Workgroup: DNSOP Working Group
Internet-Draft: draft-ietf-dnsop-svcb-https-06
Published: 16 June 2021
Intended Status: Standards Track
Expires: 18 December 2021
Authors: B. Schwartz
Google
M. Bishop
Akamai Technologies
E. Nygren
Akamai Technologies

Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)

Abstract

This document specifies the "SVCB" and "HTTPS" DNS resource record (RR) types to facilitate the lookup of information needed to make connections to network services, such as for HTTPS origins. SVCB records allow a service to be provided from multiple alternative endpoints, each with associated parameters (such as transport protocol configuration and 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 HTTPS and HTTP origins. By providing more information to the client before it attempts to establish a connection, these records offer potential benefits to both performance and privacy.

TO BE REMOVED: This document is being collaborated on in Github at: https://github.com/MikeBishop/dns-alt-svc. The most recent working version of the document, open issues, etc. should all be available there. The authors (gratefully) accept pull requests.

Status of This Memo

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

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

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

This Internet-Draft will expire on 18 December 2021.

Copyright Notice

Table of Contents

1. Introduction

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

For example, when clients need to make a connection to fetch resources associated with an HTTPS URI, they currently resolve only A and/or AAAA records for the origin hostname. This is adequate for services that use basic HTTPS (fixed port, no QUIC, no [ECH]). Going beyond basic HTTPS confers privacy, performance, and operational advantages, but it requires the client to learn additional information, and 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. delegating 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 resource record that can be applied directly and efficiently to a wide range of services (Section 2). The HTTPS RR is then defined as a special case of SVCB that improves efficiency and convenience for use with HTTPS (Section 8) by avoiding the need for an Attrleaf label [Attrleaf] (Section 8.1). Other protocols with similar needs may follow the pattern of HTTPS and assign their own SVCB-compatible RR types.

All behaviors described as applying to the SVCB RR also apply to the HTTPS RR unless explicitly stated otherwise. Section 8 describes additional behaviors specific to the HTTPS RR. Apart from Section 8 and introductory examples, much of this document refers only to the SVCB RR, but those references should be taken to apply to SVCB, HTTPS, and any future SVCB-compatible RR types.

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

1.1. Goals of the SVCB RR

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 RRs for the same name. For example, if www.example.com is a CNAME alias that switches between one of three 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 delegation of operational authority for an origin within the DNS to an alternate name.

Additional goals specific to HTTPS RRs and the HTTPS use-case include:

- Connect directly to HTTP3 (QUIC transport) alternative endpoints [HTTP3]

- Obtain the Encrypted ClientHello [ECH] keys associated with an alternative endpoint

- Support non-default TCP and UDP ports

- Enable SRV-like benefits (e.g. apex delegation, as mentioned above) for HTTP(S), where SRV [SRV] has not been widely adopted

- Provide an HSTS-like indication [HSTS] signaling that the HTTPS scheme should be used instead of HTTP for this request (see Section 8.5).

1.2. Overview of the SVCB RR

This subsection briefly describes the SVCB RR in a non-normative manner. (As mentioned above, this all applies equally to the HTTPS

RR which shares the same encoding, format, and high-level semantics.)

The SVCB RR has two modes: AliasMode, which aliases a name to another name, and ServiceMode, 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.

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

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

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

3. SvcParams (optional): A list of key=value pairs describing the alternative endpoint at TargetName (only used in ServiceMode and otherwise ignored). 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. Clients either use responses included in the additional section returned by the recursive resolver or perform necessary SVCB, A, and AAAA record resolutions. DNS authoritative servers can attach in-bailiwick SVCB, A, AAAA, and CNAME records in the Additional Section to responses for a SVCB query.

In ServiceMode, the SvcParams of the SVCB RR provide an extensible data model for describing alternative endpoints that are authoritative for the origin, along with parameters associated with each of these alternative endpoints.

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

1.3. Parameter for Encrypted ClientHello

This document also defines a parameter for Encrypted ClientHello [ECH] keys. See Section 9.

1.4. Terminology

Our terminology 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 HTTPS "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 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 format and registry are collectively known as SVCB-compatible RR types.

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 resource record (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 resource record types can also be defined as-needed. In particular, the HTTPS RR (RR type 65) provides special handling for the case of "https" origins as described in Section 8.

