Network Working Group
Internet-Draft
Updates: RFC 3658, RFC 5155, RFC 6014
(if approved)
Intended status: Standards Track
Expires: January 7, 2021

P. Hoffman
ICANN

July 06, 2020

Revised IANA Considerations for DNSSEC
draft-hoffman-dnssec-iana-cons-01

Abstract

This document changes the review requirements needed to get some DNSSEC algorithms and resource records added to IANA registries. It updates RFC 6014 to include hash algorithms for DS records and NSEC3 parameters.

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 January 7, 2021.

Table of Contents

1. Introduction

DNSSEC is primarily described in [RFC4033], [RFC4034], and [RFC4035]. DNSSEC commonly uses two resource records beyond those defined in RFC 4034: DS [RFC3658] and NSEC3 [RFC5155].

[RFC8126] describes the requirements for listing in the myriad IANA registries.

[RFC6014] updated the requirements for how DNSSEC cryptographic algorithm identifiers in the IANA registries are allocated, reducing the requirements from being "Standards Action" to "RFC Required". However, the IANA registry requirements for hash algorithms for DS records and for the hash algorithms used in NSEC3 are still "Standards Action". This document updates RFC 6014 to bring the requirements for DS records and NSEC3 hash algorithms in line with the rest of the DNSSEC cryptographic algorithms.

2. IANA Considerations

In the "Domain Name System Security (DNSSEC) NextSECure3 (NSEC3) Parameters" registry, the registration procedure for "DNSSEC NSEC3 Flags", "DNSSEC NSEC3 Hash Algorithms", and "DNSSEC NSEC3PARAM Flags" are changed from "Standards Action" to "RFC Required".

In the "Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms" registry, the registration procedure for "Digest Algorithms" is changed from "Standards Action" to "RFC Required".

3. Security Considerations

Changing the requirements for getting security algorithms added to IANA registries as described in this document will make it easier to get good algorithms added to the registries, and will make it easier to get bad algorithms added to the registries. It is impossible to weigh the security impact of those two changes.

4. Normative References

[RFC3658] Gudmundsson, O., "Delegation Signer (DS) Resource Record (RR)", RFC 3658, DOI 10.17487/RFC3658, December 2003, <https://www.rfc-editor.org/info/rfc3658>.

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

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

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

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC6014] Hoffman, P., "Cryptographic Algorithm Identifier Allocation for DNSSEC", RFC 6014, DOI 10.17487/RFC6014, November 2010, <https://www.rfc-editor.org/info/rfc6014>.

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

Appendix A. Other Options for Requirements Level

During an early discussion in the DNSOP Working Group, it was proposed that the requirements for registry allocation for DS resource records be "Specification Required". This would reduce the work required of specification authors, and of the RFC Editor, while still requiring review by an expert reviewer and a long-lived specification.

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 19, 2021

K. Fujiwara
JPRS
P. Vixie
Farsight

September 15, 2020

Fragmentation Avoidance in DNS
draft-ietf-dnsop-avoid-fragmentation-02

Abstract

EDNS0 enables a DNS server to send large responses using UDP and is widely deployed. Path MTU discovery remains widely undeployed due to security issues, and IP fragmentation has exposed weaknesses in application protocols. Currently, DNS is known to be the largest user of IP fragmentation. It is possible to avoid IP fragmentation in DNS by limiting response size where possible, and signaling the need to upgrade from UDP to TCP transport where necessary. This document proposes to avoid IP fragmentation in DNS.

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 March 19, 2021.

Table of Contents

1. Introduction

DNS has EDNS0 [RFC6891] mechanism. It enables a DNS server to send large responses using UDP. EDNS0 is now widely deployed, and DNS (over UDP) is said to be the biggest user of IP fragmentation.

However, "Fragmentation Considered Poisonous" [Herzberg2013] proposed effective off-path DNS cache poisoning attack vectors using IP fragmentation. "IP fragmentation attack on DNS" [Hlavacek2013] and "Domain Validation++ For MitM-Resilient PKI" [Brandt2018] proposed that off-path attackers can intervene in path MTU discovery [RFC1191] to perform intentionally fragmented responses from authoritative servers. [RFC7739] stated the security implications of predictable fragment identification values.

DNSSEC is a countermeasure against cache poisoning attacks that use IP fragmentation. However, DNS delegation responses are not signed with DNSSEC, and DNSSEC does not have a mechanism to get the correct response if an incorrect delegation is injected. This is a denial-of-service vulnerability that can yield failed name resolutions. If

cache poisoning attacks can be avoided, DNSSEC validation failures will be avoided.

In Section 3.2 (Message Side Guidelines) of UDP Usage Guidelines [RFC8085] we are told that an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination.

A DNS message receiver cannot trust fragmented UDP datagrams primarily due to the small amount of entropy provided by UDP port numbers and DNS message identifiers, each of which being only 16 bits in size, and both likely being in the first fragment of a packet, if fragmentation occurs. By comparison, TCP protocol stack controls packet size and avoid IP fragmentation under ICMP NEEDFRAG attacks. In TCP, fragmentation should be avoided for performance reasons, whereas for UDP, fragmentation should be avoided for resiliency and authenticity reasons.

[RFC8900] summarized that IP fragmentation introduces fragility to Internet communication. The transport of DNS messages over UDP should take account of the observations stated in that document.

TCP avoids fragmentation using its Maximum Segment Size (MSS) parameter, but each transmitted segment is header-size aware such that the size of the IP and TCP headers is known, as well as the far end's MSS parameter and the interface or path MTU, so that the segment size can be chosen so as to keep the each IP datagram below a target size. This takes advantage of the elasticity of TCP's packetizing process as to how much queued data will fit into the next segment. In contrast, DNS over UDP has little datagram size elasticity and lacks insight into IP header and option size, and so must make more conservative estimates about available UDP payload space.

This document proposes to avoid IP fragmentation in DNS/UDP.

2. Terminology

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

"Requestor" refers to the side that sends a request. "Responder" refers to an authoritative, recursive resolver or other DNS component that responds to questions. (Quoted from EDNS0 [RFC6891])

"Path MTU" is the minimum link MTU of all the links in a path between a source node and a destination node. (Quoted from [RFC8201])

"Path MTU discovery" is defined by [RFC1191], [RFC8201] and [RFC8899].

Many of the specialized terms used in this document are defined in DNS Terminology [RFC8499].

3. Proposal to avoid IP fragmentation in DNS

The methods to avoid IP fragmentation in DNS are described below:

3.1. Recommendations for UDP requestors

- UDP requestors SHOULD send DNS responses with IP_DONTFRAG / IPV6_DONTFRAG [RFC3542] options.

- UDP requestors MAY probe to discover the real MTU value per destination. If the path MTU discovery failed or is impossible, use the default path MTU described in Section 4.

- UDP reqoestors SHOULD use the requestor’s payload size to limit the path MTU value minus the IP header length and UDP header length. Of course, as in the conventional case, a specified value (1220 or 1232) as the requestor’s payload size may be used.

- UDP requestors MAY drop fragmented DNS/UDP responses without IP reassembly to avoid cache poisoning attacks.

- DNS responses may be dropped by IP fragmentation. Upon a timeout, UDP requestors may retry using TCP or UDP, per local policy.

3.2. Recommendations for UDP responders

- UDP responders SHOULD send DNS responses with IP_DONTFRAG / IPV6_DONTFRAG [RFC3542] options.

- UDP responders MAY probe to discover the real MTU value per destination. If the path MTU discovery failed or is impossible, use the default path MTU described in Section 4.

- UDP responders SHOULD compose UDP responses that result in IP packets that do not exceed the path MTU to the requestor. Of course, as in the conventional case, a specified value (1220 or 1232) as the DNS packet size limit may be used.

The cause and effect of the TC bit is unchanged from EDNS0 [RFC6891].

4. Maximum DNS/UDP payload size

Default path MTU value for IPv6 is XXXX. Default path MTU value for IPv4 is XXXX.

Discussions under here will be deleted when the discussion is over. There are many discussions for default path MTU values.

- The minimum MTU for an IPv6 interface is 1280 octets (see Section 5 of [RFC8200]). Then, we can use it as default path MTU value for IPv6.

- Most of the Internet and especially the inner core has an MTU of at least 1500 octets. An operator of a full resolver would be well advised to measure their path MTU to several authority name servers and to a random sample of their expected stub resolver client networks, to find the upper boundary on IP/UDP packet size in the average case. This limit should not be exceeded by most messages received or transmitted by a full resolver, or else fallback to TCP will occur too often. An operator of authoritative servers would also be well advised to measure their path MTU to several full-service resolvers. The Linux tool "tracepath" can be used to measure the path MTU to well known authority name servers such as [a-m].root-servers.net or [a-m].gtld-servers.net. If the reported path MTU is for example no smaller than 1460, then the maximum DNS/UDP payload would be 1432 for IP4 (which is 1460 - IP4 header(20) - UDP header(8)) and 1412 for IP6 (which is 1460 - IP6 header(40) - UDP header(8)). To allow for possible IP options and distant tunnel overhead, a useful default for maximum DNS/UDP payload size would be 1400.

- [RFC4035] defines that "A security-aware name server MUST support the EDNS0 message size extension, MUST support a message size of at least 1220 octets". Then, the smallest number of the maximum DNS/UDP payload size is 1220.

- DNS flag day 2020 proposed 1232 as an EDNS buffer size. [DNSFlagDay2020] By the above reasoning, this proposal is either too small or too large.

  It is considered that these arguments are diverted from IPv6 values because most IPv4 links have path MTU values larger than or equal to the minimum MTU value of IPv6.

5. Incremental deployment

The proposed method supports incremental deployment.

When a full-service resolver implements the proposed method, its stub resolvers (clients) and the authority server network will no longer observe IP fragmentation or reassembly from that server, and will fall back to TCP when necessary.

When an authoritative server implements the proposed method, its full service resolvers (clients) will no longer observe IP fragmentation or reassembly from that server, and will fall back to TCP when necessary.

6. Request to zone operators and DNS server operators

Large DNS responses are the result of zone configuration. Zone operators SHOULD seek configurations resulting in small responses. For example,

- Use smaller number of name servers (13 may be too large)
- Use smaller number of A/AAAA RRs for a domain name
- Use 'minimal-responses' configuration: Some implementations have 'minimal responses' configuration that causes DNS servers to make response packets smaller, containing only mandatory and required data (Appendix B).
- Use smaller signature / public key size algorithm for DNSSEC. Notably, the signature size of ECDSA or EdDSA is smaller than RSA.

7. Considerations

7.1. Protocol compliance

In prior research ([Fujiwara2018] and dns-operations mailing list discussions), there are some authoritative servers that ignore EDNS0 requestor's UDP payload size, and return large UDP responses.

It is also well known that there are some authoritative servers that do not support TCP transport.

Such non-compliant behavior cannot become implementation or configuration constraints for the rest of the DNS. If failure is the result, then that failure must be localized to the non-compliant servers.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

10. Acknowledgments

The author would like to specifically thank Paul Wouters, Mukund Sivaraman and Tony Finch for extensive review and comments.

11. References

11.1. Normative References

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[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>.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, DOI 10.17487/RFC3542, May 2003, <https://www.rfc-editor.org/info/rfc3542>.

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

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

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

[RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <https://www.rfc-editor.org/info/rfc7739>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8499] 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>.

[RFC8899] Fairhurst, G., Jones, T., Tuexen, M., Ruengeler, I., and T. Voelker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, B., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

11.2. Informative References

[Brandt2018] Brandt, M., Dai, T., Klein, A., Shulman, H., and M. Waidner, "Domain Validation++ For MitM-Resilient PKI", Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security , 2018.

[DNSFlagDay2020] "DNS flag day 2020", n.d., <https://dnsflagday.net/2020/>.

[Fujiwara2018] Fujiwara, K., "Measures against cache poisoning attacks using IP fragmentation in DNS", OARC 30 Workshop , 2019.

[Herzberg2013]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", IEEE Conference on Communications and Network Security , 2013.

[Hlavacek2013]
Hlavacek, T., "IP fragmentation attack on DNS", RIPE 67 Meeting , 2013, <https://ripe67.ripe.net/presentations/240-ipfragattack.pdf>.

Appendix A. How to retrieve path MTU value to a destination from applications

Socket options: "IP_MTU (since Linux 2.2) Retrieve the current known path MTU of the current socket. Valid only when the socket has been connected. Returns an integer. Only valid as a getsockopt(2)." (Quoted from Debian GNU Linux manual: ip(7))

"IPV6_MTU getsockopt(): Retrieve the current known path MTU of the current socket. Only valid when the socket has been connected. Returns an integer." (Quoted from Debian GNU Linux manual: ipv6(7))

Appendix B. Minimal-responses

Some implementations have 'minimal responses' configuration that causes a DNS server to make response packets smaller, containing only mandatory and required data.

Under the minimal-responses configuration, DNS servers compose response messages using only RRSets corresponding to queries. In case of delegation, DNS servers compose response packets with delegation NS RRSet in authority section and in-domain (in-zone and below-zone) glue in the additional data section. In case of non-existent domain name or non-existent type, the start of authority (SOA RR) will be placed in the Authority Section.

In addition, if the zone is DNSSEC signed and a query has the DNSSEC OK bit, signatures are added in answer section, or the corresponding DS RRSet and signatures are added in authority section. Details are defined in [RFC4035] and [RFC5155].

Authors' Addresses

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Paul Vixie
Farsight Security Inc
177 Bovet Road, Suite 180
San Mateo, CA 94402
United States of America

Phone: +1 650 393 3994
Email: vixie@fsi.io

DNSOP
Internet-Draft
Updates: 4035 (if approved)
Intended status: Informational
Expires: January 14, 2021

P. Wouters
Red Hat
W. Hardaker
USC/ISI

July 13, 2020

The DELEGATION_ONLY DNSKEY flag
draft-ietf-dnsop-delegation-only-01

Abstract

This document introduces a new DNSKEY flag called DELEGATION_ONLY that indicates that this zone will only produce delegation responses for data outside of its own apex (or _underscore labels). That is, the Answer Section for queries that do not involve the apex (or _underscore labels) of the zone is empty, and only glue records in the Authority Section and Additional Section will be acceptable data in the response message. Additionally, it indicates that it is not expected that the parent of this delegation-only zone bypasses or removes the delegation to this zone. DNSSEC Validating Resolvers can use this flag to mark any data that violates the DELEGATION_ONLY policy as Bogus.

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 January 14, 2021.

Copyright Notice

Table of Contents

1. Introduction

The DNS Security Extensions [DNSSEC] use public key cryptography to create an hierarchical trust base with the DNSSEC root public keys at the top, followed by Top Level domain (TLD) keys one level underneath. While the root and most TLD zones are assumed to be exclusively delegation-only zones, there is currently no interoperable and automatable mechanism for auditing these zones to ensure they behave as a delegation-only zone. This creates a target for malicious use of these zones - either by their owners or through coercion.

This document defines a mechanism for delegation-only zone owners to create a DNSKEY that indicates it will only delegate the remainder of the DNS tree to lower-level zones. This allows for easier delegation policy verification and logging and auditing of DNS responses served by their infrastructure.

Specifically, this document introduces a new DNSKEY flag allowing zone owners to commit to only signing records relating to delegation. If a DNSSEC validator discovers any non-delegation zone data signed by a flagged key, this data can be interpreted as Bogus.

2. Terminology

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

3. The Deep Signing problem

The hierarchical model of DNS and DNSSEC ([RFC1034], [RFC1035], [RFC4033], [RFC4034] and [RFC4035]) comes with the property that a zone at one point in the hierarchy can define, and therefor override, everything below it in the DNS tree. And this is possible to achieve on a per-client basis.

For example, the owner of the DNSSEC root key could generate a specially crafted zone file that ignores the intended NS records for ".org" and "example.org". It could place an AAAA record for "www.example.org" directly into the specially crafted root zone, with a corresponding RRSIG signed by the root DNSKEY itself. Validating resolvers would find this record perfectly acceptable, as it was signed by a key within the proper chain of trust (in this case, a root DNSKEY). This specially crafted zone could then even be served to specific clients in an effort to subvert a portion of the DNS ecosystem for a portion of the Internet.

Similarly, the TLD "example" could circumvent company.example for certain clients. It is important to note that this can be done without changing the upstream DS record or trust anchor -- the DNSKEY is (again) already in the trust path and is merely signing deeper DNS records than the zone owner’s clients may have expected it to sign.

It is important to note that this "feature" has always been present. Since the creation of the DNS, it has always been possible to serve "split zones". Specifically, it is not a problem created by DNSSEC -- DNSSEC was not designed to protect against this use case.

Exposing such targeted attacks requires a transparency audit infrastructure similar to what is deployed for PKIX ([RFC6962]). However, a DNSSEC version would need to log significantly more data, as all signed DNS data used by a DNSKEY must be recorded in order to prove that data signed by a parent zone’s DNSKEY was out of expected

policy. The very distributed nature of the DNS combined with the typically frequent zone resigning rate makes such transparency logs prohibitively expensive and nearly impossible to operate.

