DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 8, 2019

R. Bellis
ISC

November 04, 2018

A DNS Resource Record for HTTP
draft-bellis-dnsop-http-record-00

Abstract

This document specifies an "HTTP" resource record type for the DNS to facilitate the lookup of the server hostname of HTTP(s) URIs. It is intended to replace the use of CNAME records for this purpose, and in the process provides a solution for the inability of the DNS to allow a CNAME to be placed at the apex of a domain name.

Table of Contents

1. Introduction

It is very common for HTTP(s) URIs to contain a domain name that is not the same as the hostname of the actual server that hosts the content.

This is typically achieved via a CNAME record where the owner name of that record (the "Alias") is the domain name from the URI and the Canonical name field in its RDATA corresponds with the target hostname (although it should be noted that this strictly a violation of the original design semantics of the CNAME record).

It is also impossible to store a CNAME at the apex of a domain name, which causes signficant difficulties if you wish to redirect your domain name without a "www" prefix to a content delivery network (CDN). The only portable solution at the moment is to determine the IP address records of the content host and insert them directly at the apex of the zone, but this is brittle, and prevents the correct operation of typical CDN features.

While there have been previous attempts to promote the use of the SRV record instead of CNAME records, there have been concerns raised about the performance impact of the additional DNS lookup an SRV record would typically require.

To achieve equivalent end-user performance as existing CNAME-based solutions, this document permits recursive resolvers to pre-emptively look up the target of an HTTP Record and return the corresponding

records to the client. While this feature is not mandatory it is hoped that support would over time become near ubiquitous.

Also, the presence of the Port field in an SRV record is incompatible with the "Same Origin" security policy enforced by web browsers and in practise the load-balancing / fallback capabilities of the SRV record are not widely used either, and non-DNS based solutions for this are already widely deployed for HTTP traffic.

This document therefore specifies a minimal "HTTP" resource record type for the DNS to facilitate the redirection from the domain name portion of an HTTP(s) URI to the server hostname and thence to A or AAAA records. It is specifically intended to replace the use of CNAME records for this purpose, and in the process provides a solution for the inability of the DNS to allow a CNAME to be placed at the apex of a domain name.

2. Terminology

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

3. Description

The owner name of an HTTP RR is the domain name portion of an HTTP(s) URI.

The use of underscore label prefixes (e.g. _http._tcp) was considered, but rejected since it prohibits the use of wildcard records which us a valuable technique for offering per-customer domain prefixes without requiring that every prefix be individually provisioned.

3.1. Wire Format

The RDATA of an HTTP RR is a domain name in uncompressed wire format.

3.2. Presentation Format

The RDATA of an HTTP RR is presented as a domain name in standard master file format.

3.3. Server Operation

Recursive resolvers MAY on receiving a request for an HTTP record look up the A and AAAA records for the target (either from cache, or via new iterative queries) and include the results in the Additional Section of the response.

If the recursive resolver is performing DNSSEC resolution but is unable to validate the A or AAAA responses it MUST NOT include them in the response unless the client has specified the +CD (checking disabled) flag.

Where EDNS Client Subnet [RFC7871] is configured on the resolver those A and AAAA lookups MUST be performed as if the client had made those queries directly to the resolver.

3.4. Client Operation

HTTP clients supporting this specification MUST issue parallel DNS requests for the A, AAAA and HTTP records for the domain portion of an http: or https: URI.

If an HTTP record is returned, the client MUST either use the A and AAAA records contained in the Additional Section of the response, or issue further parallel requests for the A and AAAA records corresponding to the domain name in the RDATA of the HTTP record and then use those IP addresses to access the URI.

If the original A and AAAA lookups return IP addresses these MUST only be used if no HTTP record is returned.

<< the above needs more text around timing, happy eyeballs, etc. >>

4. Security Considerations

TBD

5. Implementation status

<< RFC Editor Note: Please remove this entire section prior to publication as an RFC. >>

6. Privacy Considerations

TBD (if any)

7. IANA Considerations

<< a copy of the RFC 6895 IANA RR TYPE application template will appear here >>

8. Acknowledgements

9. References

9.1. Normative References

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

[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>.

9.2. Informative References

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

Author's Address

Ray Bellis
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City CA 94063
USA

Phone: +1 650 423 1200
Email: ray@isc.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 4, 2019

P. Hoffman
ICANN

October 1, 2018

IANA Registry for Special Labels in the DNS
draft-hoffman-dns-special-labels-00

Abstract

This document defines an new IANA registry for special labels in the DNS. The registry is useful because the labels cause special handling in DNS entities such as stub resolvers, recursive resolvers, and applications that use DNS, and developers of software for those entities should be aware of the many types of special labels in use.

[[A GitHub repo for this document is at https://github.com/paulehoffman/dns-special-labels]]

Table of Contents

1. Introduction

Some DNS-related RFCs define labels that are treated specially by stub resolvers, by recursive resolvers, and by applications. It would be useful for DNS software developers to know what the entire set of such special labels are so they can determine if their software needs to handle those labels different than other labels. This document defines an IANA registry for special labels and lists the initial entries for that registry.

The special labels defined in the various RFCs were developed after extensive IETF evaluation of alternative patterns and approaches in light of the desired behavior of the associated protocols. The group designing each protocol looked at the many different ways that the protocol might be best deployed.

1.1. Terminology

In this document, "left-most label" means the label that appears at the left of a domain name when it is wire format or presentation format, as defined in [I-D.ietf-dnsop-terminology-bis]. In an application that uses IDNA [RFC5891] with one or more right-to-left labels, the order of the labels might appear different in the application.

2. Definition of the New IANA Registry

The creation of the registry is defined in Section 4.

@@ Proposed rule for getting in the registry: @@

A label or label-type can be added to the registry only by IESG approval. This approval will likely come when an Internet Draft is progressed toward publication by the RFC Editor, but can come at any time. The reason to require IESG approval as compared to something less onerous such as "expert review" is that developers who rely on the registry will expect it to contain labels and label types that are relatively stable.

The columns of the registry are as follows:

@@ Define the columns here @@

3. Existing Special Labels

The following describes the labels that are the initial contents of the registry described in Section 4.

3.1. The Root Label

According to [RFC1035], a zero-length label always indicates the root label in a domain name.

3.2. Underscore Labels

In many protocols, one or more of the left-most labels might be a label that starts with an underscore (_) character. Those labels are considered special within the context of those protocols.

The use of underscore labels is described in [I-D.ietf-dnsop-attrleaf] and [I-D.ietf-dnsop-attrleaf-fix].

3.3. IDNA

[RFC5891] defines "A-labels" as labels that begin with the characters "xn-". These labels can appear at any position in a domain name.

3.4. Sentinel

[I-D.ietf-dnsop-kskroll-sentinel] (if approved as an RFC) defines root-key-sentinel-is-ta-<key-tag> and root-key-sentinel-not-ta-<key-tag> as special labels when they are the left-most label. In these

labels, "<key-tag>" is an unsigned decimal integer that is zero-padded to five digits.

3.5. MTA-STS

[RFC8461] defines "mta-sts" as as special label when it is the left-most label.

4. IANA Considerations

@@@ Formally define the new registry here @@@

5. Security Considerations

This document doesn’t introduce any new security considerations.

6. References

6.1. Normative References

[I-D.ietf-dnsop-attrleaf]
Crocker, D., "DNS Scoped Data Through "Underscore" Naming of Attribute Leaves", draft-ietf-dnsop-attrleaf-13 (work in progress), August 2018.

[I-D.ietf-dnsop-attrleaf-fix]
Crocker, D., "DNS Attrleaf Changes: Fixing Specifications with Underscored Node Name Use", draft-ietf-dnsop-attrleaf-fix-04 (work in progress), August 2018.

[I-D.ietf-dnsop-kskroll-sentinel]
Huston, G., Damas, J., and W. Kumari, "A Root Key Trust Anchor Sentinel for DNSSEC", draft-ietf-dnsop-kskroll-sentinel-15 (work in progress), July 2018.

[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>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC8461] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <https://www.rfc-editor.org/info/rfc8461>.

6.2. Informative References

[I-D.ietf-dnsop-terminology-bis]
Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", draft-ietf-dnsop-terminology-bis-14 (work in progress), September 2018.

Appendix A. Acknowledgements

@@@ List folks who think of other new labels to add or come up with additional wording for the document @@@

Author’s Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 17, 2019

P. Hoffman
ICANN

December 14, 2018

Associating a DoH Server with a Resolver
draft-hoffman-resolver-associated-doh-07

Abstract

Browsers and web applications may want to know if there are one or more DoH servers associated with the DNS recursive resolver that the operating system is already using. This would allow them to get DNS responses from a resolver that the user (or, more likely, the user's network administrator) has already chosen. This document describes two protocols for a resolver to tell a client what its associated DoH servers are. It also describes a protocol for a client to find out the address of the resolver it is using, if it cannot find that address by an operating system API or some other means.

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 June 17, 2019.

Copyright Notice

Table of Contents

1. Introduction

DoH [RFC8484] requires that one or more DoH servers be configured for the DoH client. That document does not say how the DoH servers are found, nor how to select from a list of possible DoH servers, nor what the user interface (UI) for the configuration should be.

There is a use case for browsers and web applications to want the DNS recursive resolver(s) configured in the operating system to use DoH for DNS resolution instead of normal DNS, but to do so to at a DoH server specified by the configured resolver. For example, a recursive resolver configured by the operating system may know how to give correct answers to DNS queries that contain names that are only resolvable in the local context, or resolve differently in the local context. Similarly, the recursive resolver configured in the operating system may implement security policies such as malware prevention that are not implemented in the same way in DoH servers not affiliated with the user's organization. Users typically configure their DNS recursive resolvers with through automatic configuration from a protocol such as DHCP; much less often, they use

manual configuration (such as manually editing a /etc/named.conf file).

