

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 14, 2017

G. Huston  
APNIC  
P. Koch  
DENIC eG  
A. Durand  
ICANN  
W. Kumari  
Google  
September 10, 2016

Problem Statement for the Reservation of Special-Use Domain Names using  
RFC6761  
draft-adpkja-dnsop-special-names-problem-06

Abstract

The dominant protocol for name resolution on the Internet is the Domain Name System (DNS). However, other protocols exist that are fundamentally different from the DNS, and may or may not share the same namespace.

When an end-user triggers resolution of a name on a system that supports multiple, different protocols or resolution mechanisms, it is desirable that the protocol used is unambiguous, and that requests intended for one protocol are not inadvertently answered using another protocol.

RFC 6761 introduced a framework by which a particular domain name could be acknowledged as being special. Various challenges have become apparent with this application of the guidance provided in RFC 6761. This document focuses solely on documenting the specific challenges created by RFC 6761 in the form of a problem statement in order to facilitate further discussions of potential solutions. In particular, it refrains from proposing or promoting any solution. Also, the current document does not focus on other general issues related to the use of special use domain names.

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 <http://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 14, 2017.

#### Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

#### Table of Contents

1. Introduction: DNS, Name Space or Name Spaces, Name Resolution Protocols . . . . .	3
2. IETF RFC6761 Special Names . . . . .	4
3. Issues with RFC 6761 process . . . . .	4
4. Issues with the RFC6761 registry . . . . .	5
5. Issues Around the IETF Evaluation of Candidate Strings . . . .	6
6. Other Issues Related to RFC6761 . . . . .	6
7. Security Considerations . . . . .	6
8. IANA Considerations . . . . .	7
9. Acknowledgements . . . . .	7
10. References . . . . .	7
10.1. Normative References . . . . .	7
10.2. Informative References . . . . .	7
Appendix A. Editorial Notes . . . . .	8
A.1. Venue . . . . .	8
A.2. Change History . . . . .	8
A.2.1. draft-adpkja-special-names-problem-00 . . . . .	8
Appendix B. Change history . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction: DNS, Name Space or Name Spaces, Name Resolution Protocols

For a very long time, "DNS" and "the name space" have been perceived as the same thing. However, this has not always been the case; in the past, other name resolution protocols (such as NIS, NIS+, host files, UUCP addresses, and others) were popular. Most of those have been obsoleted by the DNS in the late 1990s. More information on the history of names and namespaces can be found in [I-D.lewis-domain-names].

More recently, new name resolution protocols have been proposed, each addressing a particular need or a particular community. For example, the DONA handle system [DONA] has been used by parts of the publication industry. The Apple "Bonjour" set of protocols, inspired by what was available on Appletalk networks, was developed to perform automatic name resolution on a local IP network. The TOR project is using the onion system to obfuscate communications, the GNU Name System (GNS) system is using block chains to build a decentralized name system to offer "privacy and censorship resistance". Many more name resolution protocols have been proposed.

These alternate name resolution protocols do not exist in a vacuum. Application developers have expressed a strong desire to build their software to function in any of those universes with minimal changes. In order to do so, the software has to deterministically recognize what kind of name it is dealing with and associate it with the corresponding name resolution protocol. An algorithmic solution frequently chosen by application developers consists simply in using a special tag padded at the end of a name to indicate an alternate name resolution method. For example, if a name ends in .local, the software uses the Apple Bonjour protocol based on multicast DNS; if the name ends in .onion, it uses the TOR protocol; if the name ends in .gnu, it uses the GNS protocol, and so on. One noteworthy exception to this approach is the DONA system that has its own interoperability mechanism with the DNS. Another noteworthy exception is the Frogans technology [FROGANS] which name space uses the character '\*' to separate network names from site names and allow any character, including dots on either side of the '\*'.

A result of the above is that a number of applications have been developed (and massively distributed) that have encoded their favorite "tag" as a DNS TLD in a free-for-all, beginning their existence by squatting on that DNS space; .local, .gnu, .onion started out like that.

## 2. IETF RFC6761 Special Names

The IETF used a provision from the IETF/ICANN MoU [RFC2860] section 4.3 that says that "(a) assignments of domain names for technical uses" is to be considered the purview of IETF (outside of the scope of ICANN) in order to create a way to reserve such names in a list of "special names". That process is documented in [RFC6761] (which, however, does not directly refer the IETF/ICANN MoU). The [RFC6761] process was first applied for .local, and the more recently for .onion.

When the [RFC6761] process was put in place, many thought it would only be used a handful of times. However, a large number of applications have since been made to the IETF. The .onion evaluation took almost a year and has started a massive (and often heated) discussion in the IETF.

[RFC6761] has a number of issues. This document groups those issues in several categories.

## 3. Issues with RFC 6761 process

- a. [RFC6761] does not mention if the protocol using the reserved name should be published as an RFC document.

Most applications have, so far, come from outside organizations, and the described protocols that have not been developed by the IETF.

- b. [RFC6761] does not provide clear enough directions as to which group of people is responsible for carrying out the evaluation for inclusion in the registry.

There are ambiguities and no formal criteria on how the IETF can (or even whether the IETF should) evaluate the merits of applicants to [RFC6761] reservations.

- c. The "seven questions" of [RFC6761] are inadequate for evaluating proposals.

Section 5 of [RFC6761] describes seven questions to be answered by an applicant for [RFC6761] status. However, running this process for the .onion application showed that those seven questions are inadequate for making the determination of whether a particular string qualifies for [RFC6761] treatment.

- d. Some organizations may want to experiment with a reserved name.

They may or may not be ready (or willing) to go through a cumbersome process and find [RFC6761] too difficult to deal with. They would like a much simpler registration process, with limited or no burden to apply.

- e. The [RFC6761] process could be abused.

In particular, the process can be abused to reserve a specific string within an existing suffix (or set of suffixes using wildcards) not under the control of IETF.

- f. [RFC6761] does not have provision for subsequent management of the registry, such as updates, deletions of entries, etc...

#### 4. Issues with the RFC6761 registry

- a. The [RFC6761] registry lacks sufficient documentation supporting a registration.

The registry may point to the RFC creating the reservation, but not to the supporting materials.

- b. The [RFC6761] registry lacks clear directions for applications to select which name resolution method to use upon seeing the special name.

In particular, it lists the reserved names but does not include direct guidance, neither in free text form nor in machine-readable instructions, for any of the seven audiences. Instead, the registry relies on a reference for each entry to the document that requested its insertion. Such documents could be difficult to read for many readers; for example, [RFC6762] is a 70-page protocol specification which is not an effective way to set expectations of non-technical end-users.

- c. Reserving a string in [RFC6761] does not guarantee queries will not leak in the DNS.

The applicants for [RFC6761] status cannot be guaranteed that leakage will not occur and will need to take this into account in their protocol design.

- d. The intended usage or protocol for which the [RFC6761] reservation is made may or may not be successful.

In the case of failure of adoption, the reserved string would be wasted.

## 5. Issues Around the IETF Evaluation of Candidate Strings

- a. The IETF does not have a formal process to evaluate candidate strings for issues such as trademark infringement and name collisions.

This leads to concerns about liability risks incurred by adding a particular string to the [RFC6761] registry.

- b. Submitting the application to IETF last call does not guarantee such issues will be always caught.

[RFC7788] describing the "home networking control protocol" was recently published. That document includes text instructing devices to use names terminating by default with the .home suffix. [RFC7788] did not reference [RFC6761] anywhere and had no IANA sections about this reservation. It was published without anyone noticing this during the entire review process. The issue was caught after the publication, and an errata was published.

## 6. Other Issues Related to RFC6761

[RFC6761] does not include a mechanism to collaborate with ICANN.

The current round of ICANN gTLD (described at [NEW-GTLD]) is, as the time of this writing, closed to new applications. It is, however, not yet completed. Some applications are still under consideration. There is a risk that, without proper collaboration between the IETF and ICANN, a new entry in the [RFC6761] registry could conflict with one of those applications still under review. It should also be noted that discussions have started about forming the next round of ICANN gTLDs, thus this issue will not go away at the conclusion of the current gTLD round.

## 7. Security Considerations

This document aims to provide a problem statement that will inform future work. While security and privacy are fundamental considerations, this document expects that future work will include such analysis, and hence no attempt is made to do so here. See [SAC-057] for further considerations.

Reserving names has been presented as a way to prevent leakage into the DNS. However, instructing resolvers to not forward the queries (and/or by instructing authoritative servers not to respond) is not a guarantee that such leakage will be prevented. The security (or

privacy) of an application MUST NOT rely on names not being exposed to the Internet DNS resolution system.

## 8. IANA Considerations

This document has no IANA actions.

## 9. Acknowledgements

Thanks to Paul Hoffman for a large amount of editing.

## 10. References

### 10.1. Normative References

#### [IANA-SPECIAL-USE]

IANA, "Special-Use Domain Names", 2016,  
<<https://www.iana.org/assignments/special-use-domain-names>>.

[RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <<http://www.rfc-editor.org/info/rfc2860>>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<http://www.rfc-editor.org/info/rfc6761>>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <<http://www.rfc-editor.org/info/rfc7788>>.

### 10.2. Informative References

[DONA] DONA, "DONA Foundation", June 2016, <<https://www.dona.net>>.

[FROGANS] Frogans Technology, "Frogans Technology", June 2016, <<https://www.frogans.org>>.

## [GUIDEBOOK]

ICANN, "gTLD Application Guidebook", June 2012,  
<<https://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf>>.

## [HUSTON]

Huston, G., "What's in a Name?", December 2015,  
<[http://www.circleid.com/posts/20151222\\_whats\\_in\\_a\\_name/](http://www.circleid.com/posts/20151222_whats_in_a_name/)>.

## [I-D.lewis-domain-names]

Lewis, E., "Domain Names", draft-lewis-domain-names-03  
(work in progress), June 2016.

## [NEW-GTLD]

ICANN, "New Generic Top-Level Domains", 2016,  
<<https://newgtlds.icann.org/>>.

## [SAC-057]

ICANN Security and Stability Advisory Committee, "SSAC  
Advisory on Internal Name Certificates", March 2013,  
<<https://www.icann.org/en/system/files/files/sac-057-en.pdf>>.

## Appendix A. Editorial Notes

This section (and sub-sections) to be removed prior to publication.

## A.1. Venue

An appropriate forum for discussion of this draft is, for now, the  
DNSOP WG.

## A.2. Change History

## A.2.1. draft-adpkja-special-names-problem-00

Initial draft circulated for comment.

## Appendix B. Change history

[ RFC Editor: Please remove this section before publication]

-06 to -05:

- o Clarify the sole focus of this draft is to document problems  
created by RFC6761, and not the larger issue of NON-DNS TLDs.
- o Split issue lists and rewrite some for clarity.

-05 to -04:



- o Added two issues to the issue list: market failure and experimental use.
- 04 to -03:
- o Minor edits to correct grammar, clarify that the current ICANN gTLD round is closed.
- 03 to -02:
- o Significant readability changes to focus the discussion.
- 01 to -02:
- o A very large number of readability / grammar / reference fixes from Paul Hoffman.
- 00 to -01:
- o Significant readability changes.
- 00:
- o Initial draft circulated for comment.

#### Authors' Addresses

Geoff Huston  
APNIC

Email: [gih@apnic.net](mailto:gih@apnic.net)

Peter Koch  
DENIC eG  
Kaiserstrasse 75-77  
Frankfurt 60329  
Germany

Email: [pk@denic.de](mailto:pk@denic.de)

Alain Durand  
ICANN

Email: [alain.durand@icann.org](mailto:alain.durand@icann.org)

Warren Kumari  
Google

Email: warren@kumari.net

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 19, 2016

K. Fujiwara  
JPRS  
A. Kato  
Keio/WIDE  
March 18, 2016

Aggressive use of NSEC/NSEC3  
draft-fujiwara-dnsop-nsec-aggressiveuse-03

Abstract

While DNS highly depends on cache, its cache usage of non-existence information has been limited to exact matching. This draft proposes the aggressive use of a NSEC/NSEC3 resource record, which is able to express non-existence of a range of names authoritatively. With this proposal, it is expected that shorter latency to many of negative responses as well as some level of mitigation of random sub-domain attacks (referred to as "Water Torture" attacks). It is also expected that non-existent TLD queries to Root DNS servers will decrease.

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 <http://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 September 19, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Problem Statement . . . . .	4
4. Proposed Solution . . . . .	4
4.1. Aggressive Negative Caching . . . . .	4
4.2. NSEC . . . . .	5
4.3. NSEC3 . . . . .	5
4.4. NSEC3 Opt-Out . . . . .	6
4.5. Wildcard . . . . .	6
4.6. Consideration on TTL . . . . .	6
5. Additional Considerations . . . . .	6
5.1. The CD Bit . . . . .	6
5.2. Detecting random subdomain attacks . . . . .	7
6. Possible side effect . . . . .	7
7. Additional proposals . . . . .	7
7.1. Partial implementation . . . . .	7
7.2. Aggressive negative caching without DNSSEC validation . .	8
7.3. Aggressive negative caching flag idea . . . . .	8
8. IANA Considerations . . . . .	8
9. Security Considerations . . . . .	8
10. Implementation Status . . . . .	9
11. Acknowledgments . . . . .	9
12. Change History . . . . .	9
12.1. Version 01 . . . . .	9
12.2. Version 02 . . . . .	9
12.3. Version 03 . . . . .	9
13. References . . . . .	10
13.1. Normative References . . . . .	10
13.2. Informative References . . . . .	10
Appendix A. Aggressive negative caching from RFC 5074 . . . . .	11
Appendix B. Detailed implementation idea . . . . .	11
Authors' Addresses . . . . .	14

## 1. Introduction

While negative (non-existence) information of DNS caching mechanism has been known as DNS negative cache [RFC2308], it requires exact matching in most cases. Assume that "example.com" zone doesn't have names such as "a.example.com" and "b.example.com". When a full-service resolver receives a query "a.example.com" , it performs a DNS

resolution process, and eventually gets NXDOMAIN and stores it into its negative cache. When the full-service resolver receives another query "b.example.com", it doesn't match with "a.example.com". So it will send a query to one of the authoritative servers of "example.com". This was because the NXDOMAIN response just says there is no such name "a.example.com" and it doesn't tell anything for "b.example.com".

Section 5 of [RFC2308] seems to show that negative answers should be cached only for the exact query name, and not (necessarily) for anything below it.

Recently, DNSSEC [RFC4035] [RFC5155] has been practically deployed. Two types of resource record (NSEC and NSEC3) along with their RRSIG records represent authentic non-existence. For a zone signed with NSEC, it would be possible to use the information carried in NSEC resource records to indicate that a range of names where no valid name exists. Such use is discouraged by Section 4.5 of RFC 4035, however.

This document proposes to make a minor change to RFC 4035 and a full-service resolver can use NSEC/NSEC3 resource records aggressively so that the resolver responds with NXDOMAIN immediately if the name in question falls into a range expressed by a NSEC/NSEC3 resource record.

Aggressive Negative Caching was first proposed in Section 6 of DNSSEC Lookaside Validation (DLV) [RFC5074] in order to find covering NSEC records efficiently. Unbound [UNBOUND] has aggressive negative caching code in its DLV validator. Unbound TODO file contains "NSEC/NSEC3 aggressive negative caching".

Section 3 of [I-D.vixie-dnsext-resimprove] ("Stopping Downward Cache Search on NXDOMAIN") proposed another approach to use NXDOMAIN information effectively.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Many of the specialized terms used in this specification are defined in DNS Terminology [RFC7719].

### 3. Problem Statement

Random sub-domain attacks (referred to as "Water Torture" attacks or NXDomain attacks) send many non-existent queries to full-service resolvers. Their query names consist of random prefixes and a target domain name. The negative cache does not work well and target full-service resolvers result in sending queries to authoritative DNS servers of the target domain name.

When number of queries is large, the full-service resolvers drop queries from both legitimate users and attackers as their outstanding queues are filled up.

For example, BIND 9.10.2 [BIND9] full-service resolvers answer SERVFAIL and Unbound 1.5.2 full-service resolvers drop most of queries under 10,000 queries per second attack.

The countermeasures implemented at this moment are rate limiting and disabling name resolution of target domain names in ad-hoc manner.

### 4. Proposed Solution

#### 4.1. Aggressive Negative Caching

If the target domain names are DNSSEC signed, aggressive use of NSEC/NSEC3 resource records mitigates the problem.

Section 4.5 of [RFC4035] shows that "In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace".

To reduce non-existent queries sent to authoritative DNS servers, it is suggested to relax this restriction as follows:

```
+-----+
| DNSSEC enabled full-service resolvers MAY use |
| NSEC/NSEC3 resource records to generate negative responses |
| until their effective TTLs or signatures on the records   |
| in question expire.                                     |
+-----+
```

If the full-service resolver's cache have enough information to validate the query, the full-service resolver MAY use NSEC/NSEC3/

wildcard records aggressively. Otherwise, the full-service resolver MUST fall back to send the query to the authoritative DNS servers.

Necessary information to validate are matching/covering NSEC/NSEC3 of the wildcards which may match the query name, matching/covering NSEC/NSEC3 of non-terminals which derive from the query name and matching/covering NSEC/NSEC3 of the query name.

If the query name has the matching NSEC/NSEC3 RR and it shows the query type does not exist, the full-service resolver is possible to respond with NODATA (empty) answer.

#### 4.2. NSEC

A full-service resolver implementation SHOULD support aggressive use of NSEC and enable it by default. It SHOULD provide a configuration knob to disable aggressive use of NSEC.

The validating resolver need to check the existence of matching wildcards which derive from the query name, covering NSEC RRs of the matching wildcards and covering NSEC RR of the query name.

If the full-service resolver's cache contains covering NSEC RRs of matching wildcards and the covering NSEC RR of the query name, the full-service resolver is possible to respond with NXDOMAIN error immediately.

#### 4.3. NSEC3

NSEC3 aggressive negative caching is more difficult. If the zone is signed with NSEC3, the validating resolver need to check the existence of non-terminals and wildcards which derive from query names.

If the full-service resolver's cache contains covering NSEC3 RRs of matching wildcards, the covering NSEC3 RRs of the non-terminals and the covering NSEC3 RR of the query name, the full-service resolver is possible to respond with NXDOMAIN error immediately.

If the validating resolver proves the non-existence of the non-terminal domain name of the query name, the query name does not exist.

To identify signing types of the zone, validating resolvers need to build separated cache of NSEC and NSEC3 resource records for each signer domain name.

When a query name is not in the regular cache, find closest enclosing NS RRset in the regular cache. The owner of the closest enclosing NS RRset may be the longest signer domain name of the query name. If there is no entry in the NSEC/NSEC3 cache of the signer domain name, aggressive negative caching is not possible at this moment. Otherwise, there is at least one NSEC or NSEC3 resource records. The record shows the signing type.

A full-service resolver implementation MAY support aggressive use of NSEC3. It SHOULD provide a configuration knob to disable aggressive use NSEC3 in this case.

#### 4.4. NSEC3 Opt-Out

If the zone is signed with NSEC3 and with Opt-Out flag set to 1, the aggressive negative caching is not possible at the zone.

#### 4.5. Wildcard

Even if a wildcard is cached, it is necessary to send a query to an authoritative server to ensure that the name in question doesn't exist as long as the name is not in the negative cache.

When aggressive use is enabled, regardless of description of Section 4.5 of [RFC4035], it is possible to send a positive response immediately when the name in question matches a NSEC/NSEC3 RRs in the negative cache.

#### 4.6. Consideration on TTL

This function needs care on the TTL value of negative information because newly added domain names cannot be used while the negative information is effective. RFC 2308 states the maximum number of negative cache TTL value is 10800 (3 hours). So the full-service resolver SHOULD limit the maximum effective TTL value of negative responses (NSEC/NSEC3 RRs) to 10800 (3 hours). It is reasonably small but still effective for the purpose of this document as it can eliminate significant amount of DNS attacks with randomly generated names.

### 5. Additional Considerations

#### 5.1. The CD Bit

The CD bit disables signature validation. It is one of the basic functions of DNSSEC protocol and it SHOULD NOT be changed. However, attackers may set the CD bit to their attack queries and the aggressive negative caching will be of no use.



Ignoring the CD bit function may break the DNSSEC protocol.

This draft proposes that the CD bit may be ignored to support aggressive negative caching when the full-service resolver is under attacks with CD bit set.

## 5.2. Detecting random subdomain attacks

Full-service resolvers should detect conditions under random subdomain attacks. When they are under attacks, their outstanding queries increase. If there are some destination addresses whose outstanding queries are many, they may contain attack target domain names. Existing countermeasures may implement attack detection.

## 6. Possible side effect

Aggressive use of NSEC/NSEC3 resource records may decrease queries to Root DNS servers.

People may generate many typos in TLD, and they will result in unnecessary DNS queries. Some implementations leak non-existent TLD queries whose second level domain are different each other. Well observed TLDs are ".local" and ".belkin". With this proposal, it is possible to return NXDOMAIN immediately to such queries without further DNS recursive resolution process. It may reduce round trip time, as well as reduce the DNS queries to corresponding authoritative servers, including Root DNS servers.

## 7. Additional proposals

There are additional proposals to the aggressive negative caching.

### 7.1. Partial implementation

It is possible to implement aggressive negative caching partially.

DLV aggressive negative caching [RFC5074] is an implementation of NSEC aggressive negative caching which targets DLV domain names.

NSEC only aggressive negative caching is easier to implement NSEC/NSEC3 aggressive negative caching (full implantation) because NSEC3 handling is hard to implement.

Root only aggressive negative caching is possible. It uses NSEC and RRSIG resource records whose signer domain name is root.

An implementation without detecting attacks is possible. It cannot ignore the CD bit and the effectiveness may be limited.

## 7.2. Aggressive negative caching without DNSSEC validation

Aggressive negative caching may be applicable to full-service resolvers without DNSSEC validation. They can set DNSSEC OK bit in query packets to obtain corresponding NSEC/NSEC3 resource records. While the full-service resolvers SHOULD validate the NSEC/NSEC3 resource records, they MAY use the records to respond NXDOMAIN error immediately without DNSSEC validation.

However, it is highly recommended to apply DNSSEC validation.

## 7.3. Aggressive negative caching flag idea

Authoritative DNS servers that dynamically generate NSEC records normally generate minimally covering NSEC Records [RFC4470]. Aggressive negative caching does not work with minimally covering NSEC records. Most of DNS operators don't want zone enumeration and zone information leaks. They prefer NSEC resource records with narrow ranges. When there is a flag that show a full-service resolver support the aggressive negative caching and a query have the aggressive negative caching flag, authoritative DNS servers can generate NSEC resource records with wider range under random subdomain attacks.

However, changing range of minimally covering NSEC Records may be implemented by detecting attacks. Authoritative DNS servers can answer any range of minimally covering NSEC Records.

## 8. IANA Considerations

This document has no IANA actions.

## 9. Security Considerations

Newly registered resource records may not be used immediately. However, choosing suitable TTL value will mitigate the problem and it is not a security problem.

It is also suggested to limit the maximum TTL value of NSEC resource records in the negative cache to, for example, 10800 seconds (3hrs), to mitigate the issue. Implementations which comply with this proposal is suggested to have a configurable maximum value of NSEC RRs in the negative cache.

Aggressive use of NSEC/NSEC3 resource records without DNSSEC validation may cause security problems.

## 10. Implementation Status

Unbound has aggressive negative caching code in its DLV validator. The author implemented NSEC aggressive caching using Unbound and its DLV validator code.

## 11. Acknowledgments

The authors gratefully acknowledge DLV [RFC5074] author Samuel Weiler and Unbound developers. Olafur Gudmundsson and Pieter Lexis proposed aggressive negative caching flag idea. Valuable comments were provided by Bob Harold, Tatuya JINMEI, Shumon Huque, Mark Andrews, and Casey Deccio.

## 12. Change History

This section is used for tracking the update of this document. Will be removed after finalize.

### 12.1. Version 01

- o Added reference to DLV [RFC5074] and imported some sentences.
- o Added Aggressive Negative Caching Flag idea.
- o Added detailed algorithms.

### 12.2. Version 02

- o Added reference to [I-D.vixie-dnsexst-resimprove]
- o Added considerations for the CD bit
- o Updated detailed algorithms.
- o Moved Aggressive Negative Caching Flag idea into Additional Proposals

### 12.3. Version 03

- o Added "Partial implementation"
- o Section 4,5,6 reorganized for better representation
- o Added NODATA answer in Section 4
- o Trivial updates

- o Updated pseudo code

### 13. References

#### 13.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<http://www.rfc-editor.org/info/rfc2308>>.
- [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, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4470] Weiler, S. and J. Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing", RFC 4470, DOI 10.17487/RFC4470, April 2006, <<http://www.rfc-editor.org/info/rfc4470>>.
- [RFC5074] Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074, DOI 10.17487/RFC5074, November 2007, <<http://www.rfc-editor.org/info/rfc5074>>.
- [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, <<http://www.rfc-editor.org/info/rfc5155>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<http://www.rfc-editor.org/info/rfc7719>>.

#### 13.2. Informative References

- [BIND9] Internet Systems Consortium, Inc., "Name Server Software", 2000, <<https://www.isc.org/downloads/bind/>>.
- [I-D.vixie-dnsext-resimprove] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsext-resimprove-00 (work in progress), June 2010.

[UNBOUND] NLnet Labs, "Unbound DNS validating resolver", 2006,  
<<http://www.unbound.net/>>.

#### Appendix A. Aggressive negative caching from RFC 5074

Imported from Section 6 of [RFC5074].

