Stream: Internet Engineering Task Force (IETF)
RFC: 9460
Category: Standards Track
Published: November 2023
ISSN: 2070-1721
Authors: B. Schwartz, *Meta Platforms, Inc.*; M. Bishop, *Akamai Technologies*; E. Nygren, *Akamai Technologies*

RFC 9460
Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)

Abstract

This document specifies the "SVCB" ("Service Binding") and "HTTPS" DNS resource record (RR) types to facilitate the lookup of information needed to make connections to network services, such as for HTTP origins. SVCB records allow a service to be provided from multiple alternative endpoints, each with associated parameters (such as transport protocol configuration), and are extensible to support future uses (such as keys for encrypting the TLS ClientHello). They also enable aliasing of apex domains, which is not possible with CNAME. The HTTPS RR is a variation of SVCB for use with HTTP (see RFC 9110, "HTTP Semantics"). By providing more information to the client before it attempts to establish a connection, these records offer potential benefits to both performance and privacy.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9460.

Copyright Notice

Table of Contents

1. Introduction 5
   1.1. Goals 5
   1.2. Overview of the SVCB RR 6
   1.3. Terminology 7
2. The SVCB Record Type 7
   2.1. Zone-File Presentation Format 8
   2.2. RDATA Wire Format 8
   2.3. SVCB Query Names 9
   2.4. Interpretation 10
      2.4.1. SvcPriority 10
      2.4.2. AliasMode 10
      2.4.3. ServiceMode 11
   2.5. Special Handling of "." in TargetName 12
      2.5.1. AliasMode 12
      2.5.2. ServiceMode 12
3. Client Behavior 12
   3.1. Handling Resolution Failures 13
   3.2. Clients Using a Proxy 14
4. DNS Server Behavior 15
   4.1. Authoritative Servers 15
   4.2. Recursive Resolvers 15
      4.2.1. DNS64 15
   4.3. General Requirements 16
   4.4. EDNS Client Subnet (ECS) 16

5. Performance Optimizations 17
5.1. Optimistic Pre-connection and Connection Reuse 17
5.2. Generating and Using Incomplete Responses 17
6. SVCB-Compatible RR Types 18
7. Initial SvcParamKeys 18
7.1. "alpn" and "no-default-alpn" 19
7.1.1. Representation 19
7.1.2. Use 20
7.2. "port" 20
7.3. "ipv4hint" and "ipv6hint" 21
7.4. "mandatory" 21
8. ServiceMode RR Compatibility and Mandatory Keys 22
9. Using Service Bindings with HTTP 22
9.1. Query Names for HTTPS RRs 23
9.2. Comparison with Alt-Svc 23
9.2.1. ALPN Usage 23
9.2.2. Untrusted Channels 24
9.2.3. Cache Lifetime 24
9.2.4. Granularity 24
9.3. Interaction with Alt-Svc 24
9.4. Requiring Server Name Indication 25
9.5. HTTP Strict Transport Security (HSTS) 25
9.6. Use of HTTPS RRs in Other Protocols 26
10. Zone Structures 26
10.1. Structuring Zones for Flexibility 26
10.2. Structuring Zones for Performance 27
10.3. Operational Considerations 27
10.4. Examples 27
10.4.1. Protocol Enhancements 27
10.4.2. Apex Aliasing 28

10.4.3. Parameter Binding 28
10.4.4. Multi-CDN Configuration 29
10.4.5. Non-HTTP Uses 30
11. Interaction with Other Standards 31
12. Security Considerations 31
13. Privacy Considerations 32
14. IANA Considerations 32
14.1. SVCB RR Type 32
14.2. HTTPS RR Type 32
14.3. New Registry for Service Parameters 32
14.3.1. Procedure 33
14.3.2. Initial Contents 33
14.4. Other Registry Updates 34
15. References 34
15.1. Normative References 34
15.2. Informative References 36
Appendix A. Decoding Text in Zone Files 37
A.1. Decoding a Comma-Separated List 38
Appendix B. HTTP Mapping Summary 39
Appendix C. Comparison with Alternatives 39
C.1. Differences from the SRV RR Type 39
C.2. Differences from the Proposed HTTP Record 40
C.3. Differences from the Proposed ANAME Record 40
C.4. Comparison with Separate RR Types for AliasMode and ServiceMode 40
Appendix D. Test Vectors 41
D.1. AliasMode 41
D.2. ServiceMode 41
D.3. Failure Cases 46
Acknowledgments and Related Proposals 47
Authors' Addresses 47

1. Introduction

The SVCB ("Service Binding") and HTTPS resource records (RRs) provide clients with complete instructions for access to a service. This information enables improved performance and privacy by avoiding transient connections to a suboptimal default server, negotiating a preferred protocol, and providing relevant public keys.

For example, HTTP clients currently resolve only A and/or AAAA records for the origin hostname, learning only its IP addresses. If an HTTP client learns more about the origin before connecting, it may be able to upgrade "http" URLs to "https", enable HTTP/3 or Encrypted ClientHello [ECH], or switch to an operationally preferable endpoint. It is highly desirable to minimize the number of round trips and lookups required to learn this additional information.

The SVCB and HTTPS RRs also help when the operator of a service wishes to delegate operational control to one or more other domains, e.g., aliasing the origin "https://example.com" to a service operator endpoint at "svc.example.net". While this case can sometimes be handled by a CNAME, that does not cover all use cases. CNAME is also inadequate when the service operator needs to provide a bound collection of consistent configuration parameters through the DNS (such as network location, protocol, and keying information).

This document first describes the SVCB RR as a general-purpose RR that can be applied directly and efficiently to a wide range of services (Section 2). It also describes the rules for defining other SVCB-compatible RR types (Section 6), starting with the HTTPS RR type (Section 9), which provides improved efficiency and convenience with HTTP by avoiding the need for an Attrleaf label [Attrleaf] (Section 9.1).

The SVCB RR has two modes: 1) "AliasMode", which simply delegates operational control for a resource and 2) "ServiceMode", which binds together configuration information for a service endpoint. ServiceMode provides additional key=value parameters within each RDATA set.

1.1. Goals

The goal of the SVCB RR is to allow clients to resolve a single additional DNS RR in a way that:

- Provides alternative endpoints that are authoritative for the service, along with parameters associated with each of these endpoints.
- Does not assume that all alternative endpoints have the same parameters or capabilities, or are even operated by the same entity. This is important, as DNS does not provide any way to tie together multiple RRsets for the same name. For example, if "www.example.com" is a CNAME alias that switches between one of three Content Delivery Networks (CDNs) or hosting environments, successive queries for that name may return records that correspond to different environments.
- Enables CNAME-like functionality at a zone apex (such as "example.com") for participating protocols and generally enables extending operational authority for a service identified by a domain name to other instances with alternate names.

Additional goals specific to HTTPS RRs and the HTTP use cases include:

- Connecting directly to HTTP/3 (QUIC transport) alternative endpoints [HTTP/3].
- Supporting non-default TCP and UDP ports.
- Enabling SRV-like benefits (e.g., apex aliasing, as mentioned above) for HTTP, where SRV [SRV] has not been widely adopted.
- Providing an indication signaling that the "https" scheme should be used instead of "http" for all HTTP requests to this host and port, similar to HTTP Strict Transport Security [HSTS] (see Section 9.5).
- Enabling the conveyance of Encrypted ClientHello keys [ECH] associated with an alternative endpoint.

1.2. Overview of the SVCB RR

This subsection briefly describes the SVCB RR with forward references to the full exposition of each component. (As discussed in Section 6, this all applies equally to the HTTPS RR, which shares the same encoding, format, and high-level semantics.)

The SVCB RR has two modes: 1) AliasMode (Section 2.4.2), which aliases a name to another name and 2) ServiceMode (Section 2.4.3), which provides connection information bound to a service endpoint domain. Placing both forms in a single RR type allows clients to fetch the relevant information with a single query (Section 2.3).

The SVCB RR has two required fields and one optional field. The fields are:

SvcPriority (Section 2.4.1): The priority of this record (relative to others, with lower values preferred). A value of 0 indicates AliasMode.

TargetName: The domain name of either the alias target (for AliasMode) or the alternative endpoint (for ServiceMode).

SvcParams (optional): A list of key=value pairs describing the alternative endpoint at TargetName (only used in ServiceMode and otherwise ignored). SvcParams are described in Section 2.1.

Cooperating DNS recursive resolvers will perform subsequent record resolution (for SVCB, A, and AAAA records) and return them in the Additional section of the response (Section 4.2). Clients either use responses included in the Additional section returned by the recursive resolver or perform necessary SVCB, A, and AAAA record resolutions (Section 3). DNS authoritative servers can attach in-bailiwick SVCB, A, AAAA, and CNAME records in the Additional section to responses for a SVCB query (Section 4.1).

In ServiceMode, the SvcParams of the SVCB RR provide an extensible data model for describing alternative endpoints that are authoritative for a service, along with parameters associated with each of these alternative endpoints (Section 7).

For HTTP use cases, the HTTPS RR (Section 9) enables many of the benefits of Alt-Svc [AltSvc] without waiting for a full HTTP connection initiation (multiple round trips) before learning of the preferred alternative, and without necessarily revealing the user's intended destination to all entities along the network path.

1.3. Terminology

Terminology in this document is based on the common case where the SVCB record is used to access a resource identified by a URI whose `authority` field contains a DNS hostname as the `host`.

- The "service" is the information source identified by the `authority` and `scheme` of the URI, capable of providing access to the resource. For "https" URIs, the "service" corresponds to an "origin" [RFC6454].
- The "service name" is the `host` portion of the authority.
- The "authority endpoint" is the authority's hostname and a port number implied by the scheme or specified in the URI.
- An "alternative endpoint" is a hostname, port number, and other associated instructions to the client on how to reach an instance of a service.

Additional DNS terminology intends to be consistent with [DNSTerm].

SVCB is a contraction of "service binding". The SVCB RR, HTTPS RR, and future RR types that share SVCB's formats and registry are collectively known as SVCB-compatible RR types. The contraction "SVCB" is also used to refer to this system as a whole.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The SVCB Record Type

The SVCB DNS RR type (RR type 64) is used to locate alternative endpoints for a service.

The algorithm for resolving SVCB records and associated address records is specified in Section 3.

Other SVCB-compatible RR types can also be defined as needed (see Section 6). In particular, the HTTPS RR (RR type 65) provides special handling for the case of "https" origins as described in Section 9.

SVCB RRs are extensible by a list of SvcParams, which are pairs consisting of a SvcParamKey and a SvcParamValue. Each SvcParamKey has a presentation name and a registered number. Values are in a format specific to the SvcParamKey. Each SvcParam has a specified presentation format (used in zone files) and wire encoding (e.g., domain names, binary data, or numeric values). The initial SvcParamKeys and their formats are defined in Section 7.