The expected use cases for DoH are browsers and web applications that would otherwise get their DNS service from the resolver configured by the operating system. The user of the client might have a preference for using a DoH server for the benefits that DoH brings, and they might need to use a DoH server that is associated with the resolver that the computer is currently using for the reasons listed above. In a common scenario, user may be required to use only resolvers that are approved by their organization's network operators.

To address these use cases, this document defines two protocols to get the list of URI templates [RFC6570] for the DoH servers associated with at least one of the resolvers being used by the operating system on the system on which the application is being run.

- The first, called "DoH servers by TXT" and described in Section 2.1, is a new special use domain name (SUDN) [RFC6761] that can be queried for a TXT RRset. This protocol is most likely useful only to browsers that can call operating system functions that in turn query the DNS for text records; web applications can only query for IP addresses.

- The second, called "DoH servers by addresses" and described in Section 2.2, is a well-known URI [I-D.nottingham-rfc5785bis] that can be resolved to return the URI templates. This is useful if a browser can call operating system functions that will return the address of the recursive DNS resolver that the operating system is currently using.

This document also defines a third protocol, called "resolver addresses by SUDN" and described in Section 3, that is a new SUDN that that can be queried for the IP address(es) of a resolver. This protocol is useful for a client that can query for the addresses associated with a domain name (such as using the POSIX "getaddrinfo()" function) but cannot use an operating system function to find those addresses. For browsers, it is only needed if the browser cannot use an API to determine the configured resolver IP address(es).

The design choices for this protocol, particularly earlier designs that were deemed unusable, are described in Section 5.

1.1. Terminology

In this document, "client" means either a web browser or application. When one or the other is named explicitly,

In this document, "DoT" is used to indicate DNS over TLS as defined in [RFC7858].

In this document, "Do53" is used to indicate DNS over UDP or TCP as defined in [RFC1035].

"DoH client" and "DoH server" are defined in [RFC8484].

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. Finding the URI Templates of the DoH Servers Associated with a Resolver

A client (a browser or web application) uses either the protocol in Section 2.1 or Section 2.2 to get a list of URI templates for the DoH servers associated with a resolver. The following sub-sections describe the protocols and have notes that are common to both protocols.

2.1. DoH Servers by TXT

To find the DoH Servers associated with a resolver, an application sends that resolver a query for "resolver-associated-doh.arpa" in class IN with the RRtype of TXT [RFC1035] (that is, the query is resolver-associated-doh.arpa/IN/TXT).

As described in Section 6, the zone resolver-associated-doh.arpa is not actually delegated and never will be. The resolver acts as if it is delegated, and adds its own TXT records to the answer. The resolver replies with its associated DoH servers as URI templates in the TXT RRset in the Answer section. The resolver can generate this reply with special code to capture queries for "resolver-associated-doh.arpa"; if the resolver can be configured to also be authoritative for some zones, it can use that configuration to actually be authoritative for "resolver-associated-doh.arpa".

A resolver that understands this protocol MUST send a TXT RRset in the Answer section. Each TXT record contains one URI template. If a resolver that understands this protocol has no associated DoH

servers, the TXT RRset contains exactly one record that has an empty string as the RDATA; that is, the RDLENGTH in that record is 1, and the RDATA contains just the byte 0x00.

The client uses the TXT records in the response to the resolver-associated-doh.arpa/IN/TXT query as a list of the URI templates of the DoH servers associated with the resolver. Note that TXT records can contain multiple "character-strings" [RFC1035]; for this protocol, all characters-strings in a TXT record are concatenated to form a single URI template.

The URI templates of the DoH servers associated with a resolver might be hosted on the resolver itself, or a resolver hosted by the same operator, or even hosted somewhere else. The latter could be used by resolver operators who don't want to host DoH servers but trust another operator to do so.

2.2. DoH Servers by Addresses

To find the DoH servers associated with a resolver, a browser or web application uses either an operating system function (if such a function is available to it) or the process described in Section 3 to find one or more IP addresses for the resolver. It uses one or more of those IP addresses as part of a well-known URI to find out the URI templates [RFC6570] to use for the DoH server(s) associated with the resolver.

To find the DoH servers associated with a resolver, the client sends a query to

https://IPADDRESSGOESHERE/.well-known/doh-servers-associated/

The resolver replies with its associated DoH servers as URI templates [RFC6570].

[[Need to describe the media types; likely JSON]]

[[Need to talk about what a response with an empty list means]]

[[Need to talk about what happens if authentication fails. This is complicated by the fact that the application doesn't know if the OS-to-resolver communication is authenticated.]]

[[Need to talk about HTTP caching]]

A client MUST try to establish a new list of DoH servers associated with a resolver every time the configured resolver in the operating system changes.

The result of sending this query can be an HTTP redirect to a different server, such as when a resolver operator doesn't want to run its own DoH server.

2.3. Issues Common to "DoH Servers by TXT" and "Resolver Addresses by SUDN"

See Section 8 for warnings about sending the DNS queries over a transport that does not assure data integrity (such as Do53), and over a transport that does assure data integrity (such as DoT) but in circumstances where the browser or web application doesn't know the type of DNS transport being used.

A client MUST re-issue the queries in {#doh_by_txt} and {#resolver_by_sudn} every time the configured resolver in the operating system changes.

[[What if there is a list of DoH servers? Pick one (how?) or jump around?]]

3. Finding the Resolver Addresses Without Operating System APIs

Browsers can often, but not always, get the IP address(es) of the resolver configured by the operating system using APIs. Browsers which cannot are still able to use an operating system function such as gethostbyname() or its equivalents to convert host names into IP addresses through the stub resolver in the operating system on which they are running. Web applications also can convert host names to IP addresses. Either can use a new protocol to find the address(es) of the resolvers configured by the operating system.

In this protocol, the browser or web application uses it normal interface for getting addresses for a hostname, and uses the SUDN "resolver-addresses.arpa" as the hostname. As described in Section 6, the zone resolver-addresses.arpa is not actually delegated and never will be. The resolver acts as if that name is delegated, and returns its own A or AAAA addresses in the records in the answer. The resolver can generate this reply with special code to capture queries for "resolver-addresses.arpa"; if the resolver can be configured to also be authoritative for some zones, it can use that configuration to actually be authoritative for "resolver-addresses.arpa".

4. User Interface

For this protocol to be useful in a browser, the browser needs to have an entry in its configuration interface where the allowed DoH servers are listed that indicates that a DoH server from the

configured Do53 or DoT resolver is allowed. That wording might say something like "DoH server associated with my current resolver" (or "servidor DoH asociado con mi resolucion actual" or "serveur DoH associe a mon resolveur actuel").

5. Design Choices

The primary use case for this protocol is a browser or web application that is getting name resolution through the stub resolver on the computer on which it is running wanting to switch its name resolution to DoH. A secondary use case is an operating system that wants to make a similar switch.

An earlier design suggestion was to use a new RRtype with a query to ./IN/NEWRRTYPE. However, it was pointed out that this would not work going through stub resolvers that validate DNSSEC.

An earlier design suggestion was to use DHCP to tell the operating system the DoH servers that the stub resolver might use. That protocol is orthogonal to the one in this document in that it addresses a different use case. If both the protocol in this document and a DHCP-based protocol are standardized, they could co-exist. However, there is no current mechanism for a stub resolver to tell a browser, or a web application, what DoH server the stub resolver is using, so DoH configuration in the stub resolver would not prevent the browser from trying to find a DoH server on its own.

An earlier design suggestion was to use an EDNS0 [RFC6891] extension. The design chosen in this document meets the use case better because applications cannot communicate EDNS0 extensions to the stub resolver.

6. IANA Considerations

IANA will record the domain name "resolver-associated-doh.arpa" in the "Special-Use Domain Names" registry [SUDN]. IANA MUST NOT delegate resolver-associated-doh.arpa in the .arpa zone.

IANA will record the domain name "resolver-addresses.arpa" in the "Special-Use Domain Names" registry [SUDN]. IANA MUST NOT delegate resolver-addresses.arpa in the .arpa zone.

Before this draft is complete, mail will be sent to wellknown-uri-review@ietf.org in order to be registered in the "Well-Known URIs" registry at IANA. The mail will contain the following:

URI suffix: doh-servers-associated
Change controller: IETF
Specification document(s): draft-hoffman-resolver-associated-doh
(or its successor, if it is adopted in a WG)
Status: permanent

7. Privacy Considerations

Allowing a user to use DoH instead of Do53 increases communication privacy because of the TLS protection.

When a Do53 or DoT server indicates that a particular DoH server is associated with it, the client might assume that the DoH server has the same information privacy policies as the Do53 or DoT server. Therefore, a Do53 or DoT server SHOULD NOT recommend a DoH server unless that DoH server has the same (or better) information privacy policy as the Do53 or DoT server.

A browser that has both a stub resolver stack and a TLS stack that is independent of HTTP could make a DOT connection to the resolver being used by the operating system.

8. Security Considerations

There is currently no way for an application to know whether the operating system's stub resolver is using a transport that assures data integrity such as DoT.

Even is an application could determine the use of a transport like DoT, the application would also need to know whether the transport was authenticated or was simply chosen opportunistically.

9. References

9.1. Normative References

[I-D.nottingham-rfc5785bis]
Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", draft-nottingham-rfc5785bis-08 (work in progress), October 2018.

[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>.

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

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

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

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

[SUDN] "Special-Use Domain Names", n.d., <https://www.iana.org/assignments/special-use-domain-names/>.

9.2. Informative References

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

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

Acknowledgments

The use case in this document was inspired by discussions and the DRIU BoF at IETF 102 and later in the DNSOP Working Group. Vladimir Cunat, Philip Homburg, Shumon Huque, Martin Thomson, Eric Rescorla, and Tony Finch offered useful advice to improve early versions of the protocol.

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

DNS Operations
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

T. Finch
University of Cambridge
E. Hunt
ISC
P. van Dijk
PowerDNS
A. Eden
DNSimple
W. Mekking
ISC

July 8, 2019

Address-specific DNS aliases (ANAME)
draft-ietf-dnsop-aname-04

Abstract

This document defines the "ANAME" DNS RR type, to provide similar functionality to CNAME, but only for address queries. Unlike CNAME, an ANAME can coexist with other record types. The ANAME RR allows zone owners to make an apex domain name into an alias in a standards compliant manner.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2020.

