovery views for internal and external agents. No changes are required to recursive or authoritative DNS server implementations beyond standard support for DNSSEC and SVCB.

This model supports opt-in adoption, allowing agent operators to publish discovery metadata without coordination with external registries or protocol maintainers. It is compatible with existing DNS tooling, including zone provisioning systems, monitoring platforms, and resolver configurations. DNS-AID is therefore suitable for deployment across enterprise, cloud, and federated environments, and may coexist with other discovery mechanisms without conflict.

2. Conventions and Definitions

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.

3. DNS Mechanisms for Agent Discovery

AI agents that need to be discovered are currently being developed to support tasks such as travel planning, etc. See Figure 1.

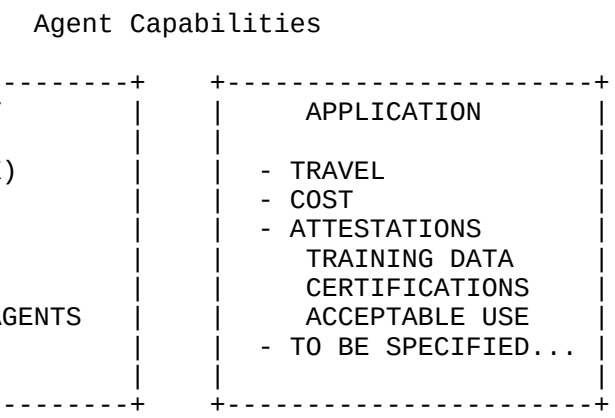


Figure 1: AI Agent Capabilities

3.1. Use of Service Binding Records in Discovery

Agents will use SVCB records as defined in [RFC9460] to discover other agent endpoints under structured leaf zones (e.g., _chat._agents.example.com). These records allow querying agents to retrieve operational parameters prior to initiating a session, including supported protocols, privacy features (e.g., Encrypted Client Hello via ECH), and failover configurations.

Agents may advertise support for QUIC, HTTP/3, or agent-to-agent protocols via ALPN declarations. Operators may specify alternate endpoints or migrate services across domains without relying on CNAME indirection, improving performance and reducing DNS resolution complexity.

When paired with attribute leaf zones and custom SVCB parameters, this mechanism enables fine-grained discovery of agent capabilities. For instance, an agent querying _a2a._agents.example.com may receive an SVCB record indicating supported modalities (e.g., chat, image-to-text), expected input formats, and cost-related metadata (e.g., token pricing). This structured discovery model supports deterministic, cacheable, and semantically rich interactions between agents.

SVCB records can indicate the protocol used to communicate with a given agent as in Figure 2:

```
_index._agents.example.com. 3600 IN SVCB 1 ai-index-svc.example.com. (
    alpn="a2a"
    port=443
    ipv4hint=192.0.2.1
    ipv6hint=2001:db8::1
)
```

Figure 2: SVCB Record Example

3.2. DNS-Based Service Discovery

DNS-SD as defined in [RFC6763], extends the capabilities of the Domain Name System to support service-type-based discovery. Unlike previous DNS resolution, which requires prior knowledge of a specific hostname, DNS-SD enables clients to discover services based on their functional type (e.g., _http._tcp, _printer._udp) within a given domain.

DNS-SD provides a mechanism for AI agents to discover other agents or services based on declared capabilities rather than explicit names. For example, an agent may issue a query for _data-cleaner._a2a._agents.example.com to locate services capable of performing data sanitization tasks. This type-based discovery model supports dynamic ecosystems where agent roles and capabilities may evolve over time.

DNS-SD operates over both multicast DNS (mDNS) and unicast DNS, allowing for flexible deployment in local networks and globally scoped domains. In enterprise or cloud environments, unicast DNS-SD enables structured and policy-compliant discovery across organizational boundaries. When combined with SVCB records, DNS-SD allows agents to retrieve detailed service metadata, including transport preferences, protocol support, and operational parameters.

This model supports federated agent architectures, where multiple agents may advertise similar capabilities under different domains. By leveraging DNS-SD, agents can perform capability-based queries and receive a list of candidate services, each accompanied by structured metadata for evaluation and selection.

DNS-SD also supports extensibility through TXT records and custom service naming conventions, enabling organizations to encode additional attributes relevant to agent interaction (e.g., supported data formats, authentication requirements, or cost models). A TXT record with the same name as the SVCB record could be used to provide additional meta-data. These features make DNS-SD a suitable foundation for scalable, interoperable, and semantically rich agent discovery workflows.

An index of agents by type of service can be provided via DNS-SD as a well known entry point to a more complete capability description via a common schema, e.g., _index._agents.example.com. This may be used in addition to or instead of querying for specific agent services such as _data-cleaner._a2a._agents.example.com. Querying specific services will shortcut communication compared to querying an index.

3.3. Domain Control Validation

DCV refers to the process by which an entity demonstrates authoritative control over a DNS domain. As described in [I-D.draft-ietf-dnsop-domain-verification-techniques], DNS-based DCV typically involves the placement of a DNS record, most commonly a TXT record, containing a challenge token or assertion at a designated location within the domain. This record is then queried by an Application Service Provider (ASP) to confirm control.

Authorization should not solely exist solely within the discovery phase, however there may be use cases in which agents acting on behalf of domains, prior to application handshake, can prove they are acting on behalf of a domain (organization). This may be a preferable mechanism, especially for ephemeral agents, to prove they act on behalf of a domain, rather than manage many temporary security mechanisms across multiple information security pillars (firewalls, certs, etc.).

In the context of DNS-AID, DCV provides a mechanism for agents to assert delegated authority on behalf of a domain. For example, an organization may publish a TXT record under a structured leaf zone (e.g., _agent-roles._a2a._agents.example.com) containing metadata such as:

ai-role=data-cleaner; crm=dummyorg; access=readonly

This record signals that a specific agent is authorized to perform scoped operations under the domain’s authority. When protected by DNSSEC [RFC9364], such assertions become cryptographically verifiable, mitigating risks of spoofing or unauthorized delegation.