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. Their definition should specify both their presentation format and wire encoding (e.g., domain

names, binary data, or numeric values). The initial SvcParamKeys and formats are defined in Section 6.

2.1. Zone file presentation format

The presentation format of the record is:

```
Name TTL IN SVCB SvcPriority TargetName SvcParams
```

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

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 subject to IANA control (Section 14.3).

Each SvcParamKey SHALL appear at most once in the SvcParams. In presentation format, SvcParamKeys are lower-case alphanumeric strings. Key names should 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
value         = *OCTET
```

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 "=" is omitted, the value is interpreted as empty.

Unrecognized keys are represented in presentation format as "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 as in 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 (ServiceMode), 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 (but constrained by the RDATA and DNS message sizes).
- an octet string of this length whose contents are 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 an 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. We term this behavior "Port Prefix Naming", and use it in the examples throughout this document.

See Section 8.1 for the 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.

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.
```

to 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" ech="..." )
```

to 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 ([RFC1035], Section 5.1).)

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. SVCB RRs with a smaller SvcPriority value SHOULD be given preference over 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 resource record in AliasMode. If multiple 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. In AliasMode, the TargetName will be the name of a domain that resolves to SVCB (or other SVCB-compatible record such as HTTPS), AAAA, and/or A records. The TargetName SHOULD NOT be equal to the owner name, as this would result in a loop.

In AliasMode, records SHOULD NOT include any SvcParams, and recipients MUST ignore any SvcParams that are present.

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, which would likely improve performance.

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.

For compatibility and performance, zone owners SHOULD NOT configure their zones to require following multiple AliasMode records.

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, nor vice-versa.

2.4.3. ServiceMode

In ServiceMode, the TargetName and SvcParams within each resource record 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 others' requirements. Zone-file implementations SHOULD enforce self-consistency. Clients MUST reject any RR whose recognized SvcParams are not self-consistent, and MAY reject the entire RRSet.

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.

For example, in the following 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 ech="..."
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 the priority-ordered 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 7) ServiceMode records are returned, these represent the alternative endpoints.

5. Otherwise, SVCB resolution has failed, and the list of known 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.

Once SVCB resolution has concluded, the client proceeds with connection establishment. Clients SHOULD try higher-priority alternatives first, with fallback to lower-priority alternatives. Clients issue AAAA and/or A queries for the selected TargetName, and MAY choose between them using an approach such as Happy Eyeballs [HappyEyeballsV2].

A client is called "SVCB-optional" if it can connect without the use of ServiceMode records, and "SVCB-reliant" otherwise. Clients for pre-existing protocols (e.g. HTTPS) SHALL implement SVCB-optional behavior (except as noted in Section 3.1 and Section 9.1).

SVCB-optional clients SHOULD issue in parallel any other DNS queries that might be needed for connection establishment. SVCB-optional clients SHALL append an alternative endpoint consisting of the final value of $QNAME, the authority endpoint's port number, and no SvcParams, to the list of alternative endpoints, which is 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.

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 SVCB resolution fails due to a SERVFAIL error, transport error, or timeout, and DNS exchanges between the client and the recursive resolver are cryptographically protected (e.g. using TLS [DoT] or HTTPS [DoH]), a SVCB-optional client SHOULD abandon the connection attempt like a SVCB-reliant client would. 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.

Similarly, if the client enforces DNSSEC validation on A/AAAA responses, it SHOULD terminate the connection if a SVCB response fails to validate.

If the client is unable to complete SVCB resolution due to its chain length limit, the client SHOULD fall back to the authority endpoint, as if the origin'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 to use named destinations, in which case the client does not perform any A or AAAA queries for destination domains. If the client is using 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. Clients that support such proxies SHOULD arrange for a separate SVCB resolution procedure with appropriate privacy properties, or disable SVCB resolution entirely if SVCB-optional.

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.

Providing the proxy with the final TargetName has several benefits:

* It allows the client to use the SvcParams, if present, which is only usable with a specific TargetName. The SvcParams may include information that enhances performance (e.g. alpn) and privacy (e.g. ech).

* It allows the service to delegate the apex domain.

* It allows the proxy to select between IPv4 and IPv6 addresses for the server according to its configuration, and receive addresses based on its network geolocation.

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 positive or negative DNSSEC responses for these records in the Additional section.

See Section 4.4 for exceptions.

4.2. Recursive resolvers

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 response. For the initial SVCB record query, this is just the normal response construction process (i.e. unknown RR type resolution under [RFC3597]). For followup resolutions performed during this procedure, we define incorporation as adding all useful RRs from the response to the Additional section without altering the response code.

Upon receiving a SVCB query, recursive resolvers SHOULD start with the standard resolution procedure, and then follow this procedure to construct the full response to the stub resolver:

1. Incorporate the results of SVCB resolution. If the chain length limit has been reached, terminate successfully (i.e. a NOERROR response).

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]).

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.

4.3. General requirements

Recursive resolvers MUST be able to convey SVCB records with unrecognized SvcParamKeys, and MAY treat the entire SvcParams portion of the record as opaque. No part of this specification requires recursive resolvers to alter their behavior based on its contents, even if the contents are invalid. Recursive resolvers MAY validate the values of recognized SvcParamKeys and reject records containing values which are invalid according to the SvcParam specification. 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 option (ECS, [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 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 followup query that can receive properly tailored ECS. (This is similar to the usage of CNAME with ECS discussed in [RFC7871], Section 7.2.1.)