Table of Contents

1. Introduction

It can be desirable to provide web sites (and other services) at a bare domain name (such as "example.com") as well as a service-specific subdomain ("www.example.com").

If the web site is hosted by a third-party provider, the ideal way to provision its name in the DNS is using a CNAME record, so that the third party provider retains control over the mapping from names to IP address(es). It is now common for name-to-address mappings to be highly dynamic, dependent on client location, server load, etc.

However, CNAME records cannot coexist with other records with the same owner name. (The reason why is explored in Appendix B). This restriction means they cannot appear at a zone apex (such as "example.com") because of the SOA, NS, and other records that have to be present there. CNAME records can also conflict at subdomains, for example, if "department.example.edu" has separately hosted mail and web servers.

Redirecting website lookups to an alternate domain name via SRV or URI resource records would be an effective solution from the DNS point of view, but to date, browser vendors have not accepted this approach.

As a result, the only widely supported and standards-compliant way to publish a web site at a bare domain is to place address records (A

and/or AAAA) at the zone apex. The flexibility afforded by CNAME is not available.

This document specifies a new RR type "ANAME", which provides similar functionality to CNAME, but only for address queries (i.e., for type A or AAAA). The basic idea is that the address records next to an ANAME record are automatically copied from and kept in sync with the ANAME target's address records. The ANAME record can be present at any DNS node, and can coexist with most other RR types, enabling it to be present at a zone apex, or any other name where the presence of other records prevents the use of a CNAME record.

Similar authoritative functionality has been implemented and deployed by a number of DNS software vendors and service providers, using names such as ALIAS, ANAME, apex CNAME, CNAME flattening, and top-level redirection. These mechanisms are proprietary, which hinders the ability of zone owners to have the same data served from multiple providers or to move from one provider to another. None of these proprietary implementations includes a mechanism for resolvers to follow the redirection chain themselves.

1.1. Overview

The core functionality of this mechanism allows zone administrators to start using ANAME records unilaterally, without requiring secondary servers or resolvers to be upgraded.

- The resource record definition in Section 2 is intended to provide zone data portability between standards-compliant DNS servers and the common core functionality of existing proprietary ANAME-like facilities.

- The zone maintenance mechanism described in Section 4 keeps the ANAME's sibling address records in sync with the ANAME target.

This definition is enough to be useful by itself. However, it can be less than optimal in certain situations: for instance, when the ANAME target uses clever tricks to provide different answers to different clients to improve latency or load balancing. The query processing rules in Section 6 require to include the ANAME record so that resolvers can use this information (as described in Section 5) to obtain answers that are tailored to the resolver rather than to the zone's primary master.

Resolver support for ANAME is not necessary, since ANAME-oblivious resolvers can get working answers from authoritative servers. It's just an optimization that can be rolled out incrementally, and that will help ANAME to work better the more widely it is deployed.

1.2. Terminology

An "address record" is a DNS resource record whose type is A or AAAA. These are referred to as "address types". "Address query" refers to a DNS query for any address type.

When talking about "address records" we mean the entire RRset, including owner name and TTL. We treat missing address records (i.e. NXDOMAIN or NODATA) the same successfully resolving as a set of zero address records, and distinct from "failure" which covers error responses such as SERVFAIL or REFUSED.

The "sibling address records" of an ANAME record are the address records at the same owner name as the ANAME, which are subject to ANAME substitution.

The "target address records" of an ANAME record are the address records obtained by resolving the ultimate target of the ANAME (see

Section 3).

During the process of looking up the target address records, one or more CNAME or ANAME records may be encountered. These records are not the final target address records, and are referred in this document as "intermediate records". The target name must be replaced with the new name provided in the RDATA and the new target is resolved.

Other DNS-related terminology can be found in [RFC8499].

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 ANAME resource record

This document defines the "ANAME" DNS resource record type, with RR TYPE value [TBD].

2.1. Presentation and wire format

The ANAME presentation format is identical to that of CNAME [RFC1033]:

```
owner ttl class ANAME target
```

The wire format is also identical to CNAME [RFC1035], except that name compression is not permitted in ANAME RDATA, per [RFC3597].

2.2. Coexistence with other types

Only one ANAME <target> can be defined per <owner>. An ANAME RRset MUST NOT contain more than one resource record.

An ANAME's sibling address records are under the control of ANAME processing (see Section 4) and are not first-class records in their own right. They MAY exist in zone files, but they can subsequently be altered by ANAME processing.

An ANAME record MAY freely coexist at the same owner name with other RR types, except they MUST NOT coexist with CNAME or any other RR type that restricts the types with which it can itself coexist. That means An ANAME record can coexist at the same owner name with A and AAAA records. These are the sibling address records that are updated with the target addresses that are retrieved through the ANAME substitution process Section 3.

Like other types, An ANAME record can coexist with DNAME records at the same owner name; in fact, the two can be used cooperatively to redirect both the owner name address records (via ANAME) and everything under it (via DNAME).

3. Substituting ANAME sibling address records

This process is used by both primary masters (see Section 4) and resolvers (see Section 5), though they vary in how they apply the edit described in the final step. However, this process is not exclusively used by primary masters and resolvers: it may be executed as a bump in the wire, as part of the query lookup, or at any other point during query resolution.

The following steps MUST be performed for each address type:

1. Starting at the ANAME owner, follow the chain of ANAME and/or CNAME records as far as possible to find the ultimate target.

2. If a loop is detected, continue with an empty RRset, otherwise get the ultimate target's address records. (Ignore any sibling address records of intermediate ANAMEs.)

3. Stop if resolution failed. (Note that NXDOMAIN and NODATA count as successfully resolving an empty RRset.)

4. If one or more address records are found, replace the owner of the target address records with the owner of the ANAME record. Set the TTL to the minimum of the ANAME TTL, the TTL of each intermediate record, and the TTL of the target address records. Drop any RRSIG records.

5. Stop if this modified RRset is the same as the sibling RRset (ignoring any RRSIG records). The comparison MAY treat nearly-equal TTLs as the same.

6. Delete the sibling address RRset (if any) and replace it with the modified RRset.

At this point, the substituted RRset is not signed. A primary master will proceed to sign the substituted RRset, whereas resolvers can only use the substituted RRset when an unsigned answer is appropriate. This is explained in more detail in the following sections.

4. ANAME processing by primary masters

Each ANAME's sibling address records are kept up-to-date as if by the following process, for each address type:

- Perform ANAME sibling address record substitution as described in Section 3. Any edit performed in the final step is applied to the ANAME's zone. A primary server MAY use Dynamic Updates (DNS UPDATE) [RFC2136] to update the zone.

- If resolution failed, wait for a period before trying again. This retry time SHOULD be configurable.

- Otherwise, wait until the target address RRset TTL has expired or is close to expiring, then repeat.

It may be more efficient to manage the polling per ANAME target rather than per ANAME as specified (for example if the same ANAME target is used by multiple zones).

Sibling address records are committed to the zone and stored in nonvolatile storage. This allows a server to restart without delays due to ANAME processing, use offline DNSSEC signing, and not implement special ANAME processing logic when handling a DNS query.

Appendix D describes how ANAME would fit in different DNS architectures that use online signing or tailored responses.

4.1. Zone transfers

ANAME is no more special than any other RRtype and does not introduce any special processing related to zone transfers.

A zone containing ANAME records that point to frequently-changing targets will itself change frequently, and may see an increased number of zone transfers. Or if a very large number of zones are sharing the same ANAME target, and that changes address, that may cause a great volume of zone transfers. Guidance on dealing with ANAME in large scale implementations is provided Appendix D.

Secondary servers rely on zone transfers to obtain sibling address

records, just like the rest of the zone, and serve them in the usual way (see Section 6). A working DNS NOTIFY [RFC1996] setup is recommended to avoid extra delays propagating updated sibling address records when they change.

4.2. DNSSEC

A zone containing ANAME records that will update address records has to do so before signing the zone with DNSSEC [RFC4033] [RFC4034] [RFC4035]. This means that for traditional DNSSEC signing the substitution of sibling address records must be done before signing and loading the zone into the name server. For servers that support online signing, the substitution may happen as part of the name server process, after loading the zone.

DNSSEC signatures on sibling address records are generated in the same way as for normal (dynamic) updates.

4.3. TTLs

Sibling address records are served from authoritative servers with a fixed TTL. Normally this TTL is expected to be the same as the target address records' TTL; however the exact mechanism for obtaining the target is unspecified, so cache effects, following ANAME and CNAME chains, or deliberate policies might make the sibling TTL smaller.

This means that when adding address records into the zone as a result of ANAME processing, the TTL to use is at most that of the TTL of the address target records. If you use a higher value, this will stretch the TTL which is undesired.

TTL stretching is hard to avoid when implementing ANAME substitution at the primary: The target address records' TTL influences the update rate of the zone, while the sibling address records' TTL determine how long a resolver may cache the address records. Thus, the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL. There is a more extended discussion of TTL handling in Appendix C.

5. ANAME processing by resolvers

When a resolver makes an address query in the usual way, it might receive a response containing ANAME information in the Answer section, as described in Section 6. This informs the resolver that it MAY resolve the ANAME target address records to get answers that are tailored to the resolver rather than the ANAME's primary master.

In order to provide tailored answers to clients that are ANAME-oblivious, the resolver MAY perform sibling address record substitution in the following situations:

- The resolver's client queries with DO=0. (As discussed in Section 8, if the resolver finds it would downgrade a secure answer to insecure, it MAY choose not to substitute the sibling address records.)

- The resolver's client queries with DO=1 and the ANAME and sibling address records are unsigned. (Note that this situation does not apply when the records are signed but insecure: the resolver might not be able to validate them because of a broken chain of trust, but its client could have an extra trust anchor that does allow it to validate them; if the resolver substitutes the sibling address records they will become bogus.)