DCV within DNS-AID supports both ephemeral and persistent validation models. Ephemeral validation may be used for short-lived agent credentials or session-based delegation, while persistent records enable long-term authorization signaling. TTL and expiration considerations, as discussed in the draft, are critical to ensuring that validation records do not persist beyond their intended scope.

Furthermore, DNS-AID encourages the use of application-specific naming conventions and token formats to avoid service confusion and collision. Validation records should be scoped to the intended service and include unguessable tokens or structured metadata to prevent unauthorized reuse across federated agent ecosystems.

4. Architecture

The DNS hierarchical namespace supports multi-tenant and federated discovery models. Organizations may delegate subdomains to tenants (e.g., customer1._agents.vendor.com) or publish agent metadata under federated zones (e.g., region1._a2a.example.org). This enables scoped discovery, policy enforcement, and operational isolation across tenants, departments, or geographic regions. Combined with DNSSEC and access controls, this model supports secure delegation and trust signaling in complex agent ecosystems, including SaaS platforms and cross-organizational collaborations.

4.1. Example Communication

Agent (org1) wants to find other agents provided by another entity (org2) and discover/verify capabilities Figure 3:

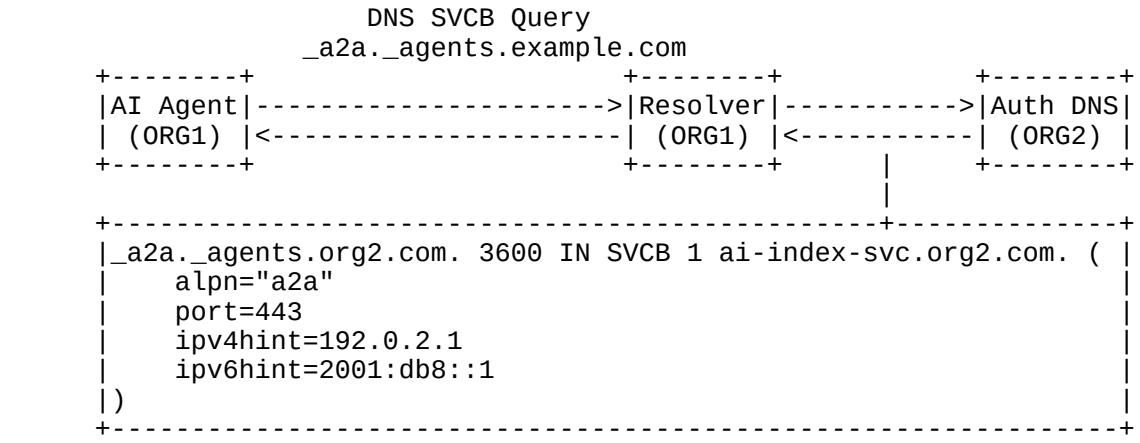


Figure 3: Agent ORG1 Querying Org2

4.2. Example Record

```
_a2a._agents.org2.com. 3600 IN SVCB 1 ai-index-svc.org2.com. (
    alpn="a2a"
    port=443
    ipv4hint=192.0.2.1
    ipv6hint=2001:db8::1
)
```

Figure 4: Example SVCB Resource Record

* _a2a._agents.example.com.: The service name, following the SVCB naming convention (_service._agents.example.com).
* 3600: TTL (Time to Live) in seconds.
* IN SVCB: The record type.
* 1: SVCB priority. 0 is for alias mode; 1+ is for service mode.
* ai-index-svc.org2.com.: The target name of the service.
* alpn="a2a": Specifies the ALPN protocol identifier. This is where your custom protocol is declared.

* port=443: The port on which the service is available.

* ipv4hint and ipv6hint: Optional hints for clients to connect directly.

4.3. Example Use Cases

1. A user instructs their internal agent to "clean up DummyCorp contacts based on this email." The agent must discover DummyCorp's authorized agents, validate its own delegation to act on behalf of the enterprise, and initiate a secure session. DNS-AID enables this by publishing agent endpoints and roles under DNS zones controlled by each organization.

2. A research consortium deploys agents across multiple institutions. Each institution publishes its agents under its own domain (e.g., _a2a._agents.universityA.example.edu), allowing collaborators to discover services based on capability (e.g., _data-annotator._a2a.universityB.example.edu) while respecting institutional boundaries and trust models.

3. A SaaS provider hosts agents for multiple customers. Each customer's agents are published under tenant-specific zones (e.g., customer1._agents.saas.com), enabling scoped discovery and policy enforcement. DNS-AID supports this model through hierarchical zone delegation and metadata-rich SVCB records.

4. Lightweight agents deployed on mobile or edge devices require low-latency, cacheable discovery mechanisms. The DNS distributed architecture and support for SVCB hints (e.g., IP addresses, preferred protocols) enable efficient resolution and connection bootstrapping in constrained environments.

5. In regulated industries, agents must operate within jurisdictional boundaries and maintain audit trails of interactions. DNS supports geographic scoping via ccTLDs and split-horizon configurations, while query logging and DNSSEC provide observability and integrity guarantees.

4.4. Zone and Other Requirements

4.4.1. Delegation and Chain of Trust

A public authoritative zone used for the purposes of agent discovery MUST use DNSSEC [RFC4033]. The zone MUST establish a complete chain of trust to a publicly recognized trust anchor. AI agents MUST use validating resolvers, or have the capability to validate records.

All DNS-AID-specific discovery records (e.g. SVCB/HTTPS [RFC9460], TXT/URI [RFC7553] used for capability descriptors, and any DNS-AID-defined RRTypes) MUST be signed and published in the DNS-AID External Zone. Where DNS-AID endpoints rely on TLS, publication of DANE TLSA records MAY be used to bind endpoint certificates to DNSSEC-validated names [RFC6698][RFC7671]. Resolver behavior consuming DNS-AID data MUST treat DNSSEC-bogus responses as failures and MUST NOT act on unsigned or invalidly signed discovery data.