Additionally, it would require zone owners to expose all their zone data to any public log operators, thereby introducing privacy implications and exposing all relevant DNS data to a public archive. This may be acceptable for some domains, such as the root, where DNS data is already considered public. However, other delegation domains have legal implications that prohibit them from participating in such a system.

Furthermore, there is no signalling mechanism that lets validating resolvers know which zones are supposedly delegation-only, and what zones can be logged. Today there are over 1500 TLDs in the root zone, some of which may be considered delegation-only while others may not be. At the time of this writing, the list of entries in the public suffix list contains over 8800 entries as well, with 73 wild-card entries (prefixed with a "*") indicating that all of their (unknown number of) children are public registration points. In the absence of an interoperable mechanism (like this draft provides), it is infeasible that a validating resolver or auditing log could know which of these zones are delegation-only without individual policy statements from each of them. [todo: xref psl]

3.1. Affected parties and their roles

Upon deployment of this specification, the following parties would be potentially benefit or be affected by:

Authoritative parent: If (and only if) an authoritative parent is a "delegation only" zone, it could generate a DNSKEY with the DELEGATION_ONLY flag set, indicating a verifiable promise to the world that will not sign any records other than delegation records.

Authoritative Child / Delegated Zone: child zones existing underneath a marked delegation-only zone get the added benefit of knowing their parent will not (and cannot) sign DNS records within the child's portion of the DNS tree using the marked DNSKEY.

Validating Resolver: A validating resolver that supports verifying the DELEGATION_ONLY flag is capable of rejecting or discarding any data from an authoritative parent that incorrectly signs non-delegation records too low in the DNS tree. If the validating resolver supports a (future-defined) DNSSEC transparency audit log as well, it may submit the appropriate data to a DNSSEC transparency log that appropriately tracks DNSSEC signatures.

DNSSEC Transparency Log (optional future): a DNSSEC transparency log would create a non-modifiable trace of log entries submitted to it, for public verification, similar to [RFC6962]. What it chooses to accept into its log might be only certain zone data, or any zone with a marked DNSKEY.

Note that without a DNSSEC Log, the DELEGATION_ONLY flag is still useful per the discussion in the Validating Resolvers role: the resolver will reject incorrectly signed, non-delegation data. However, malicious parent zones are still capable of creating two (or more) DNSKEYs, one with the DELEGATION_ONLY flag and one without. However, they would also have to publish those DS records as well, which is detectable by DNSSEC monitoring platforms, regardless of the existence of a DNSSEC Transparency Log. Any zone with multiple DS records that link to both a DELEGATION_ONLY marked and an unmarked DNSKEY would be considered suspicious (or at least in transition). Circumventing this through obfuscation would require the collusion of their parent as well. Finally, a DELEGATION_ONLY flagged DNSKEY for the root zone cannot be overridden easily, as it would require a trust anchor update in all validating resolvers.

4. The DELEGATION_ONLY DNSKEY flag

This document introduces a new DNSKEY flag called DELEGATION_ONLY. When this flag is set on a DNSKEY with its Secure Entry Point (SEP) flag set, the zone commits to only produce Authoritative Answers for the apex (and _underscore label) records. Note that DS records and its DNSSEC signatures are still allowed as this data is authoritative at the parent, not the child. This commits a parent in the DNS hierarchy to only publish signed DS records and unsigned glue records (NS and A/AAAA) for its child zones. It will no longer be able to ignore (or briefly delete, see below) a child delegation and publish authoritative data in its place.

For such a parent to take over data that belongs to its child zone, it has two choices. It can (temporarily) remove its own DNSKEY DELEGATION_ONLY flag or it can replace the NS and DS records of its child zone with its own data (destinations and key references) so it can sign DNS data that belongs to its own child zone. However, both of these actions cannot be hidden, thus exposing such malicious behaviour when combined with DNSSEC Transparency logs.

A zone that publishes a DNSKEY with the DELEGATION_ONLY flag also signifies that it is not expecting its own parent to skip it, thereby bypassing its DELEGATION_ONLY flag.

5. _underscore label exception

Some protocols, such as the DANE protocol [RFC6698] use a number of labels that start with an underscore (_) prefix to publish information about the zone itself. For example, the TLSA record for www.example.com is published at the location _443._tcp.www.example.com. These records are semantically part of the zone itself and are not delegated child zones. Any chain of labels that each start with an underscore (_) is not considered to violate the DELEGATION_ONLY flag limitation of being DELEGATION_ONLY, as this data is logically part of the zone itself and is never meant to be interpreted as an independent delegated child zone.

6. Parental Transparency

A parent zone, such as the root zone, a TLD or any public suffix list delegation point, that has published a key with the DELEGATION_ONLY flag can no longer make an exception for a single delegated zone without removing the DELEGATION_ONLY flag, switching off its published policy. This action would be highly visible, and for some domains such as the root or TLDs, require human interaction to notify the stake holders to prevent loss of trust.

Removing the DELEGATION_ONLY flag from a DNSKEY requires that the zone first publishes an additional updated DS record to its parent.

In the case of the root key, it would require updating out-of-band root key meta information and/or perform an [RFC5011] style rollover for the same key with updated DNSKEY flags. Due to the timings of such a rollover, it would take at least 30 days for the first validating resolvers to pick up this policy change. It would also be a highly visible event.

Replacing the NS and DS records of a child zone can still be done in a targeted attack mode, but these events are something that can be easily tracked by a transparency infrastructure similar to what is now in use for the WebPKI using [RFC6962](bis). With client implementations of transparency, all DELEGATION_ONLY flag changes would be logged and become visible to the owner of attacked child zones, exposing a parent’s malicious behaviour.

7. Marking zone keys DELEGATION_ONLY

Even before a parent DNSKEY (or the root key) has set the DELEGATION_ONLY flag, zones can already signal their own willingness to commit to being DELEGATION_ONLY zones. Any changes of that state in a zone DNSKEY will require those zones to submit a new DS record to their parent. Setting the DELEGATION_ONLY flag would trigger

DNSSEC Transparency clients to start monitoring for actions by the zone or its parents that would be bypassing the DELEGATION_ONLY policy of the zone. Validating resolvers would mark any data in violation of the DELEGATION_ONLY policy as Bogus.

7.1. Marking the Root DNSKEY DELEGATION_ONLY

Once the Root DNSKEY is marked with a DELEGATION_ONLY flag and deployed resolvers are configured with the new DNSKEY, all TLDs will be ensured that the Root DNSKEY can no longer be abused to override child zone data. Until the Root KSK DNSKEY sets this flag, software SHOULD imply this flag is always set, as this is the current expectation of the Root Zone.

7.2. Migrating to and from DELEGATION_ONLY

There might be multiple DNSKEYs with the SEP flag set in a zone. For the purpose of declaring a zone as DELEGATION_ONLY, only those DNSKEY's that have a corresponding DS record at the parent MUST be considered. If multiple DS records appear at the parent, some of which point to DNSKEYs with and some of which point to DNSKEYs without the DELEGATION_ONLY flag set, the zone MUST be considered DELEGATION_ONLY. This situation will occur when a zone is rolling its DNSKEY key at the same time as it is committing to a DELEGATION_ONLY zone (or the reverse).

8. Operational Considerations

Setting or unsetting the DELEGATION_ONLY flag must be handled like any other Key Signing Key rollover procedure, with the appropriate wait times to give resolvers the chance to update their caches.

Some TLDs offer a service where small domains can be hosted in-zone at the TLD zone itself. In that case, the TLD MUST NOT set the DELEGATION_ONLY flag. Another solution for such TLDs is to create delegations for these child zones with the same or different DNSKEY as used in the parent zone itself.

If a zone is publishing glue records for a number of zones, and the zone that contains the authoritative records for this glue is deleted, a resigning of the zone will make this orphaned glue authoritative within the zone. However, with the DELEGATION_ONLY flag set, this (signed) DNSSEC data will be considered Bogus as it violations the commitment to only delegate. This may impact domains that depended on this unsigned glue. Note that glue handling differs per zone. Some TLDs already remove the glue records if no authoritative child is left in its zone that matches these glue records.

For example, if "example.com" and "example.net" use NS records pointing to "ns.example.net", then if "example.net" is deleted from the ".net" zone, and the previously unsigned glue of "ns.example.net" is now signed by the ".net" zone, the "example.com" zone will lose its NS records and fail to resolve.

The use of Empty Non Terminals (ENT) is fine, as long as the ENT's ends in a proper delegation with NS (and hopefully DS) records.

Some TLDs publish their nameserver (NS) records straight within their TLD (eg "ns1.example") which makes these names indistinguishable from real delegations with respect to the DELEGATION_ONLY flag. These NS entries would have to be moved to their own delegation zone (eg "ns1.nic.example") which in itself cannot be a DELEGATION_ONLY zone.

Some TLDs have a requirement for certain Fully Qualified Domain Names (FQDN) within their TLD, such as "whois.example" or "nic.example". These usually appear as signed data of the TLD and not as a delegated child zone. These names would have to be converted to delegated zones before enabling the DELEGATION_ONLY flag.

The bind DNS software has an option called "delegation_only zones" which is an option that means something completely different. It refers to ignoring wildcard records in specified zones that are deemed delegation-only zones.

9. Security Considerations

Some parental attacks cannot be detected when the validating resolver's cache is empty. Care should be taken by resolvers to not unnecessarily empty their cache. This is specifically important for roaming clients that re-connect frequently to different wireless or mobile data networks.

Resolvers should be aware of DELEGATION_ONLY status of a parental domain and not consume Authoritative or Additional sections with data that is placed to attempt to bypass the DELEGATION_ONLY restriction. If "example.org" is a DELEGATION_ONLY zone, and a query for "www.example.org" results in non-authoritative data for A records for "www.example.org" or "mail.example.org", these records should be rejected as Bogus, irrespective of whether these were signed by the appropriate "example.org" DNSSEC key.

This DELEGATION_ONLY mechanism is not designed to completely foil attacks (since parent's can simply change a child's referral data), but rather to empower transparency logging mechanisms.

10. Privacy Considerations

Some of the protection offered by the DELEGATION_ONLY flag is only available when DNS resolvers report changes in the signing depth of high level (root or TLD) DNSKEYs to gain DNSSEC Transparency. This reporting can reveal that a particular node is trying to access a certain DNS name. Defensive measures to prevent exposing users should be taken when implementing DNSSEC Transparency. It is expected that DNSSEC Transparency behaviour will be written up in a separate document.

11. Human Rights Considerations

The DNS protocol's hierarchy limits zones authority to themselves and their child zones only. While this provides a finer grained trust model compared to a simple list of trusted entities, such as used in the WebPKI, it consolidates a lot of power in the top of the DNS hierarchy. With the increased reliance on DNSSEC for securely identifying resources, such as DANE records, it becomes very important to audit those entities high up in the hierarchy to not abuse or be co-erced into abusing control of the independent child zones. The protocol extension specifically aims at increasing parental transparency and blocks some parental attacks from those parents who have publicly claimed to never override their child zone data.

Parents using the DELEGATION_ONLY flag publication to increase their public trust are still able to remove child zones from their zone, for example in cases of legal compliance or to prevent malicious activity happening in its child zone. But these parents can only do so publicly and can no longer surreptitiously take control of their own child zones.

12. IANA Considerations

This document defines a new DNSKEY flag, the DELEGATION_ONLY flag, whose value [TBD] has been allocated by IANA from the DNSKEY FLAGS Registry.

13. Acknowledgements

The authors wishes to thank Thomas H. Ptacek for his insistence on this matter.

Thanks to the following IETF participants: Viktor Dukhovni, Shumon Huque, Geoff Huston, Rick Lamb Sam Weiler and Paul Vixie.

14. References

14.1. Normative References

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

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

[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>.

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

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <https://www.rfc-editor.org/info/rfc5011>.

[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>.

14.2. Informative References

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

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

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

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

Authors' Addresses

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

B. Schwartz
Google
M. Bishop
E. Nygren
Akamai Technologies

July 13, 2020

Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)
draft-ietf-dnsop-svcb-https-01

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 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: This document is being collaborated on in Github at: https://github.com/MikeBishop/dns-alt-svc [1]. 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.

This Internet-Draft will expire on January 14, 2021.

Table of Contents

1. Introduction

The SVCB 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 7) by avoiding the need for an Attrleaf label [Attrleaf] (Section 7.1). 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 7 describes additional behaviors specific to the HTTPS RR. 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, 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.

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 HTTPS. See [I-D.tapril-ns2] as one example.

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 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 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 2.4.3.)

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 8.

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 7.

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, 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 12.3).

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 definition of each SvcParamKey indicates that its SvcParamValue is empty, single-valued, or multi-valued. To parse a single-valued SvcParam, the parser applies the character-string decoding algorithm (Appendix A), producing a "value", and then performs key-specific processing to validate the input and produce the wire-format encoding. To parse a multi-valued SvcParam, the parser applies the value-list decoding algorithm to the "char-string" (Appendix A.1), splitting on unescaped commas to produce a list of zero or more values.

When the "=" is omitted, the "value" or value list 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. SvcParams in this form are always treated as single-valued, and the decoded "value" SHALL be used as its wire format encoding.

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 12.3.2 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 parser reaches the end of the RDATA while parsing the SvcParams.
- 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 owner 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.".

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 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" 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. Modes

When SvcPriority is 0 the SVCB record is in AliasMode. Otherwise, it is in ServiceMode.

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.

2.4.1. 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, TargetName 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.

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.

2.4.2. 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.

2.4.3. SvcPriority

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.

2.5. 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 echconfig="..."
svc2.example.net. 300   IN A     192.0.2.2
svc2.example.net. 300   IN AAAA  2001:db8::2
```

3. Client behavior

A SVCB-aware client selects an endpoint for a service using the following procedure:

1. Let $ADDR_QNAME be the service name. Let $SVCB_QNAME be the service name plus appropriate prefixes for the scheme (see Section 2.3).

2. In parallel, issue AAAA/A queries for $ADDR_QNAME and a SVCB query for $SVCB_QNAME. The answers for these may or may not include CNAME pointers before reaching one or more of these records.

3. If an AliasMode SVCB record is returned for $SVCB_QNAME, clients MUST set $ADDR_QNAME and $SVCB_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 ServiceMode records are returned for $SVCB_QNAME, clients SHOULD select the highest-priority compatible record with acceptable parameters. This TargetName and SvcParams represent the preferred endpoint. If connection to this endpoint fails, the client MAY try to connect using values from a lower-priority record. If all attempts fail, clients SHOULD go to step 5.

5. Try to use the endpoint consisting of $ADDR_QNAME, the authority endpoint's port number, and no SvcParams.

6. If all of the above connection attempts fail, clients MAY connect to the authority endpoint.

This procedure does not rely on any recursive or authoritative DNS 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 Happy Eyeballs [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 a 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 [DoT] or HTTPS [DoH]), the client SHOULD NOT fall back to $ADDR_QNAME or the authority endpoint (steps 5 and 6 above). 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 NOT fall back to steps 5 or 6 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 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 queries though a separate resolver. This might disclose the client's destinations to an additional party, creating privacy concerns. If these concerns apply, the client SHOULD disable SVCB resolution.

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. The client SHOULD 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. echconfig).

- 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 in-bailiwick TargetNames. If the zone is signed, the server SHOULD also include positive or negative DNSSEC responses for these records in the Additional section.

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 5.2 for possible optimizations of this procedure.

4.3. General requirements

Recursive resolvers SHOULD treat the SvcParams 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 a 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 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].

An 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. (
    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 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.

5.3. 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 choose TargetName to be a domain for which the client will have already issued address queries (see Section 3). For foo://foo.example.com:8080, this might look like:

```
; Origin zone
foo.example.com.            3600 IN CNAME foosvc.example.net.
_8080._foo.foo.example.com. 3600 IN CNAME foosvc.example.net.
; Service provider zone
foosvc.example.net. 3600 IN SVCB 1 . key65333=...
foosvc.example.net.  300 IN AAAA 2001:db8::1
```

Domain owners SHOULD avoid using a SvcDomainName that is below a DNAME, as this is likely unnecessary and makes responses slower and larger.

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 Negotation (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
```

"alpn" is a multi-valued SvcParamKey. Each decoded value in the "alpn" value list SHALL be an "alpn-id". The value list MUST NOT be empty.

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.

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 "SVCB ALPN set"), the client adds the default set to the list of "alpn-id"s 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. "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 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.

Each decoded value in the value list SHALL be an IP address of the appropriate family in standard textual format [RFC5952]. To enable simpler parsing, this SvcParam 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.

6.5. "mandatory"

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 implements support for all its mandatory keys. If the SVCB RRSet contains no compatible RRs, the client will generally act as if the RRSet is empty.

In presentation format, "mandatory" contains a list 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. Any listed keys MUST also appear in the SvcParams. For example, the following is a valid list of SvcParams:

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

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.)

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 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 6.5) are "port", "alpn", and "no-default-alpn".

The presence of an HTTPS RR for an origin also indicates that all HTTP resources are available over HTTPS, as discussed in Section 7.5. 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.

7.1. Owner 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 7.5, 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.

7.2. 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.

7.2.1. 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 (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 HTTPS RRs.

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 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.

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.