In these first two cases, the resolver MAY perform ANAME sibling

address record substitution as described in Section 3. Any edit performed in the final step is applied to the Answer section of the response.

If the resolver's client is querying using an API such as "getaddrinfo" [RFC3493] that does not support DNSSEC validation, the resolver MAY perform ANAME sibling address record substitution as described in Section 3. Any edits performed in the final step are applied to the addresses returned by the API. (This case is for validating stub resolvers that query an upstream recursive server with DO=1, so they cannot rely on the recursive server to do ANAME substitution for them.)

6. Query processing

6.1. Authoritative servers

6.1.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries. The ANAME record indicates to a client that it might wish to resolve the target address records itself.

6.1.2. ANAME queries

When a server receives an query for type ANAME, regardless of whether the ANAME record exists on the queried domain, any sibling address records SHOULD be added to the Additional section. Note that the sibling address records may have been substituted already.

When adding address records to the Additional section, if not all address types are present and the zone is signed, the server SHOULD include a DNSSEC proof of nonexistence for the missing address types.

6.2. Resolvers

6.2.1. Address queries

When a server receives an address query for a name that has an ANAME record, the response's Answer section MUST contain the ANAME record, in addition to the sibling address queries.

The Additional section MAY contain the target address records that match the query type (or the corresponding proof of nonexistence), if they are available in the cache and the target address RDATA fields differ from the sibling address RRset.

An ANAME target MAY resolve to address records via a chain of CNAME and/or ANAME records; any CNAME/ANAME chain MUST be included when adding target address records to a response's Additional section.

6.2.2. ANAME queries

When a resolver receives an query for type ANAME, any sibling address records SHOULD be added to the Additional section. Just like with an authoritative server, when adding address records to the Additional section, if not all address types are present and the zone is signed, the resolver SHOULD include a DNSSEC proof of nonexistence for the missing address types.

7. IANA considerations

IANA is requested to assign a DNS RR TYPE value for ANAME resource records under the "Resource Record (RR) TYPEs" subregistry under the "Domain Name System (DNS) Parameters" registry.

IANA might wish to consider the creation of a registry of address types; addition of new types to such a registry would then implicitly update this specification.

8. Security considerations

When a primary master updates an ANAME's sibling address records to match its target address records, it uses its own best information as to the correct answer. The primary master might sign the updated records, but that is not a guarantee of the actual correctness of the answer. This signing can have the effect of promoting an insecure response from the ANAME <target> to a signed response from the <owner>, which can then appear to clients to be more trustworthy than it should. DNSSEC validation SHOULD be used when resolving the ANAME <target> to mitigate this possible harm. Primary masters MAY refuse to substitute ANAME sibling address records unless the <target> node is both signed and validated.

When a resolver substitutes an ANAME's sibling address records, it can find that the sibling address records are secure but the target address records are insecure. Going ahead with the substitution will downgrade a secure answer to an insecure one. However this is likely to be the counterpart of the situation described in the previous paragraph, so the resolver is downgrading an answer that the ANAME's primary master upgraded. A resolver will only downgrade an answer in this way when its client is security-oblivious; however the client's path to the resolver is likely to be practically safer than the resolver's path to the ANAME target's servers. Resolvers MAY choose not to substitute sibling address records when they are more secure than the target address records.

9. Acknowledgments

Thanks to Mark Andrews, Ray Bellis, Stefan Buehler, Paul Ebersman, Richard Gibson, Tatuya JINMEI, Hakan Lindqvist, Mattijs Mekking, Stephen Morris, Bjorn Mott, Richard Salts, Mukund Sivaraman, Job Snijders, Jan Vcelak, Paul Vixie, Duane Wessels, and Paul Wouters, Olli Vanhoja, Brian Dickson for discussion and feedback.

10. Changes since the last revision

[This section is to be removed before publication as an RFC.]

The full history of this draft and its issue tracker can be found at https://github.com/each/draft-aname [1]

10.1. Version -04

- Split up section about Additional Section processing.
- Update Additional Section processing requirements.
- Clarify when ANAME resolution may happen [#43].
- Revisit TTL considerations [#30, #34].
- ANAME goes into the Answer section when QTYPE=A|AAAA [#62].
- Update alternative setups section with concerns (Brian Dickson) [#68].
- Add section on ANAME loops (open issue [#45]).

10.2. Version -03

- Grammar improvements (Olli Vanhoja)

- Split up Implications section, clarify text on zone transfers and dynamic updates [#39].

- Rewrite Alternative setup section and move to Appendix, add text on zone transfer scalibility concerns and GeoIP.

10.3. Version -02

Major revamp, so authoritative servers (other than primary masters) now do not do any special ANAME processing, just Additional section processing.

11. References

11.1. Normative References

[RFC1033] Lottor, M., "Domain Administrators Operations Guide", RFC 1033, DOI 10.17487/RFC1033, November 1987, <https://www.rfc-editor.org/info/rfc1033>.

[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>.

[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, <https://www.rfc-editor.org/info/rfc2136>.

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

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

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

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

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

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

11.2. Informative References

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983,

<https://www.rfc-editor.org/info/rfc882>.

[RFC0973] Mockapetris, P., "Domain system changes and observations", RFC 973, DOI 10.17487/RFC0973, January 1986, <https://www.rfc-editor.org/info/rfc973>.

[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>.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <https://www.rfc-editor.org/info/rfc2065>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <https://www.rfc-editor.org/info/rfc3493>.

11.3. URIs

[1] https://github.com/each/draft-aname

[2] https://github.com/each/draft-aname/issues/45

Appendix A. Implementation status

PowerDNS currently implements a similar authoritative-only feature using "ALIAS" records, which are expanded by the primary server and transfered as address records to secondaries.

[TODO: Add discussion of DNSimple, DNS Made Easy, EasyDNS, Cloudflare, Amazon, Dyn, and Akamai.]

Appendix B. Historical note

In the early DNS [RFC0882], CNAME records were allowed to coexist with other records. However this led to coherency problems: if a resolver had no cache entries for a given name, it would resolve queries for un-cached records at that name in the usual way; once it had cached a CNAME record for a name, it would resolve queries for un-cached records using CNAME target instead.

For example, given the zone contents below, the original CNAME behaviour meant that if you asked for "alias.example.com TXT" first, you would get the answer "owner", but if you asked for "alias.example.com A" then "alias.example.com TXT" you would get the answer "target".

```
alias.example.com.      TXT    "owner"
alias.example.com.      CNAME  canonical.example.com.
canonical.example.com.  TXT    "target"
canonical.example.com.  A      192.0.2.1
```

This coherency problem was fixed in [RFC0973] which introduced the inconvenient rule that a CNAME acts as an alias for all other RR types at a name, which prevents the coexistence of CNAME with other records.

A better fix might have been to improve the cache's awareness of which records do and do not coexist with a CNAME record. However that would have required a negative cache mechanism which was not added to the DNS until later [RFC1034] [RFC2308].

While [RFC2065] relaxed the restriction by allowing coexistence of CNAME with DNSSEC records, this exception is still not applicable to other resource records. RRSIG and NSEC exist to prove the integrity of the CNAME record; they are not intended to associate arbitrary data with the domain name. DNSSEC records avoid interoperability problems by being largely invisible to security-oblivious resolvers.

Now that the DNS has negative caching, it is tempting to amend the algorithm for resolving with CNAME records to allow them to coexist with other types. Although an amended resolver will be compatible with the rest of the DNS, it will not be of much practical use because authoritative servers which rely on coexisting CNAMEs will not interoperate well with older resolvers. Practical experiments show that the problems are particularly acute when CNAME and MX try to coexist.

Appendix C. On preserving TTLs

An ANAME's sibling address records are in an unusual situation: they are authoritative data in the owner's zone, so from that point of view the owner has the last say over what their TTL should be; on the other hand, ANAMEs are supposed to act as aliases, in which case the target should control the address record TTLs.

However there are some technical constraints that make it difficult to preserve the target address record TTLs.

The following subsections conclude that the end-to-end TTL (from the authoritative servers for the target address records to end-user DNS caches) is nearing twice the target address record TTL.

C.1. Query bunching

If the times of end-user queries for a domain name are well distributed, then (typically) queries received by the authoritative servers for that domain are also well distributed. If the domain is popular, a recursive server will re-query for it once every TTL seconds, but the periodic queries from all the various recursive servers will not be aligned, so the queries remain well distributed.

However, imagine that the TTLs of an ANAME's sibling address records are decremented in the same way as cache entries in recursive servers. Then all the recursive servers querying for the name would try to refresh their caches at the same time when the TTL reaches zero. They would become synchronized, and all the queries for the domain would be bunched into periodic spikes.

This specification says that ANAME sibling address records have a normal fixed TTL derived from (e.g. equal or nearly equal to) the target address records' original TTL. There is no cache-like decrementing TTL, so there is no bunching of queries.

C.2. Upstream caches

There are two straightforward ways to get an RRset's original TTL:

- by directly querying an authoritative server;
- using the original TTL field from the RRset's RRGIG record(s).

However, not all zones are signed, and a primary master might not be able to query other authoritative servers directly (e.g. if it is a

hidden primary behind a strict firewall). Instead it might have to obtain an ANAME's target address records via some other recursive server.

Querying via a separate recursive server means the primary master cannot trivially obtain the target address records' original TTLs. Fortunately this is likely to be a self-correcting problem for similar reasons to the query-bunching discussed in the previous subsection. The primary master can inspect the target address records just after the TTL expires when its upstream cache has just refreshed them, so the TTL will be nearly equal to the original TTL.

A related consideration is that the primary master cannot in general refresh its copies of an ANAME's target address records more frequently than their TTL, without privileged control over its resolver cache.

Combined with the requirement that sibling address records are served with a fixed TTL, this means that the end-to-end TTL will be the target address record TTL (which determines when the sibling address records are updated) plus the sibling address record TTL (which determines when end-user caches are updated). Since the sibling address record TTL is derived from the target address records' original TTL, the end-to-end TTL will be nearing twice the target address record TTL.

