deleg
Internet-Draft
Updates: 1034, 1035, 6672, 6840 (if approved)
Intended status: Standards Track
Expires: 8 January 2026

T. April
Google, LLC
P. paek
ISC
R. Weber
Akamai Technologies
D. Lawrence
Salesforce
7 July 2025

Extensible Delegation for DNS
draft-ietf-deleg-01

Abstract

A delegation in the Domain Name System (DNS) is a mechanism that enables efficient and distributed management of the DNS namespace. It involves delegating authority over subdomains to specific DNS servers via NS records, allowing for a hierarchical structure and distributing the responsibility for maintaining DNS records.

An NS record contains the hostname of the nameserver for the delegated namespace. Any facilities of that nameserver must be discovered through other mechanisms. This document proposes a new extensible DNS record type, DELEG, for delegation of the authority for a domain. Future documents then can use this mechanism to use additional information about the delegated namespace and the capabilities of authoritative nameservers for the delegated namespace.

About This Document

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

The latest revision of this draft can be found at https://github.com/ietf-wg-deleg/draft-ietf-deleg-base/tree/gh-pages. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-deleg/.

Discussion of this document takes place on the deleg Working Group mailing list (mailto:dd@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/dd/. Subscribe at https://www.ietf.org/mailman/listinfo/dd/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-deleg/draft-ietf-deleg-base/.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In the Domain Name System [STD13], subdomains within the domain name hierarchy are indicated by delegations to servers which are authoritative for their portion of the namespace. The DNS records that do this, called NS records, contain hostnames of nameservers, which resolve to addresses. No other information is available to the resolver. It is limited to connect to the authoritative servers over UDP and TCP port 53. This limitation is a barrier for efficient introduction of new DNS technology.

The proposed DELEG record type remedies this problem by providing extensible parameters to indicate capabilities and additional information, such as glue that a resolver may use for the delegated authority. It is authoritative and thus signed in the parent side of the delegation making it possible to validate all delegation parameters (names and glue records) with DNSSEC.

This document only shows how DELEG can be used instead of or along side a NS record to create a delegation. Future documents can use the extensible mechanism for more advanced features like connecting to a name server with an encrypted transport.

1.1. 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.

Terminology regarding the Domain Name System comes from [BCP219], with addition terms defined here:

* legacy name servers: An authoritative server that does not support the DELEG record.

* legacy resolvers: A resolver that does not support the DELEG record.

2. DELEG Record Type

The DELEG record uses a new resource record type, whose contents are identical to the SVCB record defined in [RFC9460]. For extensions SVCB and DELEG use Service Parameter Keys (SvcParamKeys) and new SvcParamKeys that might be needed also will use the existing IANA Registry.

2.1. Differences from SVCB

* DELEG can only have two priorities 0 indicating INCLUDE and 1 indicating a DIRECT delegation. These terms MUST be used in the presentation format of the DELEG record.

* INCLUDE and DIRECT delegation can be mixed within an RRSet.

* The final INCLUDE target is an SVCB record, though there can be further indirection using CNAME or AliasMode SVCB records.

* There can be multiple INCLUDE DELEG records, but further indirections through SVCB records have to comply with [RFC9460] in that there can be only one AliasMode SVCB record per name.

* In order to not allow unbounded indirection of DELEG records the maximum number of indirections, CNAME or AliasMode SVCB is 4.

* The SVCB IPv4hint and IPv6hint parameters keep their key values of 4 and 6, but the presentation format with DELEG MUST be Glue4 and Glue6.

* Glue4 and Glue6 records when present MUST be used to connect to the delegated name server.

* The target of any DELEG record MUST NOT be '.'

* The target of a DELEG INCLUDE record MUST be outside of the delegated domain.

* The target of a DELEG DIRECT record MUST be a domain below the delegated domain.

3. Use of DELEG record

A DELEG RRset MAY be present at a delegation point. The DELEG RRset MAY contain multiple records. DELEG RRsets MUST NOT appear at a zone's apex.

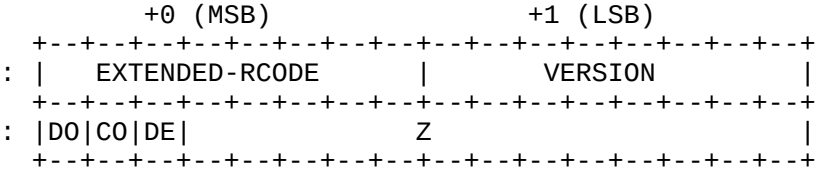
A DELEG RRset MAY be present with or without NS or DS RRsets at the delegation point.

3.1. Resolvers

3.1.1. Signaling DELEG support

A resolver that is DELEG aware MUST signal its support by sending the DE bit when iterating.

This bit is referred to as the "DELEG" (DE) bit. In the context of the EDNS0 OPT meta-RR, the DE bit is the TBD of the "extended RCODE and flags" portion of the EDNS0 OPT meta-RR, structured as follows (to be updated when assigned):



Setting the DE bit to one in a query indicates the resolver understands new DELEG semantics and does not need NS RR to follow a referral. The DE bit cleared (set to zero) indicates the resolver is unprepared to handle DELEG and hence can only be served NS, DS and glue in a delegation response.

Motivation: For a long time there will be both DELEG and NS needed for delegation. As both methods should be configured to get to a proper resolution it is not necessary to send both in a referral response. We therefore purpose an EDNS flag to be use similar to the DO Bit for DNSSEC to be used to signal that the sender understands DELEG and does not need NS or glue information in the referral.

3.1.2. Referral

The DELEG record creates a zone cut similar to the NS record.

If a DELEG record exists on a given delegation point, all record types defined as authoritative in the child zone MUST be resolved using the name servers defined in the DELEG record. In such case resolver MUST NOT use NS records even if they happen to be present in cache, even if resolution using DELEG records have failed for some reason. Such fallback from DELEG to NS would invalidate security guarantees of DELEG protocol.

If no DELEG record exists on a given delegation point resolver MUST use NS records as specified by RFC1034.

3.1.3. Parent-side types, QTYPE=DELEG

Record types defined as authoritative on the parent side of zone cut (currently DS and DELEG types) retain the same special handling as before, i.e. [RFC4035] section 2.6 applies.

DELEG unaware recursive resolvers will not be able to determine correct NS set for QTYPE=DELEG queries. This is not a bug.

3.1.4. Algorithm

This section updates instructions for step "2. Find the best servers to ask." of RFC1034 section 5.3.3 and [RFC6672] section 3.4.1.

There are two important details:

* The algorithm description should explicitly describe RR types authoritative at the parent side of a zone cut. This is implied by [RFC4035] section 3.1.4.1 for DS RR type but the text in the algorithm description was not updated. DELEG specification simply extends this existing behavior to DELEG RR type as well, and makes this special case explicit.

* When DELEG RRset exists, NS RRset is ignored on that particular zone cut by DELEG aware resolvers.

* DELEG and NS RR types can be used differently at each delegation level and resolver MUST be able follow chain of delegations which combines them in arbitrary ways.

Example of a valid delegation tree:

```
; root zone with NS-only delegations
. SOA ...
test. NS ...

; test. zone with NS+DELEG delegations
test. SOA ...
sld.test. NS ...
sld.test. DELEG ...

; sld.test. zone with NS-only delegation
sld.test. SOA ...
nssub.sld.test. NS ...

; nssub.sld.test. zone with DELEG-only delegation
delegsub.sub.sld.test. DELEG ...
```

Terms SNAME and SLIST used in the rest of this section are defined in RFC 1034 section 5.3.2.:

SNAME the domain name we are searching for.

SLIST a structure which describes the name servers and the zone which the resolver is currently trying to query.

Modified description of Step 2. Find the best servers to ask follows:

Step 2 looks for a name server to ask for the required data.

First determine deepest possible zone cut which can potentially hold the answer for given (query name, type, class) combination:

* Start with SNAME equal to QNAME.

* If QTYPE is a type authoritative at the parent side of a zone cut (DS or DELEG), remove leftmost label from SNAME. E.g. if QNAME is Test.Example. and QTYPE is DELEG or DS, set SNAME to Example.

* TODO: what to do about ". DELEG" (or DS) query? That leaves zero labels left. That by definition does not exist ...

Further general strategy is to look for locally-available DELEG and NS RRsets, starting at current SNAME. If none are found, shorten SNAME by removing leftmost label and check again. This effectivelly finds the deepest known delegation point on the path between SNAME and the root:

* For given SNAME first check existence of DELEG RRset. If it exists, resolver MUST use it's content to populate SLIST. If the DELEG RRset is known to exist but is unusable (e.g. it is found in DNSSEC BAD cache), resolver MUST NOT fallback to NS RRset, even if it is locally available. Resolver MUST treat this case as if no servers were available/reachable.

* If a given SNAME is proven to not have a DELEG RRset but has NS RRset, resolver MUST copy it into SLIST.

* If SLIST is populated, terminate walk up the DNS tree.

* If SLIST is not populated, remove leftmost label from SNAME and inspect RR types for this new SNAME.

Rest of the Step 2's description is not affected by this document.

Please note the instructions to "Bound the amount of work" further down in the original text to apply. Suitable limits MUST be enforced to limit damage EVEN IF SOMEONE HAS INCORRECTLY CONFIGURED SOME DATA.

3.2. Authoritative Servers

DELEG-aware authoritative servers act differently when handling queries from DELEG-unaware clients (those with DE=0) and queries from DELEG-aware clients (those with DE=1).

The server MUST copy the value of the DE bit from the query into the response. (TODO: not really necessary protocol-wise, but might be nice for monitoring the deployment?)

3.2.1. DELEG-unaware Clients

DELEG-unaware clients do not use DELEG records for delegation. When a DELEG-aware authoritative server responds to a DELEG-unaware client, any DELEG RR in the response does not create zone cut, is not returned in referral responses, and is not considered authoritative on the parent side of a zone cut. Because of this, DELEG-aware authoritative servers MUST answer as if they are DELEG-unaware. Please note this instruction does not affect DNSSEC signing, i.e. no special handling for NSEC type bitmap is necessary and DELEG RR type is accuratelly represented even for DELEG-unaware clients.

Two surprising narrow cases of DELEG-aware authoritative responding in DELEG-unaware manner are described here.

3.2.1.1. DELEG-unaware Clients Requesting QTYPE=DELEG

In DELEG-unaware clients, records with the DELEG RRtype are not authoritative on the parent side. Thus, queries with DE=0 and QTYPE=DELEG MUST result in a legacy referral response.

3.2.1.2. DELEG-unaware Clients with DELEG RRs Present but No NS RRs

DELEG-unaware clients might ask for a name which belongs to a zone delegated only with DELEG RRs (that is, without any NS RRs). Such zone is, by definition, not resolvable for DELEG-unaware clients. In this case the DELEG RR itself cannot create a zone cut, and the DELEG-aware authoritative server MUST return a legacy response.

The legacy response might be confusing for subdomains of zones which actually exist because DELEG-aware clients would get a different answer, namely a delegation. Example of a legacy response is in Appendix A.3.2.

The authoritative server is RECOMMENDED to supplement DELEG unaware response with Extended DNS Error "New Delegation Only".

TODO: debate if WG wants to do explicit SERVFAIL for this case instead of ’just’ EDE.

3.2.2. DELEG-aware Clients

When the client indicates that it is DELEG-aware by setting DE=1 in the query, DELEG-aware authoritative servers treat DELEG records as zone cuts, and the servers are authoritative on parent side of zone cut. This new zone cut has priority over legacy delegation with NS RRset.

3.2.2.1. DELEG-aware Clients Requesting QTYPE=DELEG

An explicit query for DELEG RR type at a delegation point behaves much like query for DS RR type: the server answers authortiatively from the parent zone. All previous specifications for special handling QTYPE=DS apply equally to QTYPE=DELEG. In summary, server either provides authoritative DELEG RRset or proves its non-existence.

3.2.2.2. Delegation with DELEG

If the delegation has a DELEG RRset, the authoritative server MUST put the DELEG RRset into the Authority section of the referral. In this case, the server MUST NOT include the NS RRset into the Authority section. Presence of the covering RRSIG follows the normal DNSSEC specification for answers with authoritative zone data.

Similarly, rules for DS RRset inclusion into referrals apply as specified by DNSSEC protocol.

3.2.2.3. DELEG-aware Clients with NS RRs Present but No DELEG RRs

If the delegation does not have a DELEG RRset, the authoritative server MUST put the NS RRset into the authority section of the referral. Absence of DELEG RRset must be proven as specified by DNSSEC protocol for authoritative data.

Similarly, rules for DS RRset inclusion into referrals apply as specified by the DNSSEC protocol. Please note in practice the same process and records are used to prove non-existence of DELEG and DS RRsets.

3.3. DNSSEC Signers

The DELEG record is authoritative on the parent side of a zone cut and needs to be signed as such. Existing rules from DNSSEC specification apply. In summary: For DNSSEC signing, treat DELEG RR type the same way as DS RR type.

In order to protect validators from downgrade attacks this draft introduces a new DNSKEY flag ADT (Authoritative Delegation Types). In zones which contain a DELEG RRset this flag MUST be set to one in at least one DNSKEYs published in the zone.

3.4. DNSSEC Validators

DELEG awareness introduces additional requirements on validators.

3.4.1. Clarifications on Nonexistence Proofs

This document updates [RFC6840] section 4.1 to include "NS or DELEG" types in type bitmap as indication of a delegation point and generalizes applicability of Ancestor delegation proof to all types authoritative at parent (i.e. DS and DELEG). Updated text follows:

An "ancestor delegation" NSEC RR (or NSEC3 RR) is one with:

* the NS and/or DELEG bit set,

* the Start of Authority (SOA) bit clear, and

* a signer field that is shorter than the owner name of the NSEC RR, or the original owner name for the NSEC3 RR.

Ancestor delegation NSEC or NSEC3 RRs MUST NOT be used to assume nonexistence of any RRs below that zone cut, which include all RRs at that (original) owner name other types authoritative at the parent-side of zone cut (DS and DELEG), and all RRs below that owner name regardless of type.

3.4.2. Insecure Delegation Proofs

This document updates [RFC6840] section 4.4 to include secure DELEG support and explicitly states Opt-Out is not applicable to DELEG. Updated text follows:

Section 5.2 of [RFC4035] specifies that a validator, when proving a delegation is not secure, needs to check for the absence of the DS and SOA bits in the NSEC (or NSEC3) type bitmap. The validator also MUST check for the presence of the NS or DELEG bit in the matching NSEC (or NSEC3) RR (proving that there is, indeed, a delegation). Alternately make sure that the delegation with NS record is covered by an NSEC3 RR with the Opt-Out flag set. Opt-Out is not applicable to DELEG RR type because this it is authoritative at the parent side of a zone cut in the same say as DS RR type.

3.4.3. Referral downgrade protection

When DNSKEY flag ADT is set to one, the DELEG aware validator MUST prove absence of a DELEG RRset in referral responses from this zone.

Without this check, an attacker could strip DELEG RRset from a referral response and replace it with an unsigned (and potentially malicious) NS RRset. A referral response with an unsigned NS and signed DS RRsets does not require additional proofs of nonexistance according to pre-DELEG DNSSEC specification and it would have been accepted as a delegation without DELEG RRset.

3.4.4. Chaining

A Validating Stub Resolver that is DELEG aware has to use a Security-Aware Resolver that is DELEG aware and if it is behind a forwarder this has to be security and DELEG aware as well.

4. IANA Considerations

IANA is requested to allocate the DELEG RR in the Resource Record (RR) TYPEs registry, with the meaning of "enhanced delegation information" and referencing this document.

IANA is requested to assign a new bit in the DNSKEY RR Flags registry ([RFC4034]) for the ADT bit (N), with the description "Authoritative Delegation Types" and referencing this document. For compatibility reasons we request the bit 14 to be used. This value has been proven to work whereas bit 0 was proven to break in practical deployments (because of bugs).

IANA is requested to assign a bit from the EDNS Header Flags registry ([RFC6891]), with the abbreviation DE, the description "DELEG enabled" and referencing this document.

IANA is requested to assign a value from the Extended DNS Error Codes ([RFC8914]), with the Purpose "New Delegation Only" and referencing this document.

For the RDATA parameters to a DELEG RR, the DNS Service Bindings (SVCB) registry ([RFC9460]) is used. This document requests no new assignments to that registry, though it is expected that future DELEG work will.

5. References

5.1. Normative References

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

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

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

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <https://www.rfc-editor.org/rfc/rfc6840>.

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

[RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <https://www.rfc-editor.org/rfc/rfc8914>.

[RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[STD13] Internet Standard 13, <https://www.rfc-editor.org/info/std13>. At the time of writing, this STD comprises the following:

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

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

5.2. Informative References

[BCP219] Best Current Practice 219, <https://www.rfc-editor.org/info/bcp219>. At the time of writing, this BCP comprises the following:

Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

[I-D.tapril-ns2] April, T., "Parameterized Nameserver Delegation with NS2 and NS2T", Work in Progress, Internet-Draft, draft-tapril-ns2-01, 13 July 2020, <https://datatracker.ietf.org/doc/html/draft-tapril-ns2-01>.

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

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

Appendix A. Examples

The following example shows an excerpt from a signed root zone. It shows the delegation point for "example." and "test."

The "example." delegation has DELEG and NS records. The "test." delegation has DELEG but no NS records.

```
example.   300 IN DELEG DIRECT a.example. Glue4=192.0.2.1 (
                        Glue6=2001:DB8::1 )
example.   300 IN DELEG INCLUDE ns2.example.net.
example.   300 IN DELEG INCLUDE ns3.example.org.
example.   300 IN RRSIG DELEG 13 4 300 20250214164848 (
                        20250207134348 21261 . HyDHYVT5KcqWc7J..= )
example.   300 IN NS    a.example.
example.   300 IN NS    b.example.net.
example.   300 IN NS    c.example.org.
example.   300 IN DS    65163 13 2 5F86F2F3AE2B02...
example.   300 IN RRSIG DS 13 4 300 20250214164848 (
                        20250207134348 21261 . O0k558jHhyrC21J..= )
example.   300 IN NSEC  a.example. NS DS RRSIG NSEC DELEG
example.   300 IN RRSIG NSEC 13 4 300 20250214164848 (
                        20250207134348 21261 . 1Kl8vab96gG21Aa..= )
a.example. 300 IN A     192.0.2.1
a.example. 300 IN AAAA  2001:DB8::1
```

The "test." delegation point has a DELEG record and no NS record.

```
test.      300 IN DELEG INCLUDE ns2.example.net
test.      300 IN RRSIG DELEG 13 4 300 20250214164848 (
                        20250207134348 21261 . 98Aac9f7A1Ac26Q..= )
test.      300 IN NSEC  a.test. RRSIG NSEC DELEG
test.      300 IN RRSIG NSEC 13 4 300  20250214164848 (
                        20250207134348 21261 . kj7YY5tr9h7UqlK..= )
```

A.1. Responses

The following sections show referral examples:

A.2. DO bit clear, DE bit clear

A.2.1. Query for foo.example

```
;; Header: QR RCODE=0 ;;

;; Question foo.example.  IN MX

;; Answer ;; (empty)

;; Authority example.  300 IN NS a.example. example.  300 IN NS
b.example.net. example.  300 IN NS c.example.org.

;; Additional a.example. 300 IN A 192.0.2.1 a.example. 300 IN AAAA
2001:DB8::1
```

A.2.2. Query for foo.test

```
;; Header: QR AA RCODE=3 ;;

;; Question foo.test.  IN MX

;; Answer ;; (empty)

;; Authority .  300 IN SOA ...

;; Additional ;; OPT with Extended DNS Error: New Delegation Only
```

A.3. DO bit set, DE bit clear

A.3.1. Query for foo.example

```
   ;; Header: QR DO RCODE=0
   ;;

   ;; Question
   foo.example.   IN MX

   ;; Answer
   ;; (empty)

   ;; Authority

   example.   300 IN NS    a.example.
   example.   300 IN NS    b.example.net.
   example.   300 IN NS    c.example.org.
   example.   300 IN DS    65163 13 2 5F86F2F3AE2B02...
   example.   300 IN RRSIG DS 13 4 300 20250214164848 (
                           20250207134348 21261 . O0k558jHhyrC21J..= )
   ;; Additional
   a.example. 300 IN A     192.0.2.1
   a.example. 300 IN AAAA  2001:DB8::1
```

A.3.2. Query for foo.test

```
   ;; Header: QR DO AA RCODE=3
   ;;

   ;; Question
   foo.test.      IN MX

   ;; Answer
   ;; (empty)

   ;; Authority
   .          300 IN SOA ...
   .          300 IN RRSIG SOA ...
   .          300 IN NSEC  aaa NS SOA RRSIG NSEC DNSKEY ZONEMD
   .          300 IN RRSIG NSEC 13 4 300
   test.      300 IN NSEC  a.test. RRSIG NSEC DELEG
   test.      300 IN RRSIG NSEC 13 4 300  20250214164848 (
                           20250207134348 21261 . aBFYask;djf7UqlK..= )

   ;; Additional
   ;; OPT with Extended DNS Error: New Delegation Only
```

A.4. DO bit clear, DE bit set

A.4.1. Query for foo.example

```
   ;; Header: QR DE RCODE=0
   ;;

   ;; Question
   foo.example.  IN MX

   ;; Answer
   ;; (empty)

   ;; Authority
   example.   300 IN DELEG DIRECT a.example. Glue4=192.0.2.1 (
                          Glue6=2001:DB8::1 )
   example.   300 IN DELEG INCLUDE ns2.example.net.
   example.   300 IN DELEG INCLUDE ns3.example.org.

   ;; Additional
   ;; (empty)
```

A.4.2. Query for foo.test

```
   ;; Header: QR AA RCODE=0
   ;;

   ;; Question
   foo.test.   IN MX

   ;; Answer
   ;; (empty)

   ;; Authority
   test.      300 IN DELEG INCLUDE ns2.example.net

   ;; Additional
   ;; (empty)
```

A.5. DO bit set, DE bit set

A.5.1. Query for foo.example

```
   ;; Header: QR DO DE RCODE=0
   ;;

   ;; Question
   foo.example.  IN MX

   ;; Answer
   ;; (empty)

   ;; Authority

   example.   300 IN DELEG DIRECT a.example. Glue4=192.0.2.1 (
                          Glue6=2001:DB8::1 )
   example.   300 IN DELEG INCLUDE ns2.example.net.
   example.   300 IN DELEG INCLUDE ns3.example.org.
   example.   300 IN RRSIG DELEG 13 4 300 20250214164848 (
                          20250207134348 21261 . HyDHYVT5KcqWc7J..= )
   example.   300 IN DS    65163 13 2 5F86F2F3AE2B02...
   example.   300 IN RRSIG DS 13 4 300 20250214164848 (
                          20250207134348 21261 . O0k558jHhyrC21J..= )

   ;; Additional
   a.example. 300 IN A     192.0.2.1
   a.example. 300 IN AAAA  2001:DB8::1
```

A.5.2. Query for foo.test

```
   ;; Header: QR DO DE AA RCODE=0
   ;;

   ;; Question
   foo.test.      IN MX

   ;; Answer
   ;; (empty)

   ;; Authority
   test.      300 IN DELEG INCLUDE ns2.example.net.
   test.      300 IN RRSIG DELEG 13 4 300 20250214164848 (
                          20250207134348 21261 . 98Aac9f7A1Ac26Q..= )
   test.      300 IN NSEC  a.test. RRSIG NSEC DELEG
   test.      300 IN RRSIG NSEC 13 4 300  20250214164848 (
                          20250207134348 21261 . kj7YY5tr9h7UqlK..= )

   ;; Additional
   ;; (empty)
```

Appendix B. Acknowledgments {:unnumbered}

This document is heavily based on past work done by Tim April in [I-D.tapril-ns2] and thus extends the thanks to the people helping on this which are: John Levine, Erik Nygren, Jon Reed, Ben Kaduk, Mashooq Muhaimen, Jason Moreau, Jerrod Wiesman, Billy Tiemann, Gordon Marx and Brian Wellington.

Appendix C. TODO

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

* Write a security considerations section
* Change the parameters form temporary to permanent once IANA assigned. Temporary use:
  - DELEG QType code is 65432
  - DELEG EDNS Flag Bit is 3
  - DELEG DNSKEY Flag Bit is 0

Appendix D. Change Log

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

D.1. since draft-wesplaap-deleg-00

* Clarified SVCB priority behavior
* Added section on differences to draft-homburg-deleg-incremental-deleg

D.2. since draft-wesplaap-deleg-01

* Reorganised and streamlined the draft to the bare minimum for DELEG as an NS replacement
* Defined codepoints for temporary testing
* Added examples

Contributors

Christian Elmerot
Cloudflare
Email: christian@elmerot.se

Edward Lewis
ICANN
Email: edward.lewis@icann.org

Roy Arends
ICANN
Email: roy.arends@icann.org

Shumon Huque
Salesforce
Email: shuque@gmail.com

Klaus Darilion
nic.at
Email: klaus.darilion@nic.at

Libor Peltan
CZ.nic
Email: libor.peltan@nic.cz

Vladimr unt
CZ.nic
Email: vladimir.cunat@nic.cz

Shane Kerr
NS1
Email: shane@time-travellers.org

David Blacka
Verisign
Email: davidb@verisign.com

George Michaelson
APNIC
Email: ggm@algebras.org

Ben Schwartz
Meta
Email: bemasc@meta.com

Jan Velk
NS1
Email: jvcelak@ns1.com

Peter van Dijk
PowerDNS
Email: peter.van.dijk@powerdns.com

Philip Homburg
NLnet Labs
Email: philip@nlnetlabs.nl

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

Vandan Adhvaryu
Team Internet
Email: vandan@adhvaryu.uk

Manu Bretelle
Meta
Email: chantr4@gmail.com

Bob Halley
Cloudflare
Email: bhalley@cloudflare.com

Authors' Addresses

Tim April
Google, LLC
Email: ietf@tapril.net

Petr paek
ISC
Email: pspacek@isc.org

Ralf Weber
Akamai Technologies
Email: rweber@akamai.com

David C Lawrence
Salesforce
Email: tale@dd.org