Previously, cached negative responses were indexed by QNAME, QCLASS, QTYPE, and the setting of the CD bit (see RFC 4035, Section 4.7), and only queries matching the index key would be answered from the cache. With aggressive negative caching, the validator, in addition to checking to see if the answer is in its cache before sending a query, checks to see whether any cached and validated NSEC record denies the existence of the sought record(s).

Using aggressive negative caching, a validator will not make queries for any name covered by a cached and validated NSEC record. Furthermore, a validator answering queries from clients will synthesize a negative answer whenever it has an applicable validated NSEC in its cache unless the CD bit was set on the incoming query.

Imported from Section 6.1 of [RFC5074].

Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC records in order to efficiently find covering NSEC records. Only NSEC records from DLV domains need to be included in this data structure.

#### Appendix B. Detailed implementation idea

Section 6.1 of [RFC5074] is expanded as follows.

Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC and NSEC3 records for each signer domain name of NSEC / NSEC3 records in order to efficiently find covering NSEC / NSEC3 records. Call the table as NSEC\_TABLE.

The aggressive negative caching may be inserted at the cache lookup part of the full-service resolvers.

If errors happen in aggressive negative caching algorithm, resolvers MUST fall back to resolve the query as usual. "Resolve the query as usual" means that the full-resolver resolve the query in Recursive-mode as if the full-service resolver does not implement aggressive negative caching.

To implement aggressive negative caching, resolver algorithm near cache lookup will be changed as follows:

```
QNAME = the query name;
QTYPE = the query type;
if ({QNAME,QTYPE} entry exists in the cache) {
    // the resolver responds the RRSet from the cache
    resolve the query as usual;
}

// if NSEC* exists, QTYPE existence is proved by type bitmap
if (matching NSEC/NSEC3 of QNAME exists in the cache) {
    if (QTYPE exists in type bitmap of NSEC/NSEC3 of QNAME) {
        // the entry exists, however, it is not in the cache.
        // need to iterate QNAME/QTYPE.
        resolve the query as usual;
    } else {
        // QNAME exists, QTYPE does not exist.
        the resolver can generate NODATA response;
    }
}

// Find closest enclosing NS RRset in the cache.
// The owner of this NS RRset will be a suffix of the QNAME
// - the longest suffix of any NS RRset in the cache.
SIGNER = closest enclosing NS RRSet of QNAME in the cache;

// Check the SOA RR of the SIGNER
if (SOA RR of SIGNER does not exist in the cache
    or SIGNER zone is not signed or not validated) {
    Resolve the query as usual;
}

if (SIGNER zone does not have NSEC_TABLE) {
    Resolve the query as usual;
}

if (SIGNER zone is signed with NSEC) { // NSEC mode

    // Check the non-existence of QNAME
    CoveringNSEC = Find the covering NSEC of QNAME;
    if (Covering NSEC doesn't exist in the cache) {
        Resolve the query as usual.
    }

    // Select the longest existing name of QNAME from covering NSEC
    LongestExistName = common part of both owner name and
                       next domain name of CoveringNSEC;
```

```
    if (*.LongestExistName entry exists in the cache) {
        the resolver can generate positive response
        // synthesize the wildcard *.TEST
    }
    if covering NSEC RR of "/*.LongestExistName" at SIGNER zone exists
        in the cache {
        the resolver can generate negative response;
    }
    /*.LongestExistName may exist. cannot generate negative response
    Resolve the query as usual.

} else
if (SIGNER zone is signed with NSEC3 and does not use Opt-Out) {
    // NSEC3 mode

    TEST = SIGNER;
    while (TEST != QNAME) {
        // if any error happens in this loop, break this loop
        UPPER = TEST;
        add a label from the QNAME to the start of TEST;
        // TEST = label.UPPER
        if (TEST name entry exist in the cache
            || matching NSEC3 of TEST exist in the cache) {
            // TEST exist
            continue; // need to check rest of QNAME
        }
        if (covering NSEC3 of TEST exist in the cache) {
            // (non-)terminal name TEST does not exist
            if (*.UPPER name entry exist in the cache) {
                // TEST does not exist and *.UPPER exist
                the resolver can generate positive response;
            } else
            if (covering NSEC3 of *.UPPER exist in the cache) {
                // TEST does not exist and *.UPPER does not exist
                the resolver can generate negative response;
            }
            break; // Lack of information (No *.UPPER information)
        }
        break; // Lack of information (No TEST information)
    }
    // no matching/covering NSEC3 of QNAME information
    Resolve the query as usual
}
```

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

Akira Kato  
Keio University/WIDE Project  
Graduate School of Media Design, 4-1-1 Hiyoshi  
Kohoku, Yokohama 223-8526  
Japan

Phone: +81 45 564 2490  
Email: kato@wide.ad.jp



DNSOP  
Internet-Draft  
Intended status: Best Current Practice  
Expires: February 12, 2017

W. Hardaker  
Parsons  
O. Gudmundsson  
CloudFlare  
S. Krishnaswamy  
Parsons  
August 11, 2016

DNSSEC Roadblock Avoidance  
draft-ietf-dnsop-dnssec-roadblock-avoidance-05.txt

Abstract

This document describes problems that a Validating DNS resolver, stub-resolver or application might run into within a non-compliant infrastructure. It outlines potential detection and mitigation techniques. The scope of the document is to create a shared approach to detect and overcome network issues that a DNSSEC software/system may face.

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 <http://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 February 12, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Notation . . . . .	3
1.2. Background . . . . .	3
1.3. Implementation experiences . . . . .	4
1.3.1. Test Zone Implementation . . . . .	4
2. Goals . . . . .	5
3. Detecting DNSSEC Non-Compliance . . . . .	5
3.1. Determining DNSSEC support in recursive resolvers . . . . .	6
3.1.1. Supports UDP answers . . . . .	6
3.1.2. Supports TCP answers . . . . .	6
3.1.3. Supports EDNS0 . . . . .	7
3.1.4. Supports the DO bit . . . . .	7
3.1.5. Supports the AD bit DNSKEY algorithm 5 and 8 . . . . .	7
3.1.6. Returns RRSig for signed answer . . . . .	8
3.1.7. Supports querying for DNSKEY records . . . . .	8
3.1.8. Supports querying for DS records . . . . .	8
3.1.9. Supports negative answers with NSEC records . . . . .	9
3.1.10. Supports negative answers with NSEC3 records . . . . .	9
3.1.11. Supports queries where DNAME records lead to an answer . . . . .	10
3.1.12. Permissive DNSSEC . . . . .	10
3.1.13. Supports Unknown RRtypes . . . . .	10
3.2. Direct Network Queries . . . . .	10
3.2.1. Support for Remote UDP Over Port 53 . . . . .	11
3.2.2. Support for Remote UDP With Fragmentation . . . . .	11
3.2.3. Support for Outbound TCP Over Port 53 . . . . .	11
3.3. Support for DNSKEY and DS combinations . . . . .	12
4. Aggregating The Results . . . . .	12
4.1. Resolver capability description . . . . .	12
5. Roadblock Avoidance . . . . .	13
5.1. Partial Resolver Usage . . . . .	16
5.1.1. Known Insecure Lookups . . . . .	16
5.1.2. Partial NSEC/NSEC3 Support . . . . .	16
6. Start-Up and Network Connectivity Issues . . . . .	16
6.1. What To Do . . . . .	17
7. Quick Test . . . . .	17
7.1. Test negative answers Algorithm 5 . . . . .	18
7.2. Test Algorithm 8 . . . . .	18
7.3. Test Algorithm 13 . . . . .	18
7.4. Fails when DNSSEC does not validate . . . . .	18
8. Security Considerations . . . . .	18

9. IANA Considerations . . . . .	18
10. Acknowledgments . . . . .	18
11. Normative References . . . . .	19
Authors' Addresses . . . . .	19

## 1. Introduction

This document describes problems observable during DNSSEC ([RFC4034], [RFC4035]) deployment that derive from non-compliant infrastructure. It poses potential detection and mitigation techniques.

### 1.1. Notation

In this document a "Host Validator" can either be a validating stub-resolver, such as library that an application has linked in, or a validating resolver daemon running on the same machine. It may or may not be trying to use upstream caching resolvers during its own resolution process; both cases are covered by the tests defined in this document.

The sub-variant of this is a "Validating Forwarding Resolver", which is a resolver that is configured to use upstream Resolvers when possible. A Validating Forward Resolver also needs to perform the tests outlined in this document before using an upstream recursive resolver.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

### 1.2. Background

Deployment of DNSSEC validation is hampered by network components that make it difficult or sometimes impossible for validating resolvers to effectively obtain the DNSSEC data they need. This can occur for many different reasons including, but not limited to:

- o Because recursive resolvers and DNS proxies [RFC5625] are not fully DNSSEC compliant
- o Because resolvers are not DNSSEC aware
- o Because "middle-boxes" actively block, modify and/or restrict outbound traffic to the DNS port (53) either UDP and/or TCP .
- o In-path network components do not allow UDP fragments

This document talks about ways that a Host Validator can detect the state of the network it is attached to, and ways to hopefully circumvent the problems associated with the network defects it discovers. The tests described in this document may be performed on any validating resolver to detect and prevent problems. While these recommendations are mainly aimed at Host Validators it is prudent to perform these tests from regular Validating Resolvers before enabling just to make sure things work.

There are situations where a host can not talk directly to a Resolver; the tests below can not address how to overcome that, and inconsistent results can be seen in such cases. This can happen, for instance, when there are DNS proxies/forwarders between the user and the actual resolvers.

### 1.3. Implementation experiences

Multiple lessons learned from multiple implementations led to the development of this document, including (in alphabetical order) DNSSEC-Tools' DNSSEC-Check, DNSSEC\_Resolver\_Check, dnssec-trigger, FCC\_Grade.

Detecting lack of support for specified DNSKEY algorithms and DS digest algorithms is outside the scope of this document but the document provides information on how to do that, see sample test tool: [https://github.com/ogud/DNSSEC\\_ALG\\_Check](https://github.com/ogud/DNSSEC_ALG_Check)

This document does describe compliance tests for algorithms 5, 7 and 13 with DS digest algorithms 1 and 2.

#### 1.3.1. Test Zone Implementation

In this document, the "test.example.com" domain is used to refer to DNS records which contain test records that have known DNSSEC properties associated with them. For example, the "badsign-a.test.example.com" domain is used below to refer to a DNS A record where the signatures published for it are invalid (i.e., they are "bad signatures" that should cause a validation failure).

At the time of this publication, the "test.dnssec-tools.org" domain implements all of these test records. Thus, it may be possible to replace "test.example.com" in this document with "test.dnssec-tools.org" when performing real-world tests.

## 2. Goals

This document is intended to show how a Host Validator can detect the capabilities of a recursive resolver, and work around any problems that could potentially affect DNSSEC resolution. This enables the Host Validator to make use of the caching functionality of the recursive resolver, which is desirable in that it decreases network traffic and improves response times.

A Host Validator has two choices: it can wait to determine that it has problems with a recursive resolver based on the results that it is getting from real-world queries issued to it, or it can proactively test for problems (Section 3) to build a work around list ahead of time (Section 5). There are pros and cons to both of these paths that are application specific, and this document does not attempt to provide guidance about whether proactive tests should or should not be used. Either way, DNSSEC roadblock avoidance techniques ought to be used when needed and if possible.

Note: Besides being useful for Host Validators, the same tests can be used for a recursive resolver to check if its upstream connections hinder DNSSEC validation.

## 3. Detecting DNSSEC Non-Compliance

A Host Validator may choose to determine early-on what roadblocks exist that may hamper its ability to perform DNSSEC look-ups. This section outlines tests that can be done to test certain features of the surrounding network.

These tests should be performed when a resolver determines its network infrastructure has changed. Certainly a resolver should perform these tests when first starting, but MAY also perform these tests when they've detected network changes (e.g. address changes, or network reattachment, etc).

NOTE: when performing these tests against an address, we make the following assumption about that address: It is a uni-cast address or an any-cast [RFC4786] cluster where all servers have identical configuration and connectivity.

NOTE: when performing these tests we also assume that the path is clear of "DNS interfering" middle-boxes, like firewalls, proxies, forwarders. Presence of such infrastructure can easily make a recursive resolver appear to be improperly performing. It is beyond the scope of the document as how to work around such interference, although the tests defined in this document may indicate when such misbehaving middle-ware is causing interference.

NOTE: This document specifies two sets of tests to perform: a comprehensive one and a fast one. The fast one will detect most common problems, thus if the fast one passes then the comprehensive MAY be considered passed as well.

### 3.1. Determining DNSSEC support in recursive resolvers

Ideally, a Host Validator can make use of the caching present in recursive resolvers. This section discusses the tests that a recursive resolver MUST pass in order to be fully usable as a DNS cache.

Unless stated otherwise, all of the following tests SHOULD have the Recursion Desired (RD) flag set when sending out a query and SHOULD be sent over UDP. Unless otherwise stated, the tests MUST NOT have the DO bit set or utilize any of the other DNSSEC related requirements, like EDNS0, unless otherwise specified. The tests are designed to check for support of one feature at a time.

#### 3.1.1. Supports UDP answers

Purpose: This tests basic DNS over UDP functionality to a resolver.

Test: A DNS request is sent to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

Note: an implementation MAY chose to not perform the rest of the tests if this test fails, as it is highly unlikely that the resolver under test will pass any of the remaining tests.

#### 3.1.2. Supports TCP answers

Purpose: This tests basic TCP functionality to a resolver.

Test: A DNS request is sent over TCP to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com.

SUCCESS: A DNS response was received that contains an A record in the answer section. (The data itself does not need to be checked.)

### 3.1.3. Supports EDNS0

Purpose: Test whether a resolver properly supports the EDNS0 extension option.

Pre-requisite: "Supports UDP or TCP".

Test: Send a request to the resolver under test for an A record for a known existing domain, such as good-a.test.example.com, with an EDNS0 OPT record in the additional section.

SUCCESS: A DNS response was received that contains an EDNS0 option with version number 0.

### 3.1.4. Supports the DO bit

Purpose: This tests whether a resolver has minimal support of the DO bit.

Pre-requisite: "Supports EDNS0".

Test: Send a request to the resolver under test for an A record for a known existing domain such as good-a.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains the DO bit set.

Note: this only tests that the resolver sets the DO bit in the response. Later tests will determine if the DO bit was actually made use of. Some resolvers successfully pass this test because they simply copy the unknown flags into the response. These resolvers will fail the later tests.

### 3.1.5. Supports the AD bit DNSKEY algorithm 5 and 8

Purpose: This tests whether the resolver is a validating resolver.

Pre-requisite: "Supports the DO bit".

Test: Send requests to the resolver under test for an A record for a known existing domain in a DNSSEC signed zone which is verifiable to a configured trust anchor, such as good-a.test.example.com using the root's published DNSKEY or DS record as a trust anchor. Set the DO bit in the outgoing query. This test should be done twice, once for a zone that contains algorithm 5 (RSASHA1) and another for algorithm 8 (RSASHA256).

SUCCESS: A DNS response was received that contains the AD bit set for algorithm 5 (RSASHA1).

BONUS: The AD bit is set for a resolver that supports Algorithm 8 RSASHA256

### 3.1.6. Returns RRSig for signed answer

Purpose: This tests whether a resolver will properly return RRSIG records when the DO bit is set.

Pre-requisite: "Supports the DO bit".

Test: Send a request to the resolver under test for an A record for a known existing domain in a DNSSEC signed zone, such as good-a.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains at least one RRSIG record.

### 3.1.7. Supports querying for DNSKEY records

Purpose: This tests whether a resolver can query for and receive a DNSKEY record from a signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an DNSKEY record which is known to exist in a signed zone, such as test.example.com/DNSKEY. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DNSKEY record in the answer section.

Note: Some DNSKEY RRset's are large and if the network path has problems with large answers this query may result in either false positive or false negative. In general the DNSKEY queried for should be small enough to fit into a 1220 byte answer, to avoid false negative result when TCP is disabled. However, querying many zones will result in answers greater than 1220 bytes so DNS over TCP MUST be available for DNSSEC use in general.

### 3.1.8. Supports querying for DS records

Purpose: This tests whether a resolver can query for and receive a DS record from a signed zone.

Pre-requisite: "Supports the DO bit."



Test: Send a request to the resolver under test for an DS record which is known to exist in a signed zone, such as test.example.com/DS. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains a DS record in the answer section.

#### 3.1.9. Supports negative answers with NSEC records

Purpose: This tests whether a resolver properly returns NSEC records for a non-existing domain in a DNSSEC signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an A record which is known to not exist in an NSEC signed zone, such as non-existent.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC record.

Note: The query issued in this test MUST be sent to a NSEC signed zone. Getting back appropriate NSEC3 records does not indicate a failure, but a bad test.

#### 3.1.10. Supports negative answers with NSEC3 records

Purpose: This tests whether a resolver properly returns NSEC3 records ([RFC5155]) for a non-existing domain in a DNSSEC signed zone.

Pre-requisite: "Supports the DO bit."

Test: Send a request to the resolver under test for an A record which is known to be non-existent in a zone signed using NSEC3, such as non-existent.nsec3-ns.test.example.com. Set the DO bit in the outgoing query.

SUCCESS: A DNS response was received that contains an NSEC3 record.

Bonus: If the AD bit is set, this validator supports algorithm 7 RSASHA1-NSEC3-SHA1

Note: The query issued in this test MUST be sent to a NSEC3 signed zone. Getting back appropriate NSEC records does not indicate a failure, but a bad test.

#### 3.1.11. Supports queries where DNAME records lead to an answer

Purpose: This tests whether a resolver can query for an A record in a zone with a known DNAME referral for the record's parent.

Test: Send a request to the resolver under test for an A record which is known to exist in a signed zone within a DNAME referral child zone, such as good-a.dname-good-ns.test.example.com.

SUCCESS: A DNS response was received that contains a DNAME in the answer section. An RRSIG MUST also be received in the answer section that covers the DNAME record.

#### 3.1.12. Permissive DNSSEC

Purpose: To see if a validating resolver is ignoring DNSSEC validation failures.

Pre-requisite: Supports the AD bit.

Test: ask for data from a broken DNSSEC delegation such as badsign-a.test.example.com.

SUCCESS: A reply was received with the Rcode set to SERVFAIL

#### 3.1.13. Supports Unknown RRtypes

Purpose: Some DNS Resolvers/gateways only support some RRtypes. This causes problems for applications that need recently defined types.

Pre-requisite: "Supports UDP or TCP".

Test: Send a request for recently defined type or unknown type in the 20000-22000 range, that resolves to a server that will return an answer for all types, such as alltypes.example.com (a server today that supports this: alltypes.res.dnssecready.org)

SUCCESS: A DNS response was retrieved that contains the type requested in the answer section.

#### 3.2. Direct Network Queries

If need be, a Host Validator may need to make direct queries to authoritative servers or known Open Recursive Resolvers in order to collect data. To do that, a number of key network features MUST be functional.

### 3.2.1. Support for Remote UDP Over Port 53

Purpose: This tests basic UDP functionality to outside the local network.

Test: A DNS request is sent to a known distant authoritative server for a record known to be within that server's authoritative data.

Example: send a query to the address of ns1.test.example.com for the good-a.test.example.com/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: an implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but does not need to be.

### 3.2.2. Support for Remote UDP With Fragmentation

Purpose: This tests if the local network can receive fragmented UDP answers

Pre-requisite: Local UDP traffic > 1500 in size is possible

Test: A DNS request is sent over UDP to a known distant DNS address asking for a record that has answer larger than 2000 bytes. For example, send a query for the test.example.com/DNSKEY record with the DO bit set in the outgoing query.

Success: A DNS response was received that contains the large answer.

Note: A failure in getting large answers over UDP is not a serious problem if TCP is working.

### 3.2.3. Support for Outbound TCP Over Port 53

Purpose: This tests basic TCP functionality to outside the local network.

Test: A DNS request is sent over TCP to a known distant authoritative server for a record known to be within that server's authoritative data. Example: send a query to the address of ns1.test.example.com for the good-a.test.example.com/A record.

SUCCESS: A DNS response was received that contains an A record in the answer section.

Note: an implementation can use the local resolvers for determining the address of the name server that is authoritative for the given zone. The recursive bit MAY be set for this request, but does not need to be.

### 3.3. Support for DNSKEY and DS combinations

Purpose: These tests can check what algorithm combinations are supported.

Pre-requisite: At least one of above tests has returned the AD bit set proving that the upstream is validating

Test: A DNS request is sent over UDP to the resolver under test for a known combination of the DS algorithm number (N) and DNSKEY algorithm number (M) of the example form ds-N.alg-M-nsec.test.example.com.

Examples:

```
ds-2.alg-13-nsec.test.example.com TXT
or
ds-4.alg-13-nsec3.test.example.com TXT.
```

SUCCESS: a DNS response is received with the AD bit set and with a matching record type in the answer section.

Note: for algorithms 6 and 7, NSEC is not defined thus query for alg-M-nsec3 is required. Similarly NSEC3 is not defined for algorithms 1, 3 and 5. Furthermore algorithms 2, 4, 9, 11 do not currently have definitions for signed zones.

## 4. Aggregating The Results

Some conclusions can be drawn from the results of the above tests in an "aggregated" form. This section defines some labels to assign to a resolver under test given the results of the tests run against them.

### 4.1. Resolver capability description

This section will group and label certain common results

Resolvers are classified into following broad behaviors:

Validator: The resolver passes all DNSSEC tests and had the AD bit appropriately set.

DNSSEC Aware: The resolver passes all DNSSEC tests, but does not appropriately set the AD bit on answers, indicating it is not

validating. A Host Validator will function fine using this resolver as a forwarder.

Non-DNSSEC capable: The resolver is not DNSSEC aware and will make it hard for a Host Validator to operate behind it. It MAY be usable for querying for data that is in known insecure sections of the DNS tree.

Not a DNS Resolver: This is a improperly behaving resolver and not should not be used at all.

While it would be great if all resolvers fell cleanly into one of the broad categories above, that is not the case. For that reason it is necessary to augment the classification with more descriptive result, this is done by adding the word "Partial" in front of Validator/DNSSEC Aware classifications, followed by sub-descriptors of what is not working.

Unknown: Failed the Unknown test

DNAME: Failed the DNAME test

NSEC3: Failed the NSEC3 test

TCP: TCP not available

SlowBig: UDP is size limited but TCP fallback works

NoBig: TCP not available and UDP is size limited

Permissive: Passes data known to fail validation

## 5. Roadblock Avoidance

The goal of this document is to tie the above tests and aggregations to avoidance practices; however the document does not specify exactly how to do that.

Once we have determined what level of support is available in the network, we can determine what must be done in order to effectively act as a validating resolver. This section discusses some of the options available given the results from the previous sections.

The general fallback approach can be described by the following sequence:

If the resolver is labeled as "Validator" or "DNSSEC aware":

Send queries through this resolver and perform local validation on the results.

If validation fails, try the next resolver

Else if the resolver is labeled "Not a DNS Resolver" or "Non-DNSSEC capable":

Mark it as unusable and try next resolver

Else if no more resolvers are configured and if direct queries are supported:

1. Try iterating from the Root
2. If the answer is SECURE/BOGUS:  
Return the result of the iteration
3. If the answer is INSECURE:  
Re-query "Non-DNSSEC capable" servers and return answers from them w/o the AD bit set to the client.

This will increase the likelihood that split-view unsigned answers are found.

Else:

Return an error code and log a failure

While attempting resolution through a particular recursive name server with a particular transport method that worked, any transport-specific parameters MUST be remembered in order to avoid any unnecessary fallback attempts.

Transport-specific parameters MUST also be remembered for each authoritative name server that is queried while performing an iterative mode lookup.

Any transport settings that are remembered for a particular name server MUST be periodically refreshed; they should also be refreshed when an error is encountered as described below.

For a stub resolver, problems with the name server can manifest themselves under the following types of error conditions:

- o No Response, error response or missing DNSSEC meta-data
- o Illegal Response: An illegal response is received, which prevents the validator from fetching all necessary records required for constructing an authentication chain. This could result when referral loops are encountered, when any of the antecedent zone delegations are lame, when aliases are erroneously followed for certain RRtypes (such as SOA, DNSKEYs or DS records), or when resource records for certain types (e.g. DS) are returned from a zone that is not authoritative for such records.
- o Bogus Response: A Bogus Response is received, when the cryptographic assertions in the authentication chain do not validate properly.

For each of the above error conditions a validator MAY adopt the following dynamic fallback technique, preferring a particular approach if it is known to work for a given name server or zone from previous attempts.

- o No response, error response, or missing DNSSEC meta-data:
  - \* Re-try with different EDNS0 sizes (4096, 1492, None)
  - \* Re-try with TCP only
  - \* Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.
  - \* Re-try using an alternative transport method, if this alternative method is known (configured) to be supported by the nameserver in question.
- o Illegal Response
  - \* Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.
  - \* Check if any of the antecedent zones up to the closest configured trust anchor are provably insecure.
- o Bogus Response
  - \* Perform an iterative query starting from the Root if the previous error was returned from a lookup that had recursion enabled.

For each fallback technique, attempts to multiple potential name servers should be skewed such that the next name server is tried when the previous one encounters an error, a timeout is reached, or whichever is earlier.

The validator SHOULD remember, in its zone-specific fallback cache, any broken behavior identified for a particular zone for a duration of that zone's SOA negative TTL.

The validator MAY place name servers that exhibit broken behavior into a blacklist, and bypass these name servers for all zones that they are authoritative for. The validator MUST time out entries in this name server blacklist periodically, where this interval could be set to be the same as the DNSSEC BAD cache default TTL.