C.3. ANAME chains

ANAME sibling address record substitution is made slightly more complicated by the requirement to follow chains of ANAME and/or CNAME records. The TTL of the substituted address records is the minimum of TTLs of the ANAME, all the intermediate records, and target records. This stops the end-to-end TTL from being inflated by each ANAME in the chain.

With CNAME records, repeat queries for "cname.example. CNAME target.example." must not be fully answered from cache after its TTL expires, but must instead be sent to name servers authoritative for "cname.example" in case the CNAME has been updated or removed. Similarly, an ANAME at "aname.example" means that repeat queries for "aname.example" must not be fully answered from cache after its TTL expire, but must instead be sent to name servers authoritative for aname.example in case the ANAME has been updated or removed.

C.4. ANAME substitution inside the name server

When ANAME substitution is performed inside the authoritative name server (as described in #alternatives) or in the resolver (as described in #resolver) the end-to-end TTL will actually be just the target address record TTL.

An authoritative server that has control over its resolver can use a cached target address RRset and decremented TTL in the response to the client rather than using the original target address records' TTL. It SHOULD however not use TTLs in the response that are nearing zero to avoid query bunching Appendix C.1.

A resolver that performs ANAME substitution is able to get the original TTL from the authoritative name server and use its own cache to store the substituted address records with the appropriate TTL, thereby honoring the TTL of target address records.

C.5. TTLs and zone transfers

When things are working properly (with secondary name servers responding to NOTIFY messages promptly) the authoritative servers will follow changes to ANAME target address records according to

their TTLs. As a result the end-to-end TTL is unchanged from the previous subsection.

If NOTIFY doesn't work, the TTLs can be stretched by the zone's SOA refresh timer. More serious breakage can stretch them up to the zone expiry time.

Appendix D. Alternative setups

If you are a large scale DNS provider, ANAME may introduce some operational concerns.

D.1. Reducing query volume

When doing ANAME target lookups, an authoritative server might want to use longer TTLs to reduce query volume, for ANAME values that do not change frequently. This is the same concern a recursive resolver may be exposed to when receiving answers with short TTLs. An authoritative server doing ANAME target lookups therefor could use the same mitigation as a recursive nameserver, that is set a configured minimum TTL usage. This may however contribute to TTL stretching as described in Section 4.3 so the configured minimum should not be too low.

D.2. Zone transfer scalability

A frequently changing ANAME target, or a ANAME target that changes its address and is used for many zones, can lead to an increased number of zone transfers. Such DNS architectures may want to consider a zone transfer mechanism outside the DNS.

Another way to deal with zone transfer scalability is to move the ANAME processing (Section 3) inside the name server daemon. This is not a requirement for ANAME to work, but may be a better solution in large scale implementations. These implementations usually already rely on online DNSSEC signing for similar reasons. If ANAME processing occurs inside the name server daemon, it MUST be done before any DNSSEC online signing happens.

For example, some existing ANAME-like implementations are based on a DNS server architecture, in which a zone's published authoritative servers all perform the duties of a primary master in a distributed manner: provisioning records from a non-DNS back-end store, refreshing DNSSEC signatures, and so forth. They don't use standard standard zone transfers, and already implement their ANAME-like processing inside the name server daemon, substituting ANAME sibling address records on demand.

D.3. Tailored responses

Some DNS providers will tailor responses based on information in the client request. Such implementations will use the source IP address or EDNS Client Subnet [RFC7871] information and use geographical data (GeoIP) or network latency measurements to decide what the best answer is for a given query. Such setups won't work with traditional DNSSEC and provide DNSSEC support usually through online signing. Similar such setups should provide ANAME support through substituting ANAME sibling records on demand.

Also, an authoritative server that uses the client address to tailor the response should obviously not use its own address when looking up ANAME targets, or it could direct clients to a suboptimal server (e.g. a wrong language, or regional restricted content). Instead the authoritative server should look up the ANAME targets on behalf of the client address. It could use for example EDNS Client Subnet for this.

In short, the exact mechanism for obtaining the target address records in such setups is unspecified; typically they will be resolved in the DNS in the usual way, but if an ANAME implementation has special knowledge of the target it can short-cut the substitution process, or it can use clever tricks such as client-dependant answers to make the answer more optimal.

Appendix E. ANAME loops

The ANAME sibling address substitution algorithm in Section 3 poses a challenge of detecting a loop between two or more ANAME records. Imagine this setup: two authoritative servers X and Y performing ANAME sibling address substition on the fly (i.e. they attempt to resolve the ANAME target when the client query arrives). If server X gets a query for FOO.TEST which is an ANAME to BAR.TEST, it will send a query to server Y for BAR.TEST which is an ANAME to FOO.TEST. Server Y will then start a new query to server X, which has no way to know that it is regarding the original FOO.TEST lookup.

The only indicator of the presence of the loop in the described setup is the network timeout. Ideally we would recognize the loop explicitly based on the exchanged DNS messages.

On-the-fly ANAME substitution is allowed and it's just the most obvious scenario where the problem can be demonstrated, but this loop can also be encountered in other situations. The root cause is that when the server gets a query it doesn't know why and that the server always attempts to fully resolve the ANAME target before sending the response.

TODO: Solve this issue [https://github.com/each/draft-aname/issues/45 [2]]

Authors' Addresses

Tony Finch
University of Cambridge
University Information Services
Roger Needham Building
7 JJ Thomson Avenue
Cambridge CB3 0RB
England

Email: dot@dotat.at

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

Email: each@isc.org

Peter van Dijk
PowerDNS.COM B.V.
Den Haag
The Netherlands

Email: peter.van.dijk@powerdns.com

Anthony Eden
DNSimple
Boston, MA USA

Email: anthony.eden@dnsimple.com
URI: https://dnsimple.com/

Matthijs Mekking
ISC
950 Charter St
Redwood City, CA 94063
USA

Email: matthijs@isc.org

Network Working Group
Internet-Draft
Obsoletes: 7816 (if approved)
Intended status: Standards Track
Expires: 30 April 2021

S. Bortzmeyer
AFNIC
R. Dolmans
NLnet Labs
P. Hoffman
ICANN

27 October 2020

DNS Query Name Minimisation to Improve Privacy
draft-ietf-dnsop-rfc7816bis-07

Abstract

This document describes a technique called "QNAME minimisation" to improve DNS privacy, where the DNS resolver no longer always sends the full original QNAME and original QTYPE to the upstream name server. This document obsoletes RFC 7816.

This document is part of the IETF DNSOP (DNS Operations) Working Group. The source of the document, as well as a list of open issues, is at <https://framagit.org/bortzmeyer/rfc7816-bis>

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 30 April 2021.

Copyright Notice

Table of Contents

1. Introduction and Background

The problem statement for this document is described in [RFC7626]. This specific solution is not intended to fully solve the DNS privacy problem; instead, it should be viewed as one tool amongst many.

QNAME minimisation follows the principle explained in Section 6.1 of [RFC6973]: the less data you send out, the fewer privacy problems you have.

Before QNAME minimisation, when a resolver received the query "What is the AAAA record for www.example.com?", it sent to the root (assuming a resolver whose cache is empty) the very same question. Sending the full QNAME to the authoritative name server was a tradition, not a protocol requirement. In a conversation with one of the authors in January 2015, Paul Mockapetris explained that this tradition comes from a desire to optimise the number of requests, when the same name server is authoritative for many zones in a given name (something that was more common in the old days, where the same name servers served .com and the root) or when the same name server is both recursive and authoritative (something that is strongly discouraged now). Whatever the merits of this choice at this time, the DNS is quite different now.

QNAME minimisation is compatible with the current DNS system and therefore can easily be deployed. Because it is only a change to the way that the resolver operates, it does not change the DNS protocol itself. The behaviour suggested here (minimising the amount of data sent in QNAMEs from the resolver) is allowed by Section 5.3.3 of [RFC1034] and Section 7.2 of [RFC1035].

1.1. Experience From RFC 7816

This document obsoletes [RFC7816]. RFC 7816 was labelled "experimental", but ideas from it were widely deployed since its publicaiton. Many resolver implementations now support QNAME minimisation. The lessons learned from implementing QNAME minimisation were used to create this new revision.

Data from DNSThought [dnsthought-qnamemin], Verisign [verisign-qnamemin] and APNIC [apnic-qnamemin] shows that a large percentage of the resolvers deployed on the Internet already support QNAME minimisation in some way.

Academic research has been performed on QNAME minimisation [devries-qnamemin]. This work shows that QNAME minimisation in relaxed mode causes almost no problems. The paper recommends using the A QTYPE, and limiting the number of queries in some way.

1.2. Terminology

The terminology used in this document is defined in [RFC8499].

In this document, a "cold" cache is one that is empty, having literally no entries in it. A "warm" cache is one that has some entries in it.

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. Description of QNAME Minimisation

The idea behind QNAME minimisation is to minimise the amount of privacy sensitive data sent from the DNS resolver to the authoritative name server. This section describes how to do QNAME minimisation. The algorithm is summarised in Section 3.

When a resolver is not able to answer a query from cache it has to send a query to an authoritative nameserver. Traditionally these queries would contain the full QNAME and the original QTYPE as received in the client query.

The full QNAME and original QTYPE are only needed at the nameserver that is authoritative for the record requested by the client. All other nameservers queried while resolving the query only need to receive enough of the QNAME to be able to answer with a delegation. The QTYPE in these queries is not relevant, as the nameserver is not able to authoritatively answer the records the client is looking for. Sending the full QNAME and original QTYPE to these nameservers therefore exposes more privacy sensitive data than necessary to resolve the client's request.

A resolver that implements QNAME minimisation changes the QNAME and QTYPE in queries directed to an authoritative nameserver that is not known to be responsible for the original QNAME. These queries contain:

* a QTYPE selected by the resolver to hide the original QTYPE

* the QNAME that is the original QNAME, stripped to just one label more than the longest matching domain name for which the nameserver is known to be authoritative

2.1. QTYPE Selection

Note that this document relaxes the recommendation in RFC 7816 to use the NS QTYPE to hide the original QTYPE. Using the NS QTYPE is still allowed. The authority of NS records lies at the child side. The parent side of the delegation will answer using a referral, like it will do for queries with other QTYPEs. Using the NS QTYPE therefore has no added value over other QTYPEs.