4.4.2. Performance Optimization(s)

Each agent service endpoint that is specifically published as a DNS record SHOULD be an SVCB record in ServiceMode (or HTTPS RR for HTTPS endpoints) to convey connection parameters and capability locators with a single lookup [RFC9460]. SVCB "address hints" (ipv4hint, ipv6hint) SHOULD be used to reduce A/AAAA follow-up queries; resolvers MAY still validate final addresses via canonical resolution. Where human-friendly names are required, SVCB AliasMode (priority 0) MAY be used to map from a stable alias name to a canonical name.

Authoritative servers MUST support EDNS(0) [RFC6891] and TCP fallback [RFC7766]. Operators SHOULD target response sizes that avoid IP fragmentation on common paths (e.g., <= 1232 bytes on IPv6), preferring compression and layered indirection over oversized RRsets. TTLs MUST be chosen to reflect the volatility of capability data; index/indirection records may use longer TTLs than frequently changing endpoint or capability descriptors. Negative caching MUST conform to [RFC2308].

A representative naming pattern is shown below; the exact label order is deployment-specific, but the leaf per service, per agent rule applies:

```
; Hashed, per-agent, per-service leaf
a4k2f9._mcp._agents.example.org.  600 IN SVCB 1 svc-a4k2f9.example.org.
    alpn="map,h2,h3" port=443 ipv6hint=2001:db8::5 ipv4hint=192.0.2.5

; Optional alias for a friendlier owner name
billing._mcp._agents.example.org. 300 IN SVCB 0 a4k2f9._mcp._agents.example.org.
```

4.4.3. Customization(s)

DNS-AID deployments MAY define additional SVCB parameters (SvcParamKeys) to convey agent-specific capability metadata, provided that interoperability safeguards in [RFC9460] are observed. During experimentation, unregistered keys MUST use the numeric keyNNNNN presentation form, and any client behavior that depends on them MUST be gated via the mandatory SvcParam to ensure downgrade safety.

This specification defines the following provisional SvcParamKeys for DNS-AID (names are illustrative; production deployments MUST register through IANA per [RFC9460] or use keyNNNNN until standardized):

The following are example use cases for additional params:

* cap - a capability descriptor locator or inline identifier (e.g., a URN or compact JSON-Ref) that identifies the agent's advertised capability schema/version.

* cap-sha256 - a base64url-encoded SHA-256 digest of the canonical capability descriptor to support integrity checks and cache revalidation.

* policy - a URI or URN identifying a policy bundle applicable to this agent (e.g., jurisdiction, data handling class) for client-side selection.

* realm - an opaque token for multi-tenant scoping or authz realm selection during protocol bootstrapping.

Clients that require any of these parameters MUST verify their presence via the mandatory key; otherwise, the SVCB record MUST be ignored. Example:

```
Single-RRTYPE publication with custom params (experimental keys shown)
a4k2f9._mcp._agents.example.org. 600 IN SVCB 1 svc-a4k2f9.example.org.
    alpn="h2" port=443 ipv4hint=192.0.2.5
    mandatory=alpn,port,key65001,key65002,key65010
    key65001="cap=urn:cap:example:mcp:invoice.v1"
    key65002="cap-sha256=yvZ0n7q8bE2gYkz8m1j1s0yQG0mC2F6qj3b9pVb6Gk0"
    key65010="bap=a2a/1,mcp/1"
```

Where endpoints are HTTPS, the HTTPS RR variant SHOULD be used to co-locate transport parameters such as ECH with capability metadata; otherwise, generic SVCB MAY be used. Future standardization SHOULD define the exact syntax (e.g., ABNF) and registry policy for these keys, including error handling for malformed or conflicting parameters. Until registration is complete, deployments MUST treat unknown keyNNNNN parameters as opaque and MUST NOT infer semantics without out-of-band agreement.

5. Implementation Guidance

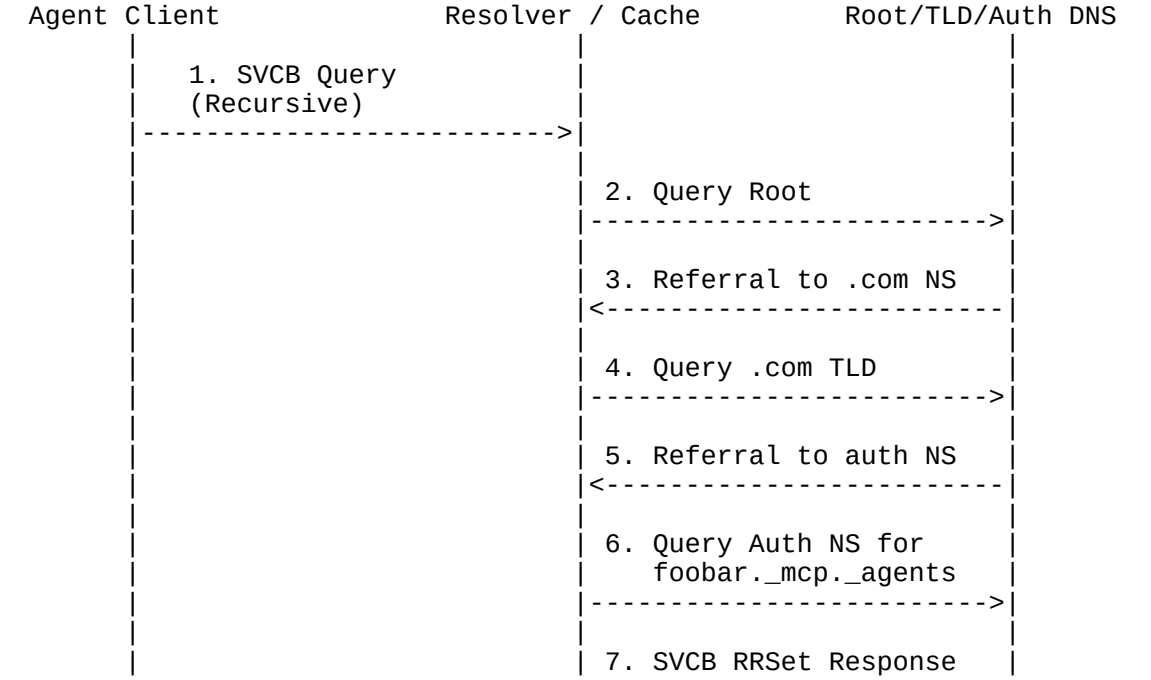
5.1. Discovery Usage Examples

5.1.1. Discovery Status 1 (Known Service and Domain)

Query foobar._mcp._agents.example.com Figure 5:

In this scenario, the AI Agent Client knows both the service type (mcp) and the authoritative domain (example.com) and performs a direct SVCB query.

AI Agent Client
wants to discover: foobar._mcp._agents.example.com
(mcp = service, foobar = agent/capability, example.com = trusted domain)



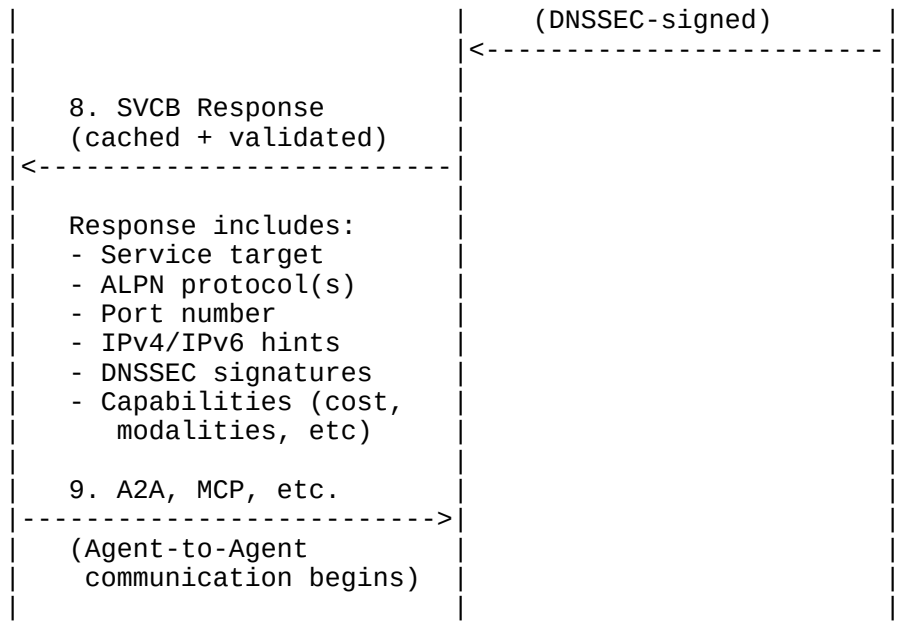


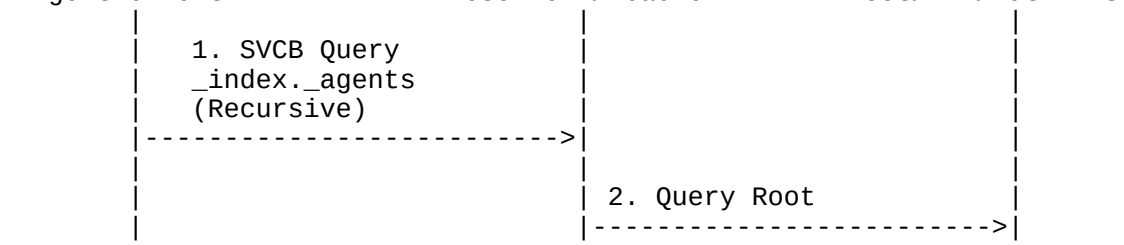
Figure 5: Discovery Status 1: SVCB Query with Full Chain Traversal

5.1.2. Discovery Status 2 (Known Service OR Domain, Trusted)

Query of the known organisation’s index if service is unknown
Figure 6:

In this scenario, the Agent Client knows the trusted domain (example.com) but the specific agent service type is unknown. The client queries a well-known index entry point to discover available agent capabilities. The index query is not DNS.

AI Agent Client
knows: example.com is trusted
wants to discover: available agents and services
queries: _index._agents.example.com (well-known entry point)



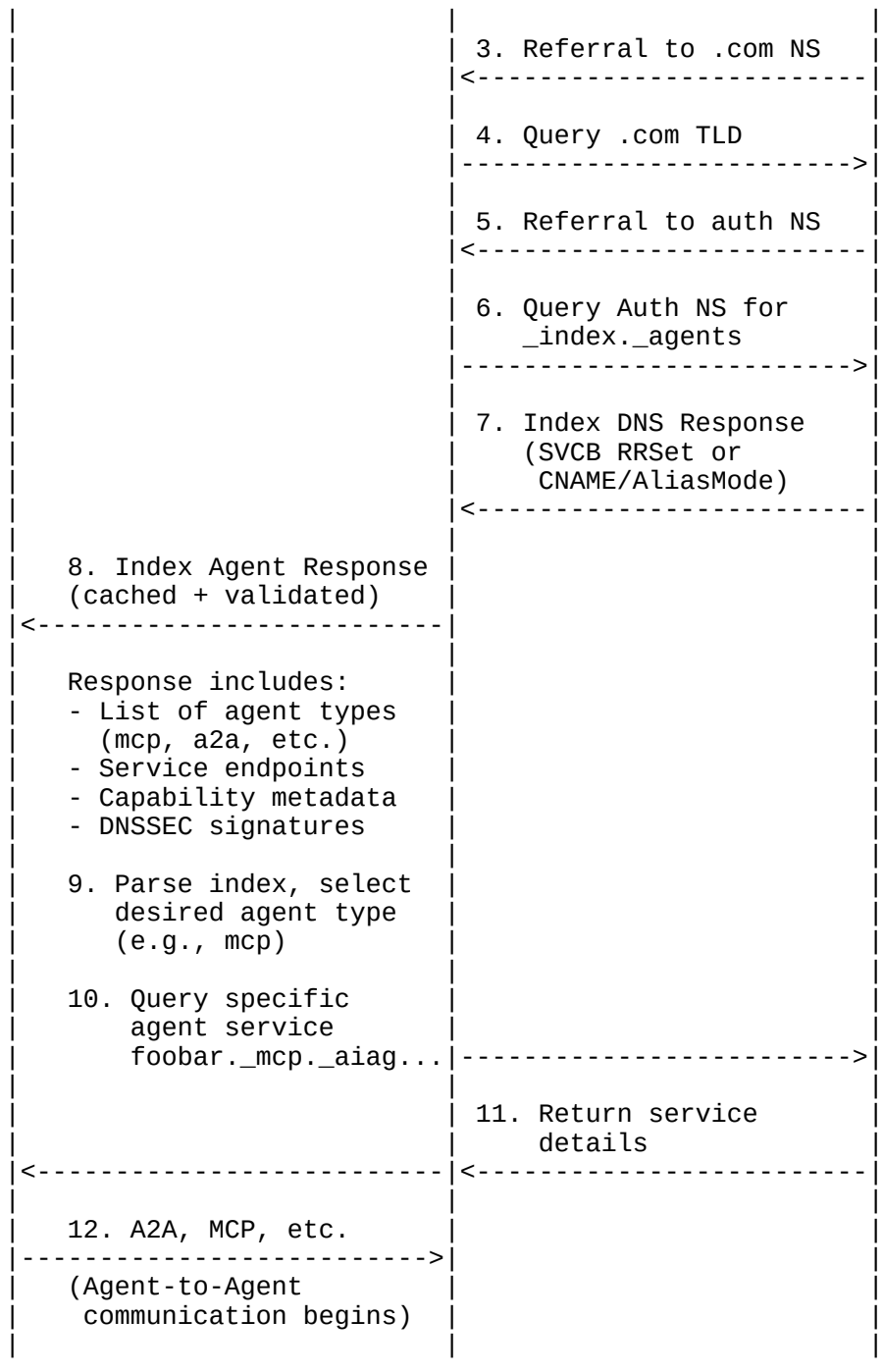
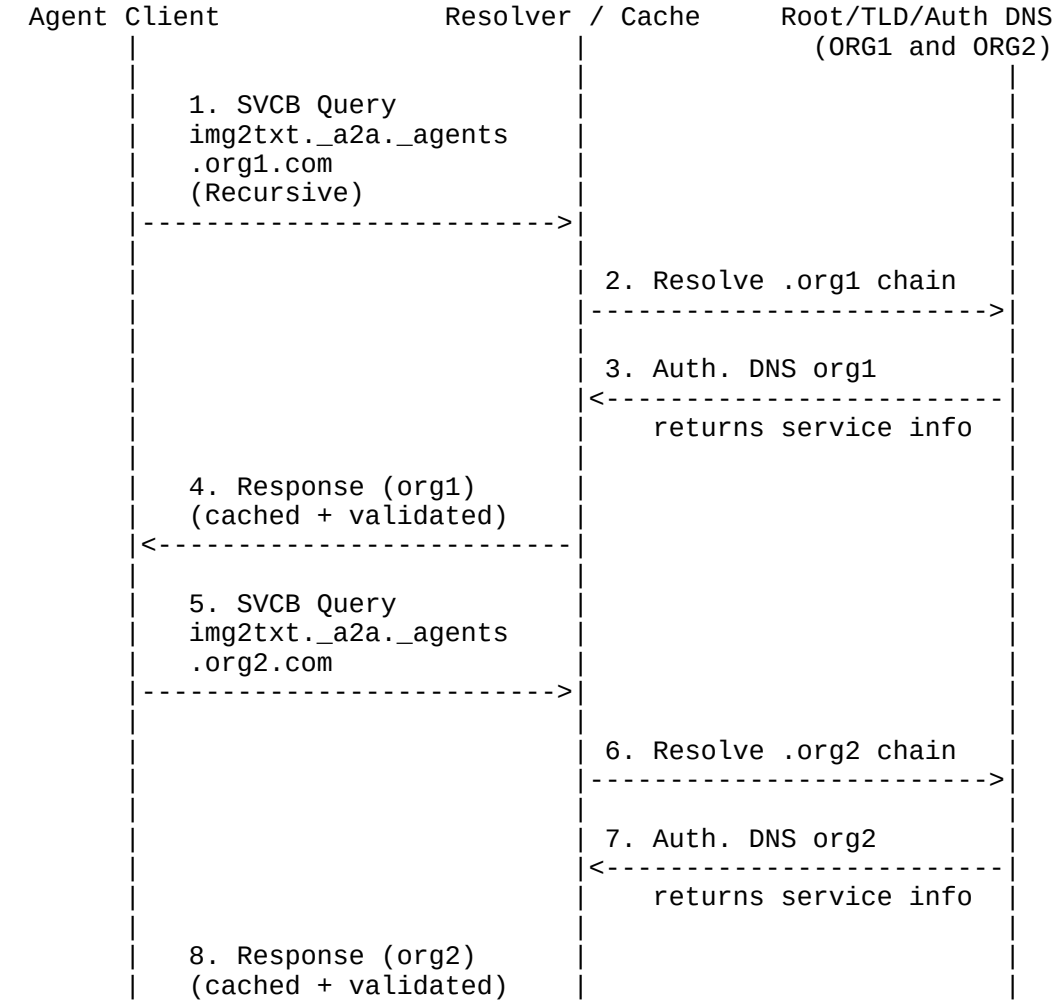