7.3. Interaction with Alt-Svc

Clients that do not implement support for Encrypted ClientHello MAY skip the HTTPS RR 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 HTTPS RR client connection procedure (Section 3).

For clients that implement support for ECH, the interaction between HTTPS RRs 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 HTTPS RR 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 TargetName.

7.5. 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 6.5), 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 making an "https" scheme request to an origin with an HTTPS RR, 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 HTTPS RRs for HTTP SHOULD also use it for this URL. 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.

8. SVCB/HTTPS RR parameter for ECH configuration

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

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 HTTPS RRs as described in Section 3, but do not fall back to address records (steps 5 and 6). If there are compatible HTTPS RRs, they all have an "echconfig" key, and attempts to connect to them 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 address records (steps 5 and 6 of Section 3) if necessary.

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

8.2. Deployment considerations

An HTTPS 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 SvcPriority) 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 HTTPS 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 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.

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 HTTPS 1 h3pool alpn=h2,h3 echconfig="123..."
              HTTPS 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 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.

9.4. 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.

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 [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.

11. 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 7.5), and disable the encryption enabled by the echconfig SvcParamKey (Section 8). 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.

12. IANA Considerations

12.1. 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

12.2. 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

12.3. 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.

12.3.1. Procedure

A registration MUST include the following fields:

- Number: SvcParamKey wire format numeric identifier (range 0-65535)
- Name: SvcParamKey presentation name
- Meaning: a short description
- Pointer to specification text

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

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

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

12.3.2. Initial contents

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

Number	Name	Meaning	Reference
0	mandatory	Mandatory keys in this RR	(This document)
1	alpn	Additional supported protocols	(This document)
2	no-default-alpn	No support for default protocol	(This document)
3	port	Port for alternative endpoint	(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 ("Invalid key")	(This document)

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

12.4. 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)

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)

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

[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>.

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

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

[ECH] Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS Encrypted Client Hello", draft-ietf-tls-esni-07 (work in progress), June 2020.

[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)", draft-ietf-quic-http-29 (work in progress), June 2020.

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

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

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

[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>.

[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>.

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

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

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

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.bellis-dnsop-http-record]
Bellis, R., "A DNS Resource Record for HTTP", draft-bellis-dnsop-http-record-00 (work in progress), November 2018.

[I-D.ietf-dnsop-aname]
Finch, T., Hunt, E., Dijk, P., Eden, A., and W. Mekking, "Address-specific DNS aliases (ANAME)", draft-ietf-dnsop-aname-04 (work in progress), July 2019.

[I-D.tapril-ns2]
April, T., "Parameterized Nameserver Delegation with NS2 and NS2T", draft-tapril-ns2-00 (work in progress), March 2020.

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

[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>.

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

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

14.3. URIs

[1] https://github.com/MikeBishop/dns-alt-svc

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 value list

In order to represent lists of values in zone files, this specification uses an extended version of character-string decoding that adds the use of "," as a delimiter after double-quote processing. When "," is not escaped (by a preceding "\" or as the escape sequence "\044"), it separates values in the output, which is a list of 1*OCTET. (For simplicity, empty values are not allowed.) We refer to this modified procedure as "value-list decoding".

```
value-list = char-string
list-value = 1*OCTET
```

For example, consider these "char-string" SvcParamValues:

```
"part1,part2\,part3"
part1,part2\044part3
```

Character-string decoding either of these inputs would produce a single "*OCTET" output:

```
part1,part2,part3
```

Value-list decoding either of these inputs would instead convert it to a list of two "list-value"s:

```
part1
part2,part3
```

Appendix B. 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.

B.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.

B.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.

B.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.

B.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 C. Change history

- 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

  o 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.

  o draft-ietf-dnsop-svcb-httpssvc-00

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

    * Relax IP hint handling requirements. (Eric Rescorla)

  o 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

v6ops
Internet-Draft
Intended status: Informational
Expires: January 29, 2021

J. Palet Martinez
The IPv6 Company
A. D'Egidio
Telecentro

July 28, 2020

464XLAT/MAT-T Optimization
draft-ietf-v6ops-464xlat-optimization-03

Abstract

IP/ICMP Translation Algorithm (SIIT) can be used to provide access for IPv4-only hosts or applications to IPv4-only or dual-stack destinations over IPv6-only infrastructure. In that case, the traffic flows are translated twice: first from IPv4 to IPv6 (stateless NAT46 at the ingress point to the IPv6-only infrastructure) and then from IPv6 back to IPv4 (stateful NAT64, at the egress point). When the destination is IPv6-enabled, the second translation might be avoided. This document describes a possible optimization to 464XLAT and MAP-T to avoid translating IPv6 flows back to IPv4 if the destination is reachable over IPv6. The proposed solution would significantly reduce the NAT64 utilization in the operator's network, increasing the performance.

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 January 29, 2021.

Copyright Notice

Table of Contents

1. Introduction

Different transition mechanisms, typically in the group of the so-called IPv6-only with IPv4aaS (IPv4-as-a-Service), such as 464XLAT ([RFC6877]) or MAP-T ([RFC7599]), allow IPv4-only hosts or applications to connect with IPv4 services in Internet over IPv6-only infrastructure, by means of a stateless NAT46 SIIT (IP/ICMP Translation Algorithm) as described by [RFC7915].

This is done by the implementation of SIIT at the CE (Customer Edge) Router or sometimes at the end-device, for example, the UE (User Equipment) in cellular networks. This functionality is the CLAT (Customer Translator) in the case of 464XLAT, while in the case of MAP-T is called NAT46.

The NAT46/CLAT (WAN side) is connected by IPv6-only to the operator network, which in turn, will have a reverse translation, the NAT64 ([RFC6146]), known as PLAT (Provider Translator) in the case of 464XLAT. This allows to translate the IPv6 flow back to IPv4, in order to forward it to Internet.

In both cases (NAT46 and NAT64), the translation of the packet headers is done using the IP/ICMP translation algorithm defined in [RFC7915]. Translation between IPv4 and IPv6 addresses is done as per [RFC6052]. The NAT64 prefix should be discovered by the CE by one or more of the existing mechanisms ([RFC7225], [RFC8781] or [RFC7050]), and sometimes it is pre-configured at the CE to the WKP.

2. Requirements Language

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Possible Optimization

In the case of 464XLAT, a DNS64 ([RFC6147]) is (optionally) in charge of the synthesis of AAAA records from the A records, so the NAT64 can be used without the need of doing a double-translation by means of the NAT46/CLAT.

However, the DNS64 is not useful for the IPv4-only hosts or applications in the LANs, as they will not be able to use the AAAA records, so they are always forced to use the double-translation.

This is the expected behavior, as the original design of NAT64 was

targeted to connect IPv6-only devices (using DNS) to IPv4-only services. 464XLAT expanded the solution to also allow IPv4-only devices (even if not using DNS) to connect to IPv4-only services by means of IPv6-only access networks.

The optimization solutions presented by this document try to avoid this double-translation, in the cases when the Internet services, are already IPv6-enabled. So, in those cases, if the NAT46 already translated the IPv4 flow to IPv6, it doesn't look necessary to translate this back to IPv6.

A typical 464XLAT deployment is depicted in Figure 1.

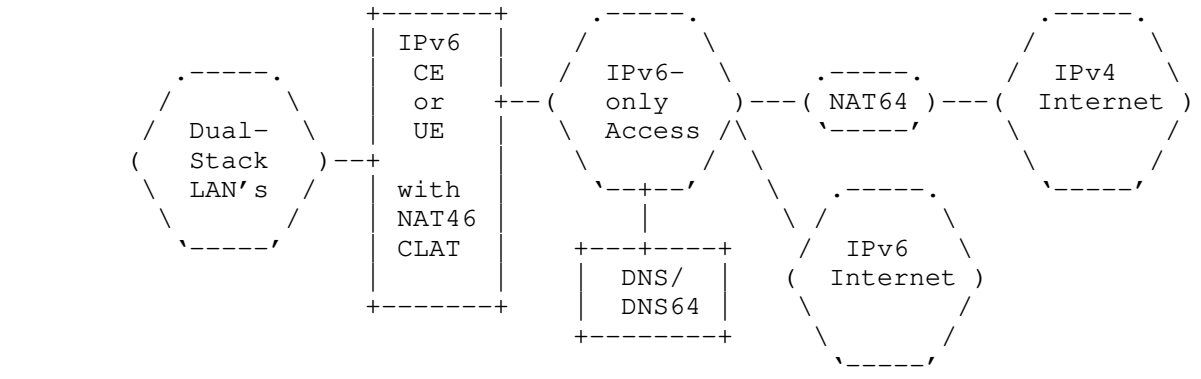


Figure 1: Typical 464XLAT Deployment

Examples of a topology shown on the above picture includes:

- An IPv6-only residential access network where the CE Router (with NAT46/CLAT) supports Dual-Stack in the customer LANs.

- An IPv6-only cellular network where a UE uses the NAT46/CLAT for dual-stack internal applications and other hosts connected via tethering.

If the operator is providing direct access, for example, to Content Delivery Networks (CDNs), caches, or other resources, and they are dual-stacked, the situation can be described as shown in Figure 2.

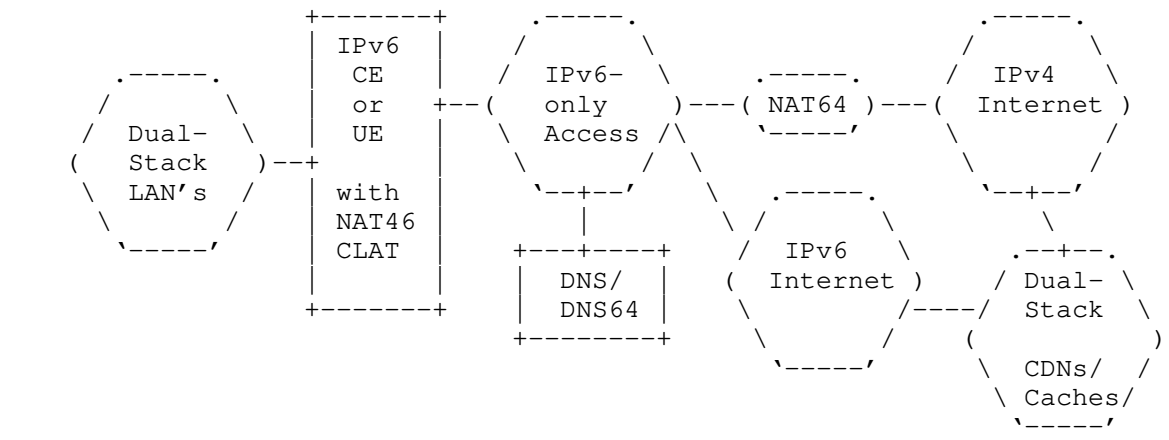


Figure 2: Typical 464XLAT Deployment with CDNs/Caches

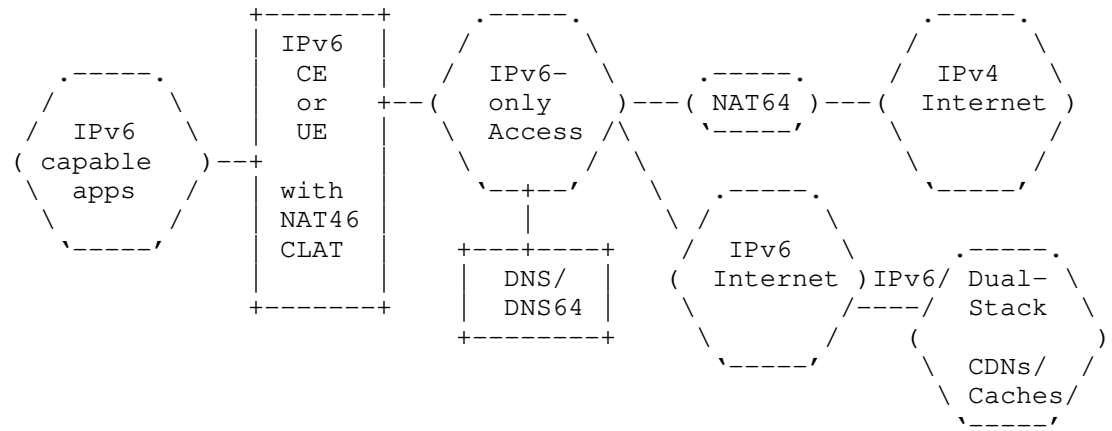
In this case if the flows initiated in the LANs come from IPv4-only hosts or applications, even if the destination resources are IPv6-enabled, the double-translation is enforced, which has the following consequences:

- More traffic needs to pass thru the NAT64 devices.
- More NAT64 devices may be needed to handle the additional traffic.
- Additional usage of IPv4 addresses.
- Additional state at the NAT64 devices.
- Additional logging, its relevant storage and processing resources.
- Increasing of delay and redduction of traffic performance.
- Unnecessary point(s) of failure for that traffic.

Clearly, all those aspects have impact in both, CapEx and OpEx. This is extremely important when considering that most of the time, the contents stored in CDNs, caches, and so on, is there for a good reason: It is frequently accessed resources and/or big. Examples such as video, audio, software and updates, are very common. So, this optimization can be highly impacting in many networks.

4. Problem Statement Summary

If the hosts or applications in the customer LAN are IPv6-capable, then the access to the CDNs, caches or other resources, will be made in an optimized way, by means of IPv6-only, not using the NAT64, as depicted in Figure 3.



However, if the hosts or applications are IPv4-only, for example, many SmartTVs and Set-Top-Boxes available today, a non-optimal double translation will occur (NAT46 at the CLAT and NAT64 at the PLAT), as illustrated in Figure 4.

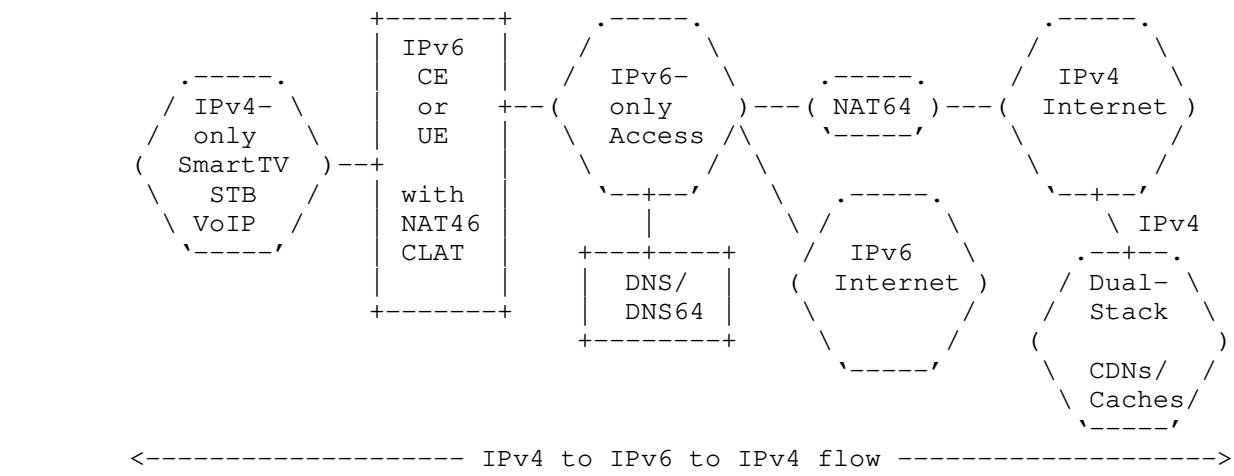


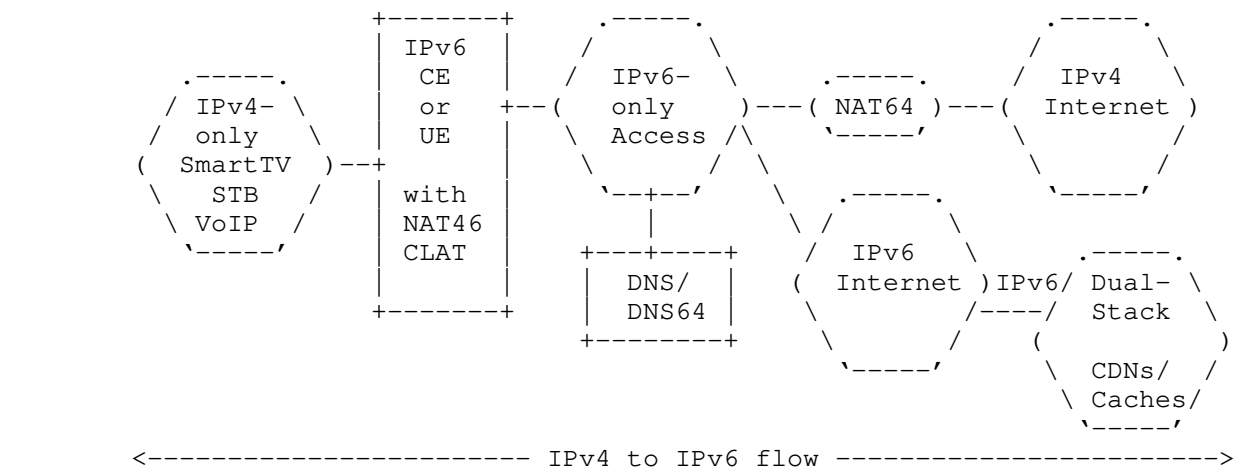
Figure 4: 464XLAT access to CDNs/Caches by IPv4-only apps

Clearly, this is a non-optimal situation, as it means that even if there is a dual-stack service, the NAT46/CLAT translated IPv4 to IPv6 traffic flow, is unnecessarily translated back to IPv4, traversing the stateful NAT64. This has a direct impact in the need to scale the NAT64 beyond what will be actually needed, if possible solutions, in order to keep using the IPv6 path towards those services, are considered.