The QTYPE to use while minimising queries can be any possible data type (as defined in [RFC6895] Section 3.1) for which the authority always lies below the zone cut (i.e. not DS, NSEC, NSEC3, OPT, TSIG, TKEY, ANY, MAILA, MAILB, AXFR, and IXFR), as long as there is no relation between the incoming QTYPE and the selection of the QTYPE to use while minimising. A good candidate is to always use the "A" QTYPE because this is the least likely to raise issues in DNS software and middleboxes that do not properly support all QTYPEs. The QTYPE=A queries will also blend into traffic from non-minimising resolvers, making it in some cases harder to observe that the resolver is using QNAME minimisation. Using the QTYPE that occurs most in incoming queries will slightly reduce the number of queries, as there is no extra check needed for delegations on non-apex records. Another potential benefit of using QTYPE=A is that [RFC8305] clients that need answers for both the A and AAAA types will send the AAAA query first. When minimising using QTYPE=A the minimised query might be useful, and now already in the cache, for the happy eyeballs query for the A QTYPE.

2.2. QNAME Selection

The minimising resolver works perfectly when it knows the zone cut (zone cuts are described in Section 6 of [RFC2181]). But zone cuts do not necessarily exist at every label boundary. In the name www.foo.bar.example, it is possible that there is a zone cut between "foo" and "bar" but not between "bar" and "example". So, assuming that the resolver already knows the name servers of example, when it receives the query "What is the AAAA record of www.foo.bar.example?", it does not always know where the zone cut will be. To find the zone cut, it will query the example name servers for a record for bar.example. It will get a non-referral answer, it has to query the example name servers again with one more label, and so on. (Section 3 describes this algorithm in deeper detail.)

2.3. Limit Number of Queries

When using QNAME minimisation, the number of labels in the received QNAME can influence the number of queries sent from the resolver. This opens an attack vector and can decrease performance. Resolvers supporting QNAME minimisation MUST implement a mechanism to limit the number of outgoing queries per user request.

Take for example an incoming QNAME with many labels, like www.host.group.department.example.com, where host.group.department.example.com is hosted on example.com's name servers. (Such deep domains are especially common under ip6.arpa.) Assume a resolver that knows only the name servers of example.com. Without QNAME minimisation, it would send these

example.com name servers a query for www.host.group.department.example.com and immediately get a specific referral or an answer, without the need for more queries to probe for the zone cut. For such a name, a cold resolver with QNAME minimisation will send more queries, one per label. Once the cache is warm, there will be less difference with a traditional resolver. Testing of this is described in [Huque-QNAME-Min].

The behaviour of sending multiple queries can be exploited by sending queries with a large number of labels in the QNAME that will be answered using a wildcard record. Take for example a record for *.example.com, hosted on example.com's name servers. An incoming query containing a QNAME with more than 100 labels, ending in example.com, will result in a query per label. By using random labels, the attacker can bypass the cache and always require the resolver to send many queries upstream. Note that [RFC8198] can limit this attack in some cases.

One mechanism to reduce this attack vector is by appending more than one label per iteration for QNAMEs with a large number of labels. To do this, a maximum number of QNAME minimisation iterations has to be selected (MAX_MINIMISE_COUNT); a good value is 10. Optionally, a value for the number of queries that should only have one label appended can be selected (MINIMISE_ONE_LAB), a good value is 4. The assumption here is that the number of labels on delegations higher in the hierarchy are rather small, therefore not exposing too may labels early on has the most privacy benefit.

When a resolver needs to send out a query, it will look for the closest known delegation point in its cache. The number of QNAME minimisation iterations is the difference between this closest nameserver and the incoming QNAME. The first MINIMISE_ONE_LAB iterations will be handles as described in Section 2. The number of labels that are not exposed yet now need to be divided over the iterations that are left (MAX_MINIMISE_COUNT - MINIMISE_ONE_LAB). The remainder of the division should be added to the last iterations. For example, when resolving a QNAME with 18 labels, the number of labels added per iteration are: 1,1,1,1,2,2,2,2,3,3.

2.4. Stub and Forwarding Resolvers

Stub and forwarding resolvers MAY implement QNAME minimisation. Minimising queries that will be sent to an upstream resolver does not help in hiding data from the upstream resolver because all information will end up there anyway. It might, however, limit the data exposure between the upstream resolver and the authoritative nameserver in the situation where the upstream resolver does not support QNAME minimisation. Using QNAME minimisation in a stub or

forwarding resolvers that does not have a mechanism to find and cache zone cuts will drastically increase the number of outgoing queries.

3. Algorithm to Perform QNAME Minimisation

This algorithm performs name resolution with QNAME minimisation in the presence of zone cuts that are not yet known.

Although a validating resolver already has the logic to find the zone cuts, implementers of resolvers may want to use this algorithm to locate the zone cuts.

(0) If the query can be answered from the cache, do so; otherwise, iterate as follows:

(1) Get the closest delegation point that can be used for the original QNAME/QTYPE combination from the cache.

(1a) For queries with QTYPE=DS this is the NS RRset with the owner matching the most labels with the QNAME stripped by one label. The QNAME will be a subdomain of (but not equal to) this NS RRset. Call this ANCESTOR.

(1b) For queries with other original QTYPEs this is the NS RRset with the owner matching the most labels with the QNAME. The QNAME will be equal to or a subdomain of this NS RRset. Call this ANCESTOR.

(2) Initialise CHILD to the same as ANCESTOR.

(3) If CHILD is the same as the QNAME, or if the CHILD is one label shorter than the QNAME and the original QTYPE is DS, resolve the original query using ANCESTOR's name servers, and finish.

(4) Otherwise, add a label from the QNAME to the start of CHILD.

(5) Look for a cache entry for the RRset at CHILD with the original QTYPE. If this entry is for an NXDOMAIN and the resolver has support for [RFC8020], the NXDOMAIN can be used in response to the original query, and stop. If the entry is for a NOERROR answer (either positive or negative), go back to step 3. If the entry is for an NXDOMAIN answer and the resolver does not support RFC 8020, go back to step 3.

(6) Query for CHILD with the selected QTYPE using ANCESTOR's name servers. The response can be:

(6a) A referral. Cache the NS RRset from the authority section,

and go back to step 1.

(6b) A NOERROR answer (either positive or negative). Cache this answer, and go back to step 3.

(6c) If the resolver is doing RFC 8020 with strict NXDOMAIN, an NXDOMAIN answer. Return an NXDOMAIN answer in response to the original query, and stop. If the resolver does not support RFC 8020, go back to step 3.

(6d) An answer with another RCODE, or no answer. Try another name server at the same delegation point. Stop if none of them are able to return a valid answer.

4. QNAME Minimisation Examples

Assume that a resolver receives a request to resolve foo.bar.baz.example. Assume that the resolver already knows that ns1.nic.example is authoritative for .example, and that the resolver does not know a more specific authoritative name server. It will send the query with QNAME=baz.example and the QTYPE selected to hide the original QTYPE to ns1.nic.example.

Here are more detailed examples of queries with QNAME minimisation:

Cold cache, traditional resolution algorithm without QNAME minimisation, request for MX record of a.b.example.org:

QTYPE	QNAME	TARGET	NOTE
MX	a.b.example.org	root nameserver	
MX	a.b.example.org	org nameserver	
MX	a.b.example.org	example.org nameserver	

Cold cache, with QNAME minimisation, request for MX record of a.b.example.org, using the A QTYPE to hide the original QTYPE:

QTYPE	QNAME	TARGET	NOTE
A	org	root nameserver	
A	example.org	org nameserver	
A	b.example.org	example.org nameserver	
A	a.b.example.org	example.org nameserver	"a" may be delegated
MX	a.b.example.org	example.org nameserver	

Note that in above example one query would have been saved if the incoming QTYPE would have been the same as the QTYPE selected by the resolver to hide the original QTYPE. Only one query for a.b.example.org would have been needed if the original QTYPE would have been A. Using the most used QTYPE to hide the original QTYPE therefore slightly reduces the number of outgoing queries.

Warm cache with only org delegation known, (example.org's NS RRset is not known), request for MX record of a.b.example.org, using A QTYPE to hide the original QTYPE:

QTYPE	QNAME	TARGET	NOTE
A	example.org	org nameserver	
A	b.example.org	example.org nameserver	
A	a.b.example.org	example.org nameserver	"a" may be delegated
MX	a.b.example.org	example.org nameserver	

5. Performance Considerations

The main goal of QNAME minimisation is to improve privacy by sending less data. However, it may have other advantages. For instance, if a resolver sends a root name server queries for A.example followed by B.example followed by C.example, the result will be three NXDOMAINs, since .example does not exist in the root zone. When using QNAME minimisation, the resolver would send only one question (for .example itself) to which they could answer NXDOMAIN. The resolver can cache this answer and use it as to prove that nothing below .example exists ([RFC8020]). A resolver now knows a priori that neither B.example nor C.example exist. Thus, in this common case, the total number of upstream queries under QNAME minimisation could counterintuitively be less than the number of queries under the traditional iteration (as described in the DNS standard).

QNAME minimisation may also improve lookup performance for TLD operators. For a TLD that is delegation-only, a two-label QNAME query may be optimal for finding the delegation owner name, depending on the way domain matching is implemented.

QNAME minimisation can increase the number of queries based on the incoming QNAME. This is described in Section 2.3.

6. Security Considerations

QNAME minimisation's benefits are clear in the case where you want to decrease exposure to the authoritative name server. But minimising the amount of data sent also, in part, addresses the case of a wire sniffer as well as the case of privacy invasion by the servers. (Encryption is of course a better defense against wire sniffers, but, unlike QNAME minimisation, it changes the protocol and cannot be deployed unilaterally. Also, the effect of QNAME minimisation on wire sniffers depends on whether the sniffer is on the DNS path.)