3. Referral to .com NS
4. Query .com TLD
5. Referral to auth NS
6. Query Auth NS for _index._agents
7. Index DNS Response (SVCB RRSet or CNAME/AliasMode)
8. Index Agent Response (cached + validated)
Response includes:
- List of agent types (mcp, a2a, etc.)
- Service endpoints
- Capability metadata
- DNSSEC signatures
9. Parse index, select desired agent type (e.g., mcp)
10. Query specific agent service foobar._mcp._aiag...
11. Return service details
12. A2A, MCP, etc.
(Agent-to-Agent communication begins)

Figure 6: Discovery Status 2: Index-Based Discovery with Service Selection

5.1.3. Discovery Status 3 (Multi-Domain Query for Known Service)

Query the same well-known agent name across multiple known domains (potential for this discovery case to be met by the registry described in Discovery Status 4):

AI Agent Client
knows: img2txt is a well-known agent type
knows: org1.com and org2.com are trusted domains
wants to discover: which orgs have img2txt agents available



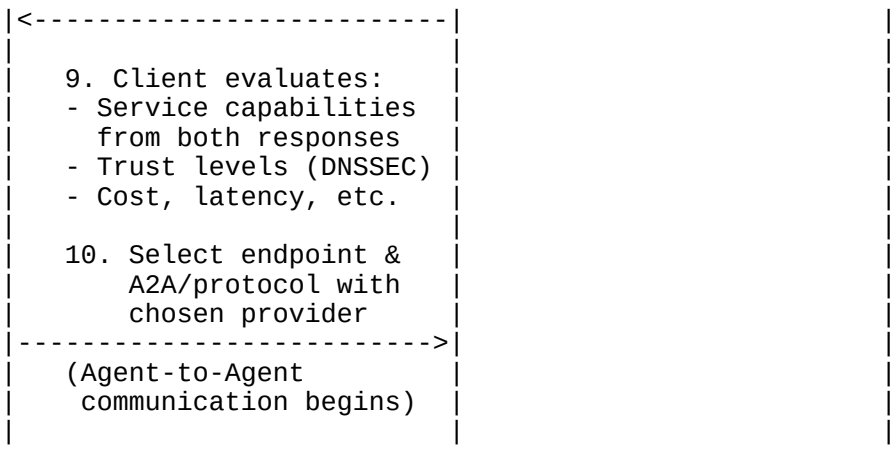
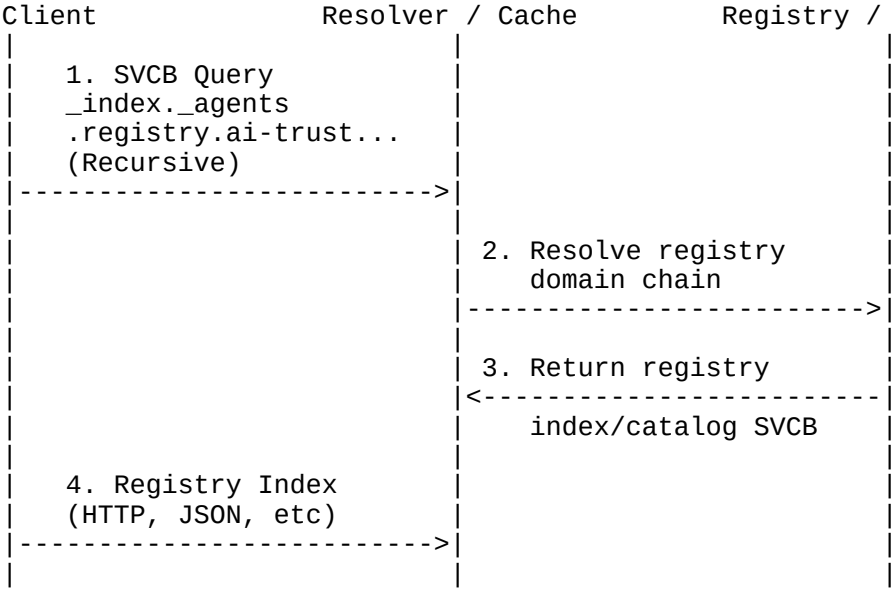


Figure 7: Discovery Status 3: Parallel Multi-Domain Service Discovery

5.1.4. Discovery Status 4 (Untrusted Domain / Consolidated Registry)

Query a consolidated (reputable) registry (not DNS) when neither domain nor comprehensive service list is known:

AI Agent Client
knows: a trusted registry (e.g., registry.ai-trust-community.org)
wants to discover: all viable img2txt agents available (any org)
scenario: agent choice driven by capability, not domain affinity