Authoritative servers that omit Additional records can avoid the added latency of a followup 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 followup queries. With this behavior, and 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, a TLS ClientHello can be altered by the "ech" value of a SVCB response (Section 6.3). 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. (
    ech="111..." ipv6hint=2001:db8::1 port=1234 )
                               SVCB 2 svc2.example.net. (
    ech="222..." 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 using ech="222...", 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 (Section 4.2 step 1), 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. Initial SvcParamKeys

A few initial SvcParamKeys are defined here. These keys are useful for HTTPS, and most are applicable to other protocols 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.

6.1. "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.

As with Alt-Svc [AltSvc], the ALPN protocol identifier is used to identify the application protocol and associated suite of protocols supported by the endpoint (the "protocol suite"). 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).

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

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

The presentation value SHALL be a comma-separated list (Appendix A.1) of one or more alpn-ids. Zone file implementations MAY disallow the "," and "\" characters 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 ALPNs that are supported by nearly all clients and servers, which MAY be empty. To determine the set of protocol suites supported by an endpoint (the "SVCB ALPN set"), the client adds the default set to the list of alpn-ids unless the "no-default-alpn" SvcParamKey is present. The presence of an ALPN protocol 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 protocol.

ALPN protocol names 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.

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.

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 RRs indicate "no-default-alpn", even if connection could have succeeded using a non-default alpn.

For compatibility with clients that require default transports, zone operators SHOULD ensure that at least one RR in each RRSet supports the default transports.

6.2. "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 SvcParam 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.

6.3. "ech"

The SvcParamKey to enable Encrypted ClientHello (ECH) is "ech". Its value is defined in Section 9. It is applicable to most TLS-based protocols.

When publishing a record containing an "ech" parameter, the publisher MUST ensure that all IP addresses of TargetName correspond to servers that have access to the corresponding private key or are authoritative for the public name. (See Section 7.2.2 of [ECH] for more details about the public name.) This yields an anonymity set of cardinality equal to the number of ECH-enabled server domains supported by a given client-facing server. Thus, even with an encrypted ClientHello, an attacker who can enumerate the set of ECH-enabled domains supported by a client-facing server can guess the correct SNI with probability at least 1/K, where K is the size of this ECH-enabled server anonymity set. This probability may be increased via traffic analysis or other mechanisms.

6.4. "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 as in 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]. 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. 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, IPv6-only clients may synthesize IPv6 addresses as specified in [RFC7050] or ignore the "ipv4hint" key and wait for AAAA resolution (Section 3). Recursive resolvers MUST NOT perform DNS64 ([RFC6147]) on parameters within a SVCB record. 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 origin hostname 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. 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" with a client if the client recognizes all the mandatory keys, and their values indicate that successful connection establishment is possible. If the SVCB RRSet contains no compatible RRs, the client will generally act as if the RRSet is empty.

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:

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

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

8. Using SVCB with HTTPS and HTTP

Use of any protocol with SVCB requires a protocol-specific mapping specification. This section specifies the mapping for HTTPS and HTTP.

To enable special handling for the HTTPS and 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 HTTPS RR wire format and presentation format are identical to SVCB, and both share the SvcParamKey registry. SVCB semantics apply equally to HTTPS RRs unless specified otherwise. The presentation format of the record is:

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

As with SVCB, the record is defined specifically within the Internet ("IN") Class [RFC1035].

All the SvcParamKeys defined in Section 6 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 7) are "port" and "no-default-alpn". (As described in Section 7, clients must either implement these keys or ignore any RR in which they appear.) Clients that restrict the HTTPS destination port (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 all HTTP resources are available over HTTPS, as discussed in Section

8.5. This allows HTTPS RRs to apply to pre-existing "http" scheme URLs, while ensuring that the client uses a secure and authenticated HTTPS connection.

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

8.1. 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 HTTPS. Reusing the service name 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.

Following of HTTPS AliasMode RRs and CNAME aliases is unchanged from SVCB.

Clients always convert "http" URLs to "https" before performing an HTTPS RR query using the process described in Section 8.5, 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.

8.2. Relationship to Alt-Svc

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

8.2.1. ALPN usage

Unlike Alt-Svc Field Values, HTTPS RRs can contain multiple ALPN IDs, and clients are encouraged to offer additional ALPNs that they support.

8.2.2. Untrusted channel

SVCB does not require or provide any assurance of authenticity. (DNSSEC signing and verification, which would provide such assurance, are OPTIONAL.) The DNS resolution process is treated as an untrusted channel that learns only the QNAME, and is prevented from mounting any attack beyond denial of service.

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.

8.2.3. 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.

8.2.4. 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.

8.3. Interaction with Alt-Svc

Clients that implement support for both Alt-Svc and HTTPS records 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 override the alt-authority. 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 ech=...
alt2.example.             IN HTTPS 1 alt2b.example. alpn=h3 ech=...
_8443._https.example.com. IN HTTPS 1 alt3.example. (
    port=9443 alpn=h2,h3 ech=... )
```