QNAME minimisation offers no protection against the recursive resolver, which still sees the full request coming from the stub resolver.

A resolver using QNAME minimisation can possibly be used to cause a query storm to be sent to servers when resolving queries containing a QNAME with a large number of labels, as described in Section 2.3. That section proposes methods to significantly dampen the effects of such attacks.

7. References

7.1. 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>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <https://www.rfc-editor.org/info/rfc7816>.

[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>.

7.2. Informative References

[apnic-qnamemin]
Huston, G. and J. Damas, "Measuring Query Name Minimization", September 2020, <https://indico.dns-oarc.net/event/34/contributions/787/attachments/777/1326/2020-09-28-oarc33-qname-minimisation.pdf>.

[devries-qnamemin]
"A First Look at QNAME Minimization in the Domain Name System", March 2019, <https://nlnetlabs.nl/downloads/publications/devries2019.pdf>.

[dnsthought-qnamemin]
"DNSThought QNAME minimisation results. Using Atlas probes", March 2020, <https://dnsthought.nlnetlabs.nl/#qnamemin>.

[Huque-QNAME-Min]
Huque, S., "Query name minimization and authoritative server behavior", May 2015, <https://indico.dns-oarc.net/event/21/contribution/9>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[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>.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <https://www.rfc-editor.org/info/rfc7626>.

[RFC8020] Bortzmeyer, S. and S. Huque, "NXDOMAIN: There Really Is Nothing Underneath", RFC 8020, DOI 10.17487/RFC8020, November 2016, <https://www.rfc-editor.org/info/rfc8020>.

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <https://www.rfc-editor.org/info/rfc8198>.

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

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

[verisign-qnamemin] Thomas, M., "Maximizing Qname Minimization: A New Chapter in DNS Protocol Evolution", September 2020, <https://blog.verisign.com/security/maximizing-qname-minimization-a-new-chapter-in-dns-protocol-evolution/>.

Acknowledgments

The acknowledgements from RFC 7816 apply here. In addition, many participants from the DNSOP Working Group helped with proposals for simplification, clarification, and general editorial help.

Changes from RFC 7816

Changed in -07

* Stopped using the term "aggressive" for the method described
* Clarified some terminology
* More reorganization

Changed in -06

* Removed lots of text from when this was experimental
* Lots of reorganization

Changed in -04

* Start structure for implementation section
* Add clarification why the used QTYPE does not matter
* Make algorithm DS QTYPE compatible

Changed in -03

* Drop recommendation to use the NS QTYPE to hide the incoming QTYPE

* Describe DoS attach vector for QNAME with large number of labels, and propose a mitigation.

* Simplify examples and change qname to a.b.example.com to show the change in number of queries.

Changed in -00, -01, and -02

* Made changes to deal with errata #4644

* Changed status to be on standards track

* Major reorganization

Authors' Addresses

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

Phone: +33 1 39 30 83 46
Email: bortzmeyer+ietf@nic.fr
URI: https://www.afnic.fr/

Ralph Dolmans
NLnet Labs

Email: ralph@nlnetlabs.nl

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

DNSOP Working Group
Internet-Draft
Updates: 1034, 1035, 2181 (if approved)
Intended status: Standards Track
Expires: June 11, 2020

D. Lawrence
Oracle
W. Kumari
P. Sood
Google

December 09, 2019

Serving Stale Data to Improve DNS Resiliency
draft-ietf-dnsop-serve-stale-10

Abstract

This draft defines a method (serve-stale) for recursive resolvers to use stale DNS data to avoid outages when authoritative nameservers cannot be reached to refresh expired data. One of the motivations for serve-stale is to make the DNS more resilient to DoS attacks, and thereby make them less attractive as an attack vector. This document updates the definitions of TTL from RFC 1034 and RFC 1035 so that data can be kept in the cache beyond the TTL expiry, updates RFC 2181 by interpreting values with the high order bit set as being positive, rather than 0, and suggests a cap of 7 days.

Table of Contents

1. Introduction

Traditionally the Time To Live (TTL) of a DNS resource record has been understood to represent the maximum number of seconds that a record can be used before it must be discarded, based on its description and usage in [RFC1035] and clarifications in [RFC2181].

This document expands the definition of the TTL to explicitly allow for expired data to be used in the exceptional circumstance that a recursive resolver is unable to refresh the information. It is predicated on the observation that authoritative answer unavailability can cause outages even when the underlying data those servers would return is typically unchanged.

We describe a method below for this use of stale data, balancing the competing needs of resiliency and freshness.

This document updates the definitions of TTL from [RFC1034] and [RFC1035] so that data can be kept in the cache beyond the TTL expiry, and also updates [RFC2181] by interpreting values with the

high order bit set as being positive, rather than 0, and also suggests a cap of 7 days.

2. Terminology

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

For a glossary of DNS terms, please see [RFC8499].

3. Background

There are a number of reasons why an authoritative server may become unreachable, including Denial of Service (DoS) attacks, network issues, and so on. If a recursive server is unable to contact the authoritative servers for a query but still has relevant data that has aged past its TTL, that information can still be useful for generating an answer under the metaphorical assumption that "stale bread is better than no bread."

[RFC1035] Section 3.2.1 says that the TTL "specifies the time interval that the resource record may be cached before the source of the information should again be consulted", and Section 4.1.3 further says the TTL, "specifies the time interval (in seconds) that the resource record may be cached before it should be discarded."

A natural English interpretation of these remarks would seem to be clear enough that records past their TTL expiration must not be used. However, [RFC1035] predates the more rigorous terminology of [RFC2119] which softened the interpretation of "may" and "should".

[RFC2181] aimed to provide "the precise definition of the Time to Live", but in Section 8 was mostly concerned with the numeric range of values rather than data expiration behavior. It does, however, close that section by noting, "The TTL specifies a maximum time to live, not a mandatory time to live." This wording again does not contain BCP 14 [RFC2119] key words, but does convey the natural language connotation that data becomes unusable past TTL expiry.

As of the time of this writing, several large-scale operators use stale data for answers in some way. A number of recursive resolver packages, including BIND, Knot, OpenDNS, and Unbound, provide options to use stale data. Apple MacOS can also use stale data as part of the Happy Eyeballs algorithms in mDNSResponder. The collective

operational experience is that using stale data can provide significant benefit with minimal downside.

4. Standards Action

The definition of TTL in [RFC1035] Sections 3.2.1 and 4.1.3 is amended to read:

> TTL a 32-bit unsigned integer number of seconds that specifies the duration that the resource record MAY be cached before the source of the information MUST again be consulted. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached. Values SHOULD be capped on the orders of days to weeks, with a recommended cap of 604,800 seconds (seven days). If the data is unable to be authoritatively refreshed when the TTL expires, the record MAY be used as though it is unexpired. See [RFC Editor: replace by RFC number] Section 5 and Section 6 for details.

Interpreting values which have the high-order bit set as being positive, rather than 0, is a change from [RFC2181], the rationale for which is explained in Section 6. Suggesting a cap of seven days, rather than the 68 years allowed by [RFC2181], reflects the current practice of major modern DNS resolvers.

When returning a response containing stale records, a recursive resolver MUST set the TTL of each expired record in the message to a value greater than 0, with a RECOMMENDED value of 30 seconds. See Section 6 for explanation.

Answers from authoritative servers that have a DNS Response Code of either 0 (NoError) or 3 (NXDomain) and the Authoritative Answers (AA) bit set MUST be considered to have refreshed the data at the resolver. Answers from authoritative servers that have any other response code SHOULD be considered a failure to refresh the data and therefore leave any previous state intact. See Section 6 for a discussion.

5. Example Method

There is more than one way a recursive resolver could responsibly implement this resiliency feature while still respecting the intent of the TTL as a signal for when data is to be refreshed.

In this example method four notable timers drive considerations for the use of stale data:

- A client response timer, which is the maximum amount of time a recursive resolver should allow between the receipt of a resolution request and sending its response.

- A query resolution timer, which caps the total amount of time a recursive resolver spends processing the query.

- A failure recheck timer, which limits the frequency at which a failed lookup will be attempted again.

- A maximum stale timer, which caps the amount of time that records will be kept past their expiration.

Most recursive resolvers already have the query resolution timer, and effectively some kind of failure recheck timer. The client response timer and maximum stale timer are new concepts for this mechanism.

When a recursive resolver receives a request, it should start the client response timer. This timer is used to avoid client timeouts. It should be configurable, with a recommended value of 1.8 seconds as being just under a common timeout value of 2 seconds while still giving the resolver a fair shot at resolving the name.

The resolver then checks its cache for any unexpired records that satisfy the request and returns them if available. If it finds no relevant unexpired data and the Recursion Desired flag is not set in the request, it should immediately return the response without consulting the cache for expired records. Typically this response would be a referral to authoritative nameservers covering the zone, but the specifics are implementation-dependent.

If iterative lookups will be done, then the failure recheck timer is consulted. Attempts to refresh from non-responsive or otherwise failing authoritative nameservers are recommended to be done no more frequently than every 30 seconds. If this request was received within this period, the cache may be immediately consulted for stale data to satisfy the request.

Outside the period of the failure recheck timer, the resolver should start the query resolution timer and begin the iterative resolution process. This timer bounds the work done by the resolver when contacting external authorities, and is commonly around 10 to 30 seconds. If this timer expires on an attempted lookup that is still being processed, the resolution effort is abandoned.

If the answer has not been completely determined by the time the client response timer has elapsed, the resolver should then check its cache to see whether there is expired data that would satisfy the

request. If so, it adds that data to the response message with a TTL greater than 0 (as specified in Section 4). The response is then sent to the client while the resolver continues its attempt to refresh the data.

When no authorities are able to be reached during a resolution attempt, the resolver should attempt to refresh the delegation and restart the iterative lookup process with the remaining time on the query resolution timer. This resumption should be done only once per resolution effort.