2.1. Zone-File Presentation Format

The presentation format `<RDATA>` of the record ([RFC1035], Section 5.1) has the form:

```
SvcPriority TargetName SvcParams
```

The SVCB record is defined specifically within the Internet ("IN") Class ([RFC1035], Section 3.2.4).

SvcPriority is a number in the range 0-65535, TargetName is a `<domain-name>` ([RFC1035], Section 5.1), and the SvcParams are a whitespace-separated list with each SvcParam consisting of a SvcParamKey=SvcParamValue pair or a standalone SvcParamKey. SvcParamKeys are registered by IANA (Section 14.3).

Each SvcParamKey **SHALL** appear at most once in the SvcParams. In presentation format, SvcParamKeys are lowercase alphanumeric strings. Key names contain 1-63 characters from the ranges "a"-"z", "0"-"9", and "-". In ABNF [RFC5234],

```
alpha-lc      = %x61-7A   ; a-z
SvcParamKey   = 1*63(alpha-lc / DIGIT / "-")
SvcParam      = SvcParamKey ["=" SvcParamValue]
SvcParamValue = char-string ; See Appendix A.
value         = *OCTET ; Value before key-specific parsing
```

The SvcParamValue is parsed using the character-string decoding algorithm (Appendix A), producing a `value`. The `value` is then validated and converted into wire format in a manner specific to each key.

When the optional "=" and SvcParamValue are omitted, the `value` is interpreted as empty.

Arbitrary keys can be represented using the unknown-key presentation format "keyNNNNN" where NNNNN is the numeric value of the key type without leading zeros. A SvcParam in this form **SHALL** be parsed as specified above, and the decoded `value` **SHALL** be used as its wire-format encoding.

For some SvcParamKeys, the `value` corresponds to a list or set of items. Presentation formats for such keys **SHOULD** use a comma-separated list (Appendix A.1).

SvcParams in presentation format **MAY** appear in any order, but keys **MUST NOT** be repeated.

2.2. RDATA Wire Format

The RDATA for the SVCB RR consists of:

- a 2-octet field for SvcPriority as an integer in network byte order.
- the uncompressed, fully qualified TargetName, represented as a sequence of length-prefixed labels per Section 3.1 of [RFC1035].

- the SvcParams, consuming the remainder of the record (so smaller than 65535 octets and constrained by the RDATA and DNS message sizes).

When the list of SvcParams is non-empty, it contains a series of SvcParamKey=SvcParamValue pairs, represented as:

- a 2-octet field containing the SvcParamKey as an integer in network byte order. (See Section 14.3.2 for the defined values.)
- a 2-octet field containing the length of the SvcParamValue as an integer between 0 and 65535 in network byte order.
- an octet string of this length whose contents are the SvcParamValue in a format determined by the SvcParamKey.

SvcParamKeys **SHALL** appear in increasing numeric order.

Clients **MUST** consider an RR malformed if:

- the end of the RDATA occurs within a SvcParam.
- SvcParamKeys are not in strictly increasing numeric order.
- the SvcParamValue for a SvcParamKey does not have the expected format.

Note that the second condition implies that there are no duplicate SvcParamKeys.

If any RRs are malformed, the client **MUST** reject the entire RRset and fall back to non-SVCB connection establishment.

2.3. SVCB Query Names

When querying the SVCB RR, a service is translated into a QNAME by prepending the service name with a label indicating the scheme, prefixed with an underscore, resulting in a domain name like "_examplescheme.api.example.com.". This follows the Attrleaf naming pattern [Attrleaf], so the scheme **MUST** be registered appropriately with IANA (see Section 11).

Protocol mapping documents **MAY** specify additional underscore-prefixed labels to be prepended. For schemes that specify a port (Section 3.2.3 of [URI]), one reasonable possibility is to prepend the indicated port number if a non-default port number is specified. This document terms this behavior "Port Prefix Naming" and uses it in the examples throughout.

See Section 9.1 for information regarding HTTPS RR behavior.

When a prior CNAME or SVCB record has aliased to a SVCB record, each RR **SHALL** be returned under its own owner name, as in ordinary CNAME processing ([RFC1034], Section 3.6.2). For details, see the recommendations regarding aliases for clients (Section 3), servers (Section 4), and zones (Section 10).

Note that none of these forms alter the origin or authority for validation purposes. For example, TLS clients **MUST** continue to validate TLS certificates for the original service name.

As an example, the owner of "example.com" could publish this record:

```
_8443._foo.api.example.com. 7200 IN SVCB 0 svc4.example.net.
```

This record would indicate that "foo://api.example.com:8443" is aliased to "svc4.example.net". The owner of "example.net", in turn, could publish this record:

```
svc4.example.net.  7200  IN SVCB 3 svc4.example.net. (
    alpn="bar" port="8004" )
```

This record would indicate that these services are served on port number 8004, which supports the protocol "bar" and its associated transport in addition to the default transport protocol for "foo://".

(Parentheses are used to ignore a line break in DNS zone-file presentation format, per Section 5.1 of [RFC1035].)

2.4. Interpretation

2.4.1. SvcPriority

When SvcPriority is 0, the SVCB record is in AliasMode (Section 2.4.2). Otherwise, it is in ServiceMode (Section 2.4.3).

Within a SVCB RRset, all RRs **SHOULD** have the same mode. If an RRset contains a record in AliasMode, the recipient **MUST** ignore any ServiceMode records in the set.

RRsets are explicitly unordered collections, so the SvcPriority field is used to impose an ordering on SVCB RRs. A smaller SvcPriority indicates that the domain owner recommends the use of this record over ServiceMode RRs with a larger SvcPriority value.

When receiving an RRset containing multiple SVCB records with the same SvcPriority value, clients **SHOULD** apply a random shuffle within a priority level to the records before using them, to ensure uniform load balancing.

2.4.2. AliasMode

In AliasMode, the SVCB record aliases a service to a TargetName. SVCB RRsets **SHOULD** only have a single RR in AliasMode. If multiple AliasMode RRs are present, clients or recursive resolvers **SHOULD** pick one at random.

The primary purpose of AliasMode is to allow aliasing at the zone apex, where CNAME is not allowed (see, for example, [RFC1912], Section 2.4). In AliasMode, the TargetName will be the name of a domain that resolves to SVCB, AAAA, and/or A records. (See Section 6 for aliasing of SVCB-compatible RR types.) Unlike CNAME, AliasMode records do not affect the resolution of other RR types and apply only to a specific service, not an entire domain name.

The AliasMode TargetName **SHOULD NOT** be equal to the owner name, as this would result in a loop. In AliasMode, recipients **MUST** ignore any SvcParams that are present. Zone-file parsers **MAY** emit a warning if an AliasMode record has SvcParams. The use of SvcParams in AliasMode records is currently not defined, but a future specification could extend AliasMode records to include SvcParams.

For example, the operator of "foo://example.com:8080" could point requests to a service operating at "foosvc.example.net" by publishing:

```
_8080._foo.example.com. 3600 IN SVCB 0 foosvc.example.net.
```

Using AliasMode maintains a separation of concerns: the owner of "foosvc.example.net" can add or remove ServiceMode SVCB records without requiring a corresponding change to "example.com". Note that if "foosvc.example.net" promises to always publish a SVCB record, this AliasMode record can be replaced by a CNAME at the same owner name.

AliasMode is especially useful for SVCB-compatible RR types that do not require an underscore prefix, such as the HTTPS RR type. For example, the operator of "https://example.com" could point requests to a server at "svc.example.net" by publishing this record at the zone apex:

```
example.com. 3600 IN HTTPS 0 svc.example.net.
```

Note that the SVCB record's owner name **MAY** be the canonical name of a CNAME record, and the TargetName **MAY** be the owner of a CNAME record. Clients and recursive resolvers **MUST** follow CNAMEs as normal.

To avoid unbounded alias chains, clients and recursive resolvers **MUST** impose a limit on the total number of SVCB aliases they will follow for each resolution request. This limit **MUST NOT** be zero, i.e., implementations **MUST** be able to follow at least one AliasMode record. The exact value of this limit is left to implementations.

Zones that require following multiple AliasMode records could encounter compatibility and performance issues.

As legacy clients will not know to use this record, service operators will likely need to retain fallback AAAA and A records alongside this SVCB record, although in a common case the target of the SVCB record might offer better performance, and therefore would be preferable for clients implementing this specification to use.

AliasMode records only apply to queries for the specific RR type. For example, a SVCB record cannot alias to an HTTPS record or vice versa.

2.4.3. ServiceMode

In ServiceMode, the TargetName and SvcParams within each RR associate an alternative endpoint for the service with its connection parameters.

Each protocol scheme that uses SVCB **MUST** define a protocol mapping that explains how SvcParams are applied for connections of that scheme. Unless specified otherwise by the protocol mapping, clients **MUST** ignore any SvcParam that they do not recognize.

Some SvcParams impose requirements on other SvcParams in the RR. A ServiceMode RR is called "self-consistent" if its SvcParams all comply with each other's requirements. Clients **MUST** reject any RR whose recognized SvcParams are not self-consistent and **MAY** reject the entire RRset. To help zone operators avoid this condition, zone-file implementations **SHOULD** enforce self-consistency as well.

2.5. Special Handling of "." in TargetName

If TargetName has the value "." (represented in the wire format as a zero-length label), special rules apply.

2.5.1. AliasMode

For AliasMode SVCB RRs, a TargetName of "." indicates that the service is not available or does not exist. This indication is advisory: clients encountering this indication **MAY** ignore it and attempt to connect without the use of SVCB.

2.5.2. ServiceMode

For ServiceMode SVCB RRs, if TargetName has the value ".", then the owner name of this record **MUST** be used as the effective TargetName. If the record has a wildcard owner name in the zone file, the recipient **SHALL** use the response's synthesized owner name as the effective TargetName.

Here, for example, "svc2.example.net" is the effective TargetName:

```
example.com.      7200  IN HTTPS 0 svc.example.net.
svc.example.net.  7200  IN CNAME svc2.example.net.
svc2.example.net. 7200  IN HTTPS 1 . port=8002
svc2.example.net. 300   IN A     192.0.2.2
svc2.example.net. 300   IN AAAA  2001:db8::2
```

3. Client Behavior

"SVCB resolution" is the process of enumerating and ordering the available endpoints for a service, as performed by the client. SVCB resolution is implemented as follows:

1. Let $QNAME be the service name plus appropriate prefixes for the scheme (see Section 2.3).
2. Issue a SVCB query for $QNAME.
3. If an AliasMode SVCB record is returned for $QNAME (after following CNAMEs as normal), set $QNAME to its TargetName (without additional prefixes) and loop back to Step 2, subject to chain length limits and loop detection heuristics (see Section 3.1).
4. If one or more "compatible" (Section 8) ServiceMode records are returned, these represent the alternative endpoints. Sort the records by ascending SvcPriority.

5. Otherwise, SVCB resolution has failed, and the list of available endpoints is empty.

This procedure does not rely on any recursive or authoritative DNS server to comply with this specification or have any awareness of SVCB.

A client is called "SVCB-optional" if it can connect without the use of ServiceMode records; otherwise, it is called "SVCB-reliant". Clients for pre-existing protocols (e.g., HTTP) **SHALL** implement SVCB-optional behavior (except as noted in Section 3.1 or when modified by future specifications).

SVCB-optional clients **SHOULD** issue in parallel any other DNS queries that might be needed for connection establishment if the SVCB record is absent, in order to minimize delay in that case and enable the optimizations discussed in Section 5.

Once SVCB resolution has concluded, whether successful or not, if at least one AliasMode record was processed, SVCB-optional clients **SHALL** append to the list of endpoints an endpoint consisting of the final value of $QNAME, the authority endpoint's port number, and no SvcParams. (This endpoint will be attempted before falling back to non-SVCB connection modes. This ensures that SVCB-optional clients will make use of an AliasMode record whose TargetName has A and/or AAAA records but no SVCB records.)

The client proceeds with connection establishment using this list of endpoints. Clients **SHOULD** try higher-priority alternatives first, with fallback to lower-priority alternatives. Clients resolve AAAA and/or A records for the selected TargetName and **MAY** choose between them using an approach such as Happy Eyeballs [HappyEyeballsV2].

If the client is SVCB-optional and connecting using this list of endpoints has failed, the client now attempts to use non-SVCB connection modes.

Some important optimizations are discussed in Section 5 to avoid additional latency in comparison to ordinary AAAA/A lookups.

3.1. Handling Resolution Failures

If DNS responses are cryptographically protected (e.g., using DNSSEC or TLS [DoT] [DoH]) and SVCB resolution fails due to an authentication error, SERVFAIL response, transport error, or timeout, the client **SHOULD** abandon its attempt to reach the service, even if the client is SVCB-optional. Otherwise, an active attacker could mount a downgrade attack by denying the user access to the SvcParams.

A SERVFAIL error can occur if the domain is DNSSEC-signed, the recursive resolver is DNSSEC-validating, and the attacker is between the recursive resolver and the authoritative DNS server. A transport error or timeout can occur if an active attacker between the client and the recursive resolver is selectively dropping SVCB queries or responses, based on their size or other observable patterns.

If the client enforces DNSSEC validation on A/AAAA responses, it **SHOULD** apply the same validation policy to SVCB. Otherwise, an attacker could defeat the A/AAAA protection by forging SVCB responses that direct the client to other IP addresses.

If DNS responses are not cryptographically protected, clients **MAY** treat SVCB resolution failure as fatal or nonfatal.

If the client is unable to complete SVCB resolution due to its chain length limit, the client **MUST** fall back to the authority endpoint, as if the service's SVCB record did not exist.

3.2. Clients Using a Proxy

Clients using a domain-oriented transport proxy like HTTP CONNECT ([RFC7231], Section 4.3.6) or SOCKS5 [RFC1928] have the option of using named destinations, in which case the client does not perform any A or AAAA queries for destination domains. If the client is configured to use named destinations with a proxy that does not provide SVCB query capability (e.g., through an affiliated DNS resolver), the client would have to perform SVCB resolution separately, likely disclosing the destinations to additional parties and not just the proxy. Clients in this configuration **SHOULD** arrange for a separate SVCB resolution procedure with appropriate privacy properties. If this is not possible, SVCB-optional clients **MUST** disable SVCB resolution entirely, and SVCB-reliant clients **MUST** treat the configuration as invalid.

If the client does use SVCB and named destinations, the client **SHOULD** follow the standard SVCB resolution process, selecting the smallest-SvcPriority option that is compatible with the client and the proxy. When connecting using a SVCB record, clients **MUST** provide the final TargetName and port to the proxy, which will perform any required A and AAAA lookups.

This arrangement has several benefits:

- Compared to disabling SVCB:
  - It allows the client to use the SvcParams, if present, which are only usable with a specific TargetName. The SvcParams may include information that enhances performance (e.g., supported protocols) and privacy.
  - It allows a service on an apex domain to use aliasing.
- Compared to providing the proxy with an IP address:
  - It allows the proxy to select between IPv4 and IPv6 addresses for the server according to its configuration.
  - It ensures that the proxy receives addresses based on its network geolocation, not the client's.
  - It enables faster fallback for TCP destinations with multiple addresses of the same family.

4. DNS Server Behavior

4.1. Authoritative Servers

When replying to a SVCB query, authoritative DNS servers **SHOULD** return A, AAAA, and SVCB records in the Additional section for any TargetNames that are in the zone. If the zone is signed, the server **SHOULD** also include DNSSEC records authenticating the existence or nonexistence of these records in the Additional section.

See Section 4.4 for exceptions.

4.2. Recursive Resolvers

Whether the recursive resolver is aware of SVCB or not, the normal response construction process used for unknown RR types [RFC3597] generates the Answer section of the response. Recursive resolvers that are aware of SVCB **SHOULD** help the client to execute the procedure in Section 3 with minimum overall latency by incorporating additional useful information into the Additional section of the response as follows:

1. Incorporate the results of SVCB resolution. If the recursive resolver's local chain length limit (which may be different from the client's limit) has been reached, terminate.
2. If any of the resolved SVCB records are in AliasMode, choose one of them at random, and resolve SVCB, A, and AAAA records for its TargetName.
   - If any SVCB records are resolved, go to Step 1.
   - Otherwise, incorporate the results of A and AAAA resolution, and terminate.
3. All the resolved SVCB records are in ServiceMode. Resolve A and AAAA queries for each TargetName (or for the owner name if TargetName is "."), incorporate all the results, and terminate.

In this procedure, "resolve" means the resolver's ordinary recursive resolution procedure, as if processing a query for that RRset. This includes following any aliases that the resolver would ordinarily follow (e.g., CNAME, DNAME [DNAME]). Errors or anomalies in obtaining additional records **MAY** cause this process to terminate but **MUST NOT** themselves cause the resolver to send a failure response.

See Section 2.4.2 for additional safeguards for recursive resolvers to implement to mitigate loops.

See Section 5.2 for possible optimizations of this procedure.

4.2.1. DNS64

DNS64 resolvers synthesize responses to AAAA queries for names that only have an A record (Section 5.1.7 of [RFC6147]). SVCB-aware DNS64 resolvers **SHOULD** apply the same synthesis logic when resolving AAAA records for the TargetName for inclusion in the Additional section (Step 2 in Section 4.2) and **MAY** omit the A records from this section.

DNS64 resolvers **MUST NOT** extrapolate the AAAA synthesis logic to the IP hints in the SvcParams (Section 7.3). Modifying the IP hints would break DNSSEC validation for the SVCB record and would not improve performance when the above recommendation is implemented.

4.3. General Requirements

Recursive resolvers **MUST** be able to convey SVCB records with unrecognized SvcParamKeys. Resolvers **MAY** accomplish this by treating the entire SvcParams portion of the record as opaque, even if the contents are invalid. If a recognized SvcParamKey is followed by a value that is invalid according to the SvcParam's specification, a recursive resolver **MAY** report an error such as SERVFAIL instead of returning the record. For complex value types whose interpretation might differ between implementations or have additional future allowed values added (e.g., URIs or "alpn"), resolvers **SHOULD** limit validation to specified constraints.

When responding to a query that includes the DNSSEC OK bit [RFC3225], DNSSEC-capable recursive and authoritative DNS servers **MUST** accompany each RRset in the Additional section with the same DNSSEC-related records that they would send when providing that RRset as an Answer (e.g., RRSIG, NSEC, NSEC3).

According to Section 5.4.1 of [RFC2181], "Unauthenticated RRs received and cached from ... the additional data section ... should not be cached in such a way that they would ever be returned as answers to a received query. They may be returned as additional information where appropriate." Recursive resolvers therefore **MAY** cache records from the Additional section for use in populating Additional section responses and **MAY** cache them for general use if they are authenticated by DNSSEC.

4.4. EDNS Client Subnet (ECS)

The EDNS Client Subnet (ECS) option [RFC7871] allows recursive resolvers to request IP addresses that are suitable for a particular client IP range. SVCB records may contain IP addresses (in ipv*hint SvcParams) or direct users to a subnet-specific TargetName, so recursive resolvers **SHOULD** include the same ECS option in SVCB queries as in A/AAAA queries.

According to Section 7.3.1 of [RFC7871], "Any records from [the Additional section] **MUST NOT** be tied to a network." Accordingly, when processing a response whose QTYPE is SVCB-compatible, resolvers **SHOULD** treat any records in the Additional section as having SOURCE PREFIX-LENGTH set to zero and SCOPE PREFIX-LENGTH as specified in the ECS option. Authoritative servers **MUST** omit such records if they are not suitable for use by any stub resolvers that set SOURCE PREFIX-LENGTH to zero. This will cause the resolver to perform a follow-up query that can receive a properly tailored ECS. (This is similar to the usage of CNAME with the ECS option as discussed in [RFC7871], Section 7.2.1.)

Authoritative servers that omit Additional records can avoid the added latency of a follow-up query by following the advice in Section 10.2.

5. Performance Optimizations

For optimal performance (i.e., minimum connection setup time), clients **SHOULD** implement a client-side DNS cache. Responses in the Additional section of a SVCB response **SHOULD** be placed in cache before performing any follow-up queries. With this behavior, and with conforming DNS servers, using SVCB does not add network latency to connection setup.

To improve performance when using a non-conforming recursive resolver, clients **SHOULD** issue speculative A and/or AAAA queries in parallel with each SVCB query, based on a predicted value of TargetName (see Section 10.2).

After a ServiceMode RRset is received, clients **MAY** try more than one option in parallel and **MAY** prefetch A and AAAA records for multiple TargetNames.

5.1. Optimistic Pre-connection and Connection Reuse

If an address response arrives before the corresponding SVCB response, the client **MAY** initiate a connection as if the SVCB query returned NODATA but **MUST NOT** transmit any information that could be altered by the SVCB response until it arrives. For example, future SvcParamKeys could be defined that alter the TLS ClientHello.

Clients implementing this optimization **SHOULD** wait for 50 milliseconds before starting optimistic pre-connection, as per the guidance in [HappyEyeballsV2].

A SVCB record is consistent with a connection if the client would attempt an equivalent connection when making use of that record. If a SVCB record is consistent with an active or in-progress connection C, the client **MAY** prefer that record and use C as its connection. For example, suppose the client receives this SVCB RRset for a protocol that uses TLS over TCP:

```
_1234._bar.example.com. 300 IN SVCB 1 svc1.example.net. (
    ipv6hint=2001:db8::1 port=1234 )
                               SVCB 2 svc2.example.net. (
    ipv6hint=2001:db8::2 port=1234 )
```

If the client has an in-progress TCP connection to `[2001:db8::2]:1234`, it **MAY** proceed with TLS on that connection, even though the other record in the RRset has higher priority.

If none of the SVCB records are consistent with any active or in-progress connection, clients proceed with connection establishment as described in Section 3.

5.2. Generating and Using Incomplete Responses

When following the procedure in Section 4.2, recursive resolvers **MAY** terminate the procedure early and produce a reply that omits some of the associated RRsets. This is **REQUIRED** when the chain length limit is reached (Step 1 in Section 4.2) but might also be appropriate when the

maximum response size is reached or when responding before fully chasing dependencies would improve performance. When omitting certain RRsets, recursive resolvers **SHOULD** prioritize information for smaller-SvcPriority records.

As discussed in Section 3, clients **MUST** be able to fetch additional information that is required to use a SVCB record, if it is not included in the initial response. As a performance optimization, if some of the SVCB records in the response can be used without requiring additional DNS queries, the client **MAY** prefer those records, regardless of their priorities.

6. SVCB-Compatible RR Types

An RR type is called "SVCB-compatible" if it permits an implementation that is identical to SVCB in its:

- RDATA presentation format
- RDATA wire format
- IANA registry used for SvcParamKeys
- Authoritative server Additional section processing
- Recursive resolution process
- Relevant Class (i.e., Internet ("IN") [RFC1035])

This allows authoritative and recursive DNS servers to apply identical processing to all SVCB-compatible RR types.

All other behaviors described as applying to the SVCB RR also apply to all SVCB-compatible RR types unless explicitly stated otherwise. When following an AliasMode record (Section 2.4.2) of RR type $T, the follow-up query to the TargetName **MUST** also be for type $T.

This document defines one SVCB-compatible RR type (other than SVCB itself): the HTTPS RR type (Section 9), which avoids Attrleaf label prefixes [Attrleaf] in order to improve compatibility with wildcards and CNAMEs, which are widely used with HTTP.

Standards authors should consider carefully whether to use SVCB or define a new SVCB-compatible RR type, as this choice cannot easily be reversed after deployment.

7. Initial SvcParamKeys

A few initial SvcParamKeys are defined here. These keys are useful for the "https" scheme, and most are expected to be generally applicable to other schemes as well.

Each new protocol mapping document **MUST** specify which keys are applicable and safe to use. Protocol mappings **MAY** alter the interpretation of SvcParamKeys but **MUST NOT** alter their presentation or wire formats.

7.1. "alpn" and "no-default-alpn"

The "alpn" and "no-default-alpn" SvcParamKeys together indicate the set of Application-Layer Protocol Negotiation (ALPN) protocol identifiers [ALPN] and associated transport protocols supported by this service endpoint (the "SVCB ALPN set").

As with Alt-Svc [AltSvc], each ALPN protocol identifier is used to identify the application protocol and associated suite of protocols supported by the endpoint (the "protocol suite"). The presence of an ALPN protocol identifier in the SVCB ALPN set indicates that this service endpoint, described by TargetName and the other parameters (e.g., "port"), offers service with the protocol suite associated with this ALPN identifier.

Clients filter the set of ALPN identifiers to match the protocol suites they support, and this informs the underlying transport protocol used (such as QUIC over UDP or TLS over TCP). ALPN protocol identifiers that do not uniquely identify a protocol suite (e.g., an Identification Sequence that can be used with both TLS and DTLS) are not compatible with this SvcParamKey and **MUST NOT** be included in the SVCB ALPN set.

7.1.1. Representation

ALPNs are identified by their registered "Identification Sequence" (`alpn-id`), which is a sequence of 1-255 octets.

```
alpn-id = 1*255OCTET
```

For "alpn", the presentation `value` **SHALL** be a comma-separated list (Appendix A.1) of one or more `alpn-id`s. Zone-file implementations **MAY** disallow the "," and "\" characters in ALPN IDs instead of implementing the `value-list` escaping procedure, relying on the opaque key format (e.g., `key1=\002h2`) in the event that these characters are needed.

The wire-format value for "alpn" consists of at least one `alpn-id` prefixed by its length as a single octet, and these length-value pairs are concatenated to form the SvcParamValue. These pairs **MUST** exactly fill the SvcParamValue; otherwise, the SvcParamValue is malformed.

For "no-default-alpn", the presentation and wire-format values **MUST** be empty. When "no-default-alpn" is specified in an RR, "alpn" must also be specified in order for the RR to be "self-consistent" (Section 2.4.3).

Each scheme that uses this SvcParamKey defines a "default set" of ALPN IDs that are supported by nearly all clients and servers; this set **MAY** be empty. To determine the SVCB ALPN set, the client starts with the list of `alpn-id`s from the "alpn" SvcParamKey, and it adds the default set unless the "no-default-alpn" SvcParamKey is present.

7.1.2. Use

To establish a connection to the endpoint, clients **MUST**

1. Let SVCB-ALPN-Intersection be the set of protocols in the SVCB ALPN set that the client supports.
2. Let Intersection-Transports be the set of transports (e.g., TLS, DTLS, QUIC) implied by the protocols in SVCB-ALPN-Intersection.
3. For each transport in Intersection-Transports, construct a ProtocolNameList containing the Identification Sequences of all the client's supported ALPN protocols for that transport, without regard to the SVCB ALPN set.

For example, if the SVCB ALPN set is ["http/1.1", "h3"] and the client supports HTTP/1.1, HTTP/2, and HTTP/3, the client could attempt to connect using TLS over TCP with a ProtocolNameList of ["http/1.1", "h2"] and could also attempt a connection using QUIC with a ProtocolNameList of ["h3"].

Once the client has constructed a ClientHello, protocol negotiation in that handshake proceeds as specified in [ALPN], without regard to the SVCB ALPN set.

Clients **MAY** implement a fallback procedure, using a less-preferred transport if more-preferred transports fail to connect. This fallback behavior is vulnerable to manipulation by a network attacker who blocks the more-preferred transports, but it may be necessary for compatibility with existing networks.

With this procedure in place, an attacker who can modify DNS and network traffic can prevent a successful transport connection but cannot otherwise interfere with ALPN protocol selection. This procedure also ensures that each ProtocolNameList includes at least one protocol from the SVCB ALPN set.

Clients **SHOULD NOT** attempt connection to a service endpoint whose SVCB ALPN set does not contain any supported protocols.

To ensure consistency of behavior, clients **MAY** reject the entire SVCB RRset and fall back to basic connection establishment if all of the compatible RRs indicate "no-default-alpn", even if connection could have succeeded using a non-default ALPN protocol.

Zone operators **SHOULD** ensure that at least one RR in each RRset supports the default transports. This enables compatibility with the greatest number of clients.

7.2. "port"

The "port" SvcParamKey defines the TCP or UDP port that should be used to reach this alternative endpoint. If this key is not present, clients **SHALL** use the authority endpoint's port number.

The presentation `value` of the SvcParamValue is a single decimal integer between 0 and 65535 in ASCII. Any other `value` (e.g., an empty value) is a syntax error. To enable simpler parsing, this SvcParamValue **MUST NOT** contain escape sequences.

The wire format of the SvcParamValue is the corresponding 2-octet numeric value in network byte order.

If a port-restricting firewall is in place between some client and the service endpoint, changing the port number might cause that client to lose access to the service, so operators should exercise caution when using this SvcParamKey to specify a non-default port.

7.3. "ipv4hint" and "ipv6hint"

The "ipv4hint" and "ipv6hint" keys convey IP addresses that clients **MAY** use to reach the service. If A and AAAA records for TargetName are locally available, the client **SHOULD** ignore these hints. Otherwise, clients **SHOULD** perform A and/or AAAA queries for TargetName per Section 3, and clients **SHOULD** use the IP address in those responses for future connections. Clients **MAY** opt to terminate any connections using the addresses in hints and instead switch to the addresses in response to the TargetName query. Failure to use A and/or AAAA response addresses could negatively impact load balancing or other geo-aware features and thereby degrade client performance.

The presentation `value` **SHALL** be a comma-separated list (Appendix A.1) of one or more IP addresses of the appropriate family in standard textual format [RFC5952] [RFC4001]. To enable simpler parsing, this SvcParamValue **MUST NOT** contain escape sequences.

The wire format for each parameter is a sequence of IP addresses in network byte order (for the respective address family). Like an A or AAAA RRset, the list of addresses represents an unordered collection, and clients **SHOULD** pick addresses to use in a random order. An empty list of addresses is invalid.

When selecting between IPv4 and IPv6 addresses to use, clients may use an approach such as Happy Eyeballs [HappyEyeballsV2]. When only "ipv4hint" is present, NAT64 clients may synthesize IPv6 addresses as specified in [RFC7050] or ignore the "ipv4hint" key and wait for AAAA resolution (Section 3). For best performance, server operators **SHOULD** include an "ipv6hint" parameter whenever they include an "ipv4hint" parameter.

These parameters are intended to minimize additional connection latency when a recursive resolver is not compliant with the requirements in Section 4 and **SHOULD NOT** be included if most clients are using compliant recursive resolvers. When TargetName is the service name or the owner name (which can be written as "."), server operators **SHOULD NOT** include these hints, because they are unlikely to convey any performance benefit.

7.4. "mandatory"

See Section 8.

8. ServiceMode RR Compatibility and Mandatory Keys

In a ServiceMode RR, a SvcParamKey is considered "mandatory" if the RR will not function correctly for clients that ignore this SvcParamKey. Each SVCB protocol mapping **SHOULD** specify a set of keys that are "automatically mandatory", i.e., mandatory if they are present in an RR. The SvcParamKey "mandatory" is used to indicate any mandatory keys for this RR, in addition to any automatically mandatory keys that are present.

A ServiceMode RR is considered "compatible" by a client if the client recognizes all the mandatory keys and their values indicate that successful connection establishment is possible. Incompatible RRs are ignored (see step 5 of the procedure defined in Section 3).

The presentation `value` **SHALL** be a comma-separated list (Appendix A.1) of one or more valid SvcParamKeys, either by their registered name or in the unknown-key format (Section 2.1). Keys **MAY** appear in any order but **MUST NOT** appear more than once. For self-consistency (Section 2.4.3), listed keys **MUST** also appear in the SvcParams.

To enable simpler parsing, this SvcParamValue **MUST NOT** contain escape sequences.

For example, the following is a valid list of SvcParams:

```
ipv6hint=... key65333=ex1 key65444=ex2 mandatory=key65444,ipv6hint
```

In wire format, the keys are represented by their numeric values in network byte order, concatenated in strictly increasing numeric order.

This SvcParamKey is always automatically mandatory and **MUST NOT** appear in its own value-list. Other automatically mandatory keys **SHOULD NOT** appear in the list either. (Including them wastes space and otherwise has no effect.)

9. Using Service Bindings with HTTP

The use of any protocol with SVCB requires a protocol-specific mapping specification. This section specifies the mapping for the "http" and "https" URI schemes [HTTP].

To enable special handling for HTTP use cases, the HTTPS RR type is defined as a SVCB-compatible RR type, specific to the "https" and "http" schemes. Clients **MUST NOT** perform SVCB queries or accept SVCB responses for "https" or "http" schemes.

The presentation format of the record is:

```
Name TTL IN HTTPS SvcPriority TargetName SvcParams
```

All the SvcParamKeys defined in Section 7 are permitted for use in HTTPS RRs. The default set of ALPN IDs is the single value "http/1.1". The "automatically mandatory" keys (Section 8) are "port" and "no-default-alpn". (As described in Section 8, clients must either implement these keys or ignore any RR in which they appear.) Clients that restrict the destination port in "https" URIs (e.g., using the "bad ports" list from [FETCH]) **SHOULD** apply the same restriction to the "port" SvcParam.

The presence of an HTTPS RR for an origin also indicates that clients should connect securely and use the "https" scheme, as discussed in Section 9.5. This allows HTTPS RRs to apply to pre-existing "http" scheme URLs, while ensuring that the client uses a secure and authenticated connection.

The HTTPS RR parallels the concepts introduced in "HTTP Alternative Services" [AltSvc]. Clients and servers that implement HTTPS RRs are not required to implement Alt-Svc.

9.1. Query Names for HTTPS RRs

The HTTPS RR uses Port Prefix Naming (Section 2.3), with one modification: if the scheme is "https" and the port is 443, then the client's original QNAME is equal to the service name (i.e., the origin's hostname), without any prefix labels.

By removing the Attrleaf labels [Attrleaf] used in SVCB, this construction enables offline DNSSEC signing of wildcard domains, which are commonly used with HTTP. Using the service name as the owner name of the HTTPS record, without prefixes, also allows the targets of existing CNAME chains (e.g., CDN hosts) to start returning HTTPS RR responses without requiring origin domains to configure and maintain an additional delegation.

The procedure for following HTTPS AliasMode RRs and CNAME aliases is unchanged from SVCB (as described in Sections 2.4.2 and 3).

Clients always convert "http" URLs to "https" before performing an HTTPS RR query using the process described in Section 9.5, so domain owners **MUST NOT** publish HTTPS RRs with a prefix of "_http".

Note that none of these forms alter the HTTPS origin or authority. For example, clients **MUST** continue to validate TLS certificate hostnames based on the origin.

9.2. Comparison with Alt-Svc

Publishing a ServiceMode HTTPS RR in DNS is intended to be similar to transmitting an Alt-Svc field value over HTTP, and receiving an HTTPS RR is intended to be similar to receiving that field value over HTTP. However, there are some differences in the intended client and server behavior.

9.2.1. ALPN Usage

Unlike Alt-Svc field values, HTTPS RRs can contain multiple ALPN IDs. The meaning and use of these IDs are discussed in Section 7.1.2.

9.2.2. Untrusted Channels

HTTPS records do not require or provide any assurance of authenticity. (DNSSEC signing and verification, which would provide such assurance, are **OPTIONAL**.) The DNS resolution process is modeled as an untrusted channel that might be controlled by an attacker, so Alt-Svc parameters that cannot be safely received in this model **MUST NOT** have a corresponding defined SvcParamKey. For example, there is no SvcParamKey corresponding to the Alt-Svc "persist" parameter, because this parameter is not safe to accept over an untrusted channel.

9.2.3. Cache Lifetime

There is no SvcParamKey corresponding to the Alt-Svc "ma" (max age) parameter. Instead, server operators encode the expiration time in the DNS TTL.

The appropriate TTL value might be different from the "ma" value used for Alt-Svc, depending on the desired efficiency and agility. Some DNS caches incorrectly extend the lifetime of DNS records beyond the stated TTL, so server operators cannot rely on HTTPS RRs expiring on time. Shortening the TTL to compensate for incorrect caching is **NOT RECOMMENDED**, as this practice impairs the performance of correctly functioning caches and does not guarantee faster expiration from incorrect caches. Instead, server operators **SHOULD** maintain compatibility with expired records until they observe that nearly all connections have migrated to the new configuration.

9.2.4. Granularity

Sending Alt-Svc over HTTP allows the server to tailor the Alt-Svc field value specifically to the client. When using an HTTPS RR, groups of clients will necessarily receive the same SvcParams. Therefore, HTTPS RRs are not suitable for uses that require single-client granularity.

9.3. Interaction with Alt-Svc

Clients that implement support for both Alt-Svc and HTTPS records and are making a connection based on a cached Alt-Svc response **SHOULD** retrieve any HTTPS records for the Alt-Svc alt-authority and ensure that their connection attempts are consistent with both the Alt-Svc parameters and any received HTTPS SvcParams. If present, the HTTPS record's TargetName and port are used for connection establishment (per Section 3). For example, suppose that "https://example.com" sends an Alt-Svc field value of:

```
Alt-Svc: h2="alt.example:443", h2="alt2.example:443", h3=":8443"
```

The client would retrieve the following HTTPS records:

```
alt.example.               IN HTTPS 1 . alpn=h2,h3 foo=...
alt2.example.              IN HTTPS 1 alt2b.example. alpn=h3 foo=...
_8443._https.example.com. IN HTTPS 1 alt3.example. (
    port=9443 alpn=h2,h3 foo=... )
```

Based on these inputs, the following connection attempts would always be allowed:

- HTTP/2 to `alt.example:443`
- HTTP/3 to `alt3.example:9443`
- Fallback to the client's non-Alt-Svc connection behavior

The following connection attempts would not be allowed:

- HTTP/3 to `alt.example:443` (not consistent with Alt-Svc)
- Any connection to `alt2b.example` (no ALPN ID consistent with both the HTTPS record and Alt-Svc)
- HTTPS over TCP to any port on `alt3.example` (not consistent with Alt-Svc)

Suppose that "foo" is a SvcParamKey that renders the client SVCB-reliant. The following Alt-Svc-only connection attempts would be allowed only if the client does not support "foo", as they rely on SVCB-optional fallback behavior:

- HTTP/2 to `alt2.example:443`
- HTTP/3 to `example.com:8443`

Alt-authorities **SHOULD** carry the same SvcParams as the origin unless a deviation is specifically known to be safe. As noted in Section 2.4 of [AltSvc], clients **MAY** disallow any Alt-Svc connection according to their own criteria, e.g., disallowing Alt-Svc connections that lack support for privacy features that are available on the authority endpoint.

9.4. Requiring Server Name Indication

Clients **MUST NOT** use an HTTPS RR response unless the client supports the TLS Server Name Indication (SNI) extension and indicates the origin name in the TLS ClientHello (which might be encrypted via a future specification such as [ECH]). This supports the conservation of IP addresses.

Note that the TLS SNI (and also the HTTP "Host" or ":authority") will indicate the origin, not the TargetName.

9.5. HTTP Strict Transport Security (HSTS)

An HTTPS RR directs the client to communicate with this host only over a secure transport, similar to HSTS [HSTS]. Prior to making an "http" scheme request, the client **SHOULD** perform a lookup to determine if any HTTPS RRs exist for that origin. To do so, the client **SHOULD** construct a corresponding "https" URL as follows:

1. Replace the "http" scheme with "https".
2. If the "http" URL explicitly specifies port 80, specify port 443.
3. Do not alter any other aspect of the URL.

This construction is equivalent to Section 8.3 of [HSTS], Step 5.

If an HTTPS RR query for this "https" URL returns any AliasMode HTTPS RRs or any compatible ServiceMode HTTPS RRs (see Section 8), the client **SHOULD** behave as if it has received an HTTP 307 (Temporary Redirect) status code with this "https" URL in the "Location" field. (Receipt of an incompatible ServiceMode RR does not trigger the redirect behavior.) Because HTTPS RRs are received over an often-insecure channel (DNS), clients **MUST NOT** place any more trust in this signal than if they had received a 307 (Temporary Redirect) response over cleartext HTTP.

Publishing an HTTPS RR can potentially lead to unexpected results or a loss in functionality in cases where the "http" resource neither redirects to the "https" resource nor references the same underlying resource.

When an "https" connection fails due to an error in the underlying secure transport, such as an error in certificate validation, some clients currently offer a "user recourse" that allows the user to bypass the security error and connect anyway. When making an "https" scheme request to an origin with an HTTPS RR, either directly or via the above redirect, such a client **MAY** remove the user recourse option. Origins that publish HTTPS RRs therefore **MUST NOT** rely on user recourse for access. For more information, see Sections 8.4 and 12.1 of [HSTS].

9.6. Use of HTTPS RRs in Other Protocols

All HTTP connections to named origins are eligible to use HTTPS RRs, even when HTTP is used as part of another protocol or without an explicit HTTP-related URI scheme (Section 4.2 of [HTTP]). For example, clients that support HTTPS RRs and implement [WebSocket] using the altered opening handshake from [FETCH-WEBSOCKETS] **SHOULD** use HTTPS RRs for the `requestURL`.

When HTTP is used in a context where URLs or redirects are not applicable (e.g., connections to an HTTP proxy), clients that find a corresponding HTTPS RR **SHOULD** implement security upgrade behavior equivalent to that specified in Section 9.5.

Such protocols **MAY** define their own SVCB mappings, which **MAY** be defined to take precedence over HTTPS RRs.

10. Zone Structures

10.1. Structuring Zones for Flexibility

Each ServiceMode RRset can only serve a single scheme. The scheme is indicated by the owner name and the RR type. For the generic SVCB RR type, this means that each owner name can only be used for a single scheme. The underscore prefixing requirement (Section 2.3) ensures that this is true for the initial query, but it is the responsibility of zone owners to choose names that satisfy this constraint when using aliases, including CNAME and AliasMode records.

When using the generic SVCB RR type with aliasing, zone owners **SHOULD** choose alias target names that indicate the scheme in use (e.g., "foosvc.example.net" for "foo" schemes). This will help to avoid confusion when another scheme needs to be added to the configuration. When multiple port numbers are in use, it may be helpful to repeat the prefix labels in the alias target name (e.g., "_1234._foo.svc.example.net").

10.2. Structuring Zones for Performance

To avoid a delay for clients using a non-conforming recursive resolver, domain owners **SHOULD** minimize the use of AliasMode records and **SHOULD** choose TargetName according to a predictable convention that is known to the client, so that clients can issue A and/or AAAA queries for TargetName in advance (see Section 5). Unless otherwise specified, the convention is to set TargetName to the service name for an initial ServiceMode record, or to "." if it is reached via an alias.

```
$ORIGIN example.com. ; Origin
foo                  3600 IN CNAME foosvc.example.net.
_8080._foo.foo       3600 IN CNAME foosvc.example.net.
bar                   300 IN AAAA 2001:db8::2
_9090._bar.bar       3600 IN SVCB 1 bar key65444=...

$ORIGIN example.net. ; Service provider zone
foosvc               3600 IN SVCB 1 . key65333=...
foosvc                300 IN AAAA 2001:db8::1
```

Figure 1: "foo://foo.example.com:8080" Is Available at "foosvc.example.net", but "bar://bar.example.com:9090" Is Served Locally

Domain owners **SHOULD** avoid using a TargetName that is below a DNAME, as this is likely unnecessary and makes responses slower and larger. Also, zone structures that require following more than eight aliases (counting both AliasMode and CNAME records) are **NOT RECOMMENDED**.

10.3. Operational Considerations

Some authoritative DNS servers may not allow A or AAAA records on names starting with an underscore (e.g., [BIND-CHECK-NAMES]). This could create an operational issue when the TargetName contains an Attrleaf label, or when using a TargetName of "." if the owner name contains an Attrleaf label.

10.4. Examples

10.4.1. Protocol Enhancements

Consider a simple zone of the form:

```
$ORIGIN simple.example. ; Simple example zone
@ 300 IN A    192.0.2.1
         AAAA 2001:db8::1
```

The domain owner could add this record:

```
@ 7200 IN HTTPS 1 . alpn=h3
```

This record would indicate that "https://simple.example" supports QUIC in addition to HTTP/1.1 over TLS over TCP (the implicit default). The record could also include other information (e.g., a non-standard port). For "https://simple.example:8443", the record would be:

```
_8443._https 7200 IN HTTPS 1 . alpn=h3
```

These records also respectively tell clients to replace the scheme with "https" when loading "http://simple.example" or "http://simple.example:8443".

10.4.2. Apex Aliasing

Consider a zone that is using CNAME aliasing:

```
$ORIGIN aliased.example. ; A zone that is using a hosting service
; Subdomain aliased to a high-performance server pool
www             7200 IN CNAME pool.svc.example.
; Apex domain on fixed IPs because CNAME is not allowed at the apex
@                300 IN A     192.0.2.1
                     IN AAAA  2001:db8::1
```

With HTTPS RRs, the owner of aliased.example could alias the apex by adding one additional record:

```
@               7200 IN HTTPS 0 pool.svc.example.
```

With this record in place, HTTPS-RR-aware clients will use the same server pool for aliased.example and www.aliased.example. (They will also upgrade "http://aliased.example/..." to "https".) Non-HTTPS-RR-aware clients will just ignore the new record.

Similar to CNAME, HTTPS RRs have no impact on the origin name. When connecting, clients will continue to treat the authoritative origins as "https://www.aliased.example" and "https://aliased.example", respectively, and will validate TLS server certificates accordingly.

10.4.3. Parameter Binding

Suppose that svc.example's primary server pool supports HTTP/3 but its backup server pool does not. This can be expressed in the following form:

```
$ORIGIN svc.example. ; A hosting provider
pool  7200 IN HTTPS 1 . alpn=h2,h3
              HTTPS 2 backup alpn=h2 port=8443
pool   300 IN A        192.0.2.2
              AAAA     2001:db8::2
backup 300 IN A        192.0.2.3
              AAAA     2001:db8::3
```

This configuration is entirely compatible with the "apex aliasing" example, whether the client supports HTTPS RRs or not. If the client does support HTTPS RRs, all connections will be upgraded to HTTPS, and clients will use HTTP/3 if they can. Parameters are "bound" to each server pool, so each server pool can have its own protocol, port number, etc.

10.4.4. Multi-CDN Configuration

The HTTPS RR is intended to support HTTPS services operated by multiple independent entities, such as different CDNs or different hosting providers. This includes the case where a service is migrated from one operator to another, as well as the case where the service is multiplexed between multiple operators for performance, redundancy, etc.

This example shows such a configuration, with www.customer.example having different DNS responses to different queries, either over time or due to logic within the authoritative DNS server:

```
; This zone contains/returns different CNAME records
; at different points in time.  The RRset for "www" can
; only ever contain a single CNAME.

; Sometimes the zone has:
$ORIGIN customer.example.  ; A multi-CDN customer domain
www 900 IN CNAME cdn1.svc1.example.

; and other times it contains:
$ORIGIN customer.example.
www 900 IN CNAME customer.svc2.example.

; and yet other times it contains:
$ORIGIN customer.example.
www 900 IN CNAME cdn3.svc3.example.

; With the following remaining constant and always included:
$ORIGIN customer.example.  ; A multi-CDN customer domain
; The apex is also aliased to www to match its configuration.
@     7200 IN HTTPS 0 www
; Non-HTTPS-aware clients use non-CDN IPs.
              A    203.0.113.82
              AAAA 2001:db8:203::2

; Resolutions following the cdn1.svc1.example
; path use these records.
; This CDN uses a different alternative service for HTTP/3.
$ORIGIN svc1.example.  ; domain for CDN 1
cdn1     1800 IN HTTPS 1 h3pool alpn=h3
                 HTTPS 2 . alpn=h2
                 A    192.0.2.2
                 AAAA 2001:db8:192::4
h3pool 300 IN A 192.0.2.3
           AAAA 2001:db8:192:7::3

; Resolutions following the customer.svc2.example
; path use these records.
; Note that this CDN only supports HTTP/2.
$ORIGIN svc2.example. ; domain operated by CDN 2
```

```
customer 300 IN HTTPS 1 . alpn=h2
          60 IN A    198.51.100.2
                A    198.51.100.3
                A    198.51.100.4
                AAAA 2001:db8:198::7
                AAAA 2001:db8:198::12

; Resolutions following the cdn3.svc3.example
; path use these records.
; Note that this CDN has no HTTPS records.
$ORIGIN svc3.example. ; domain operated by CDN 3
cdn3      60 IN A    203.0.113.8
                AAAA 2001:db8:113::8
```

Note that in the above example, the different CDNs have different configurations and different capabilities, but clients will use HTTPS RRs as a bound-together unit.

Domain owners should be cautious when using a multi-CDN configuration, as it introduces a number of complexities highlighted by this example:

- If CDN 1 supports a desired protocol or feature and CDN 2 does not, the client is vulnerable to downgrade by a network adversary who forces clients to get CDN 2 records.
- Aliasing the apex to its subdomain simplifies the zone file but likely increases resolution latency, especially when using a non-HTTPS-aware recursive resolver. An alternative would be to alias the zone apex directly to a name managed by a CDN.
- The A, AAAA, and HTTPS resolutions are independent lookups, so resolvers may observe and follow different CNAMEs to different CDNs. Clients may thus find that the A and AAAA responses do not correspond to the TargetName in the HTTPS response; these clients will need to perform additional queries to retrieve the correct IP addresses. Including ipv6hint and ipv4hint will reduce the performance impact of this case.
- If not all CDNs publish HTTPS records, clients will sometimes receive NODATA for HTTPS queries (as with cdn3.svc3.example above) but could receive A/AAAA records from a different CDN. Clients will attempt to connect to this CDN without the benefit of its HTTPS records.

10.4.5. Non-HTTP Uses

For protocols other than HTTP, the SVCB RR and an Attrleaf label [Attrleaf] will be used. For example, to reach an example resource of "baz://api.example.com:8765", the following SVCB record would be used to alias it to "svc4-baz.example.net.", which in turn could return AAAA/A records and/or SVCB records in ServiceMode:

```
_8765._baz.api.example.com. 7200 IN SVCB 0 svc4-baz.example.net.
```

HTTPS RRs use similar Attrleaf labels if the origin contains a non-default port.

11. Interaction with Other Standards

This standard is intended to reduce connection latency and improve user privacy. Server operators implementing this standard **SHOULD** also implement TLS 1.3 [RFC8446] and Online Certificate Status Protocol (OCSP) Stapling (i.e., Certificate Status Request in Section 8 of [RFC6066]), both of which confer substantial performance and privacy benefits when used in combination with SVCB records.

To realize the greatest privacy benefits, this proposal is intended for use over a privacy-preserving DNS transport (like DNS over TLS [DoT] or DNS over HTTPS [DoH]). However, performance improvements, and some modest privacy improvements, are possible without the use of those standards.

Any specification for the use of SVCB with a protocol **MUST** have an entry for its scheme under the SVCB RR type in the IANA DNS "Underscored and Globally Scoped DNS Node Names" registry [Attrleaf]. The scheme **MUST** have an entry in the "Uniform Resource Identifier (URI) Schemes" registry [RFC7595] and **MUST** have a defined specification for use with SVCB.

12. Security Considerations

SVCB/HTTPS RRs permit distribution over untrusted channels, and clients are **REQUIRED** to verify that the alternative endpoint is authoritative for the service (similar to Section 2.1 of [AltSvc]). Therefore, DNSSEC signing and validation are **OPTIONAL** for publishing and using SVCB and HTTPS RRs.

Clients **MUST** ensure that their DNS cache is partitioned for each local network, or flushed on network changes, to prevent a local adversary in one network from implanting a forged DNS record that allows them to track users or hinder their connections after they leave that network.

An attacker who can prevent SVCB resolution can deny clients any associated security benefits. A hostile recursive resolver can always deny service to SVCB queries, but network intermediaries can often prevent resolution as well, even when the client and recursive resolver validate DNSSEC and use a secure transport. These downgrade attacks can prevent the "https" upgrade provided by the HTTPS RR (Section 9.5) and can disable any other protections coordinated via SvcParams. To prevent downgrades, Section 3.1 recommends that clients abandon the connection attempt when such an attack is detected.

A hostile DNS intermediary might forge AliasMode "." records (Section 2.5.1) as a way to block clients from accessing particular services. Such an adversary could already block entire domains by forging erroneous responses, but this mechanism allows them to target particular protocols or ports within a domain. Clients that might be subject to such attacks **SHOULD** ignore AliasMode "." records.

A hostile DNS intermediary or authoritative server can return SVCB records indicating any IP address and port number, including IP addresses inside the local network and port numbers assigned to internal services. If the attacker can influence the client's payload (e.g., TLS session ticket contents) and an internal service has a sufficiently lax parser, the attacker could gain access to the internal service. (The same concerns apply to SRV records, HTTP Alt-Svc, and HTTP redirects.) As a mitigation, SVCB mapping documents **SHOULD** indicate any port number restrictions that are appropriate for the supported transports.

13. Privacy Considerations

Standard address queries reveal the user's intent to access a particular domain. This information is visible to the recursive resolver, and to many other parties when plaintext DNS transport is used. SVCB queries, like queries for SRV records and other specific RR types, additionally reveal the user's intent to use a particular protocol. This is not normally sensitive information, but it should be considered when adding SVCB support in a new context.

14. IANA Considerations

14.1. SVCB RR Type

IANA has registered the following new DNS RR type in the "Resource Record (RR) TYPEs" registry on the "Domain Name System (DNS) Parameters" page:

Type: SVCB
Value: 64
Meaning: General-purpose service binding
Reference: RFC 9460

14.2. HTTPS RR Type

IANA has registered the following new DNS RR type in the "Resource Record (RR) TYPEs" registry on the "Domain Name System (DNS) Parameters" page:

Type: HTTPS
Value: 65
Meaning: SVCB-compatible type for use with HTTP
Reference: RFC 9460

14.3. New Registry for Service Parameters

IANA has created the "Service Parameter Keys (SvcParamKeys)" registry in the "Domain Name System (DNS) Parameters" category on a new page entitled "DNS Service Bindings (SVCB)". This registry defines the namespace for parameters, including string representations and numeric SvcParamKey values. This registry is shared with other SVCB-compatible RR types, such as the HTTPS RR.

14.3.1. Procedure

A registration **MUST** include the following fields:

Number: Wire-format numeric identifier (range 0-65535)
Name: Unique presentation name
Meaning: A short description
Reference: Location of specification or registration source
Change Controller: Person or entity, with contact information if appropriate

The characters in the registered Name field entry **MUST** be lowercase alphanumeric or "-" (Section 2.1). The name **MUST NOT** start with "key" or "invalid".

The registration policy for new entries is Expert Review ([RFC8126], Section 4.5). The designated expert **MUST** ensure that the reference is stable and publicly available and that it specifies how to convert the SvcParamValue's presentation format to wire format. The reference **MAY** be any individual's Internet-Draft or a document from any other source with similar assurances of stability and availability. An entry **MAY** specify a reference of the form "Same as (other key name)" if it uses the same presentation and wire formats as an existing key.

This arrangement supports the development of new parameters while ensuring that zone files can be made interoperable.

14.3.2. Initial Contents

The "Service Parameter Keys (SvcParamKeys)" registry has been populated with the following initial registrations:

Number	Name	Meaning	Reference	Change Controller
0	mandatory	Mandatory keys in this RR	RFC 9460, Section 8	IETF
1	alpn	Additional supported protocols	RFC 9460, Section 7.1	IETF
2	no-default-alpn	No support for default protocol	RFC 9460, Section 7.1	IETF
3	port	Port for alternative endpoint	RFC 9460, Section 7.2	IETF
4	ipv4hint	IPv4 address hints	RFC 9460, Section 7.3	IETF

Number	Name	Meaning	Reference	Change Controller
5	ech	RESERVED (held for Encrypted ClientHello)	N/A	IETF
6	ipv6hint	IPv6 address hints	RFC 9460, Section 7.3	IETF
65280-65534	N/A	Reserved for Private Use	RFC 9460	IETF
65535	N/A	Reserved ("Invalid key")	RFC 9460	IETF

Table 1

14.4. Other Registry Updates

Per [Attrleaf], the following entry has been added to the DNS "Underscored and Globally Scoped DNS Node Names" registry:

RR Type	_NODE NAME	Reference
HTTPS	_https	RFC 9460

Table 2

15. References

15.1. Normative References

[ALPN] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[Attrleaf] 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/info/rfc8552>.

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

[DoT] 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/info/rfc7858>.

[HappyEyeballsV2] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

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

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <https://www.rfc-editor.org/info/rfc1928>.

[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/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, DOI 10.17487/RFC3225, December 2001, <https://www.rfc-editor.org/info/rfc3225>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, DOI 10.17487/RFC4001, February 2005, <https://www.rfc-editor.org/info/rfc4001>.

[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/info/rfc5234>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <https://www.rfc-editor.org/info/rfc7050>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[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/info/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/info/rfc8174>.

[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/info/rfc8446>.

[WebSocket] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

15.2. Informative References

[AltSvc] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/info/rfc7838>.

[ANAME-DNS-RR] Finch, T., Hunt, E., van Dijk, P., Eden, A., and W. Mekking, "Address-specific DNS aliases (ANAME)", Work in Progress, Internet-Draft, draft-ietf-dnsop-aname-04, 8 July 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-aname-04>.

[BIND-CHECK-NAMES] Internet Systems Consortium, "BIND v9.19.11 Configuration Reference: "check-names"", September 2023, <https://bind9.readthedocs.io/en/v9.19.11/reference.html#namedconf-statement-check-names>.

[DNAME] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <https://www.rfc-editor.org/info/rfc6672>.

[DNSTerm] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-17, 9 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-17>.

[FETCH] WHATWG, "Fetch Living Standard", October 2023, <https://fetch.spec.whatwg.org/>.

[FETCH-WEBSOCKETS] WHATWG, "WebSockets Living Standard", September 2023, <https://websockets.spec.whatwg.org/>.

[HSTS] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <https://www.rfc-editor.org/info/rfc6797>.

[HTTP-DNS-RR] Bellis, R., "A DNS Resource Record for HTTP", Work in Progress, Internet-Draft, draft-bellis-dnsop-http-record-00, 3 November 2018, <https://datatracker.ietf.org/doc/html/draft-bellis-dnsop-http-record-00>.

[HTTP/3] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC1912] Barr, D., "Common DNS Operational and Configuration Errors", RFC 1912, DOI 10.17487/RFC1912, February 1996, <https://www.rfc-editor.org/info/rfc1912>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[SRV] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[URI] 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/info/rfc3986>.

Appendix A. Decoding Text in Zone Files

DNS zone files are capable of representing arbitrary octet sequences in basic ASCII text, using various delimiters and encodings, according to an algorithm defined in Section 5.1 of [RFC1035]. The following summarizes some allowed inputs to that algorithm, using ABNF:

```
; non-special is VCHAR minus DQUOTE, ";", "(", ")", and "\".
non-special = %x21 / %x23-27 / %x2A-3A / %x3C-5B / %x5D-7E
; non-digit is VCHAR minus DIGIT.
non-digit   = %x21-2F / %x3A-7E
; dec-octet is a number 0-255 as a three-digit decimal number.
dec-octet   = ( "0" / "1" ) 2DIGIT /
              "2" ( ( %x30-34 DIGIT ) / ( "5" %x30-35 ) )
escaped     = "\" ( non-digit / dec-octet )
contiguous  = 1*( non-special / escaped )
quoted      = DQUOTE *( contiguous / ( ["\"] WSP ) ) DQUOTE
char-string = contiguous / quoted
```

The decoding algorithm allows `char-string` to represent any `*OCTET`, using quoting to group values (e.g., those with internal whitespace), and escaping to represent each non-printable octet as a single `escaped` sequence. In this document, this algorithm is referred to as "character-string decoding", because Section 5.1 of [RFC1035] uses this algorithm to produce a `<character-string>`. Note that while the length of a `<character-string>` is limited to 255 octets, the character-string decoding algorithm can produce output of any length.

A.1. Decoding a Comma-Separated List

In order to represent lists of items in zone files, this specification uses comma-separated lists. When the allowed items in the list cannot contain "," or "\", this is trivial. (For simplicity, empty items are not allowed.) A value-list parser that splits on "," and prohibits items containing "\" is sufficient to comply with all requirements in this document. This corresponds to the `simple-comma-separated` syntax:

```
; item-allowed is OCTET minus "," and "\".
item-allowed           = %x00-2B / %x2D-5B / %x5D-FF
simple-item            = 1*item-allowed
simple-comma-separated = [simple-item *("," simple-item)]
```

For implementations that allow "," and "\" in item values, the following escaping syntax applies:

```
item            = 1*OCTET
escaped-item    = 1*(item-allowed / "\," / "\\")
comma-separated = [escaped-item *("," escaped-item)]
```

Decoding of value-lists happens after character-string decoding. For example, consider these `char-string` SvcParamValues:

```
"part1,part2,part3\\,part4\\\\"
part1\,\p\a\r\t2\044part3\092,part4\092\\
```

These inputs are equivalent: character-string decoding either of them would produce the same `value`:

```
part1,part2,part3\,part4\\
```

Applying comma-separated list decoding to this `value` would produce a list of three `items`:

```
part1
part2
part3,part4\
```

Appendix B. HTTP Mapping Summary

This table serves as a non-normative summary of the HTTP mapping for SVCB (Section 9). Future protocol mappings may provide a similar summary table.

Mapped scheme	"https"
Other affected schemes	"http", "wss", "ws", (other HTTP-based)
RR type	HTTPS (65)
Name prefix	None for port 443, else `_$PORT._https`
Automatically mandatory keys	`port`, `no-default-alpn`
SvcParam defaults	`alpn`: ["http/1.1"]
Special behaviors	Upgrade from HTTP to HTTPS
Keys that records must include	None

Table 3

Appendix C. Comparison with Alternatives

The SVCB and HTTPS RR types closely resemble, and are inspired by, some existing record types and proposals. One complaint regarding all of the alternatives is that web clients have seemed unenthusiastic about implementing them. The hope here is that an extensible solution that solves multiple problems will overcome this inertia and have a path to achieve client implementation.

C.1. Differences from the SRV RR Type

An SRV record [SRV] can perform a function similar to that of the SVCB record, informing a client to look in a different location for a service. However, there are several differences:

- SRV records are typically mandatory, whereas SVCB is intended to be optional when used with pre-existing protocols.

- SRV records cannot instruct the client to switch or upgrade protocols, whereas SVCB can signal such an upgrade (e.g., to HTTP/2).
- SRV records are not extensible, whereas SVCB and HTTPS RRs can be extended with new parameters.
- SRV records specify a "weight" for unbalanced randomized load balancing. SVCB only supports balanced randomized load balancing, although weights could be added via a future SvcParam.