As shown in the Figure 4, this is also the case for many other services, not just CDNs or caches, such as VoIP access to the relevant operator infrastructure, which may be also dual-stack. This is true as well for many other dual-stack or IPv6-enabled services, which may be directly reachable from the operator infrastructure, even if they are not part of it, for example peering agreements, services in IXs, etc. In general, this will become a more frequent situation for many other services, which are not yet dual-stack.

For simplicity, across the rest of this document, references to CDNs/ caches, should be understood, unless otherwise stated, as any dual-stacked resources.

This document looks into different possible solution approaches in order to optimize the IPv4-only SIIT translation providing a direct path to IPv6-capable services, as depicted in Figure 5.



5. Solution Approaches

5.1. Approach 1: DNS/Routing-based Solution

Because the IPv4-only devices will not be able to query for AAAA records, the NAT46/CLAT/CE will translate the IPv4 addresses from the A record for the CDN/cache destination, using the WKP or NSP, as configured by the operator.

If the CDN/cache provider is able to configure, in the relevant interfaces of the CDN/caches, the same IPv6 addresses that will naturally result as the translated destination addresses for the queried A records, preceded by the WKP or NSP, then having more specific routing prefixes, will result in traffic to those destinations being directly forwarded towards those interfaces, instead of needing to traverse the NAT64.

For example, let's suppose a provider using the WKP (64:ff9b::/96) and a SmartTV querying for www.example.com:

www.example.com	A	192.0.2.1
NAT46/CLAT translated to		64:ff9b::192.0.2.1
CDN IPv6 interface must be		64:ff9b::192.0.2.1
Operator must have a specific route to		64:ff9b::192.0.2.1

Note: Examples using text representation as per Section 2.3 of [RFC6052] and IPv4 documentation addresses following [RFC5737].

It should be remarked that this approach requires that the path to

the destination is configured in such way (i.e., more specific routing prefixes), that doesn't traverse the NAT64 devices.

Because the WKP is non-routable, this solution will only be possible if the CDN/cache is in the same ASN as the provider network, or somehow interconnected without routing thru Internet.

This solution has the additional drawback of the operational complexity/issues added to the operation of the CDN/cache, and the need to synchronize any IPv4 interface address changes with the relevant IPv6 ones, and possibly with routing.

5.2. Approach 2: NAT46/CLAT/DNS-proxy-EAM-based Solution

If the NAT46/CLAT/CE, as commonly is the case, is also a DNS proxy/ stub resolver, it is possible to modify the behavior and create an "internal" interaction among both of them.

This approach uses the existing IPv4 and IPv6 addresses in the A and AAAA records, respectively, so no additional complexity/issues added to the CDN/caches operations.

The following sub-sections detail this approach and provide a step-by-step example case.

5.2.1. Optimization enabling

This optimization MUST be enabled by default, but only when the WAN link is IPv6-only and the NAT46/CLAT is being used. This allows the users to get CEs from the retail and take advantage of the optimization, without requiring any configuration.

The CE MUST support a way to completely disable the optimization, in order to allow the operator to turn it off in case it is required.

It is expected that the NAT64/CLAT is only enabled if the WAN link is IPv6-only.

5.2.2. Detection of IPv4-only hosts

The goal is to ensure that only IPv4-only hosts are optimized. Towards that, the CE MUST use ARP and ND (and their relevant caches, if the information of a hosts has been already learnt) each time a new host starts a connection. If it is possible to bind the same MAC address to both an IPv4 and IPv6 address, then the host is not IPv4-only (it may be IPv6-only or dual-stack), and MUST NOT be optimized.

The CE MUST maintain a table of IPv4-only hosts and ensure that if any IPv4-only hosts become IPv6-enabled, it is properly handled.

This mechanism to detect IPv4-only hosts has two drawbacks:

1. If a subscriber has intermediate dual-stack routers in between the IPv4-only host and the NAT46/CLAT, the IPv4-only hosts will be detected as dual-stack, so no optimization will be performed. This is not the most common scenario, as typically the devices that are more relevant to the optimization (in terms of those that generate more IPv4-only traffic) are directly attached to the CE, or in bridged interfaces.

2. If a host is dual-stack, but has some IPv4-only applications, because the host will be detected as dual-stack, none of the applications will be optimized. This is a good trade-off, considering the most important traffic to optimize is typically coming from real IPv4-only devices such as old SmartTVs/STBs. Furthermore, this avoid breaking other mechanisms present only in dual-stack hosts, such as Happy Eyeballs [RFC8305] and simplifies troubleshooting.

It needs to be remarked that, if the detection of the IPv4-only host is done incorrectly (either not detecting it or by a false detection), the goal is that no harm is caused. In the worst case, optimization MUST NOT be performed.

5.2.3. Detection of IPv6-enabled service

In the case of an IPv4-only detected host, the DNS proxy/stub resolver MUST actually perform an additional AAAA query, unless the information is already present in the Additional Section, as per Section 3 of [RFC3596].

Note that the NAT46/CLAT MUST already know the WKP or NSP being used in that network. If the response contains at least one IPv6 address not using the WKP/NSP, it means that the destination is IPv6-enabled (because at least one of the IPv6 addresses is not synthesized). This means that it is possible for the NAT46/CLAT, to create an Explicit Address Mapping ([RFC7757]).

5.2.4. CE DNS proxy responses

In the case of an IPv4-only detected host, the CE DNS proxy MUST only return the answer to an A query once any of the following happens:

1. An answer to the AAAA query has been received.

2. A SERVFAIL has been received.

3. The "Resolution Delay" has passed.

The Resolution Delay MUST be set to the same value (50 milliseconds) as indicated by Happy Eyeballs [RFC8305].

5.2.5. Creation of EAMT entries

An EAM Table (EAMT used for short, across the rest of this document) MUST be created/maintained automatically by the NAT46/CLAT, which is responsible to prioritize any available entries in the EAMT, versus the use of any synthetic AAAA.

The EAMT entry MUST only be created if all the following conditions are met:

1. The source host is IPv4-only.

2. The DNS proxy is ready to return the A answer (according to Section 5.2.4).

3. There is at least one non-synthesized AAAA response.

4. If DNSSEC is available, the response has been locally validated or the AD bit has been set by a trusted resolver, as per [RFC3655].

This avoids a slight NAT64 overload and flapping between destination addresses (IPv4/IPv6), which may impact some applications, at the cost of a small extra delay for the initial communication setup, when the EAMT entry doesn’t yet exist.

Each EAMT entry MUST contain, the fields already described in [RFC7757] and a few new extensions (as per section 3.1 of [RFC7757]):

1. ID: EAMT Entry Index (optional).

2. MAC address: Identify the host to which this EAMT entry belongs.

3. Destination IPv4 address/prefix: By default, the prefix length is 32 bits.

4. Destination IPv6 address/prefix: By default, the prefix length is 128 bits. Only non-synthesized addresses are allowed.

5. TTL: It MUST be set to the minimum value from the AAAA/A RR pair. In normal conditions the TTL for both A and AAAA records, of a

given FQDN, should be the same, so this ensures a proper behavior if there is any DNS mismatch.

6. FQDN: The one that originated the A query for this EAMT entry. Required in order to ensure a correct detection of cases such as the use of reverse-proxy with a single IPv4 address to multiple IPv6 addresses.

7. Valid/Stale/Invalid: When set to Stale, means that this EAMT entry MUST NOT be used for new connections. When set to Invalid, means that this EAMT entry can be deleted, unless the Auto/Static bit is also set.

8. Auto/Static: When set to 1, means that this EAMT entry has been manually/statically configured, for example by means of an explicit configuration (GUI, CLI, provisioning system, etc.).

Note that allowing destination IPv4 and IPv6 prefixes, together with the Valid/Stale/Invalid and Auto/Static flags, allows manual explicit optimization and non-optimization configuration for specific parts of Internet.

When a new EAMT entry is first automatically created, it is flagged as "Valid" and "Auto". If a subsequent A query, with a different FQDN, results in an IPv4 address that has already an EAMT entry and a different IPv6 address, it means that some reverse-proxy or similar functionality is being used by the IPv6-enabled service. In this case, the existing EAMT entry will be marked as "Stale". No new EAMT entry is created for that IPv4 address. Otherwise, the optimization will only allow to access the first set of IPv4/IPv6/FQDN, which may break the access to other FQDN that share the same IPv4 address and different IPv6 addresses.

In this case the EAMT entry will still become "Invalid" according the TTL, which allows to re-enable optimization if a new query for the A record has changed the situation. For example, maybe the reverse-proxy has been removed, or there is now only a single device using it, so at the time being, the optimization is again possible without creating troubles to other hosts.

An EAMT entry marked as "Stale" or "Invalid", only affects the relevant host. Other hosts have their own EAMT entries or they are using the regular NAT46/CLAT+NAT64 path (without the optimization).

5.2.6. Forwarding path via stateful NAT for existing EAMT entries

Following this approach, if there is a valid EAMT entry, for a given pair of source-MAC-address/IPv4-destination, the IPv6-native path pointed by the IPv6 address of that EAMT entry, will take precedence versus the NAT64 path, so the traffic will not be forwarded to the NAT64.

However, this is not sufficient to ensure that individual applications are able to keep existing connections. In many cases, audio and video streaming may use a single TCP connection lasting from minutes to hours. Instead, the CDN TTLs may be configured in the range from 10 to 300 seconds in order to allow new resolutions to switch quickly and to handle large recursive resolvers (with hundreds of thousands of clients behind them).

Consequently, the EAMT entries MUST NOT be used directly to establish a forwarding path, but instead, MUST be used to create a stateful NAT entry for the 4-tuple for the duration of the session/connection. This means that to implement the optimization the NAT46 MUST be stateful. Typically, stateful NAT46 are implemented by means of a stateful NAT44 (which often maybe hardware off-loaded), followed by a stateless NAT46. If the SoC/code is able to do stateless NAT46, this still could be used when the optimization is disabled.

5.2.7. Maintenance of the EAMT entries

An EATM entry with the Auto/Static bit set, MUST NOT be automatically modified.

An EAMT entry with the Auto/Static bit clear, MUST be set to Stale in case of:

1. TTL time-out.

2. Or the conditions for creation of the EAMT entry (Section 5.2.5) aren't longer met.

Entries in Stale state MUST be set to Invalid once existing connections time-out.

The Invalid EAMT entries MUST be deleted, unless the Auto/Static bit is set. This allows users/operators to set explicit rules for diagnostics or resolution of issues in special situations.

5.2.8. Usage example

Using the same example as in the previous approach:

www.example.com		A	192.0.2.1
		AAAA	2001:db8::a:b:c:d
EAMT entry	192.0.2.1		2001:db8::a:b:c:d
NAT46/CLAT translated to			2001:db8::a:b:c:d
CDN IPv6 interface already is			2001:db8::a:b:c:d
Operator already has a specific route to			2001:db8::a:b:c:d

The following is an example of the CE behavior after the previous case has already created an EAMT entry and a reverse-proxy is detected:

1. A query for www.example.net A RR is received

2. www.example.net A 192.0.2.1

3. www.example.net AAAA 2001:db8::e:e:f:f

4. A conflict has been detected

5. The existing EAMT entry for 192.0.2.1 is set to Stale

5.2.9. Behavior in case of multiple A/AAAA RRs

If multiple A/AAAA RRs are available, any of them could be chosen and in general, the optimization will not present any different result to the hosts compared with a situation where the optimization is not used.

Existing DNS proxy/stub resolvers already implement mechanisms for DNS Load Balancing ([RFC1794]). This don't need to be modified to implement the optimization if some degree of randomness is already secured.

To secure sufficient randomness, a possible algorithm shall ensure that different EAMT entries (for different hosts) are permuted randomly among different A/AAAA records on the A/AAAA RR set.

5.2.10. Behavior in presence/absence of DNS64

464XLAT can be deployed/used with and without a DNS64. However, as indicated in Section 5.2.3, the EAMT entry is only created when the service is IPv6-enabled, because the optimization is only relevant for destinations which already have AAAA records. In those cases DNS64 is not relevant.

5.2.11. Behavior when using literal addresses or non IPv6-compliant APIs

Because the EAMT entries are only created when the NAT46/CLAT/CE proxy/stub DNS is being used, any hosts or applications that don't use DNS, will not create the relevant entries.

They will not be optimized unless EAMT entries are statically configured.

5.2.12. Behavior in case of Foreign DNS

Hosts or applications may use DNS servers from other networks. For a complete description of reasons for that, refer to Section 4.4 of [RFC8683]. In the case the DNS is modified, or some hosts or applications use other DNS servers, the possible scenarios and the implications are:

a. Devices configured to use a DNS proxy/resolver which is not the CE/NAT46/CLAT. In this case this optimization will not work, because the EAMT entry will not be created based on their own flows. Nevertheless, the EAMT entry may be manually created. However, the lack of EAMT entry, will not impact negatively in the user's hosts/applications (the optimization is not performed). It should be noticed that users commonly, don't change the configuration of devices such as SmartTVs or STBs (if they do, some other functionalities, such as CDN/caches optimizations may not work as well), so this only happens typically if the vendor is doing it on-purpose and for good well-known reasons.

b. DNS privacy/encryption. Hosts or applications that use mechanisms for DNS privacy/encryption, such as DoT ([RFC7858], [RFC8094]), DoH ([RFC8484]) or DoQ ([I-D.huitema-dprive-dnsoquic]), will not make use of the stub/proxy resolver, so the same considerations as for the previous case applies.

c. Users that modify the DNS in their Operating Systems. This is quite frequent, however commonly Operating Systems are dual-stack, so aren't part of the problem statement described by this document and will not be adversely affected.

d. Users that modify the DNS in the CE. This is less common. In this case, this optimization is not adversely affected, because it doesn't depend on the operator DNS, it works only based on the internal CE interaction between the NAT46/CLAT and the stub/proxy resolver. Note that it may be affected if the operator offers

different "DNS views" or "split DNS", however this is not related to this optimization and will anyway impact in the other possible operator optimizations (i.e. CDN/cache features).

e. Combinations of the above ones. No further impact is observed, beyond the ones already described.

5.2.13. False detection of a dual-stack host as IPv4-only

If a dual-stack host is being detected as IPv4-only, is because it is not responding to the CE ND messages, so by all means, should be considered, at the time being, as IPv4-only, and consequently EAMT entries will be created and traffic will be optimized for IPv4 flows.

However, if this hosts suddenly become IPv6-enabled (or dual-stack), the relevant EAMT entries must be flagged by the CE as "Stale". The host will be able to complete the connections and the entries will be marked as "Invalid" and deleted.

Anyway, those EAMT entries, while "Valid", may not be actually used by the dual-stack hosts, because those hosts or applications should prefer IPv6, so most probably was either a temporary failure or done on-purpose (user, troubleshooting). If the host is preferring IPv4 for connecting to the CDN/cache or IPv6-enabled service, it will be actually using the NAT46/CLAT, including the EAMT entry and consequently IPv6, so this mechanism will be correcting an undesirable behavior. This may be also a special case, which actually seems to be an incoherent host or application implementation.

5.2.14. Behavior in presence of Happy Eyeballs

Happy Eyeballs [RFC8305] is only enabled in dual-stack hosts. Consequently, it is not affected by this optimization because both, as the host should not be detected as IPv4-only, following Section 5.2.2.

If Happy Eyeballs triggers a fallback to IPv4 for a given host, it will be actually using IPv4 without the optimization, which in turn, uses the IPv6-only WAN link of the NAT46/CLAT/CE. So even if Happy Eyeballs is present, IPv4 is expected to be slower than native IPv6 itself due to delays added by the NAT46/CLAT+NAT64 translations. This optimization reduces those delays by eliminating the second translation (NAT64) for IPv4-only detected hosts.

However, there may be cases where this may be understood as problematic. The possible reasons why Happy Eyeballs may trigger an IPv4 fallback, in the case of IPv6-only access networks with IPv4aaS,

in general, can be classified as:

1. Failure at the CE or customer LANs. It may happen that the CE or other devices in the customer LANs are showing erratic behaviors or malfunctions. It is difficult to believe that this happens only with IPv6, and if that's the case Happy Eyeballs will not resolve the issue, because IPv4 is provided as a service on top of IPv6.

2. Complete failure of the IPv6-only link or IPv6-only operator's infrastructure (up to the NAT64). In this case, IPv4 will not work for that subscriber. Happy Eyeballs will not resolve the issue, and instead will only be adding some extra delay (the attempt to fallback to IPv4 before timing-out).

3. Complete failure of both IPv4 and IPv6 links behind the operator's NAT64 towards the destination. In this case, typically both, IPv4 and IPv6 will fail (in many cases, they are dual-stack links, not different links). Again, Happy Eyeballs will also fail to resolve the issue.

4. Complete failure only in the IPv6 links behind the operator's NAT64 towards the destination. This is less frequent, bus still miss-configured AAAA RRs, or diverse paths for IPv4 and IPv6 together with outages or IPv6-only routing issues, could generate this problem. In this case, Happy Eyeballs could resolve the issue, however. It will work because the optimization is not enabled for dual-stack hosts.

5. Partial failure: Slower IPv6 vs IPv4 path end-to-end. In general, the added delay of the IPv4 translations and NATs across the path, increases the chances that IPv4 is slower than IPv6. However, it may happen that there is some IPv6 specific link congestion or packet dropping, that generates the reverse situation, so IPv4 become faster than IPv6. Because the optimization is not being used for dual-stack hosts, Happy Eyeballs will be resolving the issue.

In summary, the optimization will not change the Happy Eyeballs behaviour. Furthermore, it should be observed that IPv6 failures will also impact other operators (even if not using the optimization), and especially those using only NAT64+DNS64 instead of 464XLAT, or even more, any IPv6-only hosts and applications in any operator network across the entire Internet. It looks like it is very important to make sure that, as IPv6 is more prevalent, there is a better monitoring and failures are detected ASAP, instead of being hidden by Happy Eyeballs, specially in IPv6-only networks. It should be noticed also that in IPv6-only with IPv4aaS, the chances of

troubles in the IPv4 paths seem to be higher than in the IPv6, as there are more translations, more devices, more delays, while the optimization will precisely reduce them.

5.2.15. Troubleshooting Implications

When there is a need to troubleshoot IPv4 from the CE LANs, it may happen that there is an EAMT entry forcing the flow to a given destination(s) to use IPv6, which will distort the results, unless the host being used is dual-stack (which is the most common situation).

This can be avoided, using a CLI/GUI or provisioning procedure, to either completely disable the optimization during the troubleshooting, or create specific static EAMT entries, using the Valid/Stale/Invalid and Auto/Static flags, as described in Section 5.2.5.

Consequently, the CE MUST allow both, disabling the optimization and the setup of manual/static EAMT entries.

5.3. Approach 3: NAT46/CLAT-provider-EAM-based Solution

Instead of using the DNS proxy/stub resolver to create the EAMT entries, the operator may push this table (or parts of it) into the CE/NAT46/CLAT, by using configuration/management mechanisms.

This solution has the advantage of not being affected by any DNS changes from the user (the EAMT is created by the operator) and ensures a complete control from the operator. However, it may impact the cases of devices with a DNS configured by the vendor.

In general, most of the considerations from the previous approach will apply.

One more advantage of this solution is that the EAMT pairs doesn't need to match the "real" IPv4/IPv6 addresses available in the A/AAAA records, as shown in the next example.

```
www.example.com                      A       192.0.2.1
                                     AAAA    2001:db8::a:b:c:d
EAMT pulled/pushed entry        192.0.2.1    2001:db8::f:e:d:c
NAT46/CLAT translated to                     2001:db8::f:e:d:c
CDN IPv6 interface already is                2001:db8::f:e:d:c
Operator already has a specific route to     2001:db8::f:e:d:c
```

EAMT may contain TTLs which probably are derived from DNS ones, or alternatively, a global TTL for the full table.

An alternative way to configure the table, is that the CE is actually pulling the table (or parts of it) from the operator infrastructure. In this case it will be mandatory that the entries have individual TTLs, again probably derived from the DNS ones.

This approach has three major drawbacks:

1. CDNs are used to do frequent changes at the DNS level, so unless the CDNs offer an API or equivalent convenient solution to keep updated the EAMT, the operator will need to cache the most frequent FQDNs being resolved in their own DNS and based on the TTLs, update the EAMT.

2. It requires a new protocol, or an extension to existing ones, in order to push or pull the EAMT updates.

3. It may generate additional bandwidth utilization in the WAN links for every CE when the EAMT needs to be update, even when a CE reboots.

6. IPv6-only Services become accessible to IPv4-only devices/apps

One of the issues with the IPv6 deployment, is that those services which become IPv6-only in Internet, aren't reachable by IPv4-only hosts and applications. This means that new content providers must support dual-stack even for new services, even while IPv4 public addresses aren't available.

If NAT46/CLAT/DNS-proxy-EAM approach (Section 5.2) is chosen, it also offers the chance to resolve this issue. This is possible if IPv6-only services get configured with an A resource record pointing to a well-known IPv4 address despite they aren't actually connected to IPv4. This is out of scope for this document, as it will require further work and a reservation by IANA, This will mean that those services will work fine if there is a NAT46/CLAT supporting the optimization. This A RR has no negative impact if the NAT46/CLAT doesn't exist, or it is not optimized, because is not reachable via IPv4-only, so is not a different situation compared with not having an A RR.

The result of this is equivalent to the approach taken by MAP-T (Section 12.3 of [RFC7599]). However, it has the advantage that the MAP-T approach is restricted to services in the MAP-T domain.

In fact, it may become an incentive for the IPv6 deployment in Internet services, as it provides the option to use a well-known IPv4 address (maybe anycast) for the "non-valid" A RR, that points, in case of port 80/443 to a web page or service that returns a warning

such as "This service is only available if the network is properly connected to Internet with IPv6".

7. Conclusions

NAT46/CLAT/DNS-proxy-EAM approach (Section 5.2) seems the right solution for optimizing the access to dual-stack services, whether they are located inside or outside the ISP. It is also the only approach which has no additional requirements for the network operators (both ISPs and CDN/cache operators).

Having this type of optimization facilitates and increases the usage of IPv6, even for IPv4-only hosts and applications, at the same time that decreases the use of the NAT64.

SIIT already has a SHOULD for EAM support, so it is not a high additional burden the support required for existing implementations to be updated for this optimization.

8. Security Considerations

This document does not have any new specific security considerations, besides the ones already known for DNS64.

Spoofed DNS responses could generate incorrect EAMT entries. However, this seems not different than if the optimization is not in place and the spoofed DNS responses are cached by the CE DNS proxy/ stub resolver or even by hosts in the CE LANs. It very much depends on how and where the attack is actually performed.

In both cases, 464XLAT and MAP-T, the CE device should contain a DNS proxy/stub resolver, which is also required for the optimization. Nevertheless, it is common that the user change DNS settings. If it happens, in the case of MAP-T, the port-set is restricted for an efficient public IPv4 address sharing, so the entropy of the source ports is significantly lowered. In this case, theoretically MAP-T is less resilient against cache poisoning ([RFC5452]) compared with 464XLAT. However, an efficient cache poisoning attack requires that the subscriber operates its own caching DNS server and the attack is performed in the service provider network, so the chances of a successful exploitation of this vulnerability are low.

9. IANA Considerations

This document does not have any new specific IANA considerations.

10. Acknowledgements

The authors would like to acknowledge the inputs of Erik Nygren, Fred Baker, Martin Hunek, Chongfeng Xie, Fernando Gont, Fernando Frediani, Jen Linkova, Eduard Vasilenko and Philip Homburg.

11. References

11.1. Normative References

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

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <https://www.rfc-editor.org/info/rfc3596>.

[RFC3655] Wellington, B. and O. Gudmundsson, "Redefinition of DNS Authenticated Data (AD) bit", RFC 3655, DOI 10.17487/RFC3655, November 2003, <https://www.rfc-editor.org/info/rfc3655>.

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <https://www.rfc-editor.org/info/rfc5452>.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <https://www.rfc-editor.org/info/rfc6052>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[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>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <https://www.rfc-editor.org/info/rfc6877>.

[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>.

[RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <https://www.rfc-editor.org/info/rfc7225>.

[RFC7599] Li, X., Bao, C., Dec, W., Ed., Troan, O., Matsushima, S., and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, DOI 10.17487/RFC7599, July 2015, <https://www.rfc-editor.org/info/rfc7599>.

[RFC7757] Anderson, T. and A. Leiva Popper, "Explicit Address Mappings for Stateless IP/ICMP Translation", RFC 7757, DOI 10.17487/RFC7757, February 2016, <https://www.rfc-editor.org/info/rfc7757>.

[RFC7915] Bao, C., Li, X., Baker, F., Anderson, T., and F. Gont, "IP/ICMP Translation Algorithm", RFC 7915, DOI 10.17487/RFC7915, June 2016, <https://www.rfc-editor.org/info/rfc7915>.

[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>.

[RFC8305] 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>.

[RFC8781] Colitti, L. and J. Linkova, "Discovering PREF64 in Router Advertisements", RFC 8781, DOI 10.17487/RFC8781, April 2020, <https://www.rfc-editor.org/info/rfc8781>.

11.2. Informative References

[I-D.huitema-dprive-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-huitema-dprive-dnsoquic-00 (work in progress), March 2020.

[RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794, DOI 10.17487/RFC1794, April 1995, <https://www.rfc-editor.org/info/rfc1794>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, DOI 10.17487/RFC5737, January 2010, <https://www.rfc-editor.org/info/rfc5737>.

[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>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/info/rfc8094>.

[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>.

[RFC8683] Palet Martinez, J., "Additional Deployment Guidelines for NAT64/464XLAT in Operator and Enterprise Networks", RFC 8683, DOI 10.17487/RFC8683, November 2019, <https://www.rfc-editor.org/info/rfc8683>.

Authors' Addresses

Jordi Palet Martinez
The IPv6 Company
Molino de la Navata, 75
La Navata - Galapagar, Madrid 28420
Spain

Email: jordi.palet@theipv6company.com
URI: http://www.theipv6company.com/

Alejandro D'Egidio
Telecentro
Argentina

Email: adegidio@telecentro.net.ar

Network Working Group
Internet-Draft
Obsoletes: 8109 (if approved)
Intended status: Best Current Practice
Expires: January 1, 2021

P. Koch
DENIC eG
M. Larson
P. Hoffman
ICANN

June 30, 2020

Initializing a DNS Resolver with Priming Queries
draft-klh-dnsop-rfc8109bis-00

Abstract

This document describes the queries that a DNS resolver should emit to initialize its cache. The result is that the resolver gets both a current NS Resource Record Set (RRset) for the root zone and the necessary address information for reaching the root servers.

This document, when published, obsoletes RFC 8109.

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 January 1, 2021.

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

Recursive DNS resolvers need a starting point to resolve queries. [RFC1034] describes a common scenario for recursive resolvers: they begin with an empty cache and some configuration for finding the names and addresses of the DNS root servers. [RFC1034] describes that configuration as a list of servers that will give authoritative answers to queries about the root. This has become a common implementation choice for recursive resolvers, and is the topic of this document.

This document describes the steps needed for this common implementation choice. Note that this is not the only way to start a recursive name server with an empty cache, but it is the only one described in [RFC1034]. Some implementers have chosen other directions, some of which work well and others of which fail (sometimes disastrously) under different conditions. For example, an implementation that only gets the addresses of the root name servers from configuration, not from the DNS as described in this document, will have stale data that could cause slower resolution.

This document only deals with recursive name servers (recursive resolvers, resolvers) for the IN class.

1.1. Changes from RFC 8109

This document obsoletes [RFC8109]. The significant changes from RFC 8109 are:

- Added section on the content of priming information.
- Added paragraph about no expectation that the TC bit in responses will be set.
- Clarified language for root server domain names as "root server identifiers".
- Added informative references to RSSAC documents.
- Added short discussion about this document and private DNS.
- Future changes noted in the current text with [[text like this]].

1.2. Terminology

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.

See [RSSAC026v2] for terminology that relates to the root server system.

2. Description of Priming

Priming is the act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers. A recursive resolver starts with no information about the root servers, and ends up with a list of their names and their addresses.

Priming is described in Sections 5.3.2 and 5.3.3 of [RFC1034]. The scenario used in that description, that of a recursive server that is also authoritative, is no longer as common.

The configured list of IP addresses for the root servers usually comes from the vendor or distributor of the recursive server software. This list is usually correct and complete when shipped, but may become out of date over time.

The domain names for the root servers are called the "root server identifiers". This list has been stable since 1997, but the IPv4 and IPv6 addresses for the root server identifiers sometimes change, Research shows that after those addresses change, some resolvers never get the new addresses. Therefore, it is important that resolvers be able to cope with change, even without relying upon configuration updates to be applied by their operator. Root server identifier and address changes are the main reasons that resolvers need to do priming instead of just going from a configured list to get a full and accurate list of root servers.

See [RSSAC023v2]for a history of the root server system.

Although this document is targeted at the global DNS, it also could apply to a private DNS as well. These terms are defined in [RFC8499].

2.1. Content of Priming Information

As described above, the configuration for priming is a list of IP addresses. The priming information in software may be in any format that gives he software the addresses associated with at least some of the root server identifiers.

Some software has configuration that also contains the root server identifiers, sometimes as comments and sometimes as data consumed by the software. For example, IANA's "Root Hints File" at <https://www.internic.net/domain/named.root> is derived directly from the root zone and contains all of the addresses of the root server identifiers found in the root zone, It is in DNS master file format, and includes the root server identifiers. Although there is no harm to adding such information, it is not useful in the root priming process.

3. Priming Queries

A priming query is a DNS query used to get the root server information in a resolver. It has a QNAME of ".", a QTYPE of NS, and a QCLASS of IN; it is sent to one of the addresses in the configuration for the recursive resolver. The priming query can be sent over either UDP or TCP. If the query is sent over UDP, the source port SHOULD be randomly selected (see [RFC5452]). The Recursion Desired (RD) bit MAY be set to 0 or 1, although the meaning of it being set to 1 is undefined for priming queries.

The recursive resolver SHOULD use EDNS0 [RFC6891] for priming queries and SHOULD announce and handle a reassembly size of at least 1024 octets [RFC3226]. Doing so allows responses that cover the size of a full priming response (see Section 4.2) for the current set of root servers. See Section 3.3 for discussion of setting the DNSSEC OK (DO) bit (defined in [RFC4033]).

3.1. Repeating Priming Queries

The recursive resolver SHOULD send a priming query only when it is needed, such as when the resolver starts with an empty cache and when the NS RRset for the root zone has expired. Because the NS records for the root are not special, the recursive resolver expires those NS records according to their TTL values. (Note that a recursive resolver MAY pre-fetch the NS RRset before it expires.)

[[Need to discuss: when pre-fetching, does the resolver send the queries to the addresses associated with the . / NS RRset in the cache, or does the resolver go back to the addresses in the configuration?]]

If a priming query does not get a response, the recursive resolver needs to retry the query with a different target address from the configuration.

3.2. Target Selection

In order to spread the load across all the root server identifiers, the recursive resolver SHOULD select the target for a priming query randomly from the list of addresses. The recursive resolver might choose either IPv4 or IPv6 addresses based on its knowledge of whether the system on which it is running has adequate connectivity on either type of address.

Note that this recommended method is not the only way to choose from the list in a recursive resolver's configuration. Two other common methods include picking the first from the list, and remembering which address in the list gave the fastest response earlier and using that one. There are probably other methods in use today. However, the random method listed above SHOULD be used for priming.

3.3. DNSSEC with Priming Queries

[[This section talks about sending the DO bit, but does not actually talk about validating the response to the priming query. This became important after the root KSK rollover in 2018 because some resolvers apparently were validating and only had the old KSK, but were still sending RFC 8145 telemetry even after failing to validate their priming response.]]

The resolver MAY set the DNSSEC OK (DO) bit. At the time of publication, there is little use to performing DNSSEC validation on the priming query. Currently, all root name server names end in "root-servers.net" and the AAAA and A RRsets for the root server names reside in the "root-servers.net" zone. All root servers are also authoritative for this zone, allowing priming responses to include the appropriate root name server A and AAAA RRsets. But, because the "root-servers.net" zone is not currently signed, these RRsets cannot be validated.

A man-in-the-middle attack on the priming query could direct a resolver to a rogue root name server. Note, however, that a validating resolver will not accept responses from rogue root name servers if they are different from the real responses because the resolver has a trust anchor for the root and the answers from the root are signed. Thus, if there is a man-in-the-middle attack on the priming query, the only result for a validating resolver will be a denial of service, not the resolver's accepting the bad responses.

If the "root-servers.net" zone is later signed, or if the root servers are named in a different zone and that zone is signed, having DNSSEC validation for the priming queries might be valuable.

4. Priming Responses

A priming query is a normal DNS query. Thus, a root name server cannot distinguish a priming query from any other query for the root NS RRset. Thus, the root server's response will also be a normal DNS response.

4.1. Expected Properties of the Priming Response

The priming response is expected to have an RCODE of NOERROR, and to have the Authoritative Answer (AA) bit set. Also, it is expected to have an NS RRset in the Answer section (because the NS RRset originates from the root zone), and an empty Authority section (because the NS RRset already appears in the Answer section). There will also be an Additional section with A and/or AAAA RRsets for the root name servers pointed at by the NS RRset.

Resolver software SHOULD treat the response to the priming query as a normal DNS response, just as it would use any other data fed to its cache. Resolver software SHOULD NOT expect exactly 13 NS RRs because, historically, some root servers have returned fewer.

4.2. Completeness of the Response

There are currently 13 root servers. All have one IPv4 address and one IPv6 address. Not even counting the NS RRset, the combined size of all the A and AAAA RRsets exceeds the original 512-octet payload limit from [RFC1035].

In the event of a response where the Additional section omits certain root server address information, re-issuing of the priming query does not help with those root name servers that respond with a fixed order of addresses in the Additional section. Instead, the recursive resolver needs to issue direct queries for A and AAAA RRsets for the remaining names. Currently, these RRsets would be authoritatively available from the root name servers.

If the Additional section is truncated, there is no expectation that the TC bit in the response will be set to 1. At the time that this document is written, many of the root servers are not setting the TC bit on responses with a truncated Additional section.

5. Post-Priming Strategies

[[Describe some common post-priming strategies for picking which RSI to use for queries sent to the root, such as "always use the fastest", "create buckets of fastness and pick randomly in the buckets", and others.]]

6. Security Considerations

Spoofing a response to a priming query can be used to redirect all of the queries originating from a victim recursive resolver to one or more servers for the attacker. Until the responses to priming queries are protected with DNSSEC, there is no definitive way to prevent such redirection.

An on-path attacker who sees a priming query coming from a resolver can inject false answers before a root server can give correct answers. If the attacker's answers are accepted, this can set up the ability to give further false answers for future queries to the resolver. False answers for root servers are more dangerous than, say, false answers for Top-Level Domains (TLDs), because the root is the highest node of the DNS. See Section 3.3 for more discussion.

In both of the scenarios above, a validating resolver will be able to detect the attack if its chain of queries comes to a zone that is signed, but not for those that are unsigned.

7. IANA Considerations

This document does not require any IANA actions.

8. References

8.1. Normative References

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

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

[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>.

[RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver message size requirements", RFC 3226, DOI 10.17487/RFC3226, December 2001, <https://www.rfc-editor.org/info/rfc3226>.

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

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, DOI 10.17487/RFC5452, January 2009, <https://www.rfc-editor.org/info/rfc5452>.

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

[RFC8109] Koch, P., Larson, M., and P. Hoffman, "Initializing a DNS Resolver with Priming Queries", BCP 209, RFC 8109, DOI 10.17487/RFC8109, March 2017, <https://www.rfc-editor.org/info/rfc8109>.

[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>.

[RFC8499] 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>.

8.2. Informative References

[RSSAC023v2] "History of the Root Server System", 2016, <https://www.icann.org/en/system/files/files/rssac-023-17jun20-en.pdf>.

[RSSAC026v2] "RSSAC Lexicon", 2020, <https://www.icann.org/en/system/files/files/rssac-026-lexicon-12mar20-en.pdf>.

Appendix A. Acknowledgements

RFC 8109 was the product of the DNSOP WG and benefitted from the reviews done there.

Authors' Addresses

Peter Koch
DENIC eG
Kaiserstrasse 75-77
Frankfurt 60329
Germany

Phone: +49 69 27235 0
Email: pk@DENIC.DE

Matt Larson
ICANN

Email: matt.larson@icann.org

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

DNSOP WG
Internet-Draft
Intended status: Standards Track
Expires: March 3, 2021

T. Reddy
McAfee
N. Cook
Open-Xchange
D. Wing
Citrix
M. Boucadair
Orange

August 30, 2020

DNS Access Denied Error page
draft-reddy-dnsop-error-page-04

Abstract

When a DNS server filters a query the response conveys no detailed explanation of why that query was blocked, leading to end-user confusion. This document defines a method to return an URI that explains the reason a DNS query was filtered.

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 March 3, 2021.

Copyright Notice

Table of Contents

1. Introduction

DNS filters are deployed for a variety of reasons including endpoint security, parental filtering, and filtering required by law enforcement. These are discussed in more detail below:

- Various network security services are provided by Enterprise networks to protect endpoints (e.g., Hosts including IoT devices). Network-based security solutions such as firewalls and Intrusion Prevention Systems (IPS) rely on network traffic inspection to implement perimeter-based security policies. The network security services may, for example, prevent malware download, block known malicious domains, block phishing sites, etc. These network security services act on DNS queries originating from endpoints. For example, DNS firewalls, a method of expressing DNS response policy information inside specially constructed DNS zones, known as Response Policy Zones (RPZs) allows DNS servers to modify DNS responses in real time to stop access to malware and phishing domains. Note that some of the commonly known types of malware are viruses, worms, trojans, bots, ransomware, backdoors, spyware, and adware.

- Network devices in a home network offer network security to protect the devices connected to the home network by performing DNS-based content filtering. The network security service may,

for example, block access to specific domains to enforce parental control, block access to malware sites, etc.

- ISPs typically block access to some domains due to a requirement imposed by an external entity (e.g., Law Enforcement Agency) by performing DNS-based content filtering.

DNS responses can be filtered by sending a bogus ("forged") A or AAAA response, NXDOMAIN error or empty answer, or an extended error code defined in [I-D.ietf-dnsop-extended-error]. Each of these have advantages and disadvantages, discussed below:

1. The DNS response is forged providing IP addresses that point to a HTTP(S) server alerting the end user of the reason for blocking access to the domain (e.g., malware). When a HTTP(S) enabled domain name is blocked, the network security device presents a block page instead of the HTTP response from the content provider. If an HTTP enabled domain name is blocked, the network security device intercepts the HTTP request and returns a block page over HTTP. If an HTTPS enabled domain is blocked, the block page is also served over HTTPS. In order to return a block page over HTTPS, man in the middle (MITM) is enabled on endpoints by generating a local root certificate and an accompanying (local) public/private key pair. The local root certificate is installed on the endpoint, and the network security device(s) store a copy of the private key. During the TLS handshake, the network security device modifies the certificate provided by the server and (re)signs it with the private key from the local root certificate.

    * However, configuring the local root certificate on endpoints is not viable option in several deployments like Home networks, Schools, Small Office/Home Office (SOHO), and Small/Medium Enterprise (SME). In these cases, the typical behavior is that the forged DNS response directs the user towards a server hosted to display the block page which breaks the TLS connection. For web-browsing this then results in an HTTPS certificate error message indicating that a secure connection could not be established, which gives no information to the end-user about the reason for the error. The typical errors are "The security certificate presented by this website was not issued by a trusted certificate authority" (Internet Explorer/Edge"), "The site’s security certificate is not trusted" (Chrome), "This Connection is Untrusted" (Firefox), "Safari can’t verify the identity of the website..." (Safari on MacOS)".

* Enterprise networks do not assume that all the devices connected to their network are managed by the IT team or Mobile Device Management (MDM) devices, especially in the quite common BYOD ("Bring Your Own Device") scenario. In addition, the local root certificate cannot be installed on IoT devices without a device management tool.

* An end user does not know why the connection was reset and, consequently, may repeatedly try to unsuccessfully reach the domain. Frustrated, the end user may use insecure interfaces to reach the domain, potentially compromising both security and privacy. Furthermore, certificate errors train users to click through certificate errors, which is poor security practice. To eliminate the need for an end user to click through certificate errors, an end user may manually install a local root certificate [Chrome-Install-Cert] on a host device. Doing so, however, is also poor security practice as it creates a security vulnerability that may be exploited by a MITM attack. When the manually installed local root certificate expires, the user has to (again) manually install the new local root certificate.

2. The DNS response is forged to provide a NXDOMAIN response to cause the DNS lookup to terminate in failure. In this case, an end user does not know why the domain cannot be reached, and may repeatedly try to unsuccessfully reach the domain. Frustrated, the end user may use insecure interfaces to reach the domain, potentially compromising both security and privacy.

3. The extended error codes Blocked, Censored, and Filtered defined in [I-D.ietf-dnsop-extended-error] can be returned by the DNS server to provide additional information about the cause of an DNS error. If the extended error code "Forged answer" defined in [I-D.ietf-dnsop-extended-error] is returned by the DNS server, the client can identify the DNS response is forged and the reason for HTTPS certificate error. These extended error codes do not suffer from the limitations discussed in (1) and (2) but the user still does not know the exact reason nor the user is aware of the exact entity blocking the access to the domain. For example, a DNS server may block access domain based on the content category like "Adult Content" to enforce parental control, "Violence & Terrorism" due to an external requirement imposed by an external entity (e.g., Law Enforcement Agency), etc. The content categories for domains cannot be standardized because the classification of domains into content categories is vendor specific, typically ranges from 40 to 100 types of categories depending on the vendor and the categories keep evolving. Further, the threat data used to categorize domains may sometimes

mis-classify domains (e.g., Domains wrongly classified as DGA (Domain Generation Algorithm) by deep learning techniques, domain wrongly classified as phishing due to crowd sourcing, new domains not categorized by the threat data). The end user needs to know the contact details of the IT/InfoSec team to raise a complaint.

No matter which type of response is generated (forged IP address, NXDOMAIN or empty answer, or an extended error code), the user who generated the query has little chance to understand which entity filtered the query, how to report a mistake in the filter, or why the entity filtered it at all. This document describes a mechanism to provide a URI which, when accessed, provides such information to the user.

One of the other benefits of this approach is to eliminate the need to "spoof" block pages for HTTPS resources, as the block page no longer needs to create a signed certificate when blocking a destination. This avoids the need to install a local root certificate authority on those IT-managed devices.

2. Terminology

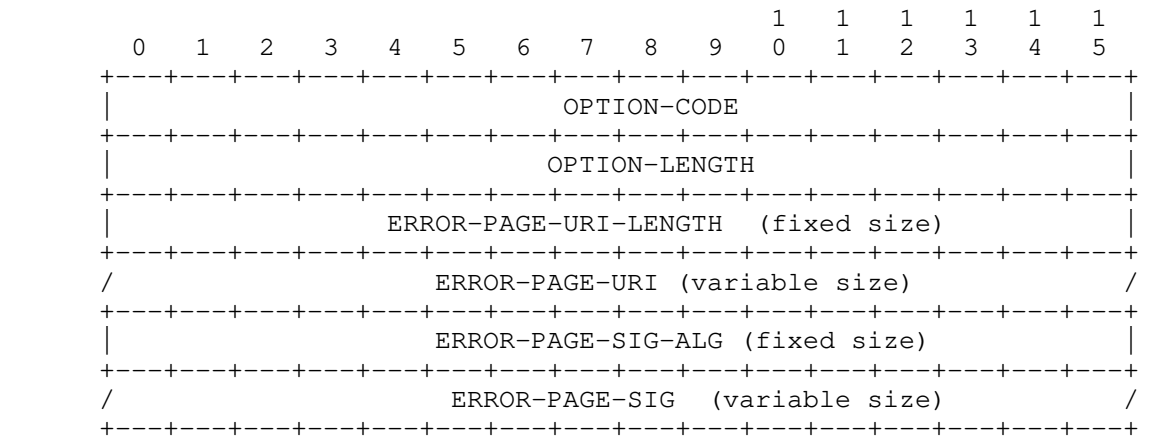
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [RFC8499] and [I-D.ietf-dnsop-terminology-ter].

'Encrypted DNS' refers to DNS-over-HTTPS [RFC8484], DNS-over-TLS [RFC7858], or DNS-over-QUIC [I-D.ietf-dprive-dnsoquic].

3. Error page URI EDNS0 option format

This document uses an EDNS0 [RFC6891] option to include the URI that gives additional information in a DNS response about the cause of blocking access to a domain. The option is structured as follows:



The description of the fields is as follows:

- OPTION-CODE: TBD, indicates the code assigned for Error page URI (Section 6.1.2 of [RFC6891]). [RFC Editor: change TBD to the proper code once assigned by IANA.]

- OPTION-LENGTH: See Section 6.1.2 of [RFC6891]. This fields contains the length of the payload (everything after OPTION-LENGTH) in octets. The variability of the option length stems from the variable-length ERROR-PAGE-URI and ERROR-PAGE-SIG fields.

- ERROR-PAGE-URI-LENGTH: This 16-bit field indicates the length of ERROR-PAGE-URI. It MUST NOT be set to 0.

- ERROR-PAGE-URI: a variable length UTF-8 encoded [RFC5198] text field containing the URI Template [RFC6570] that gives additional information about the cause of blocking access to a domain. The ERROR-PAGE-URI field MUST NOT be zero octets in length.

- ERROR-PAGE-SIG-ALG: A 16-bits field that contains the algorithm used to generate the signature for the Error page URI Template. The values are defined in the TLS SignatureScheme [TLS-SIG-SCHEME] with limitations described in Section 5.

- ERROR-PAGE-SIG, a variable length field containing the signature of the Error page URI Template. The signature generation process is discussed in Section 5.

The Error page URI option MUST only be included in error responses SERVFAIL, NXDOMAIN, or REFUSED to a query that included the OPT Pseudo-RR [RFC6891].

The URI Template defined in ERROR-PAGE-URI describes how to construct the URL to fetch the error page. The agent acting as HTTPS client on the endpoint encodes a FQDN to which access is denied into an HTTP GET request to retrieve the error page. The HTTPS server returning the error page defines the URI used by the HTTP GET request through the use of a URI Template. The URI Template is processed with a defined variable "target-domain" whose value is set to the FQDN to which access is denied. The FQDN is encoded using base64url [RFC4648] and then provided as the variable value for "target-domain" to expand the URI Template into a URI reference in the HTTP GET request. Padding characters for base64url MUST NOT be included.

An example is illustrated below:

If the URI Template is "https://block.example.net/block-page{?target-domain}" for the HTTPS server returning the error page and access to the target domain "example.com" is blocked by the encrypted DNS server, the variable "target-domain" has the value "example.com" base64url encoded into an HTTP GET request. In the above example, the expansion of the above URI Template is "https://block.example.net/block-page?target-domain=ZXhhbXBsZS5jb20".

HTTP/2 [RFC7540] is the minimum RECOMMENDED version of HTTP to use to retrieve the error page. The HTTPS client retrieving the error page MUST verify the entire certification path per [RFC5280]. The HTTPS client additionally uses validation techniques as described in [RFC6125] to compare the domain name in the error page URI to the server certificate provided in TLS handshake. See [RFC7525] for additional TLS recommendations.

4. Error page URI Processing

The DNS client MUST follow the following rules to process the Error page URI EDNS0 option:

- The Error page URI EDNS0 option is susceptible to forgery. In order to defend against this attack the DNS client MUST NOT process the DNS response with Error page URI EDNS0 option unless DNS messages exchanged are cryptographically protected using encrypted DNS.

- If an DNS client has enabled opportunistic privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client will either fallback to an encrypted connection without authenticating the DNS server provided by the local network or fallback to clear text DNS, and cannot exchange encrypted DNS messages. The fallback adversely impacts security and privacy. If the DNS client has

enabled opportunistic privacy profile for DoT, the client MUST NOT process the DNS response with Error page URI EDNS0 option.

- If an DNS client has enabled strict privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client requires an encrypted connection and successful authentication of the DNS server; this mitigates both passive eavesdropping and client redirection (at the expense of providing no DNS service if an encrypted, authenticated connection is not available). If the DNS client has enabled strict privacy profile for DoT, the client can process the DNS response with Error page URI EDNS0 option. Note that the strict and opportunistic privacy profiles as defined in [RFC8310] only applies to DoT protocol, there has been no such distinction made for DoH protocol.

- If the DNS response contains more than one Error page URI EDNS0 option, the DNS client MUST discard multiple Error page URI EDNS0 options in the DNS response.

- The Error page URI EDNS0 option MUST be processed by the DNS client for a "Censored", "Blocked" or "Filtered" extended error codes and MUST be ignored for any other type of extended DNS error code. When "Censored", "Blocked" or "Filtered" extended error code is returned in conjunction with an Error page URI EDNS0 option, any other resource records in the answer MUST be ignored by clients supporting this specification.

- If the encrypted DNS server does not offer DNS filtering service (e.g., prevent access to malware sites or objectionable content (e.g., legal obligation)), the DNS client MUST reject the Error page URI EDNS0 option. The mechanism discussed in [I-D.reddy-add-server-policy-selection] can be used by the DNS client to assess whether the encrypted DNS server performs DNS-based content filtering.

- If the Error page URI EDNS0 option is included in a NOERROR RCODE message, the DNS client MUST reject the Error page URI EDNS0 option.

- The DNS client MUST reject the error page URI if the scheme is not "https".

- The DNS client verifies the signature in the ERROR-PAGE-SIG field following the mechanism discussed in Section 5. If the signature is valid, the client can positively identify that the Error page URI EDNS0 option has been generated by the encrypted DNS server and the encrypted DNS server did not forward the Error Page URI

EDNS0 option from an upstream resolver. If signature validation fails, the DNS client MUST reject the Error page URI EDNS0 option.

Because EDNS is a hop-by-hop extension to DNS and EDNS responses are not cached (Section 6.2.1 of [RFC6891]), a DNS resolver MUST NOT propagate a received Error Page URI EDNS0 option, because an attacker could insert a bogus URI; instead, if access to a domain is denied, the DNS resolver MUST generate its own Error Page URI. Because DNS forwarders (or DNS proxies) are supposed to propagate unknown EDNS0 options (Section 4.1 and Section 4.4.1 of [RFC5625]), the Error Page URI EDNS0 option may get propagated. To detect both of these scenarios, the Error Page URI Template is protected with an object signature as described in Section 5.

5. Sign and Verify

The algorithms for generating signature for DNS resource record sets (RRsets) are defined in [DNSKEY-IANA]. The "mandatory-to-implement" algorithms are RSA, Elliptic Curve Digital Signature Algorithm (ECDSA), and Edwards-curve Digital Security Algorithm (EdDSA) [RFC8624]. Along similar lines, the encrypted DNS server's end-entity certificate's public key and the signature algorithm with which the key can be used are RSA, ECDSA, and EdDSA [RFC8446]. If ECDSA is used, it is RECOMMENDED to use the deterministic digital signature generation procedure of the ECDSA, specified in [RFC6979].

The signature is generated by the encrypted DNS server using the Error page URI Template, private key of the encrypted DNS server's end-entity certificate as inputs to the signature algorithm. The signature algorithm in the ERROR-PAGE-SIG-ALG field MUST be compatible with the key in the DNS server's end-entity certificate. The implementation MUST support the same set of algorithms in the TLS client for validating the signature in the CertificateVerify message from the server in the TLS handshake and in the DNS client to validate the signature for the Error page URI Template. As a reminder, the server's end-entity certificate's public key will be compatible with the selected authentication algorithm from the client's "signature_algorithms" TLS extension (Section 4.4.2.2 of [RFC8624]).

If the signature algorithm in the ERROR-PAGE-SIG-ALG field is not compatible with the key in the DNS server's end-entity certificate, the DNS client MUST reject the Error page URI EDNS0 option. The DNS client verifies the signature using the signature in the ERROR-PAGE-SIG field, error page URI Template and DNS server's end-entity certificate's public key as inputs to the signature algorithm. For example, if Ed25519 is used, Ed25519 signature algorithm and

verification of the Ed25519 signature are described in Sections 5.1.6 and 5.1.7 of [RFC8032], respectively.

6. ERROR Page

The following text outlines the RECOMMENDED contents of an error page to assist the operator developing the error page.

- The exact reason for blocking access to the domain. If the domain is blocked based on some threat data, the threat type associated with the blocked domain can be provided/displayed to the end user. For example, the reason can indicate the type of malware blocked like spyware and the damage it can do the security and privacy of the user.

- The domain name blocked.

- If query was blocked by regulation, a pointer to a regulatory text that mandates this query block.

- The entity (or organization) blocking the access to the domain and contact details of the IT/InfoSec team to raise a complaint.

- The blocked error page to not include Ads and dynamic content.

The content of the error page discussed above is non-normative, the above text only provides the guidelines and template for the error page and.

- Does not attempt to offer an exhaustive list for the contents of an error page.

- It is not intended to form the basis of any legal/compliance for developing the error page.

7. Usability Considerations

The error page SHOULD be returned in the user's preferred language as expressed by the Accept-Language header. If the error page is displayed in a language not known to the end user and assuming Internationalization features failed, browser extensions to translate to user's native language can be used. For example, "Google Translate" extension [Chrome-Translate] provided by Google on Chrome can be used by the user to translate the error page. The "Google Translate" extension automatically detects whether the language of a page is different from the language the user has selected. If it is in a different language, a banner appears at the top of the page.

The user can click on the Translate button in the banner to have all the text on the page appear in the language selected by the user.

8. Security Considerations

Security considerations in [I-D.ietf-dnsop-extended-error] and [RFC8624] need to be taken into consideration.

The Error page URI EDNS0 option causes an HTTPS retrieval by the client. To prevent forgery of the Error page URI EDNS0 option, this specification requires it only be sent only over an encrypted DNS channel with an authorized DNS server.

The DNS client can learn the filtering capability of an encrypted DNS server using [I-D.reddy-add-server-policy-selection]. [I-D.reddy-add-server-policy-selection] also discusses how a DNS client can authenticate it is connecting to an encrypted DNS server hosted by a specific organization (e.g., ISP). This information is cryptographically signed to attest its authenticity. It is particularly useful when the encrypted DNS server is insecurely discovered and prevents the client from connecting to an attacker's encrypted DNS server.

To reduce threat surface when retrieving the Error page URL over HTTPS, the client SHOULD perform the retrieval using an isolated environment. Such isolation should prevent transmitting cookies, block JavaScript, block auto-fill of credentials or personal information, and be isolated from the user's normal environment. Browsers perform some of these limitations today when accessing captive portals [Safari-Cookie], during private browsing, or using containerization [Facebook-Container]).

The encrypted DNS session provides transport security for the interaction between the DNS client and server, but DNSSEC signing and validation is not possible for the Error Page URI EDNS0 option returning the error page URI template. For example, if access to a doman is blocked, the encrypted DNS resolver can rewrite the response to send NXDOMAIN error and Error page URI EDNS0 option in the DNS response, it will omit DNSSEC RRsets, because the modified responses cannot be verified by DNSSEC signatures. However, the signature in the Error Page URI EDNS0 option provides data origin authentication of the Error Page URI EDNS0 option. Nevertheless, the Error Page URI EDNS0 option returning the error page URI Template should be treated only as diagnostic information and MUST NOT alter DNS protocol processing.

By design, the object referenced by the error page URL potentially exposes additional information about the DNS resolution process that

may leak information. An example of this is the reason for blocking the access to the domain name and the entity blocking access to the domain.

9. IANA Considerations

9.1. A New Error Page URI EDNS Option

This document defines a new EDNS(0) option, entitled "Error Page URI", assigned a value of TBD from the "DNS EDNS0 Option Codes (OPT)" registry [to be removed upon publication: [http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11]

Value	Name	Status	Reference
TBD	Error Page URI	Standard	[This document]

10. Acknowledgements

Thanks to Vittorio Bertola, Wes Hardaker, Ben Schwartz, Erid Orth, Viktor Dukhovni and Bob Harold for the comments.

11. References

11.1. Normative References

[I-D.ietf-dnsop-extended-error]
Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", draft-ietf-dnsop-extended-error-16 (work in progress), May 2020.

[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>.

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

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <https://www.rfc-editor.org/info/rfc5625>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

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

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[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>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

[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>.

[RFC8624] Wouters, P. and O. Sury, "Algorithm Implementation Requirements and Usage Guidance for DNSSEC", RFC 8624, DOI 10.17487/RFC8624, June 2019, <https://www.rfc-editor.org/info/rfc8624>.

[TLS-SIG-SCHEME] "IANA, TLS SignatureScheme", <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-signaturescheme>.

11.2. Informative References

[Chrome-Install-Cert] "How to manually install the Securly SSL certificate in Chrome", <support.securly.com/hc/en-us/articles/206081828-How-to-manually-install-the-Securly-SSL-certificate-in-Chrome>.

[Chrome-Translate] "Google Translate", <https://chrome.google.com/webstore/detail/google-translate/aapbdbdomjkkjkaonfhkkikfgjllcleb/RK%3D2/RS%3DBBFW_pnWkPY0xPMYsAZI5xOgQEE->.

[DNSKEY-IANA] "IANA, Domain Name System Security (DNSSEC) Algorithm Numbers", <http://www.iana.org/assignments/dns-sec-alg-numbers>.

[Facebook-Container] "Facebook container for Firefox", <https://www.mozilla.org/en-US/firefox/facebookcontainer/>.

[I-D.ietf-dnsop-terminology-ter] Hoffman, P., "Terminology for DNS Transports and Location", draft-ietf-dnsop-terminology-ter-02 (work in progress), August 2020.

[I-D.ietf-dprive-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnsoquic-00 (work in progress), April 2020.

[I-D.reddy-add-server-policy-selection]
Reddy.K, T., Wing, D., Richardson, M., and M. Boucadair, "DNS Server Selection: DNS Server Information with Assertion Token", draft-reddy-add-server-policy-selection-04 (work in progress), July 2020.

[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>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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>.

[RFC8499] 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>.

[Safari-Cookie]
"Isolated cookie store (CVE-2016-1730)", <https://support.apple.com/en-us/HT205732>.

Authors' Addresses

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: kondtir@gmail.com

Neil Cook
Open-Xchange
UK

Email: neil.cook@noware.co.uk

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

dnsop
Internet-Draft
Intended status: Informational
Expires: 14 January 2021

T. April
Akamai Technologies

13 July 2020

Parameterized Nameserver Delegation with NS2 and NS2T
draft-tapril-ns2-01

Abstract

Within the DNS, there is no mechanism for authoritative servers to advertise which transport methods they are capable of. If secure transport methods are adopted by authoritative operators, transport signaling would be required to negotiate how authoritative servers would be contacted by resolvers. This document provides two new Resource Record Types, NS2 and NS2T, to facilitate this negotiation by allowing zone owners to signal how the authoritative nameserver(s) for their zone(s) may accept queries.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/timapril/ns2 (https://github.com/timapril/ns2).

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 14 January 2021.

Copyright Notice

Table of Contents

1. Introduction

Resolvers currently rely on the NS records in the parent and child zones to provide and confirm the nameservers that are authoritative for each zone. The Nameserver version 2 (NS2) record extends the functionality of the NS record to include additional information about how authoritative zone information can be queried, whether that be over alternate protocols or by using alternate protocol parameters. The NS2 record may be present at zone cuts but can also redirect resolvers to other nameservers for further redirection via the Nameserver Version 2 Target (NS2T) record, which does not indicate a zone cut.

The NS2 and NS2T records uses the SVCB record format defined in [I-D.draft-ietf-dnsop-svcb-https-00], using a subset of the already defined service parameters as well as new parameters described in this document. Some, but not all, of the existing service parameters will also be available for NS2 and NS2T records. This document will outline the available parameters and their usage.

1.1. Introductory Examples

To introduce the NS2 and NS2T records, this example shows a possible response from an authoritative in the authority section of the DNS response when delegating to another nameserver.

```
example.com.  86400  IN NS2    2 ns2.example.com. ( transports=dot,
                dnsTlsFingerprints=["MIIS987SSLKJ...123===",
                    "MII3SODKSLKJ...456==="] )
example.com.  86400  IN NS2    3 ns3.example.com. ( transports=doh,
                dnsDohURITemplate="https://ns.example.net/q/{?dns}" )
example.com.  86400  IN NS     ns1.example.com.
ns1.example.com.    86400   IN  NS  192.0.2.1
ns2.example.com.    86400   IN  NS  192.0.2.2
ns3.example.com.    86400   IN  NS  192.0.2.3
```

In this example, the authoritative nameserver is delegating to both a DNS-over-TLS and DNS-over-HTTPS service running on ns2.example.net and ns3.example.com respectively, for resolvers that support NS2, and also delegating to ns1.example.com which will serve unencrypted DNS over port 53 for those that do not.

Like in SVCB, NS2 and NS2T also offer the ability to use the Alias form delegation. The example below shows an example where example.com is being delegated with a NS2 AliasForm record which can then be further resolved to locate the authroitative nameserver(s).

```
example.com.  86400  IN NS2 0   ns2.example.net.
example.com.  86400  IN NS     ns1.example.com.
ns1.example.com.    86400   IN  NS  192.0.2.1
```

The example.net authoritative server may return the following NS2T records in response to a query as deirected by the above records.

```
ns2.example.net 3600    IN NS2T ns2.example.org. ( transports=dot,
                dnsTlsFingerprints=["MIIS987SSLKJ...123==="] )
ns2.example.net 3600    IN NS2T ns3.example.org. ( transports=doh,
                ddnsDohURITemplate="https://ns.example.net/q/{?dns}")
```

The avbove records indicate to the client that the authoritative nameservers for zones that Alias to ns2.example.net are ns1.example.org and ns2.example.rg with the configuration provided.

Later sections of this document will go into more detail on the resolution process using these records.

1.2. Goal of the NS2 and NS2T Records

The primary goal of the NS2 and NS2T records is to provide zone owners a way to signal to clients how they may receive queries for the records for which they are authoritative. The NS2 and NS2T records are machine readable, can coexist with NS records in the same zone, and do not break software that does not support them.

NS2 and NS2T are designed to allow child zones to publish NS2 and NS2T records, even without support in the parent zone. Lack of parent zone support for NS2 records may arise for technical or policy reasons.

1.3. Terminology

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. NS2 and NS2T Record Types

The SVCB record allows for two types of records, the AliasForm and the ServiceForm. The NS2 record takes advantage of both and each will be described below in depth.

2.1. Difference between the records

This document introduces two different resource record types. Both records have the same functionality, with the difference between them being that the NS2 record MUST only be used at a delegation point while the NS2T, is NS2 Target, record does not indicate that the label is being delegated. For example, take the following NS2 record:

example.com. 86400 IN NS2 1 ns2.example.net. (transports=dot)

When a client receives the above record, the resolver should send queries for any name under example.com to the nameserver at ns2.example.com unless further delegated. By contrast, when presented with the records below:

```
example.com. 86400 IN NS2 0 customer.example.org.
customer.example.org. 86400 IN NS2T 1 ns2.example.net. ( transports=dot )
```

A resolver trying to resolve a name under example.com would get the first record above from the parent authoritative server, .COM, indicating that the NS2T records found at customer.example.org should be used to locate the authoritative nameservers of example.com. The second record above dictates that the authoritative nameserver from records that have aliased to customer.example.org is ns2.example.net, which only accepts queries over DNS-over-TLS. The primary difference between the two records is that the NS2 record means that anything under the record label should be queried at the delegated server while the NS2T record is just for redirection purposes, and any names under the record's label should still be resolved using the same server unless otherwise delegated.

It should be noted that both NS2 and NS2T records may exist for the same label. Below is an example of this:

```
example.com.  86400  IN NS2 0   c1.example.org.
c1.example.org.  86400  IN NS2T 1   ns2.example.net. ( transports=dot )
c1.example.org.  86400  IN NS2  1   ns3.example.net. ( transports=dot )
test.c1.example.org. 600 IN A 192.0.2.1
```

In the above case, the NS2 record for c1.example.org would only be used when trying to resolve names below c1.example.org. This reason is why when an AliasForm NS2 or NS2T record is encountered, the resolver MUST query for the NS2T record associated with the given name.

Since the NS2 record is indicative of a zone cut while NS2T is not, names MUST NOT have but NS2 and NS2T records at the same time.

2.2. AliasForm Record Type

In order to take full advantage of the AliasForm of NS2 and NS2T, the parent, child and resolver must support these records. When supported, the use of the AliasForm will allow zone owners to delegate their zones to another operator with a single record in the parent. AliasForm NS2 records SHOULD appear in the child zone when used in the parent. If a resolver were to encounter an AliasForm NS2 or NS2T record, it would then resolve the name in the SvcDomainName of the original record using NS2T RR type to receive the either another AliasForm record or a ServiceForm NS2T record.

For example, if the name www.example.com was being resolved, the .com zone may issue a referral by returning the following record:

```
example.com.    86400    IN  NS2     0   customer1.example.net.
```

The above record would indicate to the resolver that in order to obtain the authoritative nameserver records for example.com, the resolver should resolve the RR type NS2T for the name customer1.example.net.

2.2.1. Multiple Service Providers

Some zone owners may wish to use multiple providers to serve their zone, in which case multiple NS2 AliasForm records can be used. In the event that multiple NS2 AliasForm records are encountered, the resolver SHOULD pick one of the records at random. For example, to split traffic between two providers, the zone owner for example.com could have the following NS2 records:

```
example.com.    86400    IN  NS2     0   customer1.example.net.
example.com.    86400    IN  NS2     0   customer1.example.org.
```

DRAFT NOTE: SVCB says that there "SHOULD only have a single RR". This ignores that but keep the randomization part. Section 2.5p1 of SVCB

2.2.2. Loop Prevention

Special care should be taken by both the zone owner and the delegated zone operator to ensure that a lookup loop is not created by having two AliasForm records rely on each other to serve the zone. Doing so may result in a resolution loop, and likely a denial of service. Any clients implementing NS2 and NS2T SHOULD implement a per-resolution limit of how many AliasForm records may be traversed when looking up a delegation to prevent infinite looping. When a loop is detected, like with the handling of CNAME or NS, the server should respond to the client with SERVFAIL.

2.3. ServiceForm Record Type

The ServiceForm of the NS2 and NS2T records are likely to be the more popular of the two. They work the same way as the SVCB or HTTPSSVC records do by providing a priority and list of parameters associated with the SvcDomainName. In addition to being able to identify which protocols are supported by the authoriatative server, the ServiceForm record will also allow providers to operate different protocols on different addresses.

2.3.1. SvcFieldPriority

As defined in the DNS SVCB document [I-D.draft-ietf-dnsop-svcb-httpssvc-00], the SvcFieldPriority values SHOULD be used to dictate the priority when multiple NS2 or NS2T records are returned.

2.3.2. SvcDomainName

As defined in the SVCB document [I-D.draft-ietf-dnsop-svcb-httpssvc-00], the SvcDomainName provides the hostname to which the NS2 or NS2T record refers. The wire format must be the a-label format of the name. When presented, the u-label may be presented, but the the a-label should also be displayed displaying the u-label. The SvcDomainName field MUST be set and MUST NOT be "." for NS2 records. Records with a SvcDomainName of "." SHOULD be discarded.

DRAFT NOTE: Should u-label and a-label be expanded? I'm leaning towards not expanding.

2.3.3. SvcParamKeys

The following DNS SVCB parameters are defined for the NS2 and NS2T ServiceForms.

2.3.3.1. "transports"

The "transports" SvcParamKey defines the list of transports offered by the nameserver named in the SvcDomainName.

The existing "alpn" SvcParamKey was not reused for NS2 and NS2T due to the restriction that the "alpn" SvcParamValues are limited to those defined in the TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs registry. Plaintext DNS traffic is not, and should not be listed in that registry, but is required to support the transition to encrypted transport in NS2 and NS2T records.

Below is a list of the transport values defined in this document:

* "do53": indicates that a server supports plaintext, unencrypted DNS traffic over UDP or UDP as defined in [RFC1035] and [RFC1034] and the updates to those RFCs.

* "dot": indicates that the server supports encrypted DNS traffic over DNS-over-TLS as defined in [RFC7858].

* "doh": indicates that the server supports encrypted DNS traffic over DNS-over-HTTPS as defined in [RFC8484]. Records that use the DoH service form may be further redirected with HTTPSSVC records in the delegated zone.

* "doq": indicates that the server supports encrypted DNS traffic over DNS over Dedicated QUIC Connections [I-D.draft-huitema-quic-dnsoquic-07]

The order of the keys in the list dictate the order which the nameserver SHOULD be contacted in. The client SHOULD compare the order of available transports with the set of transports it supports to determine how to contact the selected nameserver.

The presentation format of the SvcParamValue is a comma delimited quoted string of the available transport names. The wire format for the SvcParamValue is a string of 16-bit integers representing the TransportKey values as described in the "NS2/NS2T Transport Parameter Registry".

2.3.3.2. "dnsDotEarlyData"

The "dnsDotEarlyData" SvcParamKey indicates if the server will accept requests with TLS 1.3 Early Data as described in [I-D.draft-ghedini-dprive-early-data-01]. If the "dot" transport is enabled on the record but this value is not present, the default value is that the server will not accept early data.

The presentation format of the SvcParamValue is the string "true" or "false". The wire format of the SvcParamValue is to not have the key present or a single octet with the value of 0x00 when early data is not allowed and a 0x01 value when early data is allowed. All other values should be treated as an error and revert the value to the default of not supported.

DRAFT NOTE: Should this be "ns2flags" and just have a 16 bit field for boolean values?

2.3.3.3. "dnsDohURITemplate"

The "dnsDohURITemplate" SvcParamKey defines the URI template to be used for issuing DNS-over-HTTPS queries to the nameserver defined in the record. The host portion of the "dnsDohURITemplate" value SHOULD match the SvcDomainName field.

In the event that the host portion of the "dnsDohURITemplate" SvcParamValues and SvcDomainName field do not match, the SvcDomainName value SHOULD be used for resolving the host and provide host portion of the "dnsDohURITemplate" template SvcParamValue for the TLS ServerNameIndication header and the HTTP Host header. For example, in the below NS2 delegation, the client SHOULD resolve the name ns.example.net and provide the host header and TLS ServerNameIndication header of doh.example.org:

example.com. 86400 IN NS2 3 ns.example.net. (transports=doh, dnsDohURITemplate="https://doh.example.org/q/{?dns}")

The presentation format of the SvcParamValue is a quoted string. The wire form of the SvcParamValue is an octet string of the URI template as defined in [RFC8484].

2.3.3.4. "esniconfig"

The "esnikeys" SvcParamKey is defined in [I-D.draft-ietf-dnsop-svcb-httpssvc-00]. It can be used to provide the ESNI key for DoT, DoH and/or future protocols which may make use of ESNI for session establishment. See [I-D.draft-ietf-dnsop-svcb-httpssvc-00] for the usage, wire, and display formatting for this SvcParamKey.

2.3.3.5. "dnsTlsFingerprints"

The "dnsTlsFingerprints" SvcParamKey is a list of fingerprints for the TLS certificates that may be presented by the nameserver. This record SHOULD match the TLSA record as described in [RFC6698]. Due to bootstrapping concerns, this SvcParamKey has been added to the NS2 record as the TLSA records would only be resolveable after the initial connetion to the delegated nameserver was established. When this field is not present, certificate validation should be performed by either DANE or by traditional TLS certification validation using trusted root certification authorities.

The presentation and wire format of the SvcParamValue is the same as the presentation and wire format described for the TLSA record as defined in [RFC6698], sections 2.1 and 2.2 respectively.

2.4. Deployment Considerations

The NS2 and NS2T records intends to replace the NS record while also adding additional functionality in order to support additional transports for the DNS. Below are discussions of considerations for deployment.

2.4.1. AliasForm and ServiceForm in the Parent

Both the AliasForm and ServiceForm records MUST NOT be returned for the same record when not in delegated zone. In the case where both are present, the ServiceForm MUST be used and the AliasForm ignored.

DRAFT NOTE: This is in direct conflict with SVCB. I’m currently of the opinion that it should stay as it is for reliability reasons. If it is decided that this should contradict SVCB, maybe we should try to change SVCB.

2.4.2. Rollout

When introduced, the NS2 and NS2T record will likely not be supported by the Root or second level operators, and may not be for some time. In the interim, zone owners may place these records into their zones, both for their own zone and any of their child zones. If a resolver supports alternative transports, it MAY, when delegated to another server, issue a query for NS2 or NS2T records and potentially use those records for further query processing.

DRAFT NOTE: Should we include something here that an authoritative MAY include NS2 records in the additionals section of responses to encourage resolvers to upgrade? What about an ECS option from the authority to signal that it is capable of alternat transports?

2.4.3. Availability

If a zone operator removes all NS records before NS2 and NS2T records are implemented by all clients, the availability of their zones will be impacted for the clients that are using non-supporting resolvers. In some cases, this may be a desired quality, but should be noted by zone owners and operators.

2.4.4. Multiple ServiceForm records for the same host or IP address

As described in the "transport" SvcParamKey section above, a host or IP address may support multiple different transport methods. This can be represented in two ways. The first is to list all supported transports in the order of diminishing desire in the same record. The second is to use multiple NS2 or NS2T records.

When those records have different SvcFieldPriority values, as in [I-D.draft-ietf-dnsop-svcb-httpssvc-00], lower-numbered priorities express a higher preference for that record.

NS2 and NS2T records may have multiple values for the "dnsTlsFingerprints" SvcParamKey. Records that are identical other than the "dnsTlsFingerprints" SvcParamValues may be joined together including multiple "dnsTlsFingerprints" as seen in this example:

```
example.com.  86400  IN NS2    2 ns.example.net. ( transports=dot,
                dnsTlsFingerprints="MIIS987SSLKJ...123===" )
example.com.  86400  IN NS2    2 ns.example.net. ( transports=dot,
                dnsTlsFingerprints="MII3SODKSLKJ...456===" )
example.com.  86400  IN NS2    3 ns.example.net. ( transports=doh,
                dnsDohURITemplate="https://dns.example.org/q/{?dns}" )
```

are the same as:

```
example.com.  86400  IN NS2    2 ns.example.net. ( transports=dot,
                dnsTlsFingerprints=["MIIS987SSLKJ...123===",
                    "MII3SODKSLKJ...456==="] )
example.com.  86400  IN NS2    3 ns.example.net. ( transports=doh,
                dnsDohURITemplate="https://dns.example.org/q/{?dns}" )
```

3. Responses with NS2

The NS2 and NS2T records are intended to supersede the NS record. As such, the NS2 records should be included in the response in the following situations, assuming the records exist in the servers zone:

1) If the qtype is for NS2 or NS2T, the server should respond with NS2 or NS2T respectively in the authority section of the response.

2) For queries over unencrypted TCP port 53, any of the NS2 and NS2T records SHOULD be included in the additional section of the response.

3) For queries over unencrypted UDP port 53, any of the NS2 and NS2T records SHOULD be included in the additional section of the response unless doing so would result in a truncated response. For responses that would require truncation, the resolver operator and/or implementor may decide to truncate the response or exclude the records from the response.

4) If encrypted transports are supported on the authority, the any NS2 and NS2T records should be included in the authority section of the respose.

DRAFT NOTE: It is unknown how resolvers will handle multiple authoritative RRTypes in the authority section of the response, leaving 2 and 3 as the additional section until either testing is done or a consensus is determined in DNSOP/DPRIVE.

DRAFT NOTE: Suggesting authority for encrypted transport since it would more closely align with the NS record.

3.1. Response Size Considerations

For latency-conscious zones, the overall packet size of the delegation records from a parent zone to child zone should be taken into account when configuring the NS, NS2 and NS2T records. Resolvers that wish to receive NS2 and NS2T records in response SHOULD advertise and support a buffer size that is as large as possible, ideally 4096 bytes, to allow the authoritative server to respond without truncating whenever possible.

3.2. When to include glue

Like with NS, when a parent is delegating to a child that is in bailiwick, glue records should be included with responses to enable the client to continue to resolve the names.

The ServiceForm version of NS2 returnes sufficent information to the client communicate with the delegated nameserver over a transport other than Do53, but the AliasForm does not. For this reason, the most common use of the AliasForm record would be to alias to a name that is out of bailiwick to the requested zone, which SHOULD be resolveable without relying on the originally queried zone. In the event of an in-bailiwick AliasForm record, the client must either have a pre-agreed upon configuration for the server or attempt opprotunisitic upgrade of the connection to use non-Do53 for the initial setup.

4. DNSSEC and NS2

Like with NS records, the NS2 records in the child zone SHOULD be signed when the zone is DNSSEC signed. The NS2 records that appear in the parent zone are glue and would not be signed, as is the case with NS records.

NS2T records, SHOULD be signed in a zone which is signed by DNSSEC.

5. Privact Considerations

All of the information handled or transmitted by this protocol is public information published in the DNS.

While the records are transmitting public infomration, resolvers which are making use of records may be attemping to keep the information they are querying private from on-path observers. Privacy consious resolvers should query for this record at the apex of a zone when delegated from the parent to help establish an encrypted channel to the authority.

6. Security Considerations

TODO: Fill this section out

6.1. Availability of zones without NS

6.2. Reflection Attacks

6.3. Parsing

6.4. Availability

6.5. Connetion Failures

When a resolver attempts to access nameserver delegated by a NS2 or NST2 record, if a connection error occurs, such as a certificate mismatch or unreachable server, the resolver SHOULD attempt to connect to the other nameservers delegated to until either exhausting the list or the resolver's policy indicates that they should treat the resoltion as failed.

The failure action when failing to resolve a name with NS2/NS2T due to connection erros is dependant on the resolver operators policies. For resolvers which strongly favor privacy, the operators may wish to return a SERVFAIL when the NS2/NS2T resoltion process completes without successfully contacting a delegated nameserver(s) while opportunisitic privacy resolvers may wish to attempt resolution using any NS records that may be present.

7. IANA Considerations

7.1. New registry for NS2 transports

The "NS2/NS2T Transport Parameter Registry" defines the namespace for parameters, including string representations and numeric values. This registry applies to the "transports" DNS SVCB format, currently impacting the NS2 RR Type.

ACTION: create and include a reference to this registry.

7.1.1. Procedure

A registration MUST include the following fields:

* Name: The transport type key name
* TransportKey: A numeric identifier (range 0-65535)
* Meaning: a short description
* Protocol Specification
* Pointer to specification text

7.1.2. Initial Contents

The "NS2/NS2T Transport Parameter Registry" shall initially be populated with the registrations below:

TransportKey	Name	Meaning	Protocol Specification	Reference
0	key0	Reserved	Reserved	(This Document)
1	do53	Unencrypted, Plaintext DNS over UDP or TCP	RFC1035	(This Document)
2	dot	DNS-over-TLS	RFC7858	(This Document)
3	doh	DNS-over-HTTPS	RFC8484	(This Document)
4	doq	DNS over Dedicated QUIC Connections	[DOQ-I-D]	
65280-65534	keyNNNNN	Private Use	Private Use	(This Document)
65535	key65535	Reserved	Reserved	(This Document)

Table 1

7.2. New SvcParamKey Values

This document defines new SvcParamKey values in the "Service Binding (SVCB) Parameter Registry".

SvcParamKey	NAME	Meaning	Reference
TBD1	transports		(This Document)
TBD2	dnsDotEarlyData		(This Document)
TBD3	dnsDohURITemplate		(This Document)
TBD4	dnsTlsFingerprints		(This Document)

Table 2

8. Informative References

[I-D.draft-ietf-dnsop-svcb-https-00]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-00, 12 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-svcb-https-00.txt>.

[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>.

[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>.

[I-D.draft-ietf-dnsop-svcb-httpssvc-00]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-httpssvc-00, 31 October 2019, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-svcb-httpssvc-00.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>.

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

[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>.

[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>.

[I-D.draft-huitema-quic-dnsoquic-07]
Huitema, C., Shore, M., Mankin, A., Dickinson, S., and J. Iyengar, "Specification of DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-huitema-quic-dnsoquic-07, 7 September 2019, <http://www.ietf.org/internet-drafts/draft-huitema-quic-dnsoquic-07.txt>.

[I-D.draft-ghedini-dprive-early-data-01]
Ghedini, A., "Using Early Data in DNS over TLS", Work in Progress, Internet-Draft, draft-ghedini-dprive-early-data-01, 6 July 2019, <http://www.ietf.org/internet-drafts/draft-ghedini-dprive-early-data-01.txt>.

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

[DOQ-I-D] Huitema, C., Shore, M., Mankin, A., Dickinson, S., and J. Iyengar, "Specification of DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-huitema-quic-dnsoquic-07, 7 September 2019, <http://www.ietf.org/internet-drafts/draft-huitema-quic-dnsoquic-07.txt>.

Appendix A. Acknowledgements

Thank you to John Levine, Erik Nygren, Ralf Weber, Jon Reed, Ben Kaduk, Mashooq Muhaimen, Jason Moreau, Jerrod Wiesman, Billy Tiemann, Gordon Marx and Brian Wellington for their comments and suggestions on this draft.

Appendix B. TODO

RFC EDITOR: PLEASE REMOVE THE THIS SECTION PRIOR TO PUBLICATION.

* Discussion about TTLs and what they should be and how that might impact performance

* Discuss the cacheability of the Alias form records.

* Remove all "DRAFT NOTES:" in the document.

* Write a security considerations section

* add prohibition of AliasForm referring to AliasForm

* add out-of-bailiwick requirement for AliasForm

* worked out resolution example including alias form delegation

* DoH URI teamplte does not include post

* If NS2 is authoritative in the parent, does that mean that it will not be a referral anymore? Probably a question for the working group

Appendix C. Change Log

RFC EDITOR: PLEASE REMOVE THE THIS SECTION PRIOR TO PUBLICATION.

pre-00

* Initial draft.

* Wire and Presentation formats for all new SvcParamKeys and SvcParamValues

* IANA considerations first pass

* Added a section about SvcFieldPriority

* Added the "ds" and "dnskey" SvcParamKey options to support the deprecation of the DS in the parent.

* Added notes on DNSSEC signing of NS2

* Removed multi-provider sharding example, with performance measurements the distribution probably wouldn't work

* Reworked the introduction to try and make it easier to parse

* Removed the port fields for each transport option

* Added a discussion about when to include NS2 records.

* Add an example early in the draft. Introduction area

* Add section about port numbers discussion

* collapse udp and tcp to the do53

* added a second record (NS2 and NS2T)

-01

* Removed DS and DNSKEY SvcParamFields. Avoids issues with DNSSEC signing in the parent.

* Removed IPv{4,6}Hints SvcParamFields. There was a discussion on DNSOP about how glue is required

* Updating when to sign the NS2 / NS2T records (removed signing in the parent)

* Attempting to clean up the introduction, goals and motivations of the document

* Adding a privacy considerations section

* Adding more clairty around when to include/expect the NS2/NS2T records

* Adding this note that CNS2 will not be included in this draft

* Prohibiting NS2 and NS2T from existing at the same name

* Making the statement about the parent records being glue and should not be signed

Appendix D. Discussions

Editor Note: Remove this full section before publication.

D.1. Port Numbers

Originally, I had added SvcParamKeys for port numbers for each of the protocols. There was a discussion that resulted in removing the port numbers, since it was added complexity that had little perceived use in the wild. These can be added back if there is a desire to have them. The original author included them as a way to provide the nameserver operator a way to differentiate incoming traffic when using the aliasform with lower TTLs and intelligent responses based on the client IP.

D.2. CNS2

Client NS2, similar to CDS might be a way to provide support for getting NS2 records into the parent zone before going through the registrars, but that might be a tough thing to agree on at this point.

D.3. Second Record Name

Selected NS2T for Nameserver 2 Target since the record defines the target authoritative servers.

~~~ 01234567890123456789012345678901234567890123456789012345678901234567891

Author's Address

Tim April
Akamai Technologies

Email: ietf@tapril.net
~~~