#### 5.1. Partial Resolver Usage

It may be possible to use Non-DNSSEC Capable caching resolvers in careful ways if maximum optimization is desired. This section describes some of the advanced techniques that could be used to use a resolver in at least a minimal way. Most of the time this would be unnecessary, except in the case where none of the resolvers are fully compliant and thus the choices would be to use them at least minimally or not at all (and no caching benefits would be available).

##### 5.1.1. Known Insecure Lookups

If a resolver is Non-DNSSEC Capable but a section of the DNS tree has been determined to be Provably Insecure [RFC4035], then queries to this section of the tree MAY be sent through Non-DNSSEC Capable caching resolver.

##### 5.1.2. Partial NSEC/NSEC3 Support

Resolvers that understand DNSSEC generally, and understand NSEC but not NSEC3 are partially usable. These resolvers generally also lack support for Unknown types, rendering them mostly useless and to be avoided.

#### 6. Start-Up and Network Connectivity Issues

A number of scenarios will produce either short-term or long-term connectivity issues with respect to DNSSEC validation. Consider the following cases:

Time Synchronization: Time synchronization problems can occur when a device which has been off for a period of time and the clock is no longer in close synchronization with "real time" or when a



device always has clock set to the same time during start-up. This will cause problems when the device needs to resolve their source of time synchronization, such as "ntp.example.com".

Changing Network Properties: A newly established network connection may change state shortly after a HTTP-based pay-wall authentication system has been used. This is especially common in hotel, airport and coffee-shop style networks, where DNSSEC, validation and even DNS are not functional until the user proceeds through a series of forced web pages used to enable their network. The tests in Section 3 will produce very different results before and after the network authorization has succeeded. APIs exist on many operating systems to detect initial network device status changes, such as right after DHCP has finished, but few (none?) exist to detect that authentication through a pay-wall has succeeded.

There are only two choices when situations like this happen:

Continue to perform DNSSEC processing, which will likely result in all DNS requests failing. This is the most secure route, but causes the most operational grief for users.

Turn off DNSSEC support until the network proves to be usable. This allows the user to continue using the network, at the sacrifice of security. It also allows for a denial of security-service attack if a man-in-the-middle can convince a device that DNSSEC is impossible.

#### 6.1. What To Do

If the Host Validator detects that DNSSEC resolution is not possible it SHOULD log the event and/or SHOULD report an error to the user. In the case there is no user, then no reporting can be performed and thus the device MAY have a policy of action, like continue or fail. Until middle boxes allow DNSSEC protected information to traverse them consistently, software implementations may need to offer this choice to let users pick the security level they require. Note that continuing without DNSSEC protection in the absence of a notification or report could lead to situations where users assume a level of security that does not exist.

#### 7. Quick Test

The quick tests defined below make the assumption that the questions to be asked are of a real resolver and the only real question is: "how complete is the DNSSEC support?". This quick test has been implemented in few programs developed at IETF hackthons at IETF-91

and IETF-92. The programs use a common grading method. For each question that returns expected answer the resolver gets a point. If the AD bit is set as expected the resolver gets a second point.

#### 7.1. Test negative answers Algorithm 5

Query: really-doesnotexist.test.example.com. A

Answer: RCODE= NXDOMAIN, Empty Answer, Authority: NSEC proof

#### 7.2. Test Algorithm 8

Query: alg-8-nsec3.test.example.com. SOA

Answer: RCODE= 0, Answer: SOA record

#### 7.3. Test Algorithm 13

Query: alg-13-nsec.test.example.com. SOA

Answer: RCODE= 0, Answer: SOA record

#### 7.4. Fails when DNSSEC does not validate

Query: dnssec-failed.test.example.com. SOA

Answer: RCODE= SERVFAIL, empty answer, and authority, AD=0

### 8. Security Considerations

This document discusses problems that may occur while deploying the DNSSEC protocol. It describes what may be possible to help detect and mitigate these problems. Following the outlined suggestions will result in a more secure DNSSEC operational environment than if DNSSEC was simply disabled.

### 9. IANA Considerations

No IANA actions are required.

### 10. Acknowledgments

We thank the IESG and DNSOP working group members for their extensive comments and suggestions.

## 11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <<http://www.rfc-editor.org/info/rfc4786>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<http://www.rfc-editor.org/info/rfc5625>>.

## Authors' Addresses

Wes Hardaker  
Parsons  
P.O. Box 382  
Davis, CA 95617  
US

Email: [ietf@hardakers.net](mailto:ietf@hardakers.net)

Olafur Gudmundsson  
CloudFlare  
San Francisco, CA 94107  
USA

Email: [olafur+ietf@cloudflare.com](mailto:olafur+ietf@cloudflare.com)

Suresh Krishnaswamy  
Parsons  
7110 Samuel Morse Dr  
Columbia, MD 21046  
US

Email: suresh@tislabs.com

dnsop  
Internet-Draft  
Intended status: Standards Track  
Expires: July 14, 2017

O. Gudmundsson  
CloudFlare  
P. Wouters  
Red Hat  
January 10, 2017

Managing DS records from parent via CDS/CDNSKEY  
draft-ietf-dnsop-maintain-ds-06

Abstract

RFC7344 specifies how DNS trust can be maintained across key rollovers in-band between parent and child. This document elevates RFC7344 from informational to standards track and adds a standard track method for initial trust setup and removal of secure entry point.

Changing a domain's DNSSEC status can be a complicated matter involving multiple unrelated parties. Some of these parties, such as the DNS operator, might not even be known by all the organizations involved. The inability to disable DNSSEC via in-band signaling is seen as a problem or liability that prevents some DNSSEC adoption at large scale. This document adds a method for in-band signaling of these DNSSEC status changes.

This document describes reasonable policies to ease deployment of the initial acceptance of new secure entry points (DS records)

It is preferable that operators collaborate on the transfer or move of a domain. The best method is to perform a Key Signing Key ("KSK") plus Zone Signing Key ("ZSK") rollover. If that is not possible, the method using an unsigned intermediate state described in this document can be used to move the domain between two parties. This leaves the domain temporarily unsigned and vulnerable to DNS spoofing, but that is preferred over the alternative of validation failures due to a mismatched DS and DNSKEY record.

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 <http://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 July 14, 2017.

#### Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Introducing a DS record . . . . .	3
1.2. Removing a DS Record . . . . .	3
1.3. Notation . . . . .	4
1.4. Terminology . . . . .	4
2. The Three Uses of CDS . . . . .	4
2.1. The meaning of the CDS RRset . . . . .	5
3. Enabling DNSSEC via CDS/CDNSKEY . . . . .	5
3.1. Accept policy via authenticated channel . . . . .	6
3.2. Accept with extra checks . . . . .	6
3.3. Accept after delay . . . . .	6
3.4. Accept with challenge . . . . .	6
3.5. Accept from inception . . . . .	7
4. DNSSEC Delete Algorithm . . . . .	7
5. Security considerations . . . . .	8
6. IANA considerations . . . . .	8
6.1. Promoting RFC7344 to standards track . . . . .	9
7. References . . . . .	9
7.1. Normative References . . . . .	9
7.2. Informative References . . . . .	9
Appendix A. Acknowledgments . . . . .	9
Authors' Addresses . . . . .	9

## 1. Introduction

CDS/CDNSKEY [RFC7344] records are used to signal changes in secure entry points. This is one method to maintain delegations that can be used when the DNS operator has no other way to inform the parent that changes are needed. This document elevates [RFC7344] from informational to standards track RFC.

In addition, [RFC7344] is lacking two different options for full automated operation to be possible. It did not define a method for the Initial Trust establishment, leaving it open to each parent to come up with an acceptance policy. Additionally, [RFC7344] did not provide a "delete" signal for the child to inform the parent that the DNSSEC security for its domain must be removed.

### 1.1. Introducing a DS record

Automated insertion of DS records has been limited for many zones by the requirement that all changes pass through a "registry" of the child zone's parent. This has significantly hindered deployment of DNSSEC at large scale for DNS hosters, as the child zone owner is often not aware or able to update DNS records such as the DS record.

This document describes a few possible methods for the parent to accept a request by the child to add a DS record to its zone. These methods have different security properties that addresses different deployment scenarios, all resulting in an automated method of trust introduction.

### 1.2. Removing a DS Record

This document introduces the delete option for both CDS and CDNSKEY, allowing a child to signal to the parent to turn off DNSSEC. When a domain is moved from one DNS operator to another, sometimes it is necessary to turn off DNSSEC to facilitate the change of DNS operator. Common scenarios include:

- 1 Alternative to doing a proper DNSSEC algorithm rollover due to operational limitations such as software limitations.
- 2 Moving from a DNSSEC operator to a non-DNSSEC capable operator.
- 3 Moving to an operator that cannot/does-not-want to do a proper DNSSEC rollover.
- 4 When moving between two DNS operators that use disjoint sets of algorithms to sign the zone, thus an algorithm rollover can not be performed.

5 The domain holder no longer wants DNSSEC enabled.

The lack of a "remove my DNSSEC" option is cited as a reason why some operators cannot deploy DNSSEC, as this is seen as an operational risk.

Turning off DNSSEC reduces the security of the domain and thus should only be done carefully, and that decision should be fully under the child domain's control.

### 1.3. Notation

Signaling can happen via CDS or CDNSKEY records. The only differences between the two records is how information is represented, and who calculates the DS digest. For clarity, this document uses the term "CDS" throughout the document to mean "either CDS or CDNSKEY".

When the document uses the word "parent" it implies an entity that is authorized to insert DS records into the parent zone on behalf of the child domain. Which entity this exactly is does not matter. It could be the Registrar or Reseller that the child domain was purchased from. It could be the Registry that the domain is registered in when allowed. Or it could be some other entity.

### 1.4. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. The Three Uses of CDS

In general there are three operations that a domain wants to instruct their parent to perform:

- 1 Enable DNSSEC validation, i.e. place an initial DS RRset in the parent.
- 2 Roll over the Key Signing Key ("KSK"), this means updating the DS records in the parent to reflect the new set of KSK's at the child. This could be an ADD operation, a DELETE operation on one or more records while keeping at least one DS RR, or a full REPLACE operation.
- 3 Turn off DNSSEC validation, i.e. delete all the DS records.



Rolling the KSK is covered in [RFC7344]. It is considered the safest use case of a CDS/CDNSKEY record as it makes no change to the trust relationship between parent and child. Introduction and removal of DS records are defined in this document. As these CDS/CDNSKEY use cases create or end the trust relationship between the parent and child, these use cases should be carefully implemented and monitored.

### 2.1. The meaning of the CDS RRset

The semantic meaning of publishing a CDS RRset is interpreted to mean:

"Publishing a CDS or CDNSKEY record signals to the parent that the child desires that the corresponding DS records be synchronized. Every parent or parental agent should have an acceptance policy of these records for the three different use cases involved: Initial DS publication, Key rollover, and Returning to Insecure."

In short, the CDS RRset is an instruction to the parent to modify the DS RRset if the CDS and DS Reset's differ.

The acceptance policy for CDS in the rollover case is "seeing" according to [RFC7344]. The acceptance policy in the Delete case is seeing a (validly signed) CDS RRset with the delete operation specified in this document.

### 3. Enabling DNSSEC via CDS/CDNSKEY

There are number of different models for managing initial trust, but in the general case, the child wants to enable global validation. As long as the child is insecure, DNS answers can be forged. The goal is to promote the child from insecure to secure as soon as reasonably possible by the parent. This means that the period from the child's publication of CDS/CDNSKEY RRset to the parent publishing the synchronized DS RRset should be as short as possible.

One important use case is how a third party DNS operator can upload its DNSSEC information to the parent, so the parent can publish a DS record for the child. In this case there is a possibility of setting up some kind of authentication mechanism and submission mechanism that is outside the scope of this document.

Below are some policies that parents can use. These policies assume that the notifications can be verified or authenticated.

### 3.1. Accept policy via authenticated channel

In this case the parent is notified via authenticated channel UI/API that a CDS/CDNSKEY RRset exists. In the case of a CDS RRset the parent retrieves the CDS RRset and inserts the corresponding DS RRset as requested. In the case of CDNSKEY the parent retrieves the CDNSKEY RRset and calculates the DS record(s). Parents may limit the DS record type based on local policy. Parents SHOULD NOT refuse CDS/CDNSKEY updates that do not (yet) have a matching DNSKEY in the child zone. This will allow the child to prepublish a spare (and potentially offline) DNSKEY.

### 3.2. Accept with extra checks

In this case the parent checks that the source of the notification is allowed to request the DS insertion. The checks could include whether this is a trusted entity, whether the nameservers correspond to the requester, whether there have been any changes in registration in the last few days, etc. The parent can also send a notification requesting a confirmation, for example by sending email to the registrant requesting a confirmation. The end result is that the CDS RRset is accepted at the end of the checks or when the out-of-band confirmation is received. Any extra checks should have proper rate limiting in place to prevent abuse.

### 3.3. Accept after delay

In this case, if the parent deems the request valid, it starts monitoring the CDS RRset at the child nameservers over period of time to make sure nothing changes. After some time or after a number of checks, preferably from different vantage points in the network, the parent accepts the CDS RRset as a valid signal to update its DS RRset for this child.

### 3.4. Accept with challenge

In this case the parent instructs the requester to insert some record into the child domain to prove it has the ability to do so (i.e., it is the operator of the zone). This method imposes a new task on the parent to monitor the child zone to see if the challenge has been added to the zone. The parent should verify the challenge is published by all the child's nameservers and should test for this challenge from various diverse network locations to increase the security of this method as much as possible.

### 3.5. Accept from inception

If a parent is adding a new child domain that is not currently delegated at all, it could use the child CDS/CDNSKEY RRset to immediately publish a DS RRset along with the new NS RRset. This would ensure that the new child domain is never active in an insecure state.

## 4. DNSSEC Delete Algorithm

This document defines the previously reserved DNS Security Algorithm Number of value 0 in the context of CDS and CDNSKEY records to mean that the entire DS RRset at the parent must be removed. The value 0 remains reserved for the DS and DNSKEY records.

No DNSSEC validator can treat algorithm 0 as a valid signature algorithm. If a validator sees a DNSKEY or DS record with this algorithm value, it must treat it as unknown. Accordingly, the zone is treated as unsigned unless there are other algorithms present. In general the value 0 should never be used in the context of DNSKEY and DS records.

The CERT record [RFC4398] defines the value 0 similarly to mean the algorithm in the CERT record is not defined in DNSSEC.

The contents of the CDS or CDNSKEY RRset MUST contain one RR and only contain the exactly the fields as shown below.

```
1  CDS 0 0 0 0
```

```
2  CDNSKEY 0 3 0 0
```

The keying material payload is represented by a single 0. This record is signed in the same way as regular CDS/CDNSKEY RRsets are signed. This is a change in format from strict interpretation of [RFC7344] and may cause problems with some deployed software.

Strictly speaking the CDS record could be "CDS X 0 X 0" as only the DNSKEY algorithm is what signals the DELETE operation, but for clarity the "0 0 0 0" notation is mandated - this is not a definition of DS Digest algorithm 0. The same argument applies to "CDNSKEY 0 3 0 0", the value 3 in second field is mandated by [RFC4034] section 2.1.2.

Once the parent has verified the CDS/CDNSKEY RRset and it has passed other acceptance tests, the parent MUST remove the DS RRset. After waiting a sufficient amount of time - depending on the parental TTL's - the child can start the process of turning off DNSSEC.

## 5. Security considerations

Turning off DNSSEC reduces the security of the domain and thus should only be done as a last resort in preventing DNSSEC validation errors due to mismatched DS and DNSKEY records.

Users should keep in mind that re-establishing trust in delegation can be hard and takes time. Before deciding to complete the rollover via an unsigned state, all other options should be considered first.

A parent SHOULD ensure that when it is allowing a child to become securely delegated, that it has a reasonable assurance that the CDS/CDNSKEY RRset that is used to bootstrap the security is visible from a geographically and topologically diverse view. It SHOULD also ensure that the zone validates correctly if the parent publishes the DS record. A parent zone might also consider sending an email to its contact addresses to give the child zone a warning that security will be enabled after a certain amount of wait time - thus allowing a child administrator to cancel the request.

This document describes a few possible acceptance criteria for the Initial Trust establishment. Due to a large variety of legal frameworks surrounding parent domains (TLDs in particular) this document cannot give a definitive list of valid acceptance criteria. Parental zones should look at the listed methods and pick the most secure method possible within their legal and technical scenario, possibly further securing the acceptance criteria, as long as the deployed method still enables a fully automated method for non-direct parties such as third party DNS hosters.

## 6. IANA considerations

This document updates entry number 0 of the "DNS Security Algorithm Numbers" IANA Registry as follows:

Numb er	Descrip tion	Mnemo nic	Zone Signi ng	Trans . Sec.	Reference
0	Delete DS	DELET E	N	N	[RFC4034][RFC4398]RFC- THIS-DOCUMENT]

### 6.1. Promoting RFC7344 to standards track

Experience has shown that CDS/CDNSKEY are useful in the deployment of DNSSEC. [RFC7344] was published as Informational, this document elevates RFC7344 to standards track.

## 7. References

### 7.1. 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, <<http://www.rfc-editor.org/info/rfc4034>>.
- [RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <<http://www.rfc-editor.org/info/rfc7344>>.

### 7.2. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, DOI 10.17487/RFC4398, March 2006, <<http://www.rfc-editor.org/info/rfc4398>>.

## Appendix A. Acknowledgments

This document is generated using the mmark tool that Miek Gieben has developed. We thank number of people that have provided feedback and useful comments including Bob Harold, John Levine, Matthijs Mekking(!), Dan York, Shane Kerr, Jacques Latour.

## Authors' Addresses

Olafur Gudmundsson  
CloudFlare

Email: [olafur+ietf@cloudflare.com](mailto:olafur+ietf@cloudflare.com)

Paul Wouters  
Red Hat

Email: [pwouters@redhat.com](mailto:pwouters@redhat.com)

Domain Name System Operations (dnsop) Working Group  
Internet-Draft  
Updates: 1034, 2308 (if approved)  
Intended status: Standards Track  
Expires: March 22, 2017

S. Bortzmeyer  
AFNIC  
S. Huque  
Verisign Labs  
September 18, 2016

NXDOMAIN really means there is nothing underneath  
draft-ietf-dnsop-nxdomain-cut-05

## Abstract

This document states clearly that when a DNS resolver receives a response with response code of NXDOMAIN, it means that the domain name which is thus denied AND ALL THE NAMES UNDER IT do not exist.

REMOVE BEFORE PUBLICATION: this document should be discussed in the IETF DNSOP (DNS Operations) group, through its mailing list. The source of the document, as well as a list of open issues, is currently kept at Github [1].

This documents clarifies RFC 1034 and modifies a portion of RFC 2308, so it updates both of them.

## 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 <http://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 22, 2017.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction and background . . . . .	2
1.1. Terminology . . . . .	3
2. Rule . . . . .	3
3. Updates to RFCs . . . . .	5
3.1. Updates to RFC1034 . . . . .	5
3.2. Updates to RFC2308 . . . . .	5
4. Benefits . . . . .	5
5. Possible issues . . . . .	6
6. Implementation considerations . . . . .	6
7. IANA Considerations . . . . .	7
8. Security Considerations . . . . .	7
9. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION	7
10. Acknowledgments . . . . .	8
11. References . . . . .	8
11.1. Normative References . . . . .	8
11.2. Informative References . . . . .	9
11.3. URIs . . . . .	10
Appendix A. Why can't we just use the owner name of the returned SOA? . . . . .	10
Appendix B. Related approaches . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction and background

The DNS protocol [RFC1035] defines response code 3 as "Name Error", or "NXDOMAIN" [RFC2308], which means that the queried domain name does not exist in the DNS. Since domain names are represented as a tree of labels ([RFC1034], Section 3.1), non-existence of a node implies non-existence of the entire sub-tree rooted at this node.

The DNS iterative resolution algorithm precisely interprets the NXDOMAIN signal in this manner. If it encounters an NXDOMAIN response code from an authoritative server, it immediately stops iteration and returns the NXDOMAIN response to the querier.

However, in most known existing resolvers today, a cached non-existence for a domain is not considered "proof" that there can be no child domains underneath. This is due to an ambiguity in [RFC1034]



that failed to distinguish Empty Non-Terminal names (ENT) ([RFC7719]) from nonexistent names (Section 3.1). The distinction became especially important for the development of DNSSEC, which provides proof of non-existence. [RFC4035], section 3.1.3.2, describes how security-aware authoritative name servers make the distinction, but no existing RFCs describe the behavior for recursive name servers.

This document specifies that an NXDOMAIN response for a domain name means that no child domains underneath the queried name exist either. And furthermore, that DNS resolvers should interpret cached non-existence in this manner. Since the domain names are organized in a tree, it is a simple consequence of the tree structure: non-existence of a node implies non-existence of the entire sub-tree rooted at this node.

### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

"Denied name": the domain name whose existence has been denied by a response of rcode NXDOMAIN. In most cases, it is the QNAME but, because of [RFC6604], it is not always the case.

Other terms are defined in [RFC1034] or [RFC1035] or (like NXDOMAIN itself) in the more recent [RFC7719].

The domain name space is conceptually defined in terms of a tree structure. The implementation of a DNS resolver/cache MAY use a tree or other data structures. The cache being a subset of the data in the domain name space, it is much easier to reason about it in terms of that tree structure and to describe things in those terms (names under/above, descendent names, subtrees etc). In fact, the DNS algorithm description in [RFC1034] even states an assumption that the cache is a tree structure, so the precedent is already well established: see its section 4.3.2 which says "The following algorithm assumes that the RRs are organized in several tree structures, one for each zone, and another for the cache..." So, in this document, each time we talk of a tree or tree operations, it refers to the model, not to the actual implementation.

## 2. Rule

When an iterative caching DNS resolver receives a response NXDOMAIN, it SHOULD store it in its cache and all names and RRsets at or below that node SHOULD then be considered to be unreachable. Subsequent queries for such names SHOULD elicit an NXDOMAIN response.

But if a resolver has cached data under the NXDOMAIN cut, it MAY continue to send it as a reply (until the TTL of this cached data expires), since this may avoid additional processing when a query is received. Section 6 provides more information about this.

Another exception is that a validating resolver MAY decide to implement the "NXDOMAIN cut" behaviour (described in the first paragraph of this section) only when the NXDOMAIN response has been validated with DNSSEC. See Section 8 for the rationale.

The fact that a subtree does not exist is not forever: [RFC2308], section 3, already describes the amount of time that an NXDOMAIN response may be cached (the "negative TTL").

If the NXDOMAIN response due to a cached non-existence is from a DNSSEC signed zone, then it will have accompanying NSEC or NSEC3 records that authenticate the non-existence of the name. For a descendant name of the original NXDOMAIN name, the same set of NSEC or NSEC3 records proves the non-existence of the descendant name. The iterative, caching resolver MUST return these NSEC or NSEC3 records in the response to the triggering query if the query had the DNSSEC OK (DO) bit set.

Warning: if there is a chain of CNAME (or DNAME), the name which does not exist is the last of the chain ([RFC6604]) and not the QNAME. The NXDOMAIN stored in the cache is for the denied name, not always for the QNAME.

As an example of the consequence of these rules, consider two successive queries to a resolver, with a non-existing domain 'foo.example': the first is for 'foo.example' (which results in an NXDOMAIN) and the second for 'bar.foo.example' (which also results in an NXDOMAIN). Many resolvers today will forward both queries, as noticed in Section 9. However, following the rules in this document ("NXDOMAIN cut"), a resolver would cache the first NXDOMAIN response, as a sign of non-existence, and then immediately return an NXDOMAIN response for the second query, without transmitting it to an authoritative server.

If the first request is for 'bar.foo.example' and the second for 'baz.foo.example', the first NXDOMAIN response won't tell anything about 'baz.foo.example' and therefore the second query will be transmitted as it was before the use of "NXDOMAIN cut" optimisation (see Appendix A).

### 3. Updates to RFCs

#### 3.1. Updates to RFC1034

This document clarifies possible ambiguities in [RFC1034] that did not clearly distinguish Empty Non-Terminal names (ENT) ([RFC7719]) from nonexistent names, and refers to subsequent documents that do. Empty Non-Terminals are nodes in the DNS that have no resource record sets associated with them, but have descendant nodes that do. The correct response to Empty Non-Terminals is NODATA (i.e. a response code of NOERROR and an empty ANSWER section). Additional clarifying language on these points is provided in section 7.16 of [RFC2136] and sections 2.2.2 and 2.2.3 of [RFC4592].

#### 3.2. Updates to RFC2308

The second paragraph of section 5 of [RFC2308] states the following: "A negative answer that resulted from a name error (NXDOMAIN) should be cached such that it can be retrieved and returned in response to another query for the same <QNAME, QCLASS> that resulted in the cached negative response."

This document revises that paragraph to the following: "A negative answer that resulted from a name error (NXDOMAIN) should be cached such that it can be retrieved and returned in response to another query for the same <QNAME, QCLASS> that resulted in the cached negative response, or where QNAME is a descendant of the original QNAME, and QCLASS is the same."

The above section Section 2 elaborates on the revised rule and specifies when it may be reasonable to relax or ignore it.

### 4. Benefits

The main benefit is a better efficiency of the caches. In the example above, the resolver sends only one query instead of two, the second one being answered from the cache. This will benefit the entire DNS ecosystem, since the authoritative name servers will have less unnecessary traffic to process.

The correct behavior (in [RFC1034] and made clearer in this document) is specially useful when combined with QNAME minimisation [RFC7816] since it will allow a resolver to stop searching as soon as an NXDOMAIN is encountered.