Outside the resolution process, the maximum stale timer is used for cache management and is independent of the query resolution process. This timer is conceptually different from the maximum cache TTL that exists in many resolvers, the latter being a clamp on the value of TTLs as received from authoritative servers and recommended to be seven days in the TTL definition in Section 4. The maximum stale timer should be configurable, and defines the length of time after a record expires that it should be retained in the cache. The suggested value is between 1 and 3 days.

6. Implementation Considerations

This document mainly describes the issues behind serving stale data and intentionally does not provide a formal algorithm. The concept is not overly complex, and the details are best left to resolver authors to implement in their codebases. The processing of serve-stale is a local operation, and consistent variables between deployments are not needed for interoperability. However, we would like to highlight the impact of various implementation choices, starting with the timers involved.

The most obvious of these is the maximum stale timer. If this variable is too large it could cause excessive cache memory usage, but if it is too small, the serve-stale technique becomes less effective, as the record may not be in the cache to be used if needed. Shorter values, even less than a day, can effectively handle the vast majority of outages. Longer values, as much as a week, give time for monitoring systems to notice a resolution problem and for human intervention to fix it; operational experience has been that sometimes the right people can be hard to track down and unfortunately slow to remedy the situation.

Increased memory consumption could be mitigated by prioritizing removal of stale records over non-expired records during cache exhaustion. Implementations may also wish to consider whether to track the names in requests for their last time of use or their popularity, using that as an additional factor when considering cache

eviction. A feature to manually flush only stale records could also be useful.

The client response timer is another variable which deserves consideration. If this value is too short, there exists the risk that stale answers may be used even when the authoritative server is actually reachable but slow; this may result in undesirable answers being returned. Conversely, waiting too long will negatively impact user experience.

The balance for the failure recheck timer is responsiveness in detecting the renewed availability of authorities versus the extra resource use for resolution. If this variable is set too large, stale answers may continue to be returned even after the authoritative server is reachable; per [RFC2308], Section 7, this should be no more than five minutes. If this variable is too small, authoritative servers may be targeted with a significant amount of excess traffic.

Regarding the TTL to set on stale records in the response, historically TTLs of zero seconds have been problematic for some implementations, and negative values can't effectively be communicated to existing software. Other very short TTLs could lead to congestive collapse as TTL-respecting clients rapidly try to refresh. The recommended value of 30 seconds not only sidesteps those potential problems with no practical negative consequences, it also rate limits further queries from any client that honors the TTL, such as a forwarding resolver.

As for the change to treat a TTL with the high-order bit set as positive and then clamping it, as opposed to [RFC2181] treating it as zero, the rationale here is basically one of engineering simplicity versus an inconsequential operational history. Negative TTLs had no rational intentional meaning that wouldn't have been satisfied by just sending 0 instead, and similarly there was realistically no practical purpose for sending TTLs of 2^25 seconds (1 year) or more. There's also no record of TTLs in the wild having the most significant bit set in DNS-OARC's "Day in the Life" samples [DITL]. With no apparent reason for operators to use them intentionally, that leaves either errors or non-standard experiments as explanations as to why such TTLs might be encountered, with neither providing an obviously compelling reason as to why having the leading bit set should be treated differently from having any of the next eleven bits set and then capped per Section 4.

Another implementation consideration is the use of stale nameserver addresses for lookups. This is mentioned explicitly because, in some resolvers, getting the addresses for nameservers is a separate path

from a normal cache lookup. If authoritative server addresses are not able to be refreshed, resolution can possibly still be successful if the authoritative servers themselves are up. For instance, consider an attack on a top-level domain that takes its nameservers offline; serve-stale resolvers that had expired glue addresses for subdomains within that TLD would still be able to resolve names within those subdomains, even those it had not previously looked up.

The directive in Section 4 that only NoError and NXDomain responses should invalidate any previously associated answer stems from the fact that no other RCODEs that a resolver normally encounters make any assertions regarding the name in the question or any data associated with it. This comports with existing resolver behavior where a failed lookup (say, during pre-fetching) doesn't impact the existing cache state. Some authoritative server operators have said that they would prefer stale answers to be used in the event that their servers are responding with errors like ServFail instead of giving true authoritative answers. Implementers MAY decide to return stale answers in this situation.

Since the goal of serve-stale is to provide resiliency for all obvious errors to refresh data, these other RCODEs are treated as though they are equivalent to not getting an authoritative response. Although NXDomain for a previously existing name might well be an error, it is not handled that way because there is no effective way to distinguish operator intent for legitimate cases versus error cases.

During discussion in the IETF, it was suggested that, if all authorities return responses with RCODE of Refused, it may be an explicit signal to take down the zone from servers that still have the zone's delegation pointed to them. Refused, however, is also overloaded to mean multiple possible failures which could represent transient configuration failures. Operational experience has shown that purposely returning Refused is a poor way to achieve an explicit takedown of a zone compared to either updating the delegation or returning NXDomain with a suitable SOA for extended negative caching. Implementers MAY nonetheless consider whether to treat all authorities returning Refused as preempting the use of stale data.

7. Implementation Caveats

Stale data is used only when refreshing has failed in order to adhere to the original intent of the design of the DNS and the behaviour expected by operators. If stale data were to always be used immediately and then a cache refresh attempted after the client response has been sent, the resolver would frequently be sending data that it would have had no trouble refreshing. Because modern

resolvers use techniques like pre-fetching and request coalescing for efficiency, it is not necessary that every client request needs to trigger a new lookup flow in the presence of stale data, but rather that a good-faith effort has been recently made to refresh the stale data before it is delivered to any client.

It is important to continue the resolution attempt after the stale response has been sent, until the query resolution timeout, because some pathological resolutions can take many seconds to succeed as they cope with unavailable servers, bad networks, and other problems. Stopping the resolution attempt when the response with expired data has been sent would mean that answers in these pathological cases would never be refreshed.

The continuing prohibition against using data with a 0 second TTL beyond the current transaction explicitly extends to it being unusable even for stale fallback, as it is not to be cached at all.

Be aware that Canonical Name (CNAME) and DNAME [RFC6672] records mingled in the expired cache with other records at the same owner name can cause surprising results. This was observed with an initial implementation in BIND when a hostname changed from having an IPv4 Address (A) record to a CNAME. The version of BIND being used did not evict other types in the cache when a CNAME was received, which in normal operations is not a significant issue. However, after both records expired and the authorities became unavailable, the fallback to stale answers returned the older A instead of the newer CNAME.

8. Implementation Status

The algorithm described in Section 5 was originally implemented as a patch to BIND 9.7.0. It has been in use on Akamai's production network since 2011, and effectively smoothed over transient failures and longer outages that would have resulted in major incidents. The patch was contributed to Internet Systems Consortium and the functionality is now available in BIND 9.12 and later via the options stale-answer-enable, stale-answer-ttl, and max-stale-ttl.

Unbound has a similar feature for serving stale answers, and will respond with stale data immediately if it has recently tried and failed to refresh the answer by pre-fetching.

Knot Resolver has a demo module here: https://knot-resolver.readthedocs.io/en/stable/modules.html#serve-stale

Apple's system resolvers are also known to use stale answers, but the details are not readily available.

In the research paper "When the Dike Breaks: Dissecting DNS Defenses During DDoS" [DikeBreaks], the authors detected some use of stale answers by resolvers when authorities came under attack. Their research results suggest that more widespread adoption of the technique would significantly improve resiliency for the large number of requests that fail or experience abnormally long resolution times during an attack.

9. EDNS Option

During the discussion of serve-stale in the IETF, it was suggested that an EDNS option should be available to either explicitly opt-in to getting data that is possibly stale, or at least as a debugging tool to indicate when stale data has been used for a response.

The opt-in use case was rejected as the technique was meant to be immediately useful in improving DNS resiliency for all clients.

The reporting case was ultimately also rejected because even the simpler version of a proposed option was still too much bother to implement for too little perceived value.

10. Security Considerations

The most obvious security issue is the increased likelihood of DNSSEC validation failures when using stale data because signatures could be returned outside their validity period. Stale negative records can increase the time window where newly published TLSA or DS RRs may not be used due to cached NSEC or NSEC3 records. These scenarios would only be an issue if the authoritative servers are unreachable, the only time the techniques in this document are used, and thus does not introduce a new failure in place of what would have otherwise been success.

Additionally, bad actors have been known to use DNS caches to keep records alive even after their authorities have gone away. The serve stale feature potentially makes the attack easier, although without introducing a new risk. In addition, attackers could combine this with a DDoS attack on authoritative servers with the explicit intent of having stale information cached for longer. But if attackers have this capacity, they probably could do much worse than prolonging the life of old data.

In [CloudStrife], it was demonstrated how stale DNS data, namely hostnames pointing to addresses that are no longer in use by the owner of the name, can be used to co-opt security such as to get domain-validated certificates fraudulently issued to an attacker. While this document does not create a new vulnerability in this area,

it does potentially enlarge the window in which such an attack could be made. A proposed mitigation is that certificate authorities should fully look up each name starting at the DNS root for every name lookup. Alternatively, CAs should use a resolver that is not serving stale data.

11. Privacy Considerations

This document does not add any practical new privacy issues.

12. NAT Considerations

The method described here is not affected by the use of NAT devices.

13. IANA Considerations

There are no IANA considerations.

14. Acknowledgements

The authors wish to thank Brian Carpenter, Robert Edmonds, Tony Finch, Bob Harold, Tatuya Jinmei, Matti Klock, Jason Moreau, Giovane Moura, Jean Roy, Mukund Sivaraman, Davey Song, Paul Vixie, Ralf Weber and Paul Wouters for their review and feedback. Paul Hoffman deserves special thanks for submitting a number of Pull Requests.

Thank you also to the following members of the IESG for their final review: Roman Danyliw, Benjamin Kaduk, Suresh Krishnan, Mirja Kuehlewind, and Adam Roach.

15. References

15.1. 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>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[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>.

15.2. Informative References

[CloudStrife] Borgolte, K., Fiebig, T., Hