Workgroup: DNSOP Working Group
Internet-Draft:
draft-ietf-dnsop-svcb-httpssvc-03
Published: 11 June 2020
Intended Status: Standards Track
Expires: 13 December 2020
Authors: B. Schwartz   M. Bishop
Google   Akamai Technologies
E. Nygren
Akamai Technologies

Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)

Abstract

This document specifies the "SVCB" and "HTTPSSVC" DNS resource record types to facilitate the lookup of information needed to make connections for origin resources, such as for HTTPS URLs. SVCB records allow an origin to be served from multiple network locations, 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 HTTPSSVC DNS 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 proposal is inspired by and based on recent DNS usage proposals such as ALTSVC, ANAME, and ESNIKEYS (as well as long standing desires to have SRV or a functional equivalent implemented for HTTP). These proposals each provide an important function but are potentially incompatible with each other, such as when an origin is load-balanced across multiple hosting providers (multi-CDN). Furthermore, these each add potential cases for adding additional record lookups in addition to AAAA/A lookups. This design attempts to provide a unified framework that encompasses the key functionality of these proposals, as well as providing some extensibility for addressing similar future challenges.

TO BE REMOVED: The specific name for this RR type is an open topic for discussion. "SVCB" and "HTTPSSVC" are meant as placeholders as they are easy to replace. Other names might include "B", "SRV2", "SVCHTTPS", "HTTPS", and "ALTSVC".

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 will expire on 13 December 2020.

Copyright Notice

Table of Contents

1. Introduction

The SVCB and HTTPSSVC RRs provide clients with complete instructions for access to an origin. 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 HTTPSSVC RRs also help when the operator of an origin wishes to delegate operational control to one or more other domains, e.g. delegating the origin resource "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 HTTPSSVC RR is then defined as a special case of SVCB that improves efficiency and convenience for use with HTTPS (Section 7) by avoiding the need for an [Attrleaf] label (Section 7.1). Other protocols with similar needs may follow the pattern of HTTPSSVC and assign their own SVCB-compatible RR types.

All behaviors described as applying to the SVCB RR also apply to the HTTPSSVC RR unless explicitly stated otherwise. Section 7 describes additional behaviors specific to the HTTPSSVC record. Apart from Section 7 and introductory examples, much of this document refers only to the SVCB RR, but those references should be taken to apply to SVCB, HTTPSSVC, and any future SVCB-compatible RR types.

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

TO BE REMOVED: If we use this for providing configuration for DNS authorities, it is likely we'd specify a distinct "NS2" RR type that is an instantiation of SVCB for authoritative nameserver delegation and parameter specification, similar to HTTPSSVC.

TO BE REMOVED: Another open question is whether SVCB records should be self-descriptive and include the service name (eg, "https") in the RDATA section to avoid ambiguity. Perhaps this could be included as an svc="baz" parameter for protocols that are not the default for the RR type? Current inclination is to not do so.

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 service endpoints authoritative for the service, along with parameters associated with each of these endpoints.

* Does not assume that all alternative service 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 HTTPSSVC and the HTTPS use-case include:

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

* Obtain the [ECH] keys associated with an alternative service endpoint

* Support non-default TCP and UDP ports

* Address a set of long-standing issues due to HTTP(S) clients not implementing support for SRV records, as well as due to a limitation that a DNS name can not have both CNAME and NS RRs (as is the case for zone apex names)

* Provide an HSTS-like indication signaling for the duration of the DNS RR TTL that the HTTPS scheme should be used instead of HTTP (see Section 7.5).

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

The SVCB RR has two forms: AliasForm, which aliases a name to another name, and ServiceForm, 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. SvcFieldPriority: The priority of this record (relative to others, with lower values preferred). A value of 0 indicates AliasForm. (Described in Section 2.6.2.)

2. SvcDomainName: The domain name of either the alias target (for AliasForm) or the alternative service endpoint (for ServiceForm).

3. SvcFieldValue (optional): A list of key=value pairs describing the alternative service endpoint for the domain name specified in SvcDomainName (only used in ServiceForm and otherwise ignored). Described in Section 2.1.1.

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 must 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 may attach in-bailiwick SVCB, A, AAAA, and CNAME records in the Additional Section to responses for an SVCB query.

When in the ServiceForm, the SvcFieldValue of the SVCB RR provides an extensible data model for describing network endpoints that are authoritative for the origin, along with parameters associated with each of these endpoints.

For the HTTPS use-case, the HTTPSSVC RR enables many of the benefits of [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 8.

1.4. Terminology

For consistency with [AltSvc], we adopt the following definitions:

* An "origin" is an information source as in [RFC6454]. For services other than HTTPS, the exact definition will need to be provided by the document mapping that service onto the SVCB RR.
* The "origin server" is the server that the client would reach when accessing the origin in the absence of the SVCB record or an HTTPS Alt-Svc.
* An "alternative service" is a different server that can serve the origin over a specified protocol.

For example within HTTPS, the origin consists of a scheme (typically "https"), a hostname, and a port (typically "443").

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

SVCB is a contraction of "service binding". HTTPSSVC is a contraction of "HTTPS service". SVCB, HTTPSSVC, 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 ???) is used to locate endpoints that can service an origin. There is special handling for the case of "https" origins.

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

2.1. Presentation format

The presentation format of the record is:

```
Name TTL IN SVCB SvcFieldPriority SvcDomainName SvcFieldValue
```

The SVCB record is defined specifically within the Internet ("IN") Class ([RFC1035]). SvcFieldPriority is a number in the range 0-65535, SvcDomainName is a domain name (absolute or relative), and SvcFieldValue is a set of key=value pairs present for the ServiceForm. Each key SHALL appear at most once in an SvcFieldValue. The SvcFieldValue is empty for the AliasForm.

2.1.1. Presentation format for SvcFieldValue key=value pairs

In ServiceForm, the SvcFieldValue consists of zero or more elements separated by whitespace. Each element represents a key=value pair.

Keys are IANA-registered SvcParamKeys (Section 12.1) with both a case-insensitive string representation and a numeric representation in the range 0-65535. Registered key names should only contain characters from the ranges "a"-"z", "0"-"9", and "-". In ABNF [RFC5234],

```
ALPHA-LC    = %x61-7A   ;  a-z
key         = 1*(ALPHA-LC / DIGIT / "-")
display-key = 1*(ALPHA / DIGIT / "-")
```

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 keys and formats are defined in Section 6.

The presentation format for SvcFieldValue is a whitespace-separated list of key=value pairs. When the value is omitted, or both the value and the "=" are omitted, the presentation value is the empty string.

```
; basic-visible is VCHAR minus DQUOTE, ";", "(", ")", and "\".
basic-visible = %x21 / %x23-27 / %2A-3A / %x3C-5B / %x5D-7E
escaped-char  = "\" (VCHAR / WSP)
contiguous    = 1*(basic-visible / escaped-char)
quoted-string = DQUOTE *(contiguous / WSP) DQUOTE
value         = quoted-string / contiguous
pair          = display-key "=" value
element       = display-key / pair
```

The value format is intended to match the definition of <character-string> in [RFC1035] Section 5.1. (Unlike <character-string>, the length of a value is not limited to 255 characters.)

Unrecognized keys are represented in presentation format as "keyNNNNN" where NNNNN is the numeric value of the key type without leading zeros. In presentation format, values corresponding to unrecognized keys SHALL be represented in wire format, using decimal escape codes (e.g. \255) when necessary.

When decoding values of unrecognized keys in the presentation format:

* a character other than "\" represents its ASCII value in wire format.

* the character "\" followed by three decimal digits, up to 255, represents an octet in the wire format.

* the character "\" followed by any allowed character, except a decimal digit, represents the subsequent character's ASCII value.

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

2.2. SVCB RDATA Wire Format

The RDATA for the SVCB RR consists of:

* a 2 octet field for SvcFieldPriority as an integer in network byte order.

* the uncompressed, fully-qualified SvcDomainName, represented as a sequence of length-prefixed labels as in Section 3.1 of [RFC1035].

* the SvcFieldValue byte string, consuming the remainder of the record (so smaller than 65535 octets and constrained by the RDATA and DNS message sizes).

AliasForm is defined by SvcFieldPriority being 0.

When SvcFieldValue is non-empty (ServiceForm), 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 12.1.2 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 parser reaches the end of the RDATA while parsing an SvcFieldValue.
* 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.

TODO: decide if we want special handling for any SvcParamKey ranges? For example: range for greasing; experimental range; range-of-mandatory-to-use-the-RR vs range of ignore-just-param-if-unknown.

2.3. SVCB owner names

When querying the SVCB RR, an origin is translated into a QNAME by prepending the hostname with a label indicating the scheme, prefixed with an underscore, resulting in a domain name like "_examplescheme.api.example.com.".

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 (or the default if no port number is specified). We term this behavior "Port Prefix Naming", and use it in the examples throughout this document.

See Section 7.1 for the HTTPSSVC behavior.

When a prior CNAME or SVCB record has aliased to an 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, clients MUST continue to validate TLS certificate hostnames based on the origin host.

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" echconfig="..." )
```

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. SvcRecordType

The SvcRecordType is indicated by the SvcFieldPriority, and defines the form of the SVCB RR. When SvcFieldPriority is 0, the SVCB SvcRecordType is defined to be in AliasForm. Otherwise, the SVCB SvcRecordType is defined to be in ServiceForm.

Within an SVCB RRSet, all RRs should have the same SvcRecordType. If an RRSet contains a record in AliasForm, the client MUST ignore any records in the set with ServiceForm.

2.5. SVCB records: AliasForm

When SvcRecordType is AliasForm, the SVCB record is to be treated similar to a CNAME alias pointing to SvcDomainName. SVCB RRSets SHOULD only have a single resource record in this form. If multiple are present, clients or recursive resolvers SHOULD pick one at random.

The AliasForm's primary purpose is to allow aliasing at the zone apex, where CNAME is not allowed. For example, if an operator of https://example.com wanted to point HTTPS requests to a service operating at svc.example.net, they would publish a record such as:

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

In AliasForm, SvcDomainName MUST be the name of a domain that has SVCB, AAAA, or A records. It MUST NOT be equal to the owner name, as this would cause a loop.

Note that the SVCB record's owner name MAY be the canonical name of a CNAME record, and the SvcDomainName 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 AliasForm 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 AliasForm 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.

Note that SVCB AliasForm RRs do not alias to RR types other than address records (AAAA and A), CNAMEs, and ServiceForm SVCB records. For example, an AliasForm SVCB record does not alias to an HTTPSSVC record, nor vice-versa.

2.6. SVCB records: ServiceForm

When SvcRecordType is the ServiceForm, the combination of SvcDomainName and SvcFieldValue parameters within each resource record associates an alternative service location with its connection parameters.

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

2.6.1. Special handling of "." for SvcDomainName in ServiceForm

For ServiceForm SVCB RRs, if SvcDomainName has the value "." (represented in the wire format as a zero-length label), then the owner name of this record MUST be used as the effective SvcDomainName.

For example, in the following example "svc2.example.net" is the effective SvcDomainName:

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

2.6.2. SvcFieldPriority

As RRs within an RRSet are explicitly unordered collections, the SvcFieldPriority value serves to indicate priority. SVCB RRs with a smaller SvcFieldPriority value SHOULD be given preference over RRs with a larger SvcFieldPriority value.

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

3. Client behavior

An SVCB-aware client resolves an origin HOST by attempting to determine the preferred SvcFieldValue and IP addresses for its service, using the following procedure:

1. Issue parallel AAAA/A and SVCB queries for the name HOST. The answers for these may or may not include CNAME pointers before reaching one or more of these records.

2. If an SVCB record of AliasForm SvcRecordType is returned for HOST, clients MUST loop back to step 1 replacing HOST with SvcDomainName, subject to chain length limits and loop detection heuristics (see Section 3.1).

3. If one or more SVCB records of ServiceForm SvcRecordType are returned for HOST, clients should select the highest-priority option with acceptable parameters, and resolve AAAA and/or A records for its SvcDomainName if they are not already available. These are the preferred SvcFieldValue and IP addresses. If the connection fails, the client MAY try to connect using values from a lower-priority record. If none of the options succeed, the client SHOULD connect to the origin server directly.

4. If an SVCB record for HOST does not exist, the received AAAA and/or A records are the preferred IP addresses and there is no SvcFieldValue.

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

When selecting between AAAA and A records to use, clients may use an approach such as [HappyEyeballsV2].

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 an SVCB query results in a SERVFAIL error, transport error, or timeout, and DNS exchanges between the client and the recursive resolver are cryptographically protected (e.g. using TLS [RFC7858] or HTTPS [RFC8484]), the client MUST NOT fall back to non-SVCB connection establishment. This ensures that an active attacker cannot mount a downgrade attack by denying the user access to the SVCB information.

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 MUST NOT fall back to non-SVCB connection establishment if the 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 non-SVCB connection, 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]) SHOULD disable SVCB support if performing SVCB queries would violate the client's privacy intent.

If the client can safely perform SVCB queries (e.g. via the proxy or an affiliated resolver), the client SHOULD follow the standard SVCB resolution process, selecting the highest priority option that is compatible with the client and the proxy. The client SHOULD provide the final SvcDomainName and port to the proxy, which will perform any required A and AAAA lookups.

Providing the proxy with the final SvcDomainName has several benefits:

* It allows the client to use the SvcFieldValue, if present, which is only usable with a specific SvcDomainName. The SvcFieldValue

may include information that enhances performance (e.g. alpn) and privacy (e.g. echconfig).

* It allows the origin 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 an SVCB query, authoritative DNS servers SHOULD return A, AAAA, and SVCB records (as well as any relevant CNAME or [DNAME] records) in the Additional Section for any in-bailiwick SvcDomainNames.

4.2. Recursive resolvers

Recursive resolvers that are aware of SVCB SHOULD ensure that the client can execute the procedure in Section 3 without issuing a second round of queries, by incorporating all the necessary information into a single 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 Answer and Additional RRs to the Additional section, and all Authority RRs to the Authority section, without altering the response code.

Upon receiving an 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 AliasForm, choose an AliasForm record at random, and resolve SVCB, A, and AAAA records for its SvcDomainName.

    * 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 ServiceForm. Resolve A and AAAA queries for each SvcDomainName (or for the owner name if

SvcDomainName 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]).

4.3. General requirements

All DNS servers SHOULD treat the SvcFieldValue portion of the SVCB RR as opaque and SHOULD NOT try to alter their behavior based on its contents.

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).

5. Performance optimizations

For optimal performance (i.e. minimum connection setup time), clients SHOULD issue address (AAAA and/or A) and SVCB queries simultaneously, and SHOULD implement a client-side DNS cache. Responses in the Additional section of an SVCB response SHOULD be placed in cache before performing any followup queries. With these optimizations in place, and conforming DNS servers, using SVCB does not add network latency to connection setup.

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 "echconfig" value of an 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].

An SVCB record is consistent with a connection if the client would attempt an equivalent connection when making use of that record. If an 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 (
    echconfig="111..." ipv6hint=2001:db8::1 port=1234 ... )
                               SVCB 2 svc2.example.net (
    echconfig="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 echconfig="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 must proceed as described in Step 3 of the procedure 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 from higher priority ServiceForm records over lower priority ServiceForm records.

As discussed in Section 3, clients MUST be able to fetch additional information that is required to use an 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.

5.3. Structuring zones for performance

To avoid a delay for clients using a nonconforming recursive resolver, domain owners SHOULD use a single SVCB record whose SvcDomainName is "." if possible. This will ensure that the required address records are already present in the client's DNS cache as part of the responses to the address queries that were issued in parallel.

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.

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 [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*255(OCTET)
```

The presentation value of "alpn" is a comma-separated list of one or more alpn-ids. Any commas present in the protocol-id are escaped by a backslash:

```
escaped-octet = %x00-2b / "\," / %x2d-5b / "\\" / %x5D-FF
escaped-id = 1*(escaped-octet)
alpn-value = escaped-id *("," escaped-id)
```

The wire format value for "alpn" consists of at least one ALPN identifier (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.

Each scheme that uses this SvcParamKey defines a "default set" of supported ALPNs, which SHOULD NOT be empty. To determine the set of protocol suites supported by an endpoint (the "ALPN set"), the client parses the set of ALPN identifiers in the "alpn" parameter, and then adds the default set unless the "no-default-alpn" SvcParamKey is present. The presence of a value in the alpn set indicates that this service endpoint, described by SvcDomainName and the other parameters (e.g. "port") offers service with the protocol suite associated with the ALPN ID.

ALPN IDs that do not uniquely identify a protocol suite (e.g. an ID that can be used with both TLS and DTLS) are not compatible with this SvcParamKey and MUST NOT be included in the ALPN set.

Clients SHOULD NOT attempt connection to a service endpoint whose ALPN set does not contain any compatible protocol suites. 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.

Clients MUST include an application_layer_protocol_negotiation extension in their ClientHello with a ProtocolNameList that includes at least one ID from the ALPN set. Clients SHOULD also include any other values that they support and could negotiate on that connection with equivalent or better security properties. For example, if the ALPN set only contains "http/1.1", the client could include "http/1.1" and "h2" in the ProtocolNameList.

Once the client has formulated the ClientHello, protocol negotiation on that connection proceeds as specified in [ALPN], without regard to the SVCB ALPN set. To preserve the security guarantees of this process, clients MUST consolidate all compatible ALPN IDs into a single ProtocolNameList.

6.2. "port"

The "port" SvcParamKey defines the TCP or UDP port that should be used to contact this alternative service. If this key is not present, clients SHALL use the origin server's port number.

The presentation format of the SvcParamValue is a numeric value between 0 and 65535 inclusive. Any other values (e.g. the empty value) are syntax errors.

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. "echconfig"

The SvcParamKey to enable Encrypted ClientHello (ECH) is "echconfig". Its value is defined in Section 8. It is applicable to most TLS-based protocols.

When publishing a record containing an "echconfig" parameter, the publisher MUST ensure that all IP addresses of SvcDomainName 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 SvcDomainName are locally available, the client SHOULD ignore these hints. Otherwise, clients SHOULD perform A and/or AAAA queries for SvcDomainName 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 SvcDomainName 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 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 [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 an SVCB record. For best performance, server operators SHOULD include an "ipv6hint" parameter whenever they include an "ipv4hint" parameter.

The presentation format for each parameter is a comma-separated list of IP addresses in standard textual format [RFC5952].

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 SvcDomainName is ".", server operators SHOULD NOT include these hints, because they are unlikely to convey any performance benefit.

7. 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 HTTPSSVC RR type is defined as an 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 HTTPSSVC wire format and presentation format are identical to SVCB, and both share the SvcParamKey registry. SVCB semantics apply equally to HTTPSSVC unless specified otherwise.

All the SvcParamKeys defined in Section 6 are permitted for use in HTTPSSVC. The default set of ALPN IDs is the single value "http/1.1".

The presence of an HTTPSSVC record for an origin also indicates that all HTTP resources are available over HTTPS, as discussed in Section 7.5. This allows HTTPSSVC RRs to apply to pre-existing "http" scheme URLs, while ensuring that the client uses a secure and authenticated HTTPS connection.

The HTTPSSVC RR parallels the concepts introduced in the HTTP Alternative Services proposed standard [AltSvc]. Clients and servers that implement HTTPSSVC are NOT REQUIRED to implement Alt-Svc.

7.1. Owner names for HTTPSSVC records

The HTTPSSVC 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 origin hostname, without any prefix labels.

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

Following of HTTPSSVC AliasForm and CNAME aliases is unchanged from SVCB.

Clients always convert "http" URLs to "https" before performing an HTTPSSVC query using the process described in Section 7.5, so domain owners MUST NOT publish HTTPSSVC records 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 host.

7.2. Relationship to Alt-Svc

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

7.2.1. ALPN usage

Unlike Alt-Svc Field Values, HTTPSSVC records can contain multiple ALPN IDs, and clients are encouraged to offer additional ALPNs that they support (subject to security constraints).

TO BE REMOVED: The ALPN semantics in [AltSvc] are ambiguous, and problematic in some interpretations. We should update [AltSvc] to give it well-defined semantics that match HTTPSSVC.

7.2.2. 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.

7.2.3. TTL and granularity

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 will typically be similar to the "ma" value used for Alt-Svc, but may vary 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 HTTPSSVC records 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.

Sending Alt-Svc over HTTP allows the server to tailor the Alt-Svc Field Value specifically to the client. When using an HTTPSSVC DNS record, groups of clients will necessarily receive the same SvcFieldValue. Therefore, HTTPSSVC is not suitable for uses that require single-client granularity.

7.3. Interaction with Alt-Svc

Clients that do not implement support for Encrypted ClientHello MAY skip the HTTPSSVC query if a usable Alt-Svc value is available in the local cache. If Alt-Svc connection fails, these clients SHOULD fall back to the HTTPSSVC client connection procedure (Section 3).

For clients that implement support for ECH, the interaction between HTTPSSVC and Alt-Svc is described in Section 8.1.

This specification does not alter the DNS queries performed when connecting to an Alt-Svc hostname (typically A and/or AAAA only).

7.4. Requiring Server Name Indication

Clients MUST NOT use an HTTPSSVC response unless the client supports TLS Server Name Indication (SNI) and indicate 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 SvcDomainName.

7.5. HTTP Strict Transport Security

By publishing an HTTPSSVC record, the server operator indicates that all useful HTTP resources on that origin are reachable over HTTPS, similar to HTTP Strict Transport Security [HSTS]. When an HTTPSSVC record is present for an origin, all "http" scheme requests for that origin SHOULD logically be redirected to "https".

Prior to making an "http" scheme request, the client SHOULD perform a lookup to determine if any HTTPSSVC records 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 HTTPSSVC query for this "https" URL returns any HTTPSSVC records (AliasForm or ServiceForm), the client SHOULD act as if it has received an HTTP "307 Temporary Redirect" redirect to this "https" URL. Because HTTPSSVC is 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 making an "https" scheme request to an origin with an HTTPSSVC record, either directly or via the above redirect, the client SHOULD terminate the connection if there are any errors with the underlying secure transport, such as errors in certificate validation. This aligns with Section 8.4 and Section 12.1 of [HSTS].

7.6. HTTP-based protocols

We define an "HTTP-based protocol" as one that involves connecting to an "http:" or "https:" URL. When implementing an HTTP-based protocol, clients that use HTTPSSVC for HTTP SHOULD also use it for this URL. For example, clients that support HTTPSSVC and implement the altered [WebSocket] opening handshake from [FETCH] SHOULD use HTTPSSVC for the requestURL.

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

8. SVCB/HTTPSSVC parameter for ECH configuration

The SVCB "echconfig" parameter is defined for conveying the ECH configuration of an alternative service. In wire format, the value of the parameter is an ECHConfigs vector [ECH], including the redundant length prefix. In presentation format, the value is encoded in [base64].

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 origin.

8.1. Client behavior

The general client behavior specified in Section 3 permits clients to retry connection with a less preferred alternative if the preferred option fails, including falling 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 clients SHALL implement the following behavior for connection establishment.

1. Perform connection establishment using HTTPSSVC as described in Section 3, but do not fall back to the origin's A/AAAA records. If all the HTTPSSVC RRs have an "echconfig" key, and they all fail, terminate connection establishment.

2. If the client implements Alt-Svc, try to connect using any entries from the Alt-Svc cache.

3. Fall back to the origin's A/AAAA records if necessary.

As a latency optimization, clients MAY prefetch DNS records for later steps before they are needed.

8.2. Deployment considerations

An HTTPSSVC RRSet containing some RRs with "echconfig" 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 "echconfig" as more preferred (i.e. smaller SvcFieldPriority) than those without, in order to maximize the likelihood that ECH will be used in the absence of an active adversary.

9. Examples

9.1. Protocol enhancements

Consider a simple zone of the form

```
simple.example. 300 IN A    192.0.2.1
                       AAAA 2001:db8::1
```

The domain owner could add this record

```
simple.example. 7200 IN HTTPSSVC 1 . alpn=h3 ...
```

to indicate that simple.example uses HTTPS, and 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).

9.2. 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 HTTPSSVC, the owner of aliased.example could alias the apex by adding one additional record:

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

With this record in place, HTTPSSVC-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-HTTPSSVC-aware clients will just ignore the new record.

Similar to CNAME, HTTPSSVC has 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.

9.3. 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 HTTPSSVC 1 h3pool alpn=h2,h3 echconfig="123..."
              HTTPSSVC 2 .      alpn=h2 echconfig="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 HTTPSSVC or not. If the client does support HTTPSSVC, 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.

9.4. Non-HTTPS uses

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

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

HTTPSSVC records use similar [Attrleaf] labels if the origin contains a non-default port.

10. 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 [RFC7858] or DNS over HTTPS [RFC8484]). 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.

11. Security Considerations

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

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.

12. IANA Considerations

12.1. 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 HTTPSSVC.

ACTION: create and include a reference to this registry.