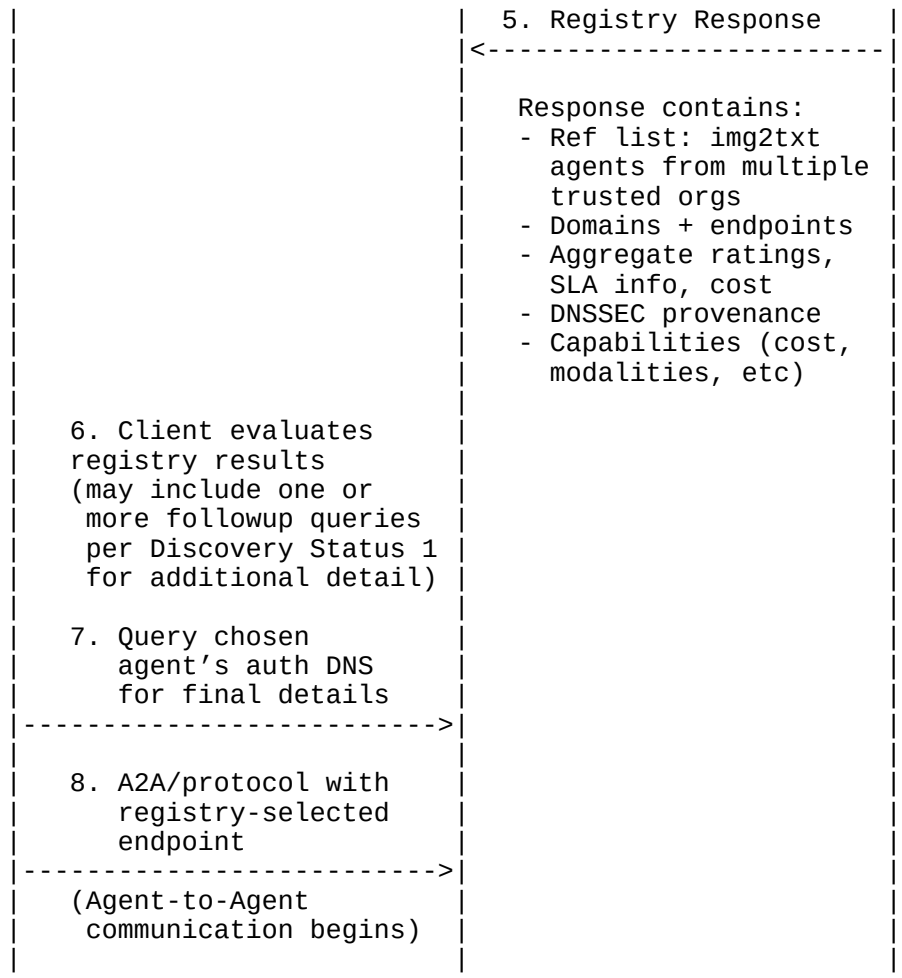


Figure 8: Discovery Status 4: Registry-Based Discovery Across Multiple Providers

5.1.5. Discovery Status 4 (Unknown / Wildcard, or status 2 untrusted?)

This case is out of scope.

5.2. AI Providers

5.2.1. Publishing Schema

Publishers SHOULD expose per-service discovery records using SVCB (or HTTPS for HTTP origins) at stable, well-scoped owner names (e.g., agent-id._mcp._agents.example.org.). SVCB ServiceMode conveys connection parameters and capability locators in a single round trip; AliasMode MAY be used to map a friendly name to a hashed leaf for sharding without duplicating RRSets. Initial connection parameter keys (alpn, port, address hints) and the mandatory key MUST be honored by clients. [RFC9460]

Minimal example:

```
; ServiceMode SVCB for an MCP-capable agent
a4k2f9._mcp._agents.example.org.  600 IN SVCB 1 svc-a4k2f9.example.net.
    alpn="h2,h3" port=443 ipv6hint=2001:db8::5 ipv4hint=192.0.2.5
    mandatory=alpn,port
```

SVCB/HTTPS semantics and SvcParam processing MUST follow [RFC9460] to ensure interop and correct fallback.

5.2.2. TTLs, Update Agility

Publishers SHOULD assign longer TTLs to relatively static indirection records (aliases, stable service labels) and shorter TTLs to volatile endpoint/capability records to balance cache efficiency with rollout safety. Consumers will apply negative caching per [RFC2308], so publishers SHOULD avoid unnecessary NXDOMAIN flaps during deployments (e.g., prefer blue/green leaves over in-place deletions).

5.2.3. Example DNS-AID Zonefile

```
$ORIGIN example.org.
$TTL 3600

; ------------------------------------------------------------------
; SOA / NS
; ------------------------------------------------------------------
@               IN SOA  ns1.example.org. hostmaster.example.org. (
                        2025091901 ; serial (YYYYMMDDnn)
                        7200       ; refresh
                        1800       ; retry
                        1209600    ; expire
                        3600 )     ; minimum
                IN NS    ns1.example.org.
                IN NS    ns2.example.org.

ns1             IN A     192.0.2.53
```

```
ns1             IN AAAA  2001:db8::53
ns2             IN A     192.0.2.54
ns2             IN AAAA  2001:db8::54

; ------------------------------------------------------------------
; DNS-AID discovery (MCP example): per-service, per-agent leaf
; - Leaf is a stable mapping from AgentID -> label (e.g. hashed).
; - Use SVCB ServiceMode (priority > 0) to bind connection parameters.
; ------------------------------------------------------------------

; AliasMode to keep a friendly name stable even if the leaf changes
billing._mcp._agents    300 IN SVCB 0 a4k2f9._mcp._agents.example.org.

; Leaf per agent/service (ServiceMode). Shorter TTL for agility.
a4k2f9._mcp._agents     600 IN SVCB 1 svc-a4k2f9.example.net. \
                            alpn="h2,h3" \
                            port=443 \
                            ipv4hint=192.0.2.5 \
                            ipv6hint=2001:db8::5 \
                            mandatory=alpn,port

; (Optional) Another service face for the same agent (A2A)
a4k2f9._a2a._agents     600 IN SVCB 1 svc-a4k2f9.example.net. \
                            alpn="h2" port=8443 \
                            mandatory=alpn,port

; (Optional) HTTPS RR at apex for web-style consumers of DNS-AID
; metadata
@                        900 IN HTTPS 1 web-gw.example.net. \
                            alpn="h2,h3" port=443

; ------------------------------------------------------------------
; Capability / policy signaling via SVCB custom keys (experimental)
; Use numeric keyNNNNN until you register IANA SvcParamKeys.
; Gate client requirements with "mandatory".
; ------------------------------------------------------------------
; Example: reference a capability descriptor and advertise supported
; app protocols.
; NOTE: Values and key numbers are illustrative.

a4k2f9._mcp._agents 600 IN SVCB 1 svc-a4k2f9.example.net. \
                        alpn="h2,h3" port=443 \
                        mandatory=alpn,port,key65001,key65010 \
                        key65001="cap=urn:cap:example:mcp:invoice.v1" \
                        key65010="bap=a2a/1,mcp/1"

; ------------------------------------------------------------------
; Domain Control Validation (DCV) for DNS-AID authorization
```

```
; Publish a short-lived token to prove control over example.org.
; Remove after successful validation. Resolver MUST DNSSEC-validate.
; ------------------------------------------------------------------------
_agents-challenge 300 IN TXT "bnd-req=svc:crm-sync@vendor.example;nonce=3Qz6l8pA;exp=2025
-09-19T06:00:00Z"

; ------------------------------------------------------------------------
; Optional DANE TLSA for the service endpoint used above.
; Owner: _‘<port>‘._tcp.‘<endpoint-FQDN>‘
; Usage 3 (DANE-EE), Selector 1 (SPKI), Match 1 (SHA-256)
; Replace the hash with the actual SPKI hash of the leaf certificate.
; ------------------------------------------------------------------------
_443._tcp.svc-a4k2f9.example.net.  1800 IN TLSA 3 1 1 (
    0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789AB )

; ------------------------------------------------------------------------
; (Optional) Address records for the service endpoint, if in-zone.
; If hosted out-of-zone (example.net), these will live there instead.
; ------------------------------------------------------------------------
svc-a4k2f9               900 IN A     192.0.2.5
svc-a4k2f9               900 IN AAAA  2001:db8::5
web-gw                   900 IN A     192.0.2.80
web-gw                   900 IN AAAA  2001:db8::80
```