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

- HTTPS over TCP to alt.example:443 (Consistent with both Alt-Svc and its HTTPS record)
- HTTP/3 to alt3.example:9443 (Consistent with both Alt-Svc and its HTTPS record)
- Fallback to the the client's non-Alt-Svc connection behavior

ECH-capable clients would use ECH when establishing any of these connections.

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 consistent with both the HTTPS record and Alt-Svc)
- HTTPS over TCP to any port on alt3.example (not consistent with Alt-Svc)

The following connection attempts would be allowed only if the client does not support ECH, as they rely on SVCB-optional fallback behavior that is disabled when the "ech" SvcParam is present (Section 9.1):

- HTTPS over TCP to alt2.example:443 (Alt-Svc only)
- HTTP/3 to example.com:8443 (Alt-Svc only)

Origins that publish an "ech" SvcParam in their HTTPS record SHOULD also publish an "ech" SvcParam for any alt-authorities. Otherwise, clients might reveal the name of the server in an unencrypted ClientHello. Similar consistency considerations could apply to future SvcParamKeys, so 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 ECH support when there is an active ECH-protected connection for this origin.

8.4. Requiring Server Name Indication

Clients MUST NOT use an HTTPS RR response unless the client supports TLS Server Name Indication (SNI) and indicates the origin name when negotiating TLS. 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.

8.5. HTTP Strict Transport Security

By publishing a usable HTTPS RR, the server operator indicates that all useful HTTP resources on that origin are reachable over HTTPS, similar to HTTP Strict Transport Security [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], point 5.

If an HTTPS RR query for this "https" URL returns any AliasMode HTTPS RRs, or any compatible ServiceMode HTTPS RRs (see Section 7), the client SHOULD act as if it has received an HTTP "307 Temporary Redirect" redirect to this "https" URL. (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 redirect over cleartext HTTP.

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 Section 8.4 and Section 12.1 of [HSTS].

8.6. HTTP-based protocols

All protocols employing "http://" or "https://" URLs SHOULD respect HTTPS RRs. For example, clients that support HTTPS RRs and implement the altered WebSocket [WebSocket] opening handshake from the W3C Fetch specification [FETCH] SHOULD use HTTPS RRs for the requestURL.

An HTTP-based protocol MAY define its own SVCB mapping. Such mappings MAY be defined to take precedence over HTTPS RRs.

9. SVCB/HTTPS RR parameter for ECH configuration

The SVCB "ech" parameter is defined for conveying the ECH configuration of an alternative endpoint. In wire format, the value of the parameter is an ECHConfigList [ECH], including the redundant length prefix. In presentation format, the value is a single ECHConfigList encoded in Base64 [base64]. Base64 is used here to simplify integration with TLS server software. To enable simpler parsing, this SvcParam MUST NOT contain escape sequences.

When ECH is in use, the TLS ClientHello is divided into an unencrypted "outer" and an encrypted "inner" ClientHello. The outer ClientHello is an implementation detail of ECH, and its contents are controlled by the ECHConfig in accordance with [ECH]. The inner ClientHello is used for establishing a connection to the service, so its contents may be influenced by other SVCB parameters. For example, the requirements on the ProtocolNameList in Section 6.1 apply only to the inner ClientHello. Similarly, it is the inner ClientHello whose Server Name Indication identifies the desired service.

9.1. Client behavior

The SVCB-optional client behavior specified in Section 3 permits clients to fall back to a direct connection if all SVCB options fail. This behavior is not suitable for ECH, because fallback would negate the privacy benefits of ECH. Accordingly, ECH-capable SVCB-optional clients MUST switch to SVCB-reliant connection establishment if SVCB resolution succeeded (following Section 3) and all alternative endpoints have an "ech" key.

As a latency optimization, clients MAY prefetch DNS records that will only be used in SVCB-optional mode.

9.2. Deployment considerations

An HTTPS RRSet containing some RRs with "ech" and some without is vulnerable to a downgrade attack. This configuration is NOT RECOMMENDED. Zone owners who do use such a mixed configuration SHOULD mark the RRs with "ech" as more preferred (i.e. smaller SvcPriority) than those without, in order to maximize the likelihood that ECH will be used in the absence of an active adversary.

10. Zone Structures

10.1. Structuring zones for flexibility

Each ServiceForm 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.

10.2. Structuring zones for performance

To avoid a delay for clients using a nonconforming 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. For foo://foo.example.com:8080, this might look like:

```
$ORIGIN example.com. ; Origin
foo                  3600 IN CNAME foosvc.example.net.
_8080._foo.foo       3600 IN CNAME foosvc.example.net.

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

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 8 aliases (counting both AliasMode and CNAME records) are NOT RECOMMENDED.

10.3. Examples

10.3.1. 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
```

to indicate that https://simple.example supports QUIC in addition to HTTPS over TCP (an implicit default). The record could also include other information (e.g. non-standard port, ECH configuration). 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.

10.3.2. 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 to HTTPS on aliased.example.) 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.

10.3.3. Parameter binding

Suppose that svc.example's default server pool supports HTTP/2, and it has deployed HTTP/3 on a new server pool with a different configuration. This can be expressed in the following form:

```
$ORIGIN svc.example. ; A hosting provider.
pool  7200 IN HTTPS 1 h3pool alpn=h2,h3 ech="123..."
              HTTPS 2 .      alpn=h2 ech="abc..."
pool   300 IN A        192.0.2.2
              AAAA     2001:db8::2
h3pool 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, ECH configuration, etc.

10.3.4. Multi-CDN

The HTTPS RR is intended to support HTTPS services operated by multiple independent entities, such as different Content Delivery Networks (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 ech="123..."
                 HTTPS 2 . alpn=h2 ech="123..."
                 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 ech="xyz..."
          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 customer.svc2.example
; path use these records.
; Note that this CDN has no HTTPS records
```

```
; and thus no ECH support.
$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 ECH 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 ECH, and CDN 2 does not, the client is vulnerable to ECH 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, HTTPS resolutions are independent lookups so clients may observe and follow different CNAMEs to different CDNs. Clients may thus find a SvcDomainName pointing to a name other than the one which returned along with the A and AAAA lookups and will need to do an additional resolution for them. 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), and thus no "ech" SvcParam, but could receive A/AAAA records from a different CDN which does support ECH. Clients will be unable to use ECH in this case.

10.3.5. Non-HTTPS uses

For services other than HTTPS, 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 OCSP Stapling [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 use of SVCB with a protocol MUST have an entry for its scheme under the SVCB RR type in the IANA DNS Underscore Global Scoped Entry Registry [Attrleaf]. The scheme SHOULD have an entry in the IANA URI Schemes Registry [RFC7595]. The scheme SHOULD have a defined specification for use with SVCB.

12. Security Considerations

SVCB/HTTPS RRs are intended for 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 8.5), and disable the encryption enabled by the "ech" SvcParamKey (Section 9). 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 AliasForm "." 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 AliasForm "." records.

A hostile DNS intermediary or origin 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, it's possible that the attacker could gain unintended access. (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 RRType

This document defines a new DNS RR type, SVCB, whose value 64 has been allocated by IANA from the "Resource Record (RR) TYPEs" subregistry of the "Domain Name System (DNS) Parameters" registry:

Type: SVCB

Value: 64

Meaning: General Purpose Service Endpoints

Reference: This document

14.2. HTTPS RRType

This document defines a new DNS RR type, HTTPS, whose value 65 has been allocated by IANA from the "Resource Record (RR) TYPEs" subregistry of the "Domain Name System (DNS) Parameters" registry:

Type: HTTPS

Value: 65

Meaning: HTTPS Specific Service Endpoints

Reference: This document

14.3. New registry for Service Parameters

The "Service Binding (SVCB) Parameter 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.

ACTION: create and include a reference to this registry.

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
- *Format Reference: pointer to specification text

The characters in the registered Name MUST be lower-case alphanumeric or "-" (Section 2.1). The name MUST NOT start with "key" or "invalid".

Entries in this registry are subject to a First Come First Served registration policy ([RFC8126], Section 4.4). The Format Reference MUST specify how to convert the SvcParamValue's presentation format to wire format and MAY detail its intended meaning and use. An entry MAY specify a Format 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 Binding (SVCB) Parameter Registry" shall initially be populated with the registrations below:

Number	Name	Meaning	Format Reference
0	mandatory	Mandatory keys in this RR	(This document) Section 7
1	alpn	Additional supported protocols	(This document) Section 6.1
2	no-default-alpn	No support for default protocol	(This document) Section 6.1
3	port		(This document) Section 6.2

Number	Name	Meaning	Format Reference
		Port for alternative endpoint	
4	ipv4hint	IPv4 address hints	(This document) Section 6.4
5	ech	Encrypted ClientHello info	(This document) Section 6.3
6	ipv6hint	IPv6 address hints	(This document) Section 6.4
65280-65534	N/A	Private Use	(This document)
65535	N/A	Reserved ("Invalid key")	(This document)

Table 1

14.4. Registry updates

Per [RFC6895], please add the following entries to the data type range of the Resource Record (RR) TYPEs registry:

TYPE	Meaning	Reference
SVCB	Service Location and Parameter Binding	(This document)
HTTPS	HTTPS Service Location and Parameter Binding	(This document)

Table 2

Per [Attrleaf], please add the following entry to the DNS Underscore Global Scoped Entry Registry:

RR TYPE	_NODE NAME	Meaning	Reference
HTTPS	_https	HTTPS SVCB info	(This document)

Table 3

15. Acknowledgments and Related Proposals

There have been a wide range of proposed solutions over the years to the "CNAME at the Zone Apex" challenge proposed. These include [I-D.bellis-dnsop-http-record], [I-D.ietf-dnsop-aname], and others.

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 draft.

16. References

16.1. 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://doi.org/10.17487/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://doi.org/10.17487/RFC8552>.

[base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://doi.org/10.17487/RFC4648>.

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

[DoH] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://doi.org/10.17487/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://doi.org/10.17487/RFC7858>.

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-11, 14 June 2021, <https://tools.ietf.org/html/draft-ietf-tls-esni-11>.

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

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

[HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-

http-34, 2 February 2021, <https://tools.ietf.org/html/draft-ietf-quic-http-34>.

[RFC1035] Mockapetris, P.V., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/RFC2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://doi.org/10.17487/RFC2181>.

[RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, DOI 10.17487/RFC3225, December 2001, <https://doi.org/10.17487/RFC3225>.

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

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

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

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/RFC8174>.

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

[WebSocket] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://doi.org/10.17487/RFC6455>.

16.2. Informative References

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

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

[FETCH] "Fetch Living Standard", May 2020, <https://fetch.spec.whatwg.org/>.

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

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

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <https://doi.org/10.17487/RFC3513>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://doi.org/10.17487/RFC6454>.

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <https://doi.org/10.17487/RFC6895>.

[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://doi.org/10.17487/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://doi.org/10.17487/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. The algorithm for decoding these character-strings is defined in Section 5.1 of [RFC1035]. Here we summarize the allowed input 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. In this document, this algorithm is referred to as "character-string decoding". The algorithm is the same as used by <character-string> in RFC 1035, although the output length in this document is not limited to 255 octets.

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.

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

```
item            = 1*OCTET
; item-allowed is OCTET minus "," and "\".
item-allowed    = %x00-2B / %x2D-5B / %x5D-FF
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 8). 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	HTTP to HTTPS upgrade

Table 4

This table does not indicate any SvcParamKeys that servers are required to publish, or that clients are required to implement, because there are none in this mapping.

Appendix C. Comparison with alternatives

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

C.1. Differences from the SRV RR type

An SRV record [SRV] can perform a similar function to 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 clients will always continue to function correctly without making use of SVCB.

* 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.

* SVCB records use 16 bit for SvcPriority for consistency with SRV and other RR types that also use 16 bit priorities.

C.2. Differences from the proposed HTTP record

Unlike [I-D.bellis-dnsop-http-record], 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 [I-D.ietf-dnsop-aname], this approach is extensible to cover Alt-Svc and ECH 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 a 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 and an AliasMode query to the authoritative. 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 wireformat is long, it is broken into several lines.

D.1. AliasForm

```
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
```

D.2. ServiceForm

The first form is the simple "use the ownername".

```
example.com.   SVCB   1 .

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

\x00\x01   # priority
\x00       # target, root label
```

This vector only has a port.

```
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 bytes
\x00\x35                                        # value
```

This example has a key that is not registered, its value is unquoted.

```
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
```

This example has a key that is not registered, its value is quoted and contains a decimal-escaped character.

```
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
```

Here, two IPv6 hints are quoted in the presentation format.

```
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
```

This example shows a single IPv6 hint in IPv4-mapped IPv6 presentation format([RFC3513]).

```
example.com.   SVCB   1 example.com. ipv6hint="::ffff:198.51.100.100"

\# 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
00 00 00 00 00 00 00 00 00 00 ff ff c6 33 64 64       ; address
)

\x00\x01                                              # priority
\x07example\x03com\x00                                # target
\x00\x06                                              # key 6
\x00\x10                                              # length 16
\x00\x00\x00\x00\x00\x00\x00\x00
     \x00\x00\xff\xff\xc6\x33\x64\x64                 # address
```

In the next vector, neither the SvcParamValues nor the mandatory keys are sorted in presentation format, but are correctly sorted in the wire-format.

```
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
```

This last vector has an alpn value with an escaped comma and an escaped backslash in two presentation formats.

```
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
```

D.3. Failure cases

In this subsection, example resource records are shown which are not compliant with this document. The various reasons for non-compliance are explained with each example.

This example has multiple instances of the same SvcParamKey Section 2.1.

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

In the next examples the SvcParamKeys are missing their values.

```
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
```

The "no-default-alpn" SvcParamKey value MUST be empty (Section 6.1).

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

In this record a mandatory SvcParam is missing (Section 7).

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

The "mandatory" SvcParamKey MUST not be included in mandatory list (Section 7).

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

Here there are multiple instances of the same SvcParamKey in the mandatory list (Section 7).

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

Appendix E. Change history

* draft-ietf-dnsop-svcb-https-06

  - Add requirements for HTTPS origins that also use Alt-Svc

  - Remove requirement for comma-escaping related to unusual ALPN values

  - Allow resolvers to reject invalid SvcParamValues, with additional guidance

* draft-ietf-dnsop-svcb-https-05

  - Specify interaction with EDNS Client Subnet and Additional section caching

  - Rename "echconfig" to "ech"

  - Add a suite of test vectors (both valid and invalid) and more examples

  - Clarify requirements for resolvers' (non-)use of SvcParams

  - Clarify guidance regarding default ALPN values

* draft-ietf-dnsop-svcb-https-04

  - Simplify the IANA instructions (pure First Come First Served)