"NXDOMAIN cut" may also help mitigate certain types of random QNAME attacks [joost-dnsterior] [balakrichen-dafa888], where there is a fixed suffix which does not exist. In these attacks against the

authoritative name server, queries are sent to resolvers for a QNAME composed of a fixed suffix ("dafa888.wf" in one of the articles above), which is typically nonexistent, and a random prefix, different for each request. A resolver receiving these requests have to forward them to the authoritative servers. With "NXDOMAIN cut", a system administrator would just have to send to the resolver a query for the fixed suffix, the resolver would get a NXDOMAIN and then would stop forwarding the queries. (It would be better if the SOA record in the NXDOMAIN response were sufficient to find the non-existing domain but it is not the case, see Appendix A.)

## 5. Possible issues

Let's assume the TLD example exists but foobar.example is not delegated (so the example's name servers will reply NXDOMAIN for a query about anything.foobar.example). A system administrator decides to name the internal machines of his organization under office.foobar.example and uses a trick of his resolver to forward requests about this zone to his local authoritative name servers. "NXDOMAIN cut" would create problems here, since, depending on the order of requests to the resolver, it may have cached the non-existence from example and therefore "deleted" everything under. This document assumes that such setup is rare and does not need to be supported.

Another issue that may happen: today, we see broken authoritative name servers which reply to ENT ([RFC7719], section 6) with NXDOMAIN instead of the normal NODATA ([RFC7719], section 3).

RFC-EDITOR: REMOVE THE PARAGRAPH BEFORE PUBLICATION. An example today is mta2.\_domainkey.cbs.nl (which exists) where querying \_domainkey.cbs.nl yields NXDOMAIN. Another example is www.upenn.edu, redirected to www.upenn.edu-dscg.edgesuite.net while a query for edu-dscg.edgesuite.net returns NXDOMAIN.

Such name servers are definitely wrong and have always been. Their behaviour is incompatible with DNSSEC. Given the advantages of "NXDOMAIN cut", there is little reason to support this behavior.

## 6. Implementation considerations

This section is non-normative, and is composed only of various things which may be useful for implementors. A recursive resolver may implement its cache in many ways. The most obvious one is a tree data structure, because it fits the data model of domain names. But, in practice, other implementations are possible, as well as various optimisations (such as a tree, augmented by an index of some common domain names).

If a resolver implements its cache as a tree (without any optimisation), one way to follow the rules of Section 2 is, when receiving the NXDOMAIN, to prune the subtree of positive cache entries at that node, or to delete all individual cache entries for names below that node. Then, when searching downward in its cache, this iterative caching DNS resolver will stop searching if it encounters a cached non-existence.

Some resolver may have a cache which is NOT organized as a tree (but, for instance, as a dictionary) and therefore have a reason to ignore the rules of Section 2. So these rules are a SHOULD and not a MUST.

#### 7. IANA Considerations

This document has no actions for IANA.

#### 8. Security Considerations

The technique described here may help against a denial-of-service attack named "random qnames" and described in Section 4.

If a resolver does not validate the answers with DNSSEC, or if the zone is not signed, the resolver can of course be poisoned with a false NXDOMAIN, thus "deleting" a part of the domain name tree. This denial-of-service attack is already possible without the rules of this document (but "NXDOMAIN cut" may increase its effects). The only solution is to use DNSSEC.

#### 9. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature.

It is up to the individual working groups to use this information as they see fit".

As of today, practically all existing DNS resolvers are conservative by default: they consider a NXDOMAIN as only significant for the denied name itself, not for the names under. Almost all the current recursive servers will send upstream a query for out-of-cache sub.example.com even if their cache contains an NXDOMAIN for example.com.

There are a few exceptions. The Unbound resolver has a configuration parameter called "harden-below-nxdomain" [2], which if set to "yes" turns on "NXDOMAIN cut" behavior ("only DNSSEC-secure nxdomains are used", see Section 8). The PowerDNS recursor has optional partial support for "NXDOMAIN cut", for the root domain only, with its "root-nx-trust" setting, described as [3] "If set, an NXDOMAIN from the root-servers will serve as a blanket NXDOMAIN for the entire TLD the query belonged to. The effect of this is far fewer queries to the root-servers.".

## 10. Acknowledgments

The main idea in this document is taken from [I-D.vixie-dnsexst-resimprove], section 3, "Stopping Downward Cache Search on NXDOMAIN". Thanks to its authors, Paul Vixie, Rodney Joffe, and Frederico Neves. Additionally Tony Finch, Ted Lemon, John Levine, Jinmei Tatuya, Bob Harold and Duane Wessels provided valuable feedback and suggestions.

## 11. References

### 11.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://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, <<http://www.rfc-editor.org/info/rfc2119>>.

- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<http://www.rfc-editor.org/info/rfc2308>>.
- [RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <<http://www.rfc-editor.org/info/rfc4592>>.
- [RFC6604] Eastlake 3rd, D., "xNAME RCODE and Status Bits Clarification", RFC 6604, DOI 10.17487/RFC6604, April 2012, <<http://www.rfc-editor.org/info/rfc6604>>.

## 11.2. Informative References

- [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, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<http://www.rfc-editor.org/info/rfc7719>>.
- [RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <<http://www.rfc-editor.org/info/rfc7816>>.
- [I-D.vixie-dnsexst-resimprove] Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsexst-resimprove-00 (work in progress), June 2010.
- [I-D.ietf-dnsop-nsec-aggressiveuse] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive use of NSEC/NSEC3", draft-ietf-dnsop-nsec-aggressiveuse-02 (work in progress), September 2016.

[joost-dnsterror]

Joost, M., "About DNS Attacks and ICMP Destination Unreachable Reports", December 2014, <<http://www.michael-joost.de/dnsterror.html>>.

[balakrichenan-dafa888]

Balakrichenan, S., "Disturbance in the DNS - "Random qnames", the dafa888 DoS attack", October 2014, <<https://indico.dns-oarc.net/event/20/session/3/contribution/37>>.

### 11.3. URIs

[1] <https://github.com/bortzmeyer/ietf-dnsop-nxdomain>

[2] <https://www.unbound.net/documentation/unbound.conf.html>

[3] <https://doc.powerdns.com/md/recursor/settings/#root-nx-trust>

### Appendix A. Why can't we just use the owner name of the returned SOA?

In this document, we deduce the non-existence of a domain only for NXDOMAIN answers where the denied name was the exact domain. If a resolver sends a query to the name servers of the TLD example, asking for the MX record for `www.foobar.example`, and subsequently receives a NXDOMAIN, it can only register the fact that `www.foobar.example` (and everything underneath) does not exist. This is true regardless if the accompanying SOA record is for the domain `example` only. One cannot infer that `foobar.example` is nonexistent. The accompanying SOA record indicates the apex of the zone, not the closest existing domain name. So, using the owner name of the SOA record in the Authoritative section to deduce "NXDOMAIN cuts" is currently definitely not OK.

RFC-EDITOR: REMOVE BEFORE PUBLICATION: to use a real example today, ask the authoritative name servers of the TLD `fr` about `anything.which.does.not.exist.gouv.fr`. The SOA will indicate `fr` (the apex) even while `gouv.fr` does exist (there is no zone cut between `gouv.fr` and `fr`).

Deducing the non-existence of a node from the SOA in the NXDOMAIN reply may certainly help with random qnames attacks but this is out-of-scope for this document. It would require to address the problems mentioned in the first paragraph of this section. A possible solution would be, when receiving a NXDOMAIN with a SOA which is more than one label up in the tree, to send requests for the domains which are between the QNAME and the owner name of the SOA. (A resolver



which does DNSSEC validation or QNAME minimisation will need to do it, anyway.)

#### Appendix B. Related approaches

The document [I-D.ietf-dnsop-nsec-aggressiveuse] describes another way to address some of the same concerns (decreasing the traffic for non-existing domain names). Unlike "NXDOMAIN cut", it requires DNSSEC but it is more powerful since it can synthesize NXDOMAINs for domains that were not queried.

#### Authors' Addresses

Stephane Bortzmeyer  
AFNIC  
1, rue Stephenson  
Montigny-le-Bretonneux 78180  
France

Phone: +33 1 39 30 83 46  
Email: [bortzmeyer+ietf@nic.fr](mailto:bortzmeyer+ietf@nic.fr)  
URI: <http://www.afnic.fr/>

Shumon Huque  
Verisign Labs  
12061 Bluemont Way  
Reston 20190  
USA

Email: [shuque@verisign.com](mailto:shuque@verisign.com)  
URI: <http://www.verisignlabs.com/>

Network Working Group  
Internet-Draft  
Updates: 1034, 1035 (if approved)  
Intended status: Standards Track  
Expires: February 15, 2019

J. Abley  
Afilias  
O. Gudmundsson  
M. Majkowski  
Cloudflare Inc.  
E. Hunt  
ISC  
August 14, 2018

Providing Minimal-Sized Responses to DNS Queries that have QTYPE=ANY  
draft-ietf-dnsop-refuse-any-07

Abstract

The Domain Name System (DNS) specifies a query type (QTYPE) "ANY". The operator of an authoritative DNS server might choose not to respond to such queries for reasons of local policy, motivated by security, performance or other reasons.

The DNS specification does not include specific guidance for the behaviour of DNS servers or clients in this situation. This document aims to provide such guidance.

This document updates RFC 1034 and RFC 1035.

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 February 15, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
2. Motivations for Use of ANY Queries . . . . .	3
3. General Approach . . . . .	4
4. Behaviour of DNS Responders . . . . .	4
4.1. Answer with a Subset of Available RRSets . . . . .	5
4.2. Answer with a Synthesised HINFO RRSet . . . . .	5
4.3. Answer with Best Guess as to Intention . . . . .	6
4.4. Behaviour with TCP Transport . . . . .	6
5. Behaviour of DNS Initiators . . . . .	6
6. HINFO Considerations . . . . .	7
7. Updates to RFC 1034 and RFC 1035 . . . . .	7
8. Implementation Experience . . . . .	8
9. Security Considerations . . . . .	8
10. IANA Considerations . . . . .	8
11. Acknowledgements . . . . .	8
12. References . . . . .	9
12.1. Normative References . . . . .	9
12.2. Informative References . . . . .	9
12.3. URIs . . . . .	9
Appendix A. Editorial Notes . . . . .	10
A.1. Change History . . . . .	10
A.1.1. draft-ietf-dnsop-refuse-any-07 . . . . .	10
A.1.2. draft-ietf-dnsop-refuse-any-06 . . . . .	10
A.1.3. draft-ietf-dnsop-refuse-any-05 . . . . .	10
A.1.4. draft-ietf-dnsop-refuse-any-04 . . . . .	10
A.1.5. draft-ietf-dnsop-refuse-any-03 . . . . .	10
A.1.6. draft-ietf-dnsop-refuse-any-02 . . . . .	10
A.1.7. draft-ietf-dnsop-refuse-any-01 . . . . .	11
A.1.8. draft-ietf-dnsop-refuse-any-00 . . . . .	11
A.1.9. draft-jabley-dnsop-refuse-any-01 . . . . .	11
A.1.10. draft-jabley-dnsop-refuse-any-00 . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

The Domain Name System (DNS) specifies a query type (QTYPE) "ANY". The operator of an authoritative DNS server might choose not to respond to such queries for reasons of local policy, motivated by security, performance or other reasons.

The DNS specification [RFC1034] [RFC1035] does not include specific guidance for the behaviour of DNS servers or clients in this situation. This document aims to provide such guidance.

### 1.1. Terminology

This document uses terminology specific to the Domain Name System (DNS), descriptions of which can be found in [RFC7719].

In this document, "ANY Query" refers to a DNS meta-query with QTYPE=ANY. An "ANY Response" is a response to such a query.

In this document, "conventional ANY response" means an ANY response that is constructed in accordance with the algorithm documented in section 4.3.2 of [RFC1034] and specifically without implementing any of the mechanisms described in this document.

In an exchange of DNS messages between two hosts, this document refers to the host sending a DNS request as the initiator, and the host sending a DNS response as the responder.

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. Motivations for Use of ANY Queries

ANY queries are legitimately used for debugging and checking the state of a DNS server for a particular name.

ANY queries are sometimes used as a attempt to reduce the number of queries needed to get information, e.g. to obtain MX, A and AAAA RRSets for a mail domain in a single query. There is no documented guidance available for this use case, however, and some implementations have been observed not to function as perhaps their developers expected. Implementers that assume that an ANY query will ultimately be received by an authoritative server and will fetch all existing RRSets, should include a fallback mechanism to use when that does not happen.

ANY queries are frequently used to exploit the amplification potential of DNS servers/resolvers using spoofed source addresses and UDP transport (see [RFC5358]). Having the ability to return small responses to such queries makes DNS servers less attractive amplifiers.

ANY queries are sometimes used to help mine authoritative-only DNS servers for zone data, since they are expected to return all RRSets for a particular query name. If a DNS operator prefers to reduce the potential for information leaks, they might choose not to send large ANY responses.

Some authoritative-only DNS server implementations require additional processing in order to send a conventional ANY response, and avoiding that processing expense might be desirable.

### 3. General Approach

This proposal provides a mechanism for an authority server to signal that conventional ANY queries are not supported for a particular QNAME, and to do so in such a way that is both compatible with and triggers desirable behaviour by unmodified clients (e.g. DNS resolvers).

Alternative proposals for dealing with ANY queries have been discussed. One approach proposed using a new RCODE to signal that an authoritative server did not answer ANY queries in the standard way. This approach was found to have an undesirable effect on both resolvers and authoritative-only servers; resolvers receiving an unknown RCODE would re-send the same query to all available authoritative servers, rather than suppress future such ANY queries for the same QNAME.

This proposal avoids that outcome by returning a non-empty RRSset in the ANY response, providing resolvers with something to cache and effectively suppressing repeat queries to the same or different authority servers.

### 4. Behaviour of DNS Responders

Below are the three different modes of behaviour by DNS responders when processing queries with QNAMEs that exist, QCLASS=IN and QTYPE=ANY. Operators/Implementers are free to choose whichever mechanism best suits their environment.

1. A DNS responder can choose to select one or a larger subset of the available RRSets at the QNAME.

2. A DNS responder can return a synthesised HINFO resource record. See Section 6 for discussion of the use of HINFO.
3. Resolver can try to give out the most likely records the requester wants. This is not always possible and the result might well be a large response.

Except as described below in this section, the DNS responder MUST follow the standard algorithms when constructing a response.

#### 4.1. Answer with a Subset of Available RRSets

A DNS responder which receives an ANY query MAY decline to provide a conventional ANY response, or MAY instead send a response with a single RRSset (or a larger subset of available RRSets) in the answer section.

The RRSets returned in the answer section of the response MAY consist of a single RRSset owned by the name specified in the QNAME. Where multiple RRSets exist, the responder SHOULD choose a small subset of those available to reduce the amplification potential of the response.

If the zone is signed, appropriate RRSIG records MUST be included in the answer.

Note that this mechanism does not provide any signalling to indicate to a client that an incomplete subset of the available RRSets has been returned.

#### 4.2. Answer with a Synthesised HINFO RRSset

If there is no CNAME present at the owner name matching the QNAME, the resource record returned in the response MAY instead be synthesised, in which case a single HINFO resource record SHOULD be returned. The CPU field of the HINFO RDATA SHOULD be set to RFCXXXX [note to RFC Editor, replace with RFC number assigned to this document]. The OS field of the HINFO RDATA SHOULD be set to the null string to minimize the size of the response.

The TTL encoded for the synthesised HINFO RR SHOULD be chosen by the operator of the DNS responder to be large enough to suppress frequent subsequent ANY queries from the same initiator with the same QNAME, understanding that a TTL that is too long might make policy changes relating to ANY queries difficult to change in the future. The specific value used is hence a familiar balance when choosing TTL for any RR in any zone, and be specified according to local policy.

If the DNS query includes DO=1 and the QNAME corresponds to a zone that is known by the responder to be signed, a valid RRSIG for the RRSets in the answer (or authority if answer is empty) section MUST be returned. In the case of DO=0, the RRSIG SHOULD be omitted.

A system that receives an HINFO response SHOULD NOT infer that the response was generated according to this specification and apply any special processing of the response, since in general it is not possible to tell with certainty whether the HINFO RRSet received was synthesised. In particular, systems SHOULD NOT rely upon the HINFO RDATA described in this section to distinguish between synthesised and non-synthesised HINFO RRSets.

#### 4.3. Answer with Best Guess as to Intention

In some cases it is possible to guess what the initiator wants in the answer (but not always). Some implementations have implemented the spirit of this document by returning all RRSets of RRTYPE CNAME, MX, A and AAAA that are present at the owner name but suppressing others. This heuristic seems to work well in practice, satisfying the needs of some applications whilst suppressing other RRSets such as TXT and DNSKEY that can often contribute to large responses. Whilst some applications may be satisfied by this behaviour, the resulting responses in the general case are larger than the approaches described in Section 4.1 and Section 4.2.

As before, if the zone is signed and the DO bit is set on the corresponding query, an RRSIG RRSet MUST be included in the response.

#### 4.4. Behaviour with TCP Transport

A DNS responder MAY behave differently when processing ANY queries received over different transport, e.g. by providing a conventional ANY response over TCP whilst using one of the other mechanisms specified in this document in the case where a query was received using UDP.

Implementers SHOULD provide configuration options to allow operators to specify different behaviour over UDP and TCP.

#### 5. Behaviour of DNS Initiators

A DNS initiator which sends a query with QTYPE=ANY and receives a response containing an HINFO resource record or a single RRset, as described in Section 4, MAY cache the response in the normal way. Such cached resource records SHOULD be retained in the cache following normal caching semantics, as it would with any other response received from a DNS responder.

A DNS initiator MAY suppress queries with QTYPE=ANY in the event that the local cache contains a matching HINFO resource record with RDATA.CPU field, as described in Section 4. A DNS initiator MAY instead respond to such queries with the contents of the local cache in the usual way.

## 6. HINFO Considerations

It is possible that the synthesised HINFO RRSset in an ANY response, once cached by the initiator, might suppress subsequent queries from the same initiator with QTYPE=HINFO. Thus the use of HINFO in this proposal would hence have effectively mask the HINFO RRSset present in the zone.

Authority-server operators who serve zones that rely upon conventional use of the HINFO RRTYPE SHOULD sensibly choose the "single RRSset" method described in this document or select another type.

The HINFO RRTYPE is believed to be rarely used in the DNS at the time of writing, based on observations made at recursive servers, authority servers and in passive DNS.

## 7. Updates to RFC 1034 and RFC 1035

This document extends the specification for processing ANY queries described in section 4.3.2 of [RFC1034].

It is important to note that returning a subset of available RRSets when processing an ANY query is legitimate and consistent with [RFC1035]; it can be argued that ANY does not always mean ALL, as used in section 3.2.3 of [RFC1035]. The main difference here is that the TC bit SHOULD NOT be set on the response indicating that this is not a complete answer.

This document describes optional behaviour for both DNS initiators and responders, and implementation of the guidance provided by this document is OPTIONAL.

RRSIG queries (i.e. queries with QTYPE=RRSIG) are similar to ANY queries in the sense that they have the potential to generate large responses as well as extra work for the responders that process them, e.g. in the case where signatures are generated on-the-fly. RRSIG RRSets are not usually obtained using such explicit queries, but are rather included in the responses for other RRSets that the RRSIGs cover. This document does not specify appropriate behaviour for RRSIG queries, but note that future such advice might well benefit



from consistency with and experience of the approaches for ANY queries described here.

## 8. Implementation Experience

In October 2015 Cloudflare Authoritative Name server implementation implemented the HINFO response. A few minor problems were reported and have since been resolved.

An implementation of the subset-mode response to ANY queries was implemented in NSD 4.1 in 2016.

An implementation of a single RRSet response to an ANY query was made for BIND9 by Tony Finch, and that functionality was subsequently made available in production releases starting in BIND 9.11.

## 9. Security Considerations

Queries with QTYPE=ANY are frequently observed as part of reflection attacks, since a relatively small query can be used to elicit a large response; this is a desirable characteristic if the goal is to maximize the amplification potential of a DNS server as part of a volumetric attack. The ability of a DNS operator to suppress such responses on a particular server makes that server a less useful amplifier.

The optional behaviour described in this document to reduce the size of responses to queries with QTYPE=ANY is compatible with the use of DNSSEC by both initiator and responder.

## 10. IANA Considerations

The IANA is requested to update the Resource Record (RR) TYPEs Registry [1] entry as follows:

Type	Value	Meaning	Reference
*	255	A request for some or all records the server has available	[RFC1035][RFC6895] [This Document]

## 11. Acknowledgements

David Lawrence provided valuable observations and concrete suggestions. Jeremy Laidman helped make the document better. Tony Finch realized that this document was valuable and implemented it

while under attack. Richard Gibson identified areas where more detail and accuracy was useful. A large number of other people also provided comments and suggestions we thank them all for the feedback.

## 12. References

### 12.1. 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>>.
- [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>>.

### 12.2. Informative References

- [RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <<https://www.rfc-editor.org/info/rfc5358>>.
- [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>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<https://www.rfc-editor.org/info/rfc7719>>.

### 12.3. URIs

- [1] <http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4>

## Appendix A. Editorial Notes

This section (and sub-sections) to be removed prior to publication.

## A.1. Change History

## A.1.1. draft-ietf-dnsop-refuse-any-07

Address AD's concerns: more colour to describe updates to 1034/1035 in the abstract; don't rely upon HINFO RDATA formatting; language cleanup around guess intent. Add Evan as author (originator of the "choose one record" response idea).

## A.1.2. draft-ietf-dnsop-refuse-any-06

Update RFC 1034 as well as RFC 1035; define the term "conventional ANY response"; soften and qualify ANY does not mean ALL; note that the subset mode response lacks signalling.

## A.1.3. draft-ietf-dnsop-refuse-any-05

Minor editorial changes. Soften advice on RRSIG queries. Version bump.

## A.1.4. draft-ietf-dnsop-refuse-any-04

These are the changes requested during WGLC. The title has been updated for readability The behavior section now contains description of three different approaches in order of preference. Text added on behavior over TCP. The document is clear in how it updates from RFC1035. Minor adjustments for readability and remove redundancy.

## A.1.5. draft-ietf-dnsop-refuse-any-03

Change section name to "Updates to RFC1034", few minor grammar changes suggested by Matthew Pounsett and Tony Finch.

Text clarifications, reflecting experience, added implementation experience.

## A.1.6. draft-ietf-dnsop-refuse-any-02

Added suggestion to call out RRSIG is optional when DO=0.

Number of text suggestions from Jeremy Laidman.

## A.1.7. draft-ietf-dnsop-refuse-any-01

Add IANA Considerations

## A.1.8. draft-ietf-dnsop-refuse-any-00

Re-submitted with a different name following adoption at the dnsop WG meeting convened at IETF 94.

## A.1.9. draft-jabley-dnsop-refuse-any-01

Make signing of RRSets in answers from signed zones mandatory.

Document the option of returning an existing RRSets in place of a synthesised one.

## A.1.10. draft-jabley-dnsop-refuse-any-00

Initial draft circulated for comment.

## Authors' Addresses

Joe Abley  
Afilias  
300-184 York Street  
London, ON N6A 1B5  
Canada

Phone: +1 519 670 9327  
Email: jabley@afilias.info

Olafur Gudmundsson  
Cloudflare Inc.

Email: olafur+ietf@cloudflare.com

Marek Majkowski  
Cloudflare Inc.

Email: marek@cloudflare.com

Evan Hunt  
ISC  
950 Charter St  
Redwood City, CA 94063  
USA

Email: [each@isc.org](mailto:each@isc.org)

Network Working Group  
Internet-Draft  
Intended status: Best Current Practice  
Expires: June 27, 2017

P. Koch  
DENIC eG  
M. Larson  
P. Hoffman  
ICANN  
December 24, 2016

Initializing a DNS Resolver with Priming Queries  
draft-ietf-dnsop-resolver-priming-11

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 RRSet for the root zone and the necessary address information for reaching the root servers.

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 <http://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 June 27, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

## 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 list of root server operators and the domain name associated with each one has been stable since 1997. However, there are address changes for the root server domain names, both for IPv4 and IPv6 addresses. However, 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 change is the main reason that resolvers need to do priming instead of just going from a configured list to get a full and accurate list of root servers.

### 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 "." and a QTYPE of NS, and 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 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.)

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 domain names, the recursive resolver SHOULD select the target for a priming query randomly from the list of addresses. The recursive resolver might choose either IPv4 and 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

The resolver MAY set the DNSSEC OK (DO) bit. At the time this document is being published, 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 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 RRSets.

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 RRSets, 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.

#### 5. 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 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.

## 6. IANA Considerations

None.

## 7. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6891>>.

## Appendix A. Acknowledgements

This document is the product of the DNSOP WG and benefitted from the reviews done there.

Authors' Addresses

Peter Koch  
DENIC eG  
Kaiserstrasse 75-77  
Frankfurt 60329  
DE

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

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: May 18, 2017

S. Kerr  
L. Song  
R. Wan  
Beijing Internet Institute  
November 14, 2016

A review of DNS over port 80/443  
draft-shane-review-dns-over-http-04

Abstract

The default DNS transport uses UDP on port 53. There are many motivations why users or operators may prefer to avoid sending DNS traffic in this way. A common solution is to use port 80 or 443; with plain TCP, TLS-encrypted TCP, or full HTTP(S). This memo reviews the possible approaches and delivers some useful information for developers.

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 <http://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 May 18, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	2
2. Different Implementations Approaches . . . . .	3
2.1. DNS over TCP on port 80/443 . . . . .	3
2.2. DNS over TLS on port 443 . . . . .	3
2.3. DNS Wire-format over HTTP(S) . . . . .	4
2.4. REST HTTP API . . . . .	5
3. Acknowledgments . . . . .	6
4. References . . . . .	6
Authors' Addresses . . . . .	7

## 1. Introduction

Name servers use port 53, on both UDP and TCP [RFC1035] [RFC5966]. However, users or operators occasionally find it useful to use an alternative way to deliver DNS information, and often pick port 80 (the default HTTP port) or 443 (the default HTTPS port) for this purpose.

There are several use cases:

- o Case 1: Firewalls or other middleboxes may interfere with normal DNS traffic [RFC3234] [RFC5625] [DOTSE] [SAC035]. In addition, some ISPs and hotels block external DNS and perform DNS rewriting to send users to advertising or other pages that they did intend, or networks may use IP addresses which cause misleading geographic location for the user [RFC7871]. Users may want DNSSEC support which is not deployed locally in such a case, and so on.
- o Case 2: Users may use DNS over TLS or HTTPS to protect privacy. This also allows the DNS client to authenticate the DNS server.
- o Case 3: Developers may want a higher level DNS API. Web developers may prefer different abstractions or familiar tools like JSON or XML, transmitted using HTTP or HTTPS.

This memo does not aim to develop standards or tools. The purpose is to review various implementation options as a reference for developers. However, it may be helpful for anyone hoping to develop specifications or implementations for DNS over 80/443.

Note that most of the implementations described in this memo are on port 80/443 and combined with TCP/TLS/HTTP(S). The main focus here

is between stub resolvers and recursive servers, and the discussion is about the stub resolver to recursive server communication.

## 2. Different Implementations Approaches

### 2.1. DNS over TCP on port 80/443

The simplest approach is just moving the DNS traffic to port 80 or 443 from 53. This approach serves the requirement use case 1.

In this way, the whole protocol is the same as current DNS transport in TCP, except the transport port is moved to port 80 or 443. The difference between port 80 and 443 is that the traffic of port 80 is often intercepted as HTTP traffic for purposes of deeper inspection, while the traffic of port 443 is usually considered to be encrypted, and typically ignored by middle-boxes. One example where DNS is transported through port 80/443 is one of the fallback cases of NLnetLabs' DNSSEC trigger [dnssec-trigger].

Transporting DNS through port 80/443 is easy to implement. Developers can simply run an existing DNS server and configure the DNS software to listen on ports 80/443. The client can also apply this change without any significant changes.

One drawback of this approach is that it might mislead the client because of the port used. For example, clients might think DNS over 443 as a secure protocol because normally the session would be encrypted. In this case, however, it is not.

### 2.2. DNS over TLS on port 443

Another approach is DNS over TLS on port 443, which is also implemented in DNSSEC trigger [dnssec-trigger]. DNS over TLS is documented in [RFC7858], which uses the well-known port 853. Using port 443 to carry the traffic instead still serves the purpose in use case 1, as some middle boxes may block traffic on the port 853. [I-D.ietf-dprive-dns-over-tls] also discusses authentication and privacy profiles.

TLS provide many benefits for DNS. First, it significantly reduces the DNS conversation's vulnerability to being hijacked. Second, like plain TCP or DNS Cookies, it prevents resolvers from being used in amplification or reflection attacks. Additionally, it provides privacy by encrypting the conversation between client and server.

One concern of DNS over TLS is its cost. Compared to UDP, DNS-over-TCP requires an additional round-trip-time (RTT) of latency to establish a TCP connection (although TCP fast open [RFC7413] may

eliminate that in some cases). Use of TLS encryption algorithms adds an additional RTT, and results in slightly higher CPU usage. Keeping a session open may amortize the latency of extra RTT, but at the cost of state on both client and resolver side. Another concern is that the DNS packet over TLS on a new port might be dropped by some middle boxes. Another concern of TLS is the deployment difficulty when authenticating the server. If servers are authenticated, certificate management is required.

### 2.3. DNS Wire-format over HTTP(S)

Different from raw DNS over TCP using port 80/443, another option is encapsulating DNS wire-format data into an HTTP body and sending it as HTTP(S) traffic. It is quite useful in use cases 1 & 2 described in the introduction. This approach has the benefit that HTTP usually makes it through even the worst coffee shop or hotel room firewalls, as working web browsing is expected by Internet users.

Using HTTP also benefits from HTTP's persistent TCP connection pool concept (see section 6.3 in [RFC7230]), which DNS on TCP port 53 does not have. Note that if HTTP/2 (see [RFC7540]) is used then there may be concurrent streams, and answers on different streams may arrive out-of-order.

Finally, as with DNS over TLS, HTTPS provides data integrity and privacy. Use of such encryption is recommended.

The basic methodology works as follows:

1. The client creates a DNS query message.
2. The client encapsulates the DNS message in a HTTP(S) message body and assigns parameters with the HTTP header.
3. The client connects to the server and issues an HTTP(S) POST request method.
4. The server decapsulates the HTTP package to DNS query, and resolves the DNS query.
5. The server encapsulates the DNS response in HTTP(S) and sends it back via the HTTP(S) session.

Note that if the original DNS query is sent by TCP, first two bits of the package is the message length and should be removed. (This is only true if some software is translating from the DNS protocol to DNS over HTTP, for example via a proxy. Native implementations will of course not need this.) There is an implementation of this



methodology in the Go Programming Language (<https://github.com/BII-Lab/DNSoverHTTPinGO>) as well as C (<https://github.com/BII-Lab/DNSoverHTTP>), maintained by BII lab.

In addition to the benefits mentioned before, the HTTP header makes DNS wire-format over HTTP(S) easy to extend. Compared to creating a new option in EDNS0, using new parameters in HTTP header is far easier to deploy, since DNS messages with EDNS0 may not pass some middle boxes.

The DNS wire-format approach has the advantage that any future changes to the DNS protocol will be transparently supported by both client and server, even while continuing to use HTTP.

One disadvantage of packaging DNS into HTTP is its cost. Packing and unpacking uses CPU and may result in higher response time. The DNS over HTTP messages also have a risk of being dropped by firewalls which intercepts HTTP packets. And it should be noted that if HTTPS is used, then all the discussion of the costs, benefits, and security recommendations about TLS in previous section is also applicable here.

#### 2.4. REST HTTP API

As mentioned in use case 3, one motivation of a REST HTTP API is for web developers who need to get DNS information but prefer not to create raw requests. They can work by creating HTTP requests other than real DNS queries.

In this style of implementation DNS data is exchanged in other formats than wire format, like JSON [I-D.hoffman-dns-in-json], or XML [I-D.mohan-dns-query-xml]. There are also lots of REST DNS API developed by DNS or cloud service providers.

Most of these APIs are developed in the scope of their own system with different specification. But a typical query is a client will requesting a special formatted URI. This may be via an HTTP POST command, or it may encode the contents of the DNS query in the URI directly. Usually there is a HTTP(S) server listening to port 80/443, which will parse the request and create a DNS query or DNS operation command towards the real DNS. Unlike wire-format DNS over HTTP(S), once the HTTP(S) server receives the response, it create the response by putting DNS data into various structured formats like JSON, XML, YAML, or even plain text.

However, such an approach may have issues, because it is not based on traditional DNS protocol. So there is no guarantee of the protocol's completeness and correctness. Support for DNSSEC might also be a

problem because the response usually do not contain RR records with the answer, making it impossible for a client to validate the reply.

As with DNS using DNS wire-format over HTTP, use of encryption is encouraged.

### 3. Acknowledgments

Thanks to Jinmei Tatuya for review. Thanks to Robert Edmonds for pushing for encryption. Thanks to Mark Delany for raising the issue of out-of-order responses. Thanks to Ted Hardie for mentioning the idea of encoding the DNS query directly in the URI.

### 4. References

- [dnssec-trigger]  
"Dnssec-Trigger", <<https://www.nlnetlabs.nl/projects/dnssec-trigger/>>.
- [DOTSE] Aehlund, J. and P. Wallstroem, "DNSSEC Tests of Consumer Broadband Routers", February 2008, <[http://www.iis.se/docs/Routertester\\_en.pdf](http://www.iis.se/docs/Routertester_en.pdf)>.
- [I-D.hoffman-dns-in-json]  
Hoffman, P., "Representing DNS Messages in JSON", draft-hoffman-dns-in-json-10 (work in progress), October 2016.
- [I-D.ietf-dprive-dns-over-tls]  
Zi, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over TLS", draft-ietf-dprive-dns-over-tls-07 (work in progress), March 2016.
- [I-D.mohan-dns-query-xml]  
Parthasarathy, M. and P. Vixie, "Representing DNS messages using XML", draft-mohan-dns-query-xml-00 (work in progress), September 2011.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <<http://www.rfc-editor.org/info/rfc3234>>.

- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<http://www.rfc-editor.org/info/rfc5625>>.
- [RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <<http://www.rfc-editor.org/info/rfc5966>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<http://www.rfc-editor.org/info/rfc7413>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<http://www.rfc-editor.org/info/rfc7540>>.
- [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, <<http://www.rfc-editor.org/info/rfc7858>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<http://www.rfc-editor.org/info/rfc7871>>.
- [SAC035] ICANN Security and Stability Advisory Committee, "DNSSEC Impact on Broadband Routers and Firewalls", 2008.

#### Authors' Addresses

Shane Kerr  
Beijing Internet Institute  
2/F, Building 5, No.58 Jinghai Road, BDA  
Beijing 100176  
CN

Email: [shane@biigroup.cn](mailto:shane@biigroup.cn)  
URI: <http://www.biigroup.com/>

Linjian Song  
Beijing Internet Institute  
2508 Room, 25th Floor, Tower A, Time Fortune  
Beijing 100028  
P. R. China

Email: [songlinjian@gmail.com](mailto:songlinjian@gmail.com)  
URI: <http://www.biigroup.com/>

Runxia Wan  
Beijing Internet Institute  
2508 Room, 25th Floor, Tower A, Time Fortune  
Beijing 100028  
P. R. China

Email: [rxwan@biigroup.cn](mailto:rxwan@biigroup.cn)  
URI: <http://www.biigroup.com/>

Internet Engineering Task Force  
Internet-Draft  
Intended status: Experimental  
Expires: December 23, 2016

L. Song  
Beijing Internet Institute  
P. Vixie  
TISF  
S. Kerr  
R. Wan  
Beijing Internet Institute  
June 21, 2016

DNS wire-format over HTTP  
draft-song-dns-wireformat-http-04

Abstract

This memo introduces a way to tunnel DNS data over HTTP. This may be useful in any situation where DNS is not working properly, such as when there is middlebox misbehavior.

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 <http://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 December 23, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	2
2. Methodology and Configuration . . . . .	3
3. DNS-over-HTTP Message Format . . . . .	4
3.1. Request Method . . . . .	4
3.2. Response Status Code . . . . .	5
3.3. Header Fields . . . . .	6
3.4. Message Body . . . . .	6
4. Security Considerations . . . . .	7
5. IANA considerations . . . . .	7
6. Acknowledgments . . . . .	8
7. References . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

RFC 1035 [RFC1035] specifies the wire format for DNS messages. It also specifies DNS transport on UDP and TCP on port 53, which is still used today. However, there are other ways to access DNS database, for example in a different data format or via alternative DNS transport. These approaches are summarized in [draft-shane-review-DNS-over-http].

One of alternative way of using DNS described in that document is to transport DNS binary data inside HTTP, with the goal of improving DNS service availability. The DNS traffic is simply sent as web traffic using port 80/443 over HTTP. It can bypass badly behaving middle boxes like firewalls, proxies or traffic shaping devices on path which might interfere with normal DNS traffic [RFC5625] [DOTSE] [SAC035].

This approach has the advantage that HTTP usually makes it through even the worst coffee shop or hotel room firewalls, as Internet users expect web browsing to always work. It also benefits from HTTP's support for persistent TCP connections (see section 6.3 in [RFC7230]). Note that 5966bis [I-D.ietf-dnsop-5966bis] specifies the persistent feature for DNS on TCP port 53, but current DNS software does not generally support this mode of operation. Finally, HTTPS provides data integrity and privacy.

One alternative idea is to simply use a VPN, rather than a custom protocol for this purpose. While this is possible, the DNS over HTTP wire format protocol presented here works on an actual HTTP server, so it can be hosted on a machine that also serves web pages. This

means that DNS over HTTP is slightly more "stealthy" than a VPN, in that it can be indistinguishable from normal web traffic.

Unlike a REST DNS API using JSON [I-D.bortzmeyer-dns-json] or XML [I-D.mohan-dns-query-xml] encoding for DNS data, in this approach wire-format data is wrapped with a HTTP header and transmitted on port 80 or 443. The protocol is intended to serve as a sort of DNS VPN, and does not introduce another format of DNS data. It is also possible as a new DNS APIs for advanced usage in application development. For example, web developers can create arbitrary HTTP/HTTPS queries and get DNS responses in their JavaScript apps.

This memo aims to describe how the DNS binary over HTTP concept works. Hopefully implementations by different developers following this memo can speak with each other.

This mechanism is designed for client stub resolver to recursive server. DNS zone transfer, DNS updates, and anything other than simple DNS queries are out-of-scope for this document.

## 2. Methodology and Configuration

As mentioned in introduction, the basic methodology is wrapping the DNS wire-format data into an HTTP header and transmitting on port 80 or 443. However, there are two different scenarios for implementation.

### Scenario 1:

The DNS server implementation handles DNS queries and responses via both UDP and TCP on port 53, and HTTP on port 80 or 443. It works as follows:

1. The client creates a DNS query message.
2. The client encapsulates the DNS message in a HTTP message body and assigns parameters with the HTTP header.
3. The client connects to the server and issues an HTTP POST request method. This may re-use an existing HTTP connection.
4. The server decapsulates the HTTP packet to get the DNS query, and resolves the DNS query.
5. The server encapsulates the DNS response in HTTP and sends it back via the HTTP session.

### Scenario 2:

In this scenario there is a DNS-HTTP proxy sitting between stub-resolver and the recursive server. The stub uses a client proxy and the recursive server uses a server proxy. This works like a DNS VPN and transmits wire-format DNS messages over HTTP between the proxy client and a server, as follows:

1. The stub-resolver sends a DNS query (over UDP or TCP) to the proxy client.
2. The proxy client encapsulates the DNS message in an HTTP message body and assigns parameters with the HTTP header.
3. The proxy client connects to the proxy server and issues an HTTP POST request method. This may re-use an existing HTTP connection.
4. The proxy server decapsulates the HTTP packet to get the DNS query, and sends it to a real DNS server over UDP/TCP.
5. The proxy server encapsulates the DNS response in HTTP and sends it back via the HTTP session.
6. The proxy client decapsulates the DNS message from the HTTP response and sends it back to the stub-resolver via previous DNS session (either UDP or TCP).

It is possible that these scenarios are mixed. The server may speak DNS over HTTP directly and the client use a proxy, or the other way around.

Note that the proxy client can be implemented listening to a loop-back address in the same host with stub-resolver. The proxy server can be implemented as a caching server as well. It is also possible to use the proxy server as a regular web server at the same time that is acting as a proxy server.

### 3. DNS-over-HTTP Message Format

DNS over HTTP is not tied to a specific version of HTTP, and should work with HTTP 1.1 [RFC7230] and HTTP/2 [RFC7540]. This section describes the details of the DNS over HTTP message format.

#### 3.1. Request Method

A DNS message is sent over HTTP from the client to the server via a properly-formed HTTP request. This is a POST method request [section 4.3.3 in RFC 7231 [RFC7231]]. If a GET method request is sent to the



server, it optionally returns a human-readable page showing information targeted at users.

Note that choosing POST (and not GET) as the request method for DNS wire-format over HTTP is mainly based on two reasons. One is that the protocol is designed using HTTP as a tunnel-like technology carrying data from one side to another, not a web service with RESTful framework. Another is that from the view of implementation some servers or middleboxes may ignore an undefined entity-body if using GET; and HTTP libraries have varying support for using GET with a payload.

The target URI is provided explicitly by the services provider. Derived from the target URI, the request-target in request-line identifies the target resource upon which to apply the request. To avoid URI conflicts and enhance interoperability, DNS wire-format over HTTP uses a well-known URI. As defined in RFC 5785 [RFC5785], it begins with the characters `"/.well-known/"` and the name `"dns-wireformat"`. So the request-target for DNS wire-format over HTTP SHOULD be `'/.well-known/dns-wireformat'`.

A DNS transaction over HTTP has no specific requirements for the transport protocol; developers can use any version of HTTP to accomplish the transaction. But developers should be aware that HTTP 1.1 [RFC7230] and HTTP/2 [RFC7540] do have differences in performance regarding multiplexing. HTTP/2 is fully multiplexed, instead of ordered and blocking. Because there is a general desire to achieve similar performance with DNS over UDP, the modern HTTP/2 is preferred for DNS over HTTP implementation. Note that there should be no problem for advanced HTTP protocol in the future deployed for DNS over HTTP.

### 3.2. Response Status Code

The status code [Section 6 of [RFC7231]], only reflects the status of the HTTP connection. If the request has succeeded, the status code is 200 (OK).

If the request fails, the proxy server will supply an appropriate error code, typically 4xx (client error) if the client has provided a query that the server cannot understand for some reasons, or 5xx (server error) if some server-side problem prevented a query from succeeding.

To be clear, a failure on the DNS side should result in a status code of 200 (OK) as long as the HTTP request is valid and can be answered. The DNS error will be returned via the encapsulated DNS message.

### 3.3. Header Fields

By definition header fields are key:value pairs that can be used to communicate data about the message, its payload, the target resource, or the connection (for example control data).

The Content-Type: header field should be set to "application/dns-wireformat".

We add one a new header field:

Proxy-DNS-Transport: xyz

Where xyz is either UDP or TCP, which is the client's indication of how it received the underlying DNS query, and which the server will use when sending the query to the far-end DNS server. This means if a stub DNS client asks for TCP, then that's what the far-end DNS server will see, and likewise for UDP.

Exposing the transport protocol of the query allows the HTTP server proxy to send DNS queries to the recursive resolver that look like those that the DNS client is sending to the client proxy. If the stub resolver sent a UDP packet, then this allows the recursive resolver to implement the logic of truncating the packet properly, instead of requiring that the HTTP client proxy somehow manage that functionality.

For a stub resolver that connects directly via HTTP to the HTTP server proxy then this flag should be set to TCP, as the entire response can always be delivered so truncation is never required.

The client MUST include this option. If it is missing then it is an error and the server should respond with HTTP code 400 (bad request).

### 3.4. Message Body

As mentioned, the message body is DNS wire-format data. It is worth mentioning that DNS messages sent over TCP connections is prefixed with a two-byte length field which gives the message length [section 4.2.2 in RFC 1035 [RFC1035]], excluding the two-byte length field. This length field allows the low-level processing to assemble a complete message before beginning to parse it. In the context of HTTP, there is content-length header field [section 3.3.2 in [RFC7230]] in which the field-value is the same with two bytes length field in DNS over TCP.

Since this two-byte length field is redundant, when the client proxy receives a DNS message over TCP, it MUST NOT include the length field

in the message sent to the server. The length in the content-length header is only the size of the DNS message itself, and MUST NOT include this two-byte length header.

#### 4. Security Considerations

This protocol does not introduce any new security considerations since it is built on the DNS and HTTP protocols.

Since this protocol transmits DNS messages, all of the security concerns that stub resolvers and recursive resolvers have with DNS messages apply. However, since HTTP uses TCP as the underlying protocol, DNS reflection or amplification attacks are not possible.

Since HTTP is used, all of the security concerns of HTTP are also present.

Servers and clients SHOULD use TLS for communication. It provides privacy and integrity for HTTP sessions. If TLS is used, then all of the security concerns of TLS are also present.

As specified in RFC 5246 [RFC5246], both the HTTP server and client can be authenticated or not authenticated. Clients SHOULD authenticate the certificate of the HTTP server they connect to. The DNS service providers can decide whether to authenticate the client side based on their own requirement for security and privacy. For example, clients of an open resolver do not require authentication.

Note that the ability to perform DNS queries in this way may allow users to bypass local DNS policy. This is problematic in any environment where administrators need to enforce specific DNS behavior, such as an enterprise environment. The protocol outlined here does not introduce any new capabilities in this area, but by creating a more standardized way of doing this it may cause operational problems for enterprise administrators.

#### 5. IANA considerations

Registration for a new Media Type: dns-wireformat

Registration for a new HTTP header field: Proxy-DNS-Transport

Registration for a new Well-Known URI: "dns-wireformat"

## 6. Acknowledgments

Thanks to Bob Harold, Paul Hoffman, and Julian Reschke for review.

## 7. References

- [DOTSE] and , "DNSSEC Tests of Consumer Broadband Routers", February 2008, <[http://www.iis.se/docs/Routertester\\_en.pdf](http://www.iis.se/docs/Routertester_en.pdf)>.
- [I-D.bortzmeyer-dns-json] Bortzmeyer, S., "JSON format to represent DNS data", draft-bortzmeyer-dns-json-01 (work in progress), February 2013.
- [I-D.ietf-dnsop-5966bis] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", draft-ietf-dnsop-5966bis-04 (work in progress), November 2015.
- [I-D.mohan-dns-query-xml] Parthasarathy, M. and P. Vixie, "Representing DNS messages using XML", draft-mohan-dns-query-xml-00 (work in progress), September 2011.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<http://www.rfc-editor.org/info/rfc5625>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.

- [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, <<http://www.rfc-editor.org/info/rfc7231>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<http://www.rfc-editor.org/info/rfc7540>>.
- [SAC035] ICANN Security and Stability Advisory Committee, "DNSSEC Impact on Broadband Routers and Firewalls", 2008.

## Authors' Addresses

Linjian Song  
Beijing Internet Institute  
2508 Room, 25th Floor, Tower A, Time Fortune  
Beijing 100028  
P. R. China

Email: [songlinjian@gmail.com](mailto:songlinjian@gmail.com)  
URI: <http://www.biigroup.com/>

Paul Vixie  
TISF  
11400 La Honda Road  
Woodside, California 94062  
US

Email: [vixie@tisf.net](mailto:vixie@tisf.net)  
URI: <http://www.redbarn.org/>

Shane Kerr  
Beijing Internet Institute  
2/F, Building 5, No.58 Jinghai Road, BDA  
Beijing 100176  
CN

Email: [shane@biigroup.cn](mailto:shane@biigroup.cn)  
URI: <http://www.biigroup.com/>

Runxia Wan  
Beijing Internet Institute  
2508 Room, 25th Floor, Tower A, Time Fortune  
Beijing 100028  
P. R. China

Email: rxwan@biigroup.cn  
URI: <http://www.biigroup.com/>

IETF  
Internet-Draft  
Updates: 6895 (if approved)  
Intended status: Best Current Practice  
Expires: January 9, 2017

A. Sullivan  
Dyn, Inc.  
July 8, 2016

The DNS Is Not Classy: DNS Classes Considered Useless  
draft-sullivan-dns-class-useless-03

Abstract

Domain Name System Resource Records are identified in part by their class. The class field is not effective, and it is not used the way it appears to have been intended. This memo suspends additions to the DNS class registry pending greater clarity on how classes might be used, and until such clarification requires those defining new RRTYPES to define them for all classes.

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 <http://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 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Classes in the Domain Name System . . . . .	2
2. Why classes are not as useful as they might be . . . . .	3
2.1. Matching rules are class-independent . . . . .	3
2.1.1. Really parallel trees . . . . .	4
2.2. Not all RRTYPEs are careful about class . . . . .	5
2.3. The DNS RRTYPE registry and meaning . . . . .	6
2.4. A new class requires new delegations . . . . .	6
3. DNS classes are effectively vestigial . . . . .	7
4. New RRTYPEs are for all classes . . . . .	7
5. Security considerations . . . . .	7
6. IANA Considerations . . . . .	7
7. Acknowledgements . . . . .	8
8. References . . . . .	8
8.1. Normative References . . . . .	8
8.2. Informative References . . . . .	8
Appendix A. Discussion Venue . . . . .	9
Appendix B. Change History . . . . .	9
Author's Address . . . . .	10

## 1. Classes in the Domain Name System

The Domain Name System (DNS) [RFC1034] [RFC1035] includes two types of division: one by class, and one by "cuts". As [RFC1034] says,

The database for any class is organized, delegated, and maintained separately from all other classes. Since, by convention, the name spaces are the same for all classes, the separate classes can be thought of as an array of parallel namespace trees. Note that the data attached to nodes will be different for these different parallel classes. The most common reasons for creating a new class are the necessity for a new data format for existing types or a desire for a separately managed version of the existing name space.

As of this writing, there are only three "ordinary" classes assigned. Class 1 is the Internet or IN class. Class 3 is the Chaos or CH class. Class 4 is the Hesiod or HS class. Class 2 is noted in [RFC1035] as the CSNET or CS class, but the current registry (at <http://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-2>) no longer includes the assignment.

There are two other assigned classes; these have special purposes. Class 255, ANY or \*, matches any class [RFC1035]. Class 254, NONE,



is used in [RFC2136] to specify certain kinds of operations on RRsets.

Of the ordinary classes, only IN is found in common use on the Internet today. Class CH is now sometimes used to carry certain kinds of name server metadata. It seems new classes might have been useful for a number of features that have been delivered in some other way. Yet classes have not been used, which might suggest that classes are less useful than they otherwise appear to be. (It is also worth observing that classes divide the database, and not the namespace itself. In other words, classes are part of the implementation of the DNS and not a natural feature of domain names as such.) In some ways, the motivations for classes are made clearer by reading [RFC0882] along with the other DNS specifications.

Nevertheless, from time to time someone comes up with a suggestion for why a new class would solve some problem or other. The purpose of this memo is to offer some considerations for those contemplating such innovation.

## 2. Why classes are not as useful as they might be

There are four (or, depending on how one counts, five) problems that make classes less useful than they might be. First, the rules for name matching are independent of the class. Second, specification of resource record types has not always attended to the handling of types across classes; this makes new classes of (at best) uncertain utility. Third, the DNS RRTYPE registry does not have a way of indicating significant differences in meaning for the same type among different classes.

Apart from those technical problems, there is an administrative problem. The Internet name space starts from a common root, which means that any resolver needs to start from the same bootstrap mechanism no matter what classes it uses. But in order for that to work, any new class would need to be delegated from the existing root name servers, or else a new set of policies about how to select the alternative roots would be required. A wrinkle (or possibly a separate problem) is that some possible uses of classes appear not really to require a shared global root. (For instance, it is not clear that Hesiod names really needed to be part of the global namespace.)

### 2.1. Matching rules are class-independent

As noted in Section 1, classes are intended to divide the DNS into separate trees. The class field does not, however, affect the

matching rules for names, so as a practical matter the namespace is primary.

The issue is made plain by considering the matching algorithm for name servers, described in [RFC1034] section 4.3.2. Suppose there are two (imaginary) classes, EG and IE. Imagine, further, that the same name has different RDATA in each class:

```
example.com EG CNAME example.net
```

```
example.com IE CNAME example.org
```

In principle, the above should work because, as noted in Section 1, each class is supposed to be "delegated separately". Therefore, when the name server for example.com in class EG receives the query for example.com, it already knows which class database to search; similarly for example.com in class IE.

Yet while the class delegations are defined to be separate, there is no way to ensure that the NS record RDATA for example.com in class IE, and the NS record RDATA for example.com in class EG, are always different. Indeed, if the different classes of name space are truly managed separately, but the name space is by convention parallel, it would not be surprising that some name server ended up authoritative for the same name in different classes. In [RFC1034], section 3.6.1, there is an example that appears to contain two classes from the same master file and for the same name. This illustrates the principle that the same name server could be authoritative for the same name in different classes. Note that the example might be a mistake, since according to [RFC1035], section 5.2, all entries in a master file for a zone should have the same class. In any case, it is plain that the name is primary, and having matched the name one can then select data according to the class. But this means that the matching rules for names cannot differ across classes, and that makes classes less useful for extending DNS capabilities than they might at first seem.

Once one describes the resolution pattern this way, and given that the IN class is so widely used and other classes so rarely, it is not too surprising that some naive implementations simply assume IN for every resource record. That assumption, of course, makes the class division in the DNS again less useful.

#### 2.1.1. Really parallel trees

It appears that the notion of "parallel namespace trees" is stronger than one might have hoped if one wanted to use classes to do something new in the DNS. When one considers how classes are treated in [RFC1034] and [RFC1035] and their predecessors, that parallelism

becomes less surprising. The examples are all of alternative networking systems to the Internet. Moreover, [RFC1034] says, "Because we want the name space to be useful in dissimilar networks and applications, we provide the ability to use the same name space with different protocol families or management."

If one thinks of classes as simply the way to resolve the same name depending on which sort of networking technology is being used, a strong expectation of completely parallel trees is not surprising. Indeed, in an environment of many different networking and internetworking technologies, it would have been surprising if, when one changed network technologies, a name referred to a completely different target system. At the same time, parallel namespace trees are not formally required.

Since the time when the DNS was defined, Internet technology has largely won out over other network technologies. In addition, the last time a fundamentally new networking technology was introduced to the Internet (with IPv6 in [RFC1883]), the designers treated it as just another part of the IN class (and introduced the AAAA record, in [RFC1886]). So, the reason for the class field in the first place has withered away; and, when the opportunity came to use classes in their originally intended way, the designers of the technology decided not to use them.

## 2.2. Not all RRTYPEs are careful about class

RRTYPEs can either be class-independent, or else they can return very different data depending on the class in question. Not every RRTYPE specification is clear about its definition under various classes. For instance, the original specification of type 55, HIP, appears not to state the class(es) for which it is defined [RFC5205]. The specification of LOC, type 29 (in [RFC1876]), says that the type is defined for classes other than IN, but also says, "The semantics of such use are not defined by this memo."

One might argue that this issue is resolved by [RFC3597], because it specifies that an unknown class and type combination is to be handled as unknown. Formally, of course, that means that every type can be handled regardless of class. But it would appear to reduce the utility of classes yet further, by increasing the probability that many RRs in every class except IN will be treated as unknown. For the purposes of resolution, that might not matter. But administrators and users will be reluctant to embrace a class that does not have good input and validation tools -- a problem that already vexes adoption of new RRTYPEs in class IN.

### 2.3. The DNS RRTYPE registry and meaning

Some RRTYPEs are defined in a class-dependent way. For instance, the A record (type 1) is defined in [RFC1035] to be for class IN only. In [RFC1034], section 3.6, the A record is also defined for class CH. Perhaps unfortunately, the IANA registry for RRTYPEs (at <http://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-4>) does not include an indicator for the class(es) in which the RRTYPE is defined.

It appears, therefore, that the "meaning" field of an RRTYPE definition is required to be class-independent, even though the RDATA for a given type may vary dramatically. For instance, in the case of the A record the RDATA is either a 32-bit IPv4 address or else a domain name and a 16-bit octal address. Across classes, even the number of fields may differ for the same type.

This appears to be yet more evidence for the "strict parallelism" explanation in Section 2.1.1. At the same time, [RFC1034] is not perfectly clear that a data type must have the same meaning in every class, and [RFC6895] does not contain clear instructions on the topic. Moreover, given the vastly different RDATA allowable for the same type across classes it is hard to be certain what is entailed by says that they all "have the same meaning", unless there is a strict requirement that a class only ever differs based on the underlying network technology.

Therefore, if classes were to be used for purposes other than alternative low-level network technologies, the RRTYPE registry ought to be altered to indicate the meaning of a type in each class for which the type is defined. Such an alteration appears to be of questionable value given the overwhelming dominance of the IN class.

### 2.4. A new class requires new delegations

The Internet's DNS system is part of the common name space of the Internet, and that common name space starts from a common root (see [RFC2826] for the arguments about why this must be true). In order to provide for the resolution of a new class, the root name servers would need to respond to resolution requests for that class and provide the delegation data. Current policies about domain name delegation in the root zone appear to apply to the IN class, and it is not clear where responsibility would lie for the policies about a new class. At the very least, a new policy of this sort would need to be worked out for any use where a class had truly global scope.

Alternatively, it is possible to imagine resolvers using a different set of root servers for different classes of query. Such a solution

merely moves the policy problem around, for it would be necessary to develop policies about root server systems for new classes whenever names in some class need to be resolvable in a globally-unambiguous way.

### 3. DNS classes are effectively vestigial

Given the considerations above, it is far from obvious that DNS classes are likely to be useful in the future. At the very least, in order that they could become useful, a number of clarifications to the DNS protocol and operation specifications would be necessary.

In the interests of encouraging interoperation, therefore, additions to the DNS CLASS registry (at <http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-2>) are suspended until such clarifications are forthcoming. New class definitions henceforth will require standards action.

Designers of new name systems should consider the design of classes in the DNS. If a similar feature is desirable, its design needs to be at least clearer and possibly different in order to be useful. Given the the way the DNS has managed to thrive without really using classes, however, it would be worth asking whether the feature is useful at all.

### 4. New RRTYPEs are for all classes

As long as the DNS CLASS registry suspension is in effect, new RRTYPE allocations are required to be defined in a class-independent way.

### 5. Security considerations

This memo creates no new security issues. It might be argued that it could in principle reduce security issues by eliminating a potential source for confusion on the Internet, but classes are so little used that there is probably no improvement in practice.

### 6. IANA Considerations

IANA is hereby requested to update the Domain Name System (DNS) Parameters registry as follows:

- o Update the DNS CLASSes sub-registry to add a reference to this document.
- o Update the DNS CLASSes sub-registry Registration Procedures field to "Standards Action" for decimal classes 1-65279 inclusive.

- o Add this document to the Resource Record (RR) TYPEs sub-registry references.

## 7. Acknowledgements

The author appreciates comments and observations from Mark Andrews, Rob Austein, Ray Bellis, Stephane Bortzmeyer, Avri Doria, John Klensin, Shane Kerr, Warren Kumari, Ed Lewis, Mukund Sivaraman, Paul Vixie, and Lixia Zhang.

## 8. References

### 8.1. Normative References

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

### 8.2. Informative References

- [RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <<http://www.rfc-editor.org/info/rfc882>>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC1876] Davis, C., Vixie, P., Goodwin, T., and I. Dickinson, "A Means for Expressing Location Information in the Domain Name System", RFC 1876, DOI 10.17487/RFC1876, January 1996, <<http://www.rfc-editor.org/info/rfc1876>>.
- [RFC1883] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, DOI 10.17487/RFC1883, December 1995, <<http://www.rfc-editor.org/info/rfc1883>>.
- [RFC1886] Thomson, S. and C. Huitema, "DNS Extensions to support IP version 6", RFC 1886, DOI 10.17487/RFC1886, December 1995, <<http://www.rfc-editor.org/info/rfc1886>>.

- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC2826] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root", RFC 2826, DOI 10.17487/RFC2826, May 2000, <<http://www.rfc-editor.org/info/rfc2826>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<http://www.rfc-editor.org/info/rfc3597>>.
- [RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, DOI 10.17487/RFC5205, April 2008, <<http://www.rfc-editor.org/info/rfc5205>>.

#### Appendix A. Discussion Venue

This Internet-Draft is discussed on the IAB Internet Names and Identifiers Program public list: [inip-discuss@iab.org](mailto:inip-discuss@iab.org).

#### Appendix B. Change History

Note to RFC Editor: this section should be removed prior to publication as an RFC.

00:

- \* Initial version

01:

- \* Clarify the distinction between database and domain names as such
- \* Address question of closing registry
- \* Minor fixes of text

02:

- \* Eliminate argument from class position in message
- \* Sharpen argument from primacy of matching rules
- \* Note network-technology history of class.

- \* Change to status: update 6895 and close class registry

03:

- \* Incorporate feedback from DNSOP meeting at IETF 95
- \* Step back from closing registry: "suspend" registration until the specification is clearer about how classes work
- \* Call for clarification of classes

Author's Address

Andrew Sullivan  
Dyn, Inc.  
150 Dow St  
Manchester, NH 03101  
U.S.A.

Email: [asullivan@dyn.com](mailto:asullivan@dyn.com)



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 20, 2017

T. Lemon  
Nominum, Inc.  
R. Droms  
Cisco, Inc.  
W. Kumari  
Google  
September 16, 2016

Special-Use Names Problem Statement  
draft-tldr-sutld-ps-04

Abstract

The Special-Use Domain Names registration policy in RFC 6761 has been shown through experience to present unanticipated challenges. This memo presents a list, intended to be comprehensive, of the problems that have been identified. In addition it reviews the history of Domain Names and summarizes current IETF publications and some publications from other standards organizations relating to special-use domain names.

[ Ed note: John Levine suggested we use 'Domain Names' instead of 'Internet Names'; this is the only change between -03 and -04. Please let us know which term you prefer. ]

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 <http://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 20, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Problems associated with Special-Use Internet Names . . . . .	3
4. Existing Practice Regarding SUINs . . . . .	6
4.1. Primary SUIN Documents . . . . .	6
4.1.1. IAB Technical Comment on the Unique DNS Root . . . . .	6
4.1.2. Special-Use Domain Names . . . . .	7
4.1.3. MoU Concerning the Technical Work of the IANA . . . . .	9
4.2. Secondary documents relating to the SUTLIN question . . . . .	10
4.2.1. Multicast DNS . . . . .	10
4.2.2. The .onion Special-Use TLD . . . . .	11
4.2.3. Locally Served DNS Zones . . . . .	11
4.2.4. Name Collision in the DNS . . . . .	11
4.2.5. Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis . . . . .	12
4.2.6. Additional Reserved Top Level Domains . . . . .	12
4.3. Summary . . . . .	12
5. History . . . . .	12
6. Contributors . . . . .	14
7. Informative References . . . . .	14
Appendix A. Change Log. . . . .	17
Authors' Addresses . . . . .	17

## 1. Introduction

One of the key services required to use the Internet is name resolution. Name resolution is the process of translating a human-readable symbolic name into some object or set of objects to which the name refers, most typically one or more IP addresses. These names are often referred to as domain names. When reading this document, care must be taken to not assume that the term Domain Name implies the particular protocol for resolving these names, the Domain Name System [RFC1034]. An excellent presentation on this topic can be found in Domain Names [I-D.lewis-domain-names].

At the time of this writing, the IETF has recently been asked to allocate several new special-use top-level Domain Names. In evaluating the process for additional special-use top-level Domain Names as documented in Special-Use Domain Names [RFC6761], the IETF encountered several different sorts of issues. Because of this, the IETF has decided to investigate the problem and decide if and how the RFC 6761 process can be improved, or whether it should be deprecated.

This document presents a list, believed to be complete, of the problems associated with the allocation of special-use names. In support of the particular set of problems described here, the document also includes documentation of existing practice as it relates to the use of Domain Names, as well as a brief history of domain names, and finally to describe the set of problems that exist as reported by various IETF participants with experience in the various aspects of the problem.

## 2. Terminology

For the sake of brevity this document uses a number of abbreviations. These are expanded here:

Domain Name A name which serves as input to a host's ordinary name resolution process, for example 'EXAMPLE.ORG'.

SUDN Special Use Domain Name

SUTLDN Special-Use Top-Level Domain Name

IANA Internet Assigned Numbers Authority

ICANN Internet Corporation for Assigned Names and Numbers

## 3. Problems associated with Special-Use Internet Names

This section presents a list of problems that have been identified with respect to the allocation of special-use names. Solutions to these problems are out of scope. Because of that, problems with solutions to these problems are also out of scope, and will be covered in a separate document.

No assertion is made that any of these problems is more or less important than any other. The point of this is simply to enumerate and briefly describe the problems that have been raised during discussions of the special-use name problem. The degree of detail is intended to be sufficient that that participants in the discussion can agree that the problems they've raised have been adequately described, and no more.

In addition, no assertion is made that all of these problems must be addressed; while each problem has one or more solutions, the solutions may in some cases be mutually contradictory, or may come with costs that do not justify the benefit that would be obtained from solving them.

This is the list of problems:

- o IETF and ICANN independently have remit to assign names out of the namespace that Domain Names represent; a formal coordination process does not exist.
- o Although IETF and ICANN nominally have authority over this namespace, neither organization can enforce that authority over any third party who wants to just start using a subset of the namespace. Reasons for doing this may include:
  - \* Unaware that a process exists for allocating such names
  - \* Intended use is covered by gTLD allocation, but no gTLD allocation is ongoing
  - \* Intended use is covered by gTLD allocation, don't want to pay fee
  - \* Intended use is covered by some IETF process, but don't want to follow process
  - \* Intended use is covered by ICANN or IETF process, but expected outcome is refusal
- o There is demand for more than one name resolution protocol for Domain Names, but Domain Names contain no metadata to indicate which protocol to use to resolve them.
- o When a top-level name is used as a means either of marking the rest of a Domain Name for resolution using a protocol other than DNS, or is used for resolution of names with no global meaning, not all software that processes such names will understand the names' special meanings. Consequently, any such use results in queries for those names being sent to authoritative servers.
- o The RFC6761 process is sufficiently uncertain that some protocol developers have assumed they could not get a name assigned; the process of assigning the first new name following RFC 6761 took more than ten years from beginning to end: longer by a factor of ten than any other part of the protocol development process. Other uses of the process have proceeded more smoothly, but there

is a reasonably justified perception that using this process is likely to be slow and difficult, with an uncertain outcome.

- o There is strong resistance within the IETF to assigning names to things outside of the DNS, for a variety of reasons:
  - \* Requires a mechanism for identifying which of a set of resolution processes is required in order to resolve a particular name.
  - \* Assertion of authority: there is a sense that the namespace is "owned" by the IETF or by ICANN, and so, if a name is allocated outside of that process, the person or entity that allocated that name should suffer some consequence that would, presumably, deter future circumvention of the official process.
  - \* More than one name resolution protocol is bad, in the sense that a single protocol is less complicated to implement and deploy.
  - \* The semantics of alternative resolution protocols may differ from the DNS protocol; DNS has the concept of RRtypes; other protocols may not support RRtypes, or may support some entirely different data structuring mechanism.
  - \* If there is an IETF process through which a name can be allocated at zero cost other than time, this process will be used as an alternative to allocating the name through ICANN.
  - \* Some names that might be allocated would be sufficiently generic that other legitimate uses of those names would overlap with a proposed use, so that assigning the name would preclude some future, better use of it.
  - \* If the IETF allocates a name that some third party or parties believes belongs to them in some way, the IETF could become embroiled in an expensive dispute with those parties.
- o In cases where the IETF has made assignments through the RFC 6761 process, technical mistakes have been made due either to insufficiently well-defined process or to a failure to follow the process that was defined in RFC 6761.
- o In principal, the RFC 6761 process could be used to document the existence of domain names that are not safe to allocate, and provide information on how those names are used in practice. However, attempts to use RFC6761 to accomplish this [I-D.chapin-additional-reserved-tlds] have not been successful.

- o No process exists for checking documents to make sure they don't accidentally assign names (e.g. RFC 7788).
- o Use of the registry is inconsistent--some SUTLIN RFCs specify registry entries; some don't; some specify delegation, some don't.
- o There exists no safe, non-process-violating mechanism for ad-hoc allocation of special-use names.
- o RFC 6761 uses the term "Domain Name" to describe the thing for which special uses are registered. This creates a great deal of confusion because some readers take "Domain Name" to imply the use of the DNS protocol.

The problems we have stated here represent the current understanding of the authors of the document as to the complete set of problems that have been identified during discussion by the working group on this topic. The remainder of this document provides additional context that will be needed for reasoning about these problems.

#### 4. Existing Practice Regarding SUINs

There are three primary and numerous secondary documents to consider when thinking about the Special-Use Domain Names process.

##### 4.1. Primary SUIN Documents

The primary documents are considered primary because they directly address the IETF's past thoughts on this topic in a general way, and also because they describe what the IETF does in practice. Only one of these documents is an IETF consensus document.

##### 4.1.1. IAB Technical Comment on the Unique DNS Root

This document [RFC2826] is not an IETF consensus document, and appears to have been written to address a different problem than the SUDN problem. However, it speaks directly to several of the key issues that must be considered, and of course coming as it does from the IAB, it is rightly given with a great deal of authority despite not being an IETF consensus document.

This document should be considered required reading for IETF participants who wish to express an informed opinion on the topic of SUDNs. The main points that appear relevant to the special use names problem are:

- o The Internet requires a globally unique namespace

- o Private networks may operate private namespaces, but still require that names in the public namespace be globally unique.
- o The Domain Name System [RFC1035] is not the only protocol that may be used for resolving domain names.
- o Users cannot be assumed to know how to distinguish between symbols that have local meaning and symbols that have global meaning. Users may therefore share symbols that incorporate domain names with no global meaning (for example, a URL of 'http://mysite.example.corp', where 'example.corp' is a domain allocated privately and informally as described in [SDO-ICANN-COLL]). Such symbols might refer to the object the user intends to share within that user's context, but either refer to some other object any recipient's context, or might not refer to any object at all in a recipient's context. The effect of this is that the user's intended communication will not be able to be understood by the recipients of the communication. This same problem can also occur simply because a single user copies a name from one context in which it has one meaning, into a different context in which it has a different meaning-- for example copying a '.onion' Domain Name out of a TOR browser, where it has meaning, and pasting this name into an ssh client that doesn't support connecting over TOR. [TODO: Consider "labels" instead of "symbols".]

To boil this down even further, we can take the following advice from this document:

- o Domain names with unambiguous global meaning are preferable to domain names with local meaning which will be ambiguous. Nevertheless both globally-meaningful and locally-special names are in use and must be supported.
- o At the time of the writing of this document the IAB was of the opinion that there might well be more than one name resolution protocol used to resolve domain names.

#### 4.1.2. Special-Use Domain Names

The second important document is Special-Use Domain Names [RFC6761]. RFC6761 represents the current IETF consensus on designating and recording SUDNs. The IETF has experienced problems with the designation process described in RFC6761; these concerns motivate this document. Again, familiarity with RFC6761 is a prerequisite for having an informed opinion on the topic of SUDNs.

RFC 6761 defines two aspects of SUDNs: designating a domain name to have a special purpose and registering that special use in the Special-Use Domain Names registry. The designation process is defined in a single sentence (RFC6761, section 4):

If it is determined that special handling of a name is required in order to implement some desired new functionality, then an IETF "Standards Action" or "IESG Approval" specification [RFC5226] MUST be published describing the new functionality.

This sentence implies that any designation of a special-use name is subject to the same open review and consensus process as used to produce and publish all other IETF specifications.

The registration process is a purely mechanical process, in which the existence of the newly designated special use name is recorded, with a pointer to a section in the relevant specification document that defines the ways in which special handling is to be applied to the name.

RFC6761 provided the process whereby Multicast DNS [RFC6762] designated ".local" as a special-use name and included it in the Special-Use Names registry. It itself also enumerated a set of names that had been previously used or defined to have special uses prior to the publication of RFC6761. Since there had been no registry for these names prior to the publication of RFC 6761, the documents defining these names could not have added them to the registry.

There are at least several important points to think of with respect to the RFC6761:

- o A special-use name may be a name that should be resolved using the DNS protocol with no special handling. An example of this is .ARPA.
- o A special-use name may be a name that is resolved using the DNS protocol, requires no special handling in the stub resolver, but requires special handling in the recursive resolver. An example of this would be "10.in-addr.arpa."
- o A special-use name may be name that requires special handling in the stub resolver. An example would be a special-use top-level name like '.local' which acts as a signal to indicate that the local stub resolver should use a non-DNS protocol for name resolution.
- o The current IETF consensus (from a process perspective, not necessarily from the perspective of what would be consensus if the



IETF were to attempt to produce a new consensus document) is that these purposes for special-use names are valid. [TODO: "Both" implies that there are only two applications; the above bullet points outline 3...]

The term "stub resolver" in this case does not mean "DNS protocol stub resolver." The stub resolver is the entity within a particular software stack that takes a question about a Domain name and answers it. One way a stub resolver can answer such a question is using the DNS protocol, but it is in the stub resolver, as we are using the term here, that the decision as to whether to use a protocol, and if so which protocol, or whether to use a local database of some sort, is made.

RFC6761 does not limit special-use names to TLDs. However, at present, all special-use names registered in the IANA Special-Use Domain Names registry [SDO-IANA-SUDR] are either intended to be resolved using the DNS protocol, or are top-level domains, or both. That is, at present there exist no special-use names which require special handling by stub resolvers and which are not at the top level of the naming hierarchy.

This does mean, however, that at present, RFC6762 requires the use of a special label, '.LOCAL', to indicate to stub resolvers that mDNS[RFC6762] be used to resolve names under that label.

#### 4.1.3. MoU Concerning the Technical Work of the IANA

There exists a Memorandum of Understanding[RFC2860] between the IETF and ICANN (Internet Corporation for Assigned Names and Numbers) which discusses how names and numbers will be managed through the IANA (Internet Assigned Numbers Authority). This document is important to the discussion of SUDNs because, while it delegates authority for managing the Domain Name System Namespace generally to ICANN, it reserves to the IETF the authority that RFC 6761 formalizes:

Note that (a) assignments of domain names for technical uses (such as domain names for inverse DNS lookup), (b) assignments of specialised address blocks (such as multicast or anycast blocks), and (c) experimental assignments are not considered to be policy issues, and shall remain subject to the provisions of this Section 4.

The above text is an addendum to the following:

Two particular assigned spaces present policy issues in addition to the technical considerations specified by the IETF: the

assignment of domain names, and the assignment of IP address blocks. These policy issues are outside the scope of this MOU.

In general, then, the assignment of names in the DNS root zone, and the management of the DNS namespace, is a function that is performed by ICANN. However, the MoU specifically exempts domain names assigned for technical use, and uses the example of 'IN-ADDR.ARPA' and 'IP6.ARPA' to illustrate. Both of these names are in the RFC 6761 registry.

The point here is not to say what the implications of this statement in the MoU are, but rather to call the reader's attention to the existence of this statement.

#### 4.2. Secondary documents relating to the SUTLIN question

In addition to these documents, there are several others with which participants in this discussion should be familiar.

##### 4.2.1. Multicast DNS

Multicast DNS [RFC6762] defines the Multicast DNS protocol, which uses the '.LOCAL' SUTLDN. Section 3 describes the semantics of "multicast DNS names." It is of considerable historical importance to note that the -00 version of this document, an individual submission, was published in July of 2001. This version contains substantially the same text in section 3, and was discussed in the DNSEXT working group at IETF 51 in August of 2001[IETF-PRO-51]. The first version of this document designated '.LOCAL.ARPA' as the special-use name. This idea was strongly opposed by DNSEXT working group participants, and as a result the author eventually switched to using '.LOCAL'.

The history of RFC 6762 is documented in substantial detail in Appendix H; some notable milestones include the initial proposal to replace Appletalk's NBP in July 1997, the chartering of the Zeroconf working group in September 1999, allocation of a multicast address for link-local name discovery in April of 2000. A companion requirements document, eventually published as [RFC6760] was first published in September of 2001.

The point of mentioning these dates is so that discussions involving the time when the '.LOCAL' domain was first deployed, and the context in which it was deployed, may be properly informed.

#### 4.2.2. The .onion Special-Use TLD

The .onion Special Use TLD [RFC7686] is important because it is the most recent IETF action on the topic of SUTLDNs; although it does not set new policy, the mere fact of its publication is worth thinking about.

Two important points to consider about this document are that:

- o The IETF gained consensus to publish it
- o The situation was somewhat forced, both by the fact of its unilateral allocation by the TOR project without following the RFC 6761 process, and because a deadline had been set by the CA/Browser Forum [SDO-CABF-INT] after which all .onion PKI certificates would expire and no new certificates would be issued, unless the .onion SUTLDN were to be recognized by the IETF.

#### 4.2.3. Locally Served DNS Zones

Locally Served DNS Zones [RFC6303] describes a particular use case for zones that exist by definition, and that are resolved using the DNS protocol, but that cannot have a global meaning, because the host IP addresses they reference are not unique. This applies to a variety of addresses, including Private IPv4 addresses [RFC1918], Unique Local IPv6 Unicast Addresses [RFC4193] (in which this practice was first described) and IANA-Reserved IPv4 Prefix for Shared Address Space [RFC6598].

This use case is distinct from the use-case for SUTLDNs like '.local' and '.onion' in that the names are resolved using the DNS protocol. But it shares the problem that such names cannot be assumed either to be unique or to be functional in all contexts for all Internet-connected hosts.

#### 4.2.4. Name Collision in the DNS

Name Collision in the DNS [SDO-ICANN-COLL] is a study commissioned by ICANN that attempts to characterize the potential risk to the Internet of adding global DNS delegations for names that were not previously delegated in the DNS, not reserved under any RFC, but also known to be (.local) or surmised to be (.corp) in significant use for special-use-type reasons (local scope DNS, or other resolution protocols altogether).

#### 4.2.5. Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis

Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis [RFC7050] is an example of a document that successfully used the RFC 6761 process to designate '.ipv4only.arpa' as a special-use name; in this case the process worked smoothly and without controversy.

#### 4.2.6. Additional Reserved Top Level Domains

Additional Reserved Top Level Domains [I-D.chapin-additional-reserved-tlds] is an example of a document that attempted to reserve several TLDs identified by ICANN as particularly at risk for collision as special-use domain names with no documented use. This attempt failed.

Although this document failed to gain consensus to publish, the need it was intended to fill still exists. Unfortunately, although a fair amount is known about the use of these names, no document exists that documents how they are used, and why it would be a problem to allocate them. Additionally, to the extent that the uses being made of these names are valid, no document exists indicating when it might make sense to use them, and when it would not make sense to use them (the simplest version of this document would of course say "never use them." If that were the IETF consensus, that would be a good reason not to bother to publish the document.

#### 4.3. Summary

The assignment of Internet Names is not under the sole control of any one organization. ICANN has authority in many cases, and could be considered in some sense the default. IETF has authority in other cases, but only with respect to protocol development. And neither of these authorities can in any practical sense exclude the practice of ad-hoc allocation of names, which can be done by any entity that has control over one or more name servers or resolvers, in the context of any hosts and services that that entity operates.

#### 5. History

Newcomers to the problem of resolving domain names may be under the mistaken impression that the DNS sprang, as in the Greek legend of Athena, directly from Paul Mockapetris' forehead. This is not the case. At the time of the writing of the IAB technical document, memories would have been fresh of the evolutionary process that led to the DNS' dominance as a protocol for domain name resolution.

In fact, in the early days of the Internet, hostnames were resolved using a text file, HOSTS.TXT, which was maintained by a central

authority, the Network Information Center, and distributed to all hosts on the Internet using the File Transfer Protocol (FTP) [RFC0959]. The inefficiency of this process is cited as a reason for the development of the DNS [RFC0882] [RFC0883] in 1983.

However, the transition from HOSTS.TXT to the DNS was not smooth. For example, Sun Microsystems's Network Information System [CORP-SUN-NIS], at the time known as Yellow Pages, was an active competitor to the DNS, although it failed to provide a complete solution to the global naming problem.

Another example was NetBIOS Name Service, also known as WINS [RFC1002]. This protocol was used mostly by Microsoft Windows machines, but also by open source BSD and Linux operating systems to do name resolution using Microsoft's own name resolution protocol.

Most modern operating systems can still use the '/etc/hosts' file for name resolution. Many still have a name service switch that can be configured on the host to resolve some domains using NIS or WINS. Most have the capability to resolve names using mDNS by recognizing the special meaning of the '.local' SUTLDN.

The Sun Microsystems model of having private domains within a corporate site, while supporting the global domain name system for off-site persisted even after the NIS protocol fell into disuse. Microsoft used to recommend that site administrators allocate a "private" top-level domain for internal use, and this practice was very much a part of the zeitgeist at the time. This attitude is the root of the widespread practice of simply picking an unused top-level domain and using it for experimental purposes, which persists even at the time of writing of this memo.

This history is being presented because discussions about special-use names in the IETF often come down to the question of why users of new name resolution protocols choose to use Domain names, rather than using some other naming concept that doesn't overlap with the namespace that, in modern times is, by default, resolved using the DNS.

The answer is that as a consequence of this long history of resolving Domain Names, Domain Names appear in a large variety of contexts in user interfaces applications programming interfaces. Any name that appears in such a context is a Domain Name. What this means is that Domain Names have a highly privileged status in deployed software. Proposals to change that will be almost impossible to get adopted in a useful or consistent way. And since in most operating systems, mechanisms already exist for implementing special handling for some Domain Names, from a practical perspective the only way to achieve

the goals of any new name resolution protocol is through the use of special-use Domain Names.

## 6. Contributors

This document came about as a result of conversations that occurred in the lobby, the weekend before IETF 95. Stuart Cheshire, Mark Andrews, David Conrad, Paul Ebersman and Aaron Falk all made helpful and insightful observations or patiently answered questions. This should not be taken as an indication that any of these folks actually agree with what the document says, but their generosity with time and thought are appreciated in any case.

Ralph started out as an innocent bystander, but discussion with him was the key motivating factor in the writing of this document, and he agreed to co-author it without too much arm-twisting. Warren spent a lot of time working with me on this document after it was first published, and joined as an author in order to make sure that the work got finished; without him the -01 and -02 versions might not have happened.

And this document owes a great deal to Ed Lewis' excellent work on what a "domain name" is [I-D.lewis-domain-names].

## 7. Informative References

- [RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <<http://www.rfc-editor.org/info/rfc882>>.
- [RFC0883] Mockapetris, P., "Domain names: Implementation specification", RFC 883, DOI 10.17487/RFC0883, November 1983, <<http://www.rfc-editor.org/info/rfc883>>.
- [RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <<http://www.rfc-editor.org/info/rfc959>>.
- [RFC1002] NetBIOS Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, and End-to-End Services Task Force, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications", STD 19, RFC 1002, DOI 10.17487/RFC1002, March 1987, <<http://www.rfc-editor.org/info/rfc1002>>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<http://www.rfc-editor.org/info/rfc1918>>.
- [RFC2826] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root", RFC 2826, DOI 10.17487/RFC2826, May 2000, <<http://www.rfc-editor.org/info/rfc2826>>.
- [RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <<http://www.rfc-editor.org/info/rfc2860>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<http://www.rfc-editor.org/info/rfc4193>>.
- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <<http://www.rfc-editor.org/info/rfc6303>>.
- [RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, DOI 10.17487/RFC6598, April 2012, <<http://www.rfc-editor.org/info/rfc6598>>.
- [RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)", RFC 6760, DOI 10.17487/RFC6760, February 2013, <<http://www.rfc-editor.org/info/rfc6760>>.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<http://www.rfc-editor.org/info/rfc6761>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.

- [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, <<http://www.rfc-editor.org/info/rfc7050>>.
- [RFC7686] Appelbaum, J. and A. Muffett, "The ".onion" Special-Use Domain Name", RFC 7686, DOI 10.17487/RFC7686, October 2015, <<http://www.rfc-editor.org/info/rfc7686>>.
- [I-D.chapin-additional-reserved-tlds]  
Chapin, L. and M. McFadden, "Additional Reserved Top Level Domains", draft-chapin-additional-reserved-tlds-02 (work in progress), March 2015.
- [I-D.lewis-domain-names]  
Lewis, E., "Domain Names", draft-lewis-domain-names-03 (work in progress), June 2016.
- [SDO-CABF-INT]  
CA/Browser Forum, "Guidance on the Deprecation of Internal Server Names and Reserved IP Addresses", June 2012, <<https://www.digicert.com/internal-names.htm>>.
- [SDO-ICANN-COLL]  
Interisle Consulting Group, LLC, "Name Collisions in the DNS", August 2013, <<https://www.icann.org/en/system/files/files/name-collision-02aug13-en.pdf>>.
- [SDO-IANA-SUDR]  
Internet Assigned Numbers Authority, "Special-Use Domain Names registry", October 2015, <<http://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml>>.
- [CORP-SUN-NIS]  
Sun Microsystems, "Large System and Network Administration", March 1990.
- [IETF-PRO-51]  
Internet Engineering Task Force, "Proceedings of the 51st IETF", August 2001, <<http://www.ietf.org/proceedings/51/51-45.htm#TopOfPage>>.



## Appendix A. Change Log.

## -03 to -04:

- o Replaced 'Internet Names' with 'Domain Names' - suggestion by John Levine.

## -02 to -03:

- o Readability fixes, small grammar updates.

## -01 to -02:

- o Cleaned up the abstract.
- o Fixed the case of Internet
- o Reference to Ed Lewis' "domain names"
- o Fleshed out the problems, primarily the coordination, collisions ones.

## -00 to -01:

- o Large refactoring, basically a rewrite. Incorporated comments, removed a bunch of unneeded text, etc.

## Authors' Addresses

Ted Lemon  
Nominum, Inc.  
800 Bridge Parkway  
Redwood City, California 94065  
United States of America

Phone: +1 650 381 6000  
Email: ted.lemon@nominum.com

Ralph Droms  
Cisco, Inc.

Email: rdroms.ietf@gmail.com

Warren Kumari  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: warren@kumari.net

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 22, 2016

M. Vavrusa  
O. Gudmundsson  
CloudFlare Inc.  
March 21, 2016

Providing AAAA records for free with QTYPE=A  
draft-vavrusa-dnsop-aaaa-for-free-00

Abstract

This document enables DNS servers to include AAAA addresses in the answer section for DNS queries with QTYPE=A in order to reduce the number of resolver round-trips during address lookups, and also provides guidance for recursive DNS servers in accepting such records.

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 <http://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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements . . . . .	2
1.2. Terminology . . . . .	2
2. Motivation . . . . .	2
3. Behaviour of authoritative DNS servers . . . . .	3
4. Behaviour of recursive DNS servers . . . . .	3
5. Examples . . . . .	3
5.1. Response to QTYPE=A with additional AAAA . . . . .	3
5.2. Response to QTYPE=A with missing AAAA . . . . .	4
5.3. Response to QTYPE=A with missing A, but added AAAA . . . . .	4
6. Security Considerations . . . . .	5
7. Performance Considerations . . . . .	5
8. Acknowledgements . . . . .	5
9. References . . . . .	5
9.1. Normative References . . . . .	5
9.2. Informative References . . . . .	6
Authors' Addresses . . . . .	6

## 1. Introduction

Over the years, there have been a number of attempts to extend DNS to allow multiple questions in a DNS query. While it is possible to place more than one query in the question section there is only one RCODE for the combined answer and there are no semantics on how to set the RCODE if there are multiple questions that have different results.

## 1.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 1.2. Terminology

The reader is assumed to be familiar with the basic DNS concepts described in [RFC1034], [RFC1035], [RFC2181] and [RFC6891]. Further DNS terminology is clarified in [RFC7719].

## 2. Motivation

The DNS specification [RFC1034] [RFC1035] doesn't provide any guidance on how to handle records in answer sections with matching QNAME, but mismatching QTYPE with the exception of CNAME and DNAME records.

The most frequently looked up types are address records, A for IPv4 addresses, and AAAA for IPv6 addresses. Stub resolvers attempt to optimize latency by issuing both queries in parallel, but both recursive and authoritative DNS servers then treat both queries independently, thus in the worst case, loss of one answer triggers requery for both. Furthermore, when client is behind an anycast resolver cluster, the two queries may go to different resolver instances. Resolvers also use queries for both record types internally when determining referral chain topology, and the loss of one answer leads either to an added round-trip if requerying, or suboptimal address selection if the recursor continues without it.

### 3. Behaviour of authoritative DNS servers

The authoritative server MAY treat a query with QTYPE=A effectively as a request for any IP address type, regardless of the address protocol with all the requirements due to [RFC1035] , [RFC4035]. Namely, the authoritative server MUST add DNSSEC signatures for any such records if the zone is signed.

However, if there is a direct answer to the original question, but no records for other address protocols, the authoritative DNS server SHOULD NOT prove their non-existence. In this respect, they are treated as additional data.

### 4. Behaviour of recursive DNS servers

The recursive resolver MAY accept RRs with TYPE=AAAA and owner equal to SNAME, therefore a direct answer to the query or matching the the final target of the CNAME chain. They MUST be treated as authoritative data as in [RFC2181], 5.4.1.

Notably, a recursive resolver MUST verify DNSSEC signatures on any such records and it MUST reject any such records if the validation fails, and the zone is not provably secure. In other words, they are subject to the same requirements as a direct answer.

A resolver SHOULD accept other IP address records even if there are no records matching the original QTYPE, given that authoritative DNS server proves non-existence of the direct answer.

### 5. Examples

#### 5.1. Response to QTYPE=A with additional AAAA

Header	QR AA RCODE=NOERROR
Question	ns1.example. IN A
Answer	ns1.example. IN A 192.0.2.1     ns1.example. IN AAAA 2001:db8::1
Authority	<empty>
Additional	<empty>

Figure 1

## 5.2. Response to QTYPE=A with missing AAAA

Header	QR AA RCODE=NOERROR
Question	ns2.example. IN A
Answer	ns2.example. IN A 192.0.2.1
Authority	<empty>
Additional	<empty>

Figure 2

## 5.3. Response to QTYPE=A with missing A, but added AAAA

Header	QR AA RCODE=NOERROR
Question	ns3.example. IN A
Answer	ns3.example. IN AAAA 2001:db8::2
Authority	example. IN SOA a.example. x.w.example.     1081539377 3600 300 3600000 3600
Additional	<empty>

Figure 3

## 6. Security Considerations

In cases where a caching resolver either doesn't validate or the authoritative answer is insecure, a successful spoofing attack may poison both address types in one successful attempt. However, the chance of successful spoofing attack is not affected.

## 7. Performance Considerations

Some resolvers might reject the answer due to "extra" records in the answer section, but more likely the resolver will discard the AAAA records, thus we are no different than today.

## 8. Acknowledgements

Dani Grant, Vicky Shrestha and Filippo Valsorda provided valuable comments on the draft.

## 9. References

### 9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<http://www.rfc-editor.org/info/rfc2181>>.
- [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, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<http://www.rfc-editor.org/info/rfc6891>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<http://www.rfc-editor.org/info/rfc7719>>.

## 9.2. Informative References

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

## Authors' Addresses

Marek Vavrusa  
CloudFlare Inc.  
101 Townsend St.  
San Francisco 94107  
USA

Email: [mvavrusa@cloudflare.com](mailto:mvavrusa@cloudflare.com)

Olafur Gudmundsson  
CloudFlare Inc.  
101 Townsend St.  
San Francisco 94107  
USA

Email: [olafur@cloudflare.com](mailto:olafur@cloudflare.com)



DNSOP  
Internet-Draft  
Intended status: Best Current Practice  
Expires: April 29, 2017

P. Wallstrom  
J. Schlyter  
Kirei AB  
October 26, 2016

DNS Delegation Requirements  
draft-wallstrom-dnsop-dns-delegation-requirements-03

Abstract

This document outlines a set of requirements on a well-behaved DNS delegation of a domain name. A large number of tools have been developed to test DNS delegations, but each tool uses a different set of requirements for what is a correct setup for a delegated domain name. However, there are few requirements on how to set up DNS in order to just make the delegation work. In order to have a well-behaved delegation that is robust to failures and also makes DNS resolvers behave consistently, there are a large number of things to consider.

Based on this document, it should be possible to set up a fully functional DNS delegation for a domain name, but also to create a set of test specifications for how to test a DNS delegation.

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 <http://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 April 29, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. DNS Terminology . . . . .	4
1.2. Reserved Words . . . . .	4
2. Basic requirements . . . . .	5
2.1. The domain name MUST be valid . . . . .	5
2.2. The domain MUST have a parent domain . . . . .	5
2.3. The domain MUST have at least one working name server . .	5
3. Address requirements . . . . .	5
3.1. Name server address MUST be globally routable . . . . .	5
3.2. The IP address of a name server MUST be delegated by IANA	6
4. Connectivity requirements . . . . .	6
4.1. All name servers MUST have UDP connectivity over port 53	7
4.2. All name servers MUST have TCP connectivity over port 53	7
5. Name server requirements . . . . .	7
5.1. Authoritative name servers SHOULD NOT be recursive . . .	7
5.2. Name servers SHOULD support EDNS0 . . . . .	7
5.3. Name servers MUST process QNAME case insensitive . . . .	8
6. Consistency requirements . . . . .	8
6.1. All name servers SHOULD respond with the same SOA serial number . . . . .	8
6.2. All name servers SHOULD respond with the same SOA RNAME .	9
6.3. All name servers SHOULD respond with the same SOA parameters . . . . .	9
6.4. All name servers MUST respond with the same NS RR Set . .	9
7. Delegation requirements . . . . .	9
7.1. The delegation SHOULD contain at least two name servers .	9
7.2. The NS RR set in the parent SHOULD be a subset of the NS RR set in the child . . . . .	10
7.3. The name servers SHOULD have network path diversity . . .	10
7.4. The name servers MUST have distinct IP addresses . . . .	10
7.5. The referral SHOULD fit into a non-truncated 512 byte UDP packet . . . . .	10
7.6. All name servers MUST be authoritative for the domain name . . . . .	11
7.7. The delegation name MUST exactly match the apex of the child zone . . . . .	11

7.8.	Glue records in delegation SHOULD exactly match records in child zone . . . . .	11
7.9.	SOA MNAME SHOULD be authoritative for the zone . . . . .	11
8.	DNSSEC requirements . . . . .	12
8.1.	The DS Digest Type MUST be assigned by IANA . . . . .	12
8.2.	The DNSKEY algorithm MUST be assigned by IANA . . . . .	12
8.3.	The chain of trust for the delegation MUST be valid . . . . .	12
8.4.	One DS MUST match a least one DNSKEY in the child zone . . . . .	12
8.5.	The number of NSEC3 iterations must not be higher than what is allowed . . . . .	13
8.6.	RRSIG validity period SHOULD NOT be too short nor too long . . . . .	13
8.7.	The name server MUST include RRSIG in all responses to DNSSEC queries . . . . .	13
8.8.	The name servers MUST include valid NSEC/NSEC3 record in NXDOMAIN responses . . . . .	13
9.	Syntax requirements . . . . .	14
9.1.	Illegal characters MUST NOT be in the domain name . . . . .	14
9.2.	Hyphens SHOULD NOT be in position 3 and 4 of the domain name . . . . .	14
9.3.	The NS names MUST be valid hostnames . . . . .	14
9.4.	The NS names MUST NOT be an alias . . . . .	14
9.5.	The SOA RNAME MUST not contain the '@' character . . . . .	14
9.6.	The SOA RNAME MUST be a legal hostname . . . . .	15
9.7.	The SOA MNAME MUST be a legal hostname . . . . .	15
9.8.	The MX record in apex MUST point to a valid hostname . . . . .	15
10.	Security Considerations . . . . .	15
11.	IANA Considerations . . . . .	15
12.	Acknowledgements . . . . .	15
13.	References . . . . .	16
13.1.	Normative References . . . . .	16
13.2.	Informative References . . . . .	17
	Authors' Addresses . . . . .	20

## 1. Introduction

This document outlines a set of requirements on a well-behaved DNS delegation of a domain name. Many domain name registries use a set of requirements on what they may consider a valid delegation. Such requirements can be used to implement tools that are used for pre- or post-delegation checks of the delegations in that registry.

To test the quality of the delegation there has been a number of different tools developed, each based on a different set of requirements. This document outlines a set of baseline requirements on a correct setup for a delegated domain name. This document is based on current RFCs and documents requirements that are protocol

specific, but also administrative policy requirements drawn from best practices and recommendations.

The DNS requirements are split into these different areas, to easier differentiate between what they are for:

- o Basic
- o Address
- o Connectivity
- o Name server
- o Consistency
- o Delegation
- o DNSSEC
- o Syntax

A secondary name server operator should follow the advice in the BCP document [RFC2182].

Nothing in this document precludes others testing servers for protocol compliance. DNS operators should test their servers to ensure that their vendors have shipped protocol compliant products. Name server vendors can use these tests as a part of this release processes. Registrants can use these tests to check their DNS operators servers.

### 1.1. DNS Terminology

This document attempts to fully follow the DNS terminology as defined in [RFC7719].

Many requirements in this document deal with the properties of a name server that is used as part of a delegation, therefore the wording mentioning the use - authoritative or recursive - of a name server as part of this is omitted.

### 1.2. Reserved Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. Basic requirements

The Basic requirements are fundamental to a working DNS delegation. Without these properties, the rest of the requirements are irrelevant.

### 2.1. The domain name MUST be valid

The domain name MUST follow the rules defined in Section 2.1 of [RFC1123] in order to be able to map the domain into a DNS packet. A domain name is normally valid if the name has been registered with a domain name registry.

Internationalized domain names, [RFC5891], are expected to be encoded using Punycode [RFC3492], thus following the rules outlined in Section 2.3.1 of [RFC1035]. Any validation of the domain name in U-label form is out of scope for this document.

### 2.2. The domain MUST have a parent domain

A DNS delegation MUST have a parent domain from which it is delegated. The concept of zone cuts was first described in [RFC1034] and later clarified in Section 6 of [RFC2182]. The only exception is the root zone, which do not have a parent zone.

### 2.3. The domain MUST have at least one working name server

A fully working DNS delegation has a parent zone delegating the zone to a set of child name servers. At least one name server MUST be able to answer DNS queries in order to be able to authoritatively serve data for the child zone.

## 3. Address requirements

A delegation in the public Internet DNS hierarchy will use the globally unique address space.

### 3.1. Name server address MUST be globally routable

In order for the domain and its resources to be accessible from the Internet, authoritative name servers must have addresses in the routable public addressing space.

IANA is responsible for global coordination of the IP addressing system. Aside its address allocation activities, it maintains reserved address ranges for special uses. There are two IANA registries for Special-Purpose Addresses, the IANA IPv6 Special-

Purpose Address Registry and the IANA IPv4 Special-Purpose Address Registry.

[RFC6890] instructs IANA on how to structure the IPv4 and IPv6 Special-Purpose Address Registries. The registries [IANA-IPv4-Special] and [IANA-IPv6-Special] are maintained by IANA, and are also described in Section 2.2 and 2.3 of [RFC7249].

A name server **MUST NOT** be using any IP address within any of these registries that are marked with False in the Global column.

### 3.2. The IP address of a name server **MUST** be delegated by IANA

IP addresses not delegated by IANA **MUST NOT** be used by a name server. Thus, IP addresses within a prefix not delegated to a RIR by IANA **MUST** be rejected.

The IANA registry [IANA-IPv6-Unicast] **SHOULD** be used to determine the status of an IPv6 prefix. Only prefixes with the status **ALLOCATED** are allowed.

The IANA registry [IANA-IPv4-Registry] **SHOULD** be used to determine the status of an IPv4 prefix. Only prefixes with the status **ALLOCATED** and **LEGACY** are allowed. Note that IPv4 **LEGACY** is not allocated to a RIR.

Martians [RFC1208] is a humorous term applied to packets that turn up unexpectedly on the wrong network because of bogus routing entries. Bogons [RFC3871] are packets sourced from addresses that have not yet been allocated by IANA or a RIR, or not delegated to a RIR by IANA as described above. Martians and Bogons **SHOULD NOT** be used as an address used by a name server.

## 4. Connectivity requirements

The use of underlying protocols for DNS is described in Section 4.2 of [RFC1035].

The Internet supports name server access using TCP on server port 53 (decimal) as well as datagram access using UDP on UDP port 53 (decimal). Today DNS is used in conjunction with both IPv4 and IPv6,

Name servers configured for a zone in a delegation **MUST** be able to answer queries using the DNS protocol.

#### 4.1. All name servers MUST have UDP connectivity over port 53

DNS queries are sent using UDP on port 53, as described in Section 4.2.1 of [RFC1035]. A name server MUST respond to DNS queries over UDP for each IP address configured for that name server.

#### 4.2. All name servers MUST have TCP connectivity over port 53

In addition to UDP, DNS queries can also be sent using TCP on port 53, as described in Section 4.2.2 of [RFC1035]. A name server MUST respond to DNS queries over TCP for each IP address configured for that name server. This requirement has also been further clarified in [RFC7766], which makes TCP a REQUIRED part of a full DNS protocol implementation.

It should be noted that even though [RFC7766] requires TCP for a DNS protocol implementation, it does not make specific recommendations to operators of DNS servers. However, it also notes that failure to support TCP (or the blocking of DNS over TCP at the network layer) may result in resolution failure and/or application-level timeouts. The operational requirements on DNS Transport over TCP are further discussed [I-D.kristoff-dnsop-dns-tcp-requirements].

### 5. Name server requirements

#### 5.1. Authoritative name servers SHOULD NOT be recursive

To ensure consistency in DNS, an authoritative name server SHOULD NOT be configured to do recursive lookups. Also, open recursive resolvers are considered bad Internet practice due to their capability of assisting in large scale DDoS attacks. The introduction to [RFC5358] elaborates on mixing recursor and authoritative functionality. Section 2.5 of [RFC2870] have very specific requirement on disabling recursion functionality on root name servers.

#### 5.2. Name servers SHOULD support EDNS0

EDNS0 is a mechanism to announce capabilities of a DNS implementation, and is now basically required by any new functionality in DNS such as DNSSEC. Initially standardized in [RFC2671] and later updated by [RFC6891], EDNS0 is a mechanism to announce capabilities of a DNS implementation.

### 5.3. Name servers MUST process QNAME case insensitive

The DNS standards require that name servers treat names with case insensitivity. That is, the names `example.com` and `EXAMPLE.COM` should resolve to the same IP address. However, in the response, most name servers echo back the name as it appeared in the request, preserving the original case. This is specified in [RFC1034] and [RFC1035], and further clarified by [RFC4343].

Therefore, another way to add entropy to requests is to randomly vary the case of letters in domain names queried. This technique, also known as "0x20" because bit 0x20 is used to set the case of of US-ASCII letters, was first proposed in [I-D.vixie-dnsext-dns0x20], Use of Bit 0x20 in DNS Labels to Improve Transaction Identity. With this technique, the name server response must match not only the query name, but the case of every letter in the name string; for example, `wWw.eXaMpLe.CoM` or `WwW.ExamPLe.COM`. This may add little or no entropy to queries for the top-level and root domains, but it's effective for most hostnames.

## 6. Consistency requirements

For DNS resolver behaviour to be consistent for a domain, it is important that the authoritative data for the domain to be consistent. All authoritative name servers for a zone should serve the same data, although it should be noted that there exists cases where authoritative name servers are configured to reply with different answers depending on the client source address and/or query options such as EDNS0 client subnet option as specified in [RFC7871].

An indicator of inconsistency is that infrastructure records (e.g., SOA and NS) differs between the authoritative name servers.

Section 4.6 in [RFC4786] advises that data synchronisation in an anycast setup should be done in a manner so that anycast nodes operate in a consistent manner.

### 6.1. All name servers SHOULD respond with the same SOA serial number

An indication that not all authoritative name servers have a consistent and updated copy of the zone is that the serial numbers differ. When querying for the SOA RR all name servers SHOULD respond with the same SOA serial number.

Section 4.3.5 in [RFC1034] explains the typical function of the serial numbers in zone maintenance and transfers.



One should note that even though different SOA serial numbers are a strong indicator of an inconsistent setup, there are several scenarios where the serial number varies between name servers. One example is a zone with frequent updates to zone data, where propagation delay between the name servers may result in limited inconsistency.

#### 6.2. All name servers SHOULD respond with the same SOA RNAME

As per Section 3.3.13 of [RFC1035], the RNAME field in the SOA RDATA refers to the mailbox of the person responsible for the zone. An indication that not all authoritative name servers have a consistent and updated copy of the zone is that the RNAME differs. When querying for the SOA RR all name servers SHOULD respond with the same SOA RNAME.

#### 6.3. All name servers SHOULD respond with the same SOA parameters

The inconsistency of the SOA parameters REFRESH, RETRY, EXPIRE and MINIMUM, defined in Section 3.3.13 of [RFC1035], might lead to operational problems for the zone. These SOA parameters SHOULD be consistent for all authoritative name servers for the zone.

#### 6.4. All name servers MUST respond with the same NS RR Set

All authoritative name servers MUST serve the same NS record set in order to ensure consistency in the zone cut as described in Section 4.2.2 of [RFC1034]. Any inconsistency of NS records described in Section 3.3.11 of RFC 1035 might result in operational failures.

### 7. Delegation requirements

[RFC2182] is a BCP on how to select and operate secondary name servers, and summarize many operational issues with the delegation of a zone. For a delegation to work continuously if one component fails, there are operational considerations to ensure this.

Section 4.2.2 [RFC1034] also adds that the administrators of both the parent and child zone should ensure that NS and glue RRs on both sides of the zone cut are consistent.

#### 7.1. The delegation SHOULD contain at least two name servers

Section 4.1 [RFC1034] states that by administrative fiat we require every zone to be available on at least two name servers. Section 5 of [RFC2182] that answers the question on how many name servers are needed, the recommendation is that "three servers be provided for

most organisation level zones, with at least one which must be well removed from the others."

In order to avoid any operational problems, a delegation SHOULD contain at least two (2) authoritative name servers.

- 7.2. The NS RR set in the parent SHOULD be a subset of the NS RR set in the child

As per the name resolving algorithm described in [RFC1034] the NS RR in the child zone is authoritative for the zone, and any delegation hints in the parent are discarded in the resolving process. The NS RR set in the parent zone SHOULD be a subset of the NS RR set in the child zone.

- 7.3. The name servers SHOULD have network path diversity

[RFC2182], Section 3.1 states that distinct authoritative name servers for a child domain should be placed in different topological and geographical locations. The objective is to minimise the likelihood of a single failure disabling all of them. Further support for this is given in Section 5:

It is recommended that three servers be provided for most organisation level zones, with at least one which must be well removed from the others.

To avoid any single point of failure in networking, the name servers SHOULD exhibit network path diversity. Using current routing technology, this means that all name servers SHOULD NOT be placed within a single routing domain, or AS (autonomous system).

- 7.4. The name servers MUST have distinct IP addresses

A common workaround to a registry policy that requires at least two name servers is to create two (2) names with the same IP address.

To avoid any operational errors and workaround such as this, all name servers used for the zone MUST use distinct IP addresses.

- 7.5. The referral SHOULD fit into a non-truncated 512 byte UDP packet

The DNS still defaults to using UDP, although efforts into requiring or transitioning to use TCP have come a long way. The UDP packet limit is 512 bytes, and although the EDNS0 [RFC6891] extension mechanism to overcome this limit have been in use for a very long time, many middleboxes and proxies still interfere with DNS packets ([RFC5625]).

To avoid any such problems with the delegation, and to avoid any unexpected truncation of a referral response, the referral containing the delegation from the parent SHOULD fit within 512 bytes.

7.6. All name servers MUST be authoritative for the domain name

A name server that does not answer authoritatively for the zone is a clear sign of misconfiguration, and is a common cause for operational problems.

Section 6.1 of [RFC2181] mandates that the name servers MUST answer authoritatively for the zone.

7.7. The delegation name MUST exactly match the apex of the child zone

The configured zone on the child name servers MUST match the delegated name of the zone. When querying the child name servers for the zone, any authoritative data for another name MUST NOT be in the response.

[RFC2181] states that the SOA RR and the NS RR indicates the origin of the zone, and both are mandatory records in a zone. Both RRs MUST be present and match the name of the zone.

7.8. Glue records in delegation SHOULD exactly match records in child zone

In-bailiwick glue for name servers listed at the parent SHOULD match the in-bailiwick glue for the name servers in the child.

If the glue address mismatch between the parent zone and the child, this is a strong indication of configuration error.

7.9. SOA MNAME SHOULD be authoritative for the zone

The hostname of the MNAME field may or may not be listed among the delegated name servers, but SHOULD still be authoritative for the zone. MNAME may be used for other services, e.g., DNS NOTIFY [RFC1996] and DNS Dynamic Updates [RFC2136].

It should be noted that there are no formal requirement that the name server listed in the SOA MNAME is reachable from the public Internet. Because of this, it may be difficult to implement a reasonable test for this requirement.

## 8. DNSSEC requirements

If DNSSEC is used for the zone, either by indicating that the zone is signed with a DS record, or the use of a DNSKEY in the zone itself, a number of things are required for a fully functional delegation.

The Domain Name System Security Extensions (DNSSEC) add data origin authentication and data integrity to the Domain Name System, and was first introduced with the RFCs [RFC4033], [RFC4034] and [RFC4035]. There are also a number of additions to DNSSEC such as NSEC3 described in [RFC5155], and a number of algorithms to the cryptographic functions.

### 8.1. The DS Digest Type MUST be assigned by IANA

The The Digest Type Field is defined as part of the DS RDATA Wire Format of Section 5.1.3 in [RFC4034]. The appendix A.2 defines the initial set of digest algorithm types with possible future algorithms. The IANA registry for DS Digest Types [IANA-DNSSEC-DS] was defined by [RFC3658].

Any DS Digest Type used for a zone MUST be assigned by IANA.

### 8.2. The DNSKEY algorithm MUST be assigned by IANA

The DNSKEY RR is defined in Section 2 of [RFC4034] as part of the DNSKEY RDATA Wire Format. The appendix A.1 defines the initial list of DNSKEY Algorithm Types. The IANA Registry for DNSKEY Algorithm Types [IANA-DNSSEC-DNSKEY] was created with [RFC3755].

Any DNSKEY algorithm number used for in a zone MUST be assigned by IANA.

### 8.3. The chain of trust for the delegation MUST be valid

A valid authentication chain from the parent DS, as described in Section 3.1 of [RFC4033], MUST exist for the SOA, DNSKEY and NS records of the child zone if a DS record is published in the parent zone.

### 8.4. One DS MUST match a least one DNSKEY in the child zone

DNS delegations from a parent to a child are secured with DNSSEC by publishing one or several Delegation Signer (DS) resource records in the parent zone, along with the NS records for the delegation.

As stated in Section 2.4 of [RFC4035], a DS RR SHOULD point to a DNSKEY RR that is present in the child's apex DNSKEY RRset. If there

is a DS RR published at the parent, there MUST be at least one DNSKEY RR in the child zone that matches at least one DS RR for every signature algorithm, otherwise the authentication of the referral will fail, as described in Section 5.2 of [RFC4035].

For each unique algorithm from the DS RRs present, there MUST be a matching DNSKEY using that algorithm in use in the child.

- 8.5. The number of NSEC3 iterations must not be higher than what is allowed

Section 10.3 of [RFC5155] specifies the max number of NSEC3 iterations allowed for different key sizes. This requirement is enforced by several resolver implementations.

The number of NSEC3 iterations MUST NOT be higher than what is allowed by Section 10.3 of [RFC5155]. It should be noted that the values in the table MUST be used independent of the key algorithm.

- 8.6. RRSIG validity period SHOULD NOT be too short nor too long

[RFC6781] describes operational considerations on the choice of validity periods for RRSIGs. Having too short validity periods might cause operational failure in case of unexpected events, but is good for protecting against replay attacks. Having too long validity periods may be good for operational security, but opens up for replay attacks.

The RRSIG validity periods in the zone SHOULD NOT be too short nor too long.

- 8.7. The name server MUST include RRSIG in all responses to DNSSEC queries

If the zone is signed, the name servers MUST be able to include RRSIG RRs as additional data in any response when the query has the DO bit set, as described in Section 3.1.1 of [RFC4035].

- 8.8. The name servers MUST include valid NSEC/NSEC3 record in NXDOMAIN responses

If the zone is signed, the name servers MUST be able to include NSEC/NSEC3 RRs as additional data in any response when the query has the DO bit set, as described in Section 3.1.1 of [RFC4035].

## 9. Syntax requirements

All domain- and host names in DNS MUST follow the rules outlined in Section 2.3.1 of [RFC1035]. The Name Syntax and LDH Label have been further clarified in Section 11 in [RFC2181] and Section 2.3.1 in [RFC5890]. From this follow the requirements below.

### 9.1. Illegal characters MUST NOT be in the domain name

There MUST NOT be any illegal characters used in the domain name. The domain name MUST follow the rules defined in Section 2.3.1 of [RFC1035], Section 2.1 of [RFC1123], Section 11 of [RFC2182] and Section 2 of [RFC3696].

### 9.2. Hyphens SHOULD NOT be in position 3 and 4 of the domain name

The effort of internationalization of domain names and the development of IDNA brought us the extension mechanism of using the string 'xn--' to have a special meaning. To allow future extensions to DNS there SHOULD be no instances of labels in the DNS that start with two characters, followed by two hyphens, where the two characters are not "xn". This has been described in Section 5 of [RFC3696].

### 9.3. The NS names MUST be valid hostnames

The Name Server name MUST be a valid hostname according to the rules defined in Section 2.3.1 of [RFC1035], in Section 2.1 in [RFC1123], Section 11 in [RFC2181] and Section 2 and 5 in [RFC3696].

### 9.4. The NS names MUST NOT be an alias

As specified in Section 10.3 of [RFC2181], the Name Server name MUST NOT be an alias (CNAME).

### 9.5. The SOA RNAME MUST not contain the '@' character

The SOA RNAME field is a mailbox address defined in Section 3.3 of [RFC1034] and Section 2.2 of [RFC1912]. The RNAME field MUST follow the rules of an e-mail address defined in Section 3.4.1 of [RFC2822], and the '@' character MUST be changed so that the whole e-mail address is converted into a single domain name as described in Section 3.3 of [RFC1034] and Section 2.1 of [RFC1123].

#### 9.6. The SOA RNAME MUST be a legal hostname

The SOA RNAME field is a mailbox address. The SOA RNAME field is defined in Section 3.3 of [RFC1034] and Section 2.2 of [RFC1912]. As a field containing a domain name, the content of the RNAME field MUST follow the rules outlined in Section 2.3.1 of [RFC1035] and Section 2.1 of [RFC1123].

#### 9.7. The SOA MNAME MUST be a legal hostname

The SOA MNAME field is a hostname. The SOA MNAME field is defined in Section 3.3.13 of [RFC1035]. As a field containing a domain name, the content of the RNAME field MUST follow the rules outlined in Section 2.3.1 of [RFC1035].

Furthermore, Section 7.3 in [RFC2181] makes it clear that the SOA MNAME field SHOULD NOT be the name of the zone itself.

#### 9.8. The MX record in apex MUST point to a valid hostname

The requirement on the existence of an MX RR in the apex of the child zone may vary by policy from different parent zones. However, it is strongly recommended in Section 7 of [RFC2142] that all domains should have a mailbox named `hostmaster@domain`. SMTP can make a delivery without the MX, using the A or AAAA record as specified in Section 5.1 of [RFC5321].

If an MX RR exists in the apex of the child zone, the hostname that the MX RR points to MUST follow the rules outlined in Section 2.3.1 of [RFC1035] and Section 2.1 of [RFC1123].

#### 10. Security Considerations

This document has no security considerations (yet).

#### 11. IANA Considerations

This document has no IANA actions.

#### 12. Acknowledgements

The requirements documented in this document were developed within the CENTR Test Requirements Task Force (TRTF). Most of the original requirements and text come from the Zonemaster project.

### 13. References

#### 13.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <<http://www.rfc-editor.org/info/rfc1123>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<http://www.rfc-editor.org/info/rfc2181>>.
- [RFC2182] Elz, R., Bush, R., Bradner, S., and M. Patton, "Selection and Operation of Secondary DNS Servers", BCP 16, RFC 2182, DOI 10.17487/RFC2182, July 1997, <<http://www.rfc-editor.org/info/rfc2182>>.
- [RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <<http://www.rfc-editor.org/info/rfc2822>>.
- [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, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<http://www.rfc-editor.org/info/rfc4343>>.
- [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <<http://www.rfc-editor.org/info/rfc7719>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<http://www.rfc-editor.org/info/rfc7766>>.



## 13.2. Informative References

- [I-D.kristoff-dnsop-dns-tcp-requirements]  
Kristoff, J., "DNS Transport over TCP - Operational Requirements", draft-kristoff-dnsop-dns-tcp-requirements-01 (work in progress), August 2016.
- [I-D.vixie-dnsext-dns0x20]  
Vixie, P. and D. Dagon, "Use of Bit 0x20 in DNS Labels to Improve Transaction Identity", draft-vixie-dnsext-dns0x20-00 (work in progress), March 2008.
- [IANA-DNSSEC-DNSKEY]  
IANA, "Domain Name System Security (DNSSEC) Algorithm Numbers", November 2003,  
<<https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>>.
- [IANA-DNSSEC-DS]  
IANA, "Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms", Oktober 2003,  
<<https://www.iana.org/assignments/ds-rr-types/ds-rr-types.xhtml>>.
- [IANA-IPv4-Registry]  
IANA, "IANA IPv4 Address Space Registry", August 2015,  
<<https://www.iana.org/assignments/ipv4-address-space>>.
- [IANA-IPv4-Special]  
IANA, "IANA IPv4 Special-Purpose Address Registry", January 2006, <<https://www.iana.org/assignments/iana-ipv4-special-registry>>.
- [IANA-IPv6-Special]  
IANA, "IANA IPv6 Special-Purpose Address Registry", January 2006, <<https://www.iana.org/assignments/iana-ipv6-special-registry>>.
- [IANA-IPv6-Unicast]  
IANA, "IPv6 Global Unicast Address Assignments", October 2015, <<http://www.iana.org/assignments/ipv6-unicast-address-assignments>>.
- [RFC1208] Jacobsen, O. and D. Lynch, "A Glossary of Networking Terms", RFC 1208, DOI 10.17487/RFC1208, March 1991, <<http://www.rfc-editor.org/info/rfc1208>>.

- [RFC1912] Barr, D., "Common DNS Operational and Configuration Errors", RFC 1912, DOI 10.17487/RFC1912, February 1996, <<http://www.rfc-editor.org/info/rfc1912>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2142] Crocker, D., "Mailbox Names for Common Services, Roles and Functions", RFC 2142, DOI 10.17487/RFC2142, May 1997, <<http://www.rfc-editor.org/info/rfc2142>>.
- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, DOI 10.17487/RFC2671, August 1999, <<http://www.rfc-editor.org/info/rfc2671>>.
- [RFC2870] Bush, R., Karrenberg, D., Koster, M., and R. Plzak, "Root Name Server Operational Requirements", RFC 2870, DOI 10.17487/RFC2870, June 2000, <<http://www.rfc-editor.org/info/rfc2870>>.
- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <<http://www.rfc-editor.org/info/rfc3492>>.
- [RFC3658] Gudmundsson, O., "Delegation Signer (DS) Resource Record (RR)", RFC 3658, DOI 10.17487/RFC3658, December 2003, <<http://www.rfc-editor.org/info/rfc3658>>.
- [RFC3696] Klensin, J., "Application Techniques for Checking and Transformation of Names", RFC 3696, DOI 10.17487/RFC3696, February 2004, <<http://www.rfc-editor.org/info/rfc3696>>.
- [RFC3755] Weiler, S., "Legacy Resolver Compatibility for Delegation Signer (DS)", RFC 3755, DOI 10.17487/RFC3755, May 2004, <<http://www.rfc-editor.org/info/rfc3755>>.
- [RFC3871] Jones, G., Ed., "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure", RFC 3871, DOI 10.17487/RFC3871, September 2004, <<http://www.rfc-editor.org/info/rfc3871>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc4034>>.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <<http://www.rfc-editor.org/info/rfc4786>>.
- [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, <<http://www.rfc-editor.org/info/rfc5155>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<http://www.rfc-editor.org/info/rfc5321>>.
- [RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <<http://www.rfc-editor.org/info/rfc5358>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<http://www.rfc-editor.org/info/rfc5625>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<http://www.rfc-editor.org/info/rfc5890>>.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<http://www.rfc-editor.org/info/rfc5891>>.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<http://www.rfc-editor.org/info/rfc6781>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<http://www.rfc-editor.org/info/rfc6890>>.

- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<http://www.rfc-editor.org/info/rfc6891>>.
- [RFC7249] Housley, R., "Internet Numbers Registries", RFC 7249, DOI 10.17487/RFC7249, May 2014, <<http://www.rfc-editor.org/info/rfc7249>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<http://www.rfc-editor.org/info/rfc7871>>.

## Authors' Addresses

Patrik Wallstrom

Email: [pawal@blipp.com](mailto:pawal@blipp.com)

Jakob Schlyter  
Kirei AB

Email: [jakob@kirei.se](mailto:jakob@kirei.se)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 27, 2016

W. Kumari  
Google  
G. Huston  
APNIC  
February 24, 2016

Believing NSEC records in the DNS root.  
draft-wkumari-dnsop-cheese-shop-01

Abstract

This document describes a method to generate negative answers from NSEC records for the special case of the DNS root. This improves performance; the resolver can answer immediately, and does not need to query the root. It also cuts down on the so-called "junk" queries.

[ Ed note: Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication.]

[ This document is being collaborated on in Github at:  
<https://github.com/wkumari/draft-wkumari-dnsop-cheese-shop>. The most recent version of the document, open issues, etc should all be available here. The authors (gratefully) accept pull requests ]

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://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 August 27, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must 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.

## Table of Contents

1. Background . . . . .	2
2. Believing NSEC records. . . . .	3
2.1. Requirements notation . . . . .	3
3. Generating negatives responses from NSEC . . . . .	3
4. IANA Considerations . . . . .	4
5. Security Considerations . . . . .	4
6. Acknowledgements . . . . .	4
7. References . . . . .	4
7.1. Normative References . . . . .	5
7.2. Informative References . . . . .	5
Appendix A. Changes / Author Notes. . . . .	5
Authors' Addresses . . . . .	5

## 1. Background

[ This section may be removed before publication... but I'd prefer not, it provides useful context ]

If a DNS resolver queries a root zone authoritative name server with the EDNS0 DNSSEC OK option set, for a name that does not exist in the root zone, it gets back an NXDOMAIN response and an NSEC record, which "proves" that the name does not exist. NSEC proves this by providing names (and signatures) for the names which do exist on either side of the queried name. For example, if a nameserver queries for .belkin, it will get back an NXDOMAIN, and an NSEC record showing that nothing exists between (currently) .beer and .bentley [Ed note: There *\*probably\** should be something between a beer and a bentley. :-P ]. This means that, if the nameserver subsequently (during the TTL of the NSEC record) gets a query for .beeswax (alphabetically between beer and bentley) it need not attempt to resolve this - it has already been given proof that the name does not exist.

The title of this draft comes from a famous Monty Python skit - "The Cheese Shop". There are some useful parallels between this problem

and the skit - watching the skit is encouraged to understand the problem - <https://www.youtube.com/watch?v=cWDdd5KKhts>

## 2. Believing NSEC records.

This is simply a refinement of [I-D.fujiwara-dnsop-nsec-aggressiveuse], for a limited use case (the root). Full credit to the authors of the aforementioned draft, and this draft does not replace that draft, nor remove the need for the broader consideration of the use of NSEC records as described in [I-D.fujiwara-dnsop-nsec-aggressiveuse].

The scope of this document is limited to the special case of recursive DNSSEC validating resolvers querying the root zone. This is because the root zone has some well known properties which make it a special case - we know it is DNSSEC signed, and uses NSEC, the majority of the queries are "junk" queries, the rate of change is relatively slow, and there are no odd corner cases such as wildcards. See Section 3 for more discussion.

If the (DNSSEC validated) answer to a query to a root server is an NXDOMAIN then the resolver SHOULD cache the NSEC record provided in the response. The resolver SHOULD NOT send further queries for names within the range of the NSEC record for the lifetime of the cached NSEC TTL. Instead, the resolver SHOULD answer these queries directly with NXDOMAIN (and NSEC records if so signalled by EDNS). They SHOULD set the AA bit and AD bits.

### 2.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 3. Generating negatives responses from NSEC

[ This section is mainly for discussion, and is more informal. It should be deleted before publication. ]

Section 4.5 of [RFC4035] says:

"In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace."

and "The reason for these recommendations is that, between the initial query and the expiration of the data from the cache, the authoritative data might have been changed (for example, via dynamic update)."

So, if a resolver generates negative answers from an NSEC record, it will not send any queries for names within that NSEC range (for the TTL). If a new name is added to the zone during this interval the resolver will not know this.

For the limited use case of this document (the DNS root) we believe that this is an acceptable trade off - the (current) TTL of the "negative cache" (in the SOA) is the same as the NSEC TTL (1 day). This means that, for a new TLD to begin resolving everywhere will require a minimum of a day - and this is true whether or not this is implemented (if someone had queried for the exact name, there would be a negatively cached answer, this simply expands the range of negative caches).

#### 4. IANA Considerations

This document contains no IANA considerations.

[ We MAY want to add something about setting the NSEC TTL appropriately?! ]

#### 5. Security Considerations

The impact of resolver caching is that the resolver will not re-query an name server for a cached response until the TTL of the cached response expires. This may lead to cases where the resolver responds with outdated information for a period of time for subsequent queries for the name name.

This draft extends the scope of this vulnerability to include queries for all names that fall within the NSEC-defined range.

#### 6. Acknowledgements

The authors wish to thank some folk, including Stephane Bortzmeyer, Bob Harold, Paul Vixie.

#### 7. References



## 7.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.

## 7.2. Informative References

- [I-D.fujiwara-dnsop-nsec-aggressiveuse] Fujiwara, K. and A. Kato, "Aggressive use of NSEC/NSEC3", draft-fujiwara-dnsop-nsec-aggressiveuse-02 (work in progress), October 2015.
- [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, <<http://www.rfc-editor.org/info/rfc4035>>.

## Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication ]

From -00 to -01.

- o Fairly significant rewrite - no substantive changes, only additional information, explanation and readability.

## Authors' Addresses

Warren Kumari  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: [warren@kumari.net](mailto:warren@kumari.net)

Geoff Huston  
APNIC  
6 Cordelia St  
South Brisbane QLD 4001  
AUS

Email: [gih@apnic.net](mailto:gih@apnic.net)