12.1.1. Procedure

A registration MUST include the following fields:

* Name: Service parameter key name

* SvcParamKey: Service parameter key numeric identifier (range 0-65535)

* Meaning: a short description

* Pointer to specification text

SvcParamKey values to be added to this namespace have different policies ([RFC5226], Section 4.1) based on their range:

SvcParamKey	IANA Policy
0-255	Standards Action
256-32767	Expert Review
32768-65280	First Come First Served
65280-65534	Private Use
65535	Standards Action

Table 1

Apart from the initial contents, the SvcParamKey name MUST NOT start with "key".

12.1.2. Initial contents

The "Service Binding (SVCB) Parameter Registry" shall initially be populated with the registrations below:

SvcParamKey	NAME	Meaning	Reference
0	(no name)	Reserved for internal use	(This document)
1	alpn	Additional supported protocols	(This document)
2	no-default-alpn	No support for default protocol	(This document)

SvcParamKey	NAME	Meaning	Reference
3	port	Port for alternative service	(This document)
4	ipv4hint	IPv4 address hints	(This document)
5	echconfig	Encrypted ClientHello info	(This document)
6	ipv6hint	IPv6 address hints	(This document)
65280-65534	keyNNNNN	Private Use	(This document)
65535	key65535	Reserved	(This document)

Table 2

TODO: do we also want to reserve a range for greasing?

12.2. 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)
HTTPSSVC	HTTPS Service Location and Parameter Binding	(This document)

Table 3

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

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

Table 4

13. 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.draft-bellis-dnsop-http-record-00], [I-D.draft-ietf-dnsop-aname-03], 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, and others for their feedback and suggestions on this draft.

14. References

14.1. Normative References

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

[Attrleaf] Crocker, D., "DNS Scoped Data Through "Underscore" Naming of Attribute Leaves", Work in Progress, Internet-Draft, draft-ietf-dnsop-attrleaf-16, 16 November 2018, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-attrleaf-16.txt>.

[base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

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

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-07, 1 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-esni-07.txt>.

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

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

[HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-20, 23 April 2019, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-20.txt>.

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

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI

10.17487/RFC1928, March 1996, <https://www.rfc-editor.org/info/rfc1928>.

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

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

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

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

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

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

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

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

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

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

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

14.2. Informative References

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

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

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

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

[I-D.draft-ietf-dnsop-aname-03]
Finch, T., Hunt, E., Dijk, P., Eden, A., and W. Mekking, "Address-specific DNS aliases (ANAME)", Work in Progress, Internet-Draft, draft-ietf-dnsop-aname-03, 15 April 2019, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-aname-03.txt>.

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

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <https://www.rfc-editor.org/info/rfc6895>.

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

Appendix A. Comparison with alternatives

The SVCB and HTTPSSVC record 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.

A.1. Differences from the SRV RR type

An SRV record [RFC2782] 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 HTTPSSVC can be extended with new parameters.

A.2. Differences from the proposed HTTP record

Unlike [I-D.draft-bellis-dnsop-http-record-00], 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.

A.3. Differences from the proposed ANAME record

Unlike [I-D.draft-ietf-dnsop-aname-03], 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 master 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.

A.4. Comparison with separate RR types for AliasForm and ServiceForm

Abstractly, functions of AliasForm and ServiceForm 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 ServiceForm query, emit both a ServiceForm and an AliasForm 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 B. Design Considerations and Open Issues

This draft is intended to be a work-in-progress for discussion. Many details are expected to change with subsequent refinement. Some known issues or topics for discussion are listed below.

B.1. Record Name

Naming is hard. "SVCB" and "HTTPSSVC" are proposed as placeholders that are easy to search for and replace when a final name is chosen. Other names for this record might include B, ALTSVC, HTTPS, HTTPSSRV, HTTPSSVC, SVCHTTPS, or something else.

B.2. Generality

The SVCB record was designed as a generalization of HTTPSSVC, based on feedback requesting a solution that applied to protocols other than HTTP. Past efforts to over-generalize have not met with broad success, but we hope that HTTPSSVC and SVCB have struck an acceptable balance between generality and focus.

B.3. Wire Format

Advice from experts in DNS wire format best practices would be greatly appreciated to refine the proposed details, overall.

B.4. Whether to include Weight

Some other similar mechanisms such as SRV have a weight in addition to priority. That is excluded here for simplicity. It could always be added as an optional SVCB parameter.

Appendix C. Change history

- 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