  - Recommend against publishing chains of >8 aliases

  - Clarify requirements for using SVCB with a transport proxy

  - Adjust guidance for Port Prefix Naming

  - Minor editorial updates

*draft-ietf-dnsop-svcb-https-03

- Simplified escaping of comma-separated values
- Reorganized client requirements
- Added a warning about port filtering for cross-protocol attacks
- Clarified self-consistency rules for SvcParams
- Added a non-normative mapping summary table for HTTPS

*draft-ietf-dnsop-svcb-https-02

- Added a Privacy Considerations section
- Adjusted resolution fallback description
- Clarified status of SvcParams in AliasMode
- Improved advice on zone structuring and use with Alt-Svc
- Improved examples, including a new Multi-CDN example
- Reorganized text on value-list parsing and SvcPriority
- Improved phrasing and other editorial improvements throughout

*draft-ietf-dnsop-svcb-https-01

- Added a "mandatory" SvcParamKey
- Added the ability to indicate that a service does not exist
- Adjusted resolution and ALPN algorithms
- Major terminology revisions for "origin" and CamelCase names
- Revised ABNF
- Include allocated RR type numbers
- Various corrections, explanations, and recommendations

*draft-ietf-dnsop-svcb-https-00

- Rename HTTPSSVC RR to HTTPS RR
- Rename "an SVCB" to "a SVCB"

- Removed "design considerations and open issues" section and some other "to be removed" text

*draft-ietf-dnsop-svcb-httpssvc-03

- Revised chain length limit requirements

- Revised IANA registry rules for SvcParamKeys

- Require HTTPS clients to implement SNI

- Update terminology for Encrypted ClientHello

- Clarifications: non-default ports, transport proxies, HSTS procedure, WebSocket behavior, wire format, IP hints, inner/outer ClientHello with ECH

- Various textual and ABNF corrections

*draft-ietf-dnsop-svcb-httpssvc-02

- All changes to Alt-Svc have been removed

- Expanded and reorganized examples

- Priority zero is now the definition of AliasForm

- Repeated SvcParamKeys are no longer allowed

- The "=" sign may be omitted in a key=value pair if the value is also empty

- In the wire format, SvcParamKeys must be in sorted order

- New text regarding how to handle resolution timeouts

- Expanded description of recursive resolver behavior

- Much more precise description of the intended ALPN behavior

- Match the HSTS specification's language on HTTPS enforcement

- Removed 'esniconfig=""' mechanism and simplified ESNI connection logic

*draft-ietf-dnsop-svcb-httpssvc-01

- Reduce the emphasis on conversion between HTTPSSVC and Alt-Svc

- Make the "untrusted channel" concept more precise.

- Make SvcFieldPriority = 0 the definition of AliasForm, instead of a requirement.

*draft-ietf-dnsop-svcb-httpssvc-00

- Document an optimization for optimistic pre-connection. (Chris Wood)

- Relax IP hint handling requirements. (Eric Rescorla)

*draft-nygren-dnsop-svcb-httpssvc-00

- Generalize to an SVCB record, with special-case handling for Alt-Svc and HTTPS separated out to dedicated sections.

- Split out a separate HTTPSSVC record for the HTTPS use-case.

- Remove the explicit SvcRecordType=0/1 and instead make the AliasForm vs ServiceForm be implicit. This was based on feedback recommending against subtyping RR type.

- Remove one optimization.

*draft-nygren-httpbis-httpssvc-03

- Change redirect type for HSTS-style behavior from 302 to 307 to reduce ambiguities.

*draft-nygren-httpbis-httpssvc-02

- Remove the redundant length fields from the wire format.

- Define a SvcDomainName of "." for SvcRecordType=1 as being the HTTPSSVC RRNAME.

- Replace "hq" with "h3".

*draft-nygren-httpbis-httpssvc-01

- Fixes of record name. Replace references to "HTTPSVC" with "HTTPSSVC".

*draft-nygren-httpbis-httpssvc-00

- Initial version

Authors' Addresses

Ben Schwartz
Google

Email: bemasc@google.com

Mike Bishop
Akamai Technologies

Email: mbishop@evequefou.be

Erik Nygren
Akamai Technologies

Email: erik+ietf@nygren.org