C.2. Differences from the Proposed HTTP Record

Unlike [HTTP-DNS-RR], this approach is extensible to cover Alt-Svc and Encrypted ClientHello use cases. Like that proposal, this addresses the zone-apex CNAME challenge.

Like that proposal, it remains necessary to continue to include address records at the zone apex for legacy clients.

C.3. Differences from the Proposed ANAME Record

Unlike [ANAME-DNS-RR], this approach is extensible to cover Alt-Svc and Encrypted ClientHello use cases. This approach also does not require any changes or special handling on either authoritative or primary servers, beyond optionally returning in-bailiwick additional records.

Like that proposal, this addresses the zone-apex CNAME challenge for clients that implement this.

However, with this SVCB proposal, it remains necessary to continue to include address records at the zone apex for legacy clients. If deployment of this standard is successful, the number of legacy clients will fall over time. As the number of legacy clients declines, the operational effort required to serve these users without the benefit of SVCB indirection should fall. Server operators can easily observe how much traffic reaches this legacy endpoint and may remove the apex's address records if the observed legacy traffic has fallen to negligible levels.

C.4. Comparison with Separate RR Types for AliasMode and ServiceMode

Abstractly, functions of AliasMode and ServiceMode are independent, so it might be tempting to specify them as separate RR types. However, this would result in serious performance impairment, because clients cannot rely on their recursive resolver to follow SVCB aliases (unlike CNAME). Thus, clients would have to issue queries for both RR types in parallel, potentially at each step of the alias chain. Recursive resolvers that implement the specification would, upon receipt of a ServiceMode query, emit both a ServiceMode query and an AliasMode query to the authoritative DNS server. Thus, splitting the RR type would double, or in some cases triple, the load on clients and servers, and would not reduce implementation complexity.

Appendix D. Test Vectors

These test vectors only contain the RDATA portion of SVCB/HTTPS records in presentation format, generic format [RFC3597], and wire format. The wire format uses hexadecimal (\xNN) for each non-ASCII byte. As the wire format is long, it is broken into several lines.

D.1. AliasMode

```
example.com.   HTTPS   0 foo.example.com.

\# 19 (
00 00                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 ; target
)

\x00\x00                                              # priority
\x03foo\x07example\x03com\x00                         # target
```

Figure 2: AliasMode

D.2. ServiceMode

```
example.com.   SVCB   1 .

\# 3 (
00 01      ; priority
00         ; target (root label)
)

\x00\x01   # priority
\x00       # target (root label)
```

Figure 3: TargetName Is "."

```
example.com.   SVCB   16 foo.example.com. port=53

\# 25 (
00 10                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 ; target
00 03                                                 ; key 3
00 02                                                 ; length 2
00 35                                                 ; value
)

\x00\x10                                              # priority
\x03foo\x07example\x03com\x00                         # target
\x00\x03                                              # key 3
\x00\x02                                              # length 2
\x00\x35                                              # value
```

Figure 4: Specifies a Port

```
example.com.   SVCB   1 foo.example.com. key667=hello

\# 28 (
00 01                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 ; target
02 9b                                                 ; key 667
00 05                                                 ; length 5
68 65 6c 6c 6f                                        ; value
)

\x00\x01                                              # priority
\x03foo\x07example\x03com\x00                         # target
\x02\x9b                                              # key 667
\x00\x05                                              # length 5
hello                                                 # value
```

Figure 5: A Generic Key and Unquoted Value

```
example.com.   SVCB   1 foo.example.com. key667="hello\210qoo"

\# 32 (
00 01                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 ; target
02 9b                                                 ; key 667
00 09                                                 ; length 9
68 65 6c 6c 6f d2 71 6f 6f                            ; value
)

\x00\x01                                              # priority
\x03foo\x07example\x03com\x00                         # target
\x02\x9b                                              # key 667
\x00\x09                                              # length 9
hello\xd2qoo                                          # value
```

Figure 6: A Generic Key and Quoted Value with a Decimal Escape

```
example.com.   SVCB   1 foo.example.com. (
                      ipv6hint="2001:db8::1,2001:db8::53:1"
                      )

\# 55 (
00 01                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 63 6f 6d 00 ; target
00 06                                                 ; key 6
00 20                                                 ; length 32
20 01 0d b8 00 00 00 00 00 00 00 00 00 00 00 01    ; first address
20 01 0d b8 00 00 00 00 00 00 00 00 00 53 00 01    ; second address
)

\x00\x01                                              # priority
\x03foo\x07example\x03com\x00                         # target
\x00\x06                                              # key 6
\x00\x20                                              # length 32
\x20\x01\x0d\xb8\x00\x00\x00\x00
     \x00\x00\x00\x00\x00\x00\x00\x01                 # first address
\x20\x01\x0d\xb8\x00\x00\x00\x00
     \x00\x00\x00\x00\x00\x53\x00\x01                 # second address
```

Figure 7: Two Quoted IPv6 Hints

```
example.com.   SVCB   1 example.com. (
                       ipv6hint="2001:db8:122:344::192.0.2.33"
                       )
\# 35 (
00 01                                                 ; priority
07 65 78 61 6d 70 6c 65 03 63 6f 6d 00                ; target
00 06                                                 ; key 6
00 10                                                 ; length 16
20 01 0d b8 01 22 03 44 00 00 00 00 c0 00 02 21       ; address
)

\x00\x01                                              # priority
\x07example\x03com\x00                                # target
\x00\x06                                              # key 6
\x00\x10                                              # length 16
\x20\x01\x0d\xb8\x01\x22\x03\x44
     \x00\x00\x00\x00\xc0\x00\x02\x21                 # address
```

Figure 8: An IPv6 Hint Using the Embedded IPv4 Syntax

```
example.com.   SVCB   16 foo.example.org. (
                       alpn=h2,h3-19 mandatory=ipv4hint,alpn
                       ipv4hint=192.0.2.1
                       )

\# 48 (
00 10                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 6f 72 67 00 ; target
00 00                                                 ; key 0
00 04                                                 ; param length 4
00 01                                                 ; value: key 1
00 04                                                 ; value: key 4
00 01                                                 ; key 1
00 09                                                 ; param length 9
02                                                    ; alpn length 2
68 32                                                 ; alpn value
05                                                    ; alpn length 5
68 33 2d 31 39                                        ; alpn value
00 04                                                 ; key 4
00 04                                                 ; param length 4
c0 00 02 01                                           ; param value
)

\x00\x10                                              # priority
\x03foo\x07example\x03org\x00                         # target
\x00\x00                                              # key 0
\x00\x04                                              # param length 4
\x00\x01                                              # value: key 1
\x00\x04                                              # value: key 4
\x00\x01                                              # key 1
\x00\x09                                              # param length 9
\x02                                                  # alpn length 2
h2                                                    # alpn value
\x05                                                  # alpn length 5
h3-19                                                 # alpn value
\x00\x04                                              # key 4
\x00\x04                                              # param length 4
\xc0\x00\x02\x01                                      # param value
```

Figure 9: SvcParamKey Ordering Is Arbitrary in Presentation Format but Sorted in Wire Format

```
example.com.   SVCB   16 foo.example.org. alpn="f\\\\oo\\,bar,h2"
example.com.   SVCB   16 foo.example.org. alpn=f\\\092oo\092,bar,h2

\# 35 (
00 10                                                 ; priority
03 66 6f 6f 07 65 78 61 6d 70 6c 65 03 6f 72 67 00 ; target
00 01                                                 ; key 1
00 0c                                                 ; param length 12
08                                                    ; alpn length 8
66 5c 6f 6f 2c 62 61 72                               ; alpn value
02                                                    ; alpn length 2
68 32                                                 ; alpn value
)

\x00\x10                                              # priority
\x03foo\x07example\x03org\x00                         # target
\x00\x01                                              # key 1
\x00\x0c                                              # param length 12
\x08                                                  # alpn length 8
f\oo,bar                                              # alpn value
\x02                                                  # alpn length 2
h2                                                    # alpn value
```

Figure 10: An "alpn" Value with an Escaped Comma and an Escaped Backslash in Two Presentation Formats

D.3. Failure Cases

This subsection contains test vectors that are not compliant with this document. The various reasons for non-compliance are explained with each example.

```
example.com.   SVCB   1 foo.example.com. (
                       key123=abc key123=def
                       )
```

Figure 11: Multiple Instances of the Same SvcParamKey

```
example.com.   SVCB   1 foo.example.com. mandatory
example.com.   SVCB   1 foo.example.com. alpn
example.com.   SVCB   1 foo.example.com. port
example.com.   SVCB   1 foo.example.com. ipv4hint
example.com.   SVCB   1 foo.example.com. ipv6hint
```

Figure 12: Missing SvcParamValues That Must Be Non-Empty

```
example.com.   SVCB   1 foo.example.com. no-default-alpn=abc
```

Figure 13: The "no-default-alpn" SvcParamKey Value Must Be Empty

```
example.com.   SVCB   1 foo.example.com. mandatory=key123
```

Figure 14: A Mandatory SvcParam Is Missing

```
example.com.   SVCB   1 foo.example.com. mandatory=mandatory
```

Figure 15: The "mandatory" SvcParamKey Must Not Be Included in the Mandatory List

```
example.com.   SVCB   1 foo.example.com. (
                      mandatory=key123,key123 key123=abc
                      )
```

Figure 16: Multiple Instances of the Same SvcParamKey in the Mandatory List

Acknowledgments and Related Proposals

Over the years, IETF participants have proposed a wide range of solutions to the "CNAME at the zone apex" challenge, including [HTTP-DNS-RR], [ANAME-DNS-RR], and others. The authors are grateful for their work to elucidate the problem and identify promising strategies to address it, some of which are reflected in this document.

Thank you to Ian Swett, Ralf Weber, Jon Reed, Martin Thomson, Lucas Pardue, Ilari Liusvaara, Tim Wicinski, Tommy Pauly, Chris Wood, David Benjamin, Mark Andrews, Emily Stark, Eric Orth, Kyle Rose, Craig Taylor, Dan McArdle, Brian Dickson, Willem Toorop, Pieter Lexis, Puneet Sood, Olivier Poitrey, Mashooq Muhaimen, Tom Carpay, and many others for their feedback and suggestions on this document.

Authors' Addresses

Ben Schwartz
Meta Platforms, Inc.
Email: ietf@bemasc.net

Mike Bishop
Akamai Technologies
Email: mbishop@evequefou.be

Erik Nygren
Akamai Technologies
Email: erik+ietf@nygren.org