5.2.4. How to use this zonefile

- Adjust TTLs for agility vs. cache efficiency. Use longer TTLs on indirection (aliases), shorter TTLs on volatile leaves and DCV. This aligns with DNS caching behavior and negative caching rules [RFC2308].

- Follow SVCB/HTTPS semantics. Clients must honor SvcPriority, AliasMode (priority 0), and ServiceMode parameters, including mandatory, alpn, port, and address hints per [RFC9460].

- DCV token flow. Issue a time-bounded token at _agents-challenge.domain and require DNSSEC-validated retrieval (pattern analogous to ACME’s DNS-01 in [RFC8555]). Remove on success.

- Bind transport to DNS with DANE (optional). If your relying parties validate DNSSEC, publish TLSA to bind the endpoint’s cert/key, using the operational guidance in [RFC7671] (TLSA record format in [RFC6698]).

- Sign the zone. This file is pre-signing. Sign with DNSSEC (DNSKEY/DS/RRSIG, etc.) before deployment; validators must treat unsigned/bogus discovery data as a failure (per your earlier spec). See DNSSEC core specs [RFC4033], [RFC4034], [RFC4035] and operational guidance.

5.3. Why these records

* SVCB/HTTPS concentrate connection metadata and allow aliasing at apex and service labels, reducing round trips and enabling policy-gated parameters via mandatory.

* Per-service, per-agent leaves avoid oversized RRsets and allow parallel administration/sharding while keeping alias names stable. This follows performance guidance in SVCB and your DNS-AID perf section.

* DCV via TXT mirrors a well-understood, automated control-proof pattern [RFC8555] that fits DNS-AID authorization workflows.

* DANE TLSA optionally removes reliance on external PKI anchors for endpoint auth where DNSSEC is deployed [RFC6698], with deployment rules in [RFC7671].

6. Future Work & Unaddressed Portions

How a consolidated registry would operate? Would it gather and publish information from individual organisations (scraping), would those organisations registery attest to security and be verified?

7. Security Considerations

DNSSEC and other security considerations apply. More work is needed to expand on this.

8. Operational Considerations

Work is needed to expand on operational considerations.

9. IANA Considerations

IANA maintains a registry of "Underscored and Globally Scoped DNS Node Names" per [RFC8552]. IANA is is requested to register an underscored attribute leaf for AI agents. _agent is suggested.

IANA maintains a registry of "DNS SVCB Service Parameter Keys (SvcParamKeys)" per [RFC9460]. IANA is requested to designate custom key-value pairs for SVCB parameters to facilitate agent-to-agent application discovery and cost optimization. The following parameters are requested:

Name	Meaning
cap	capability descriptor locator or inline identifier | (e.g., a

URN or compact JSON-Ref) cap-sha256 | capability base64url-encoded SHA-256 digest of the | canonical capability descriptor policy | URI or URN identifying a policy bundle applicable to an | agent realm | opaque token for multi-tenant scoping or authz realm | selection during protocol bootstrapping

10. References

10.1. Normative References

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

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/rfc/rfc2308>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/rfc/rfc4033>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/rfc/rfc6891>.

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/rfc/rfc7766>.

[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>.

[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>.

10.2. Informative References

[I-D.draft-ietf-dnsop-domain-verification-techniques] Sahib, S. K., Huque, S., Wouters, P., Nygren, E., and T. Wicinski, "Domain Control Validation using DNS", Work in

Progress, Internet-Draft, draft-ietf-dnsop-domain-verification-techniques-11, 1 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-domain-verification-techniques-11>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/rfc/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/rfc/rfc4035>.

[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>.

[RFC7553] Faltstrom, P. and O. Kolkman, "The Uniform Resource Identifier (URI) DNS Resource Record", RFC 7553, DOI 10.17487/RFC7553, June 2015, <https://www.rfc-editor.org/rfc/rfc7553>.

[RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <https://www.rfc-editor.org/rfc/rfc7671>.

[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>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/rfc/rfc8555>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/rfc/rfc9364>.

Acknowledgments

The authors thank Aijun Wang, Ben Schwartz and Ross Gibson for their contributions, questions, and comments.

Authors' Addresses

Jim Mozley
Infoblox, Inc.
Email: jmozley@infoblox.com

Nic Williams
Infoblox, Inc.
Email: nic@infoblox.com

Behcet Sarikaya
Unaffiliated
Email: sarikaya@ieee.org

Roland Schott
Deutsche Telekom
Email: roland.schott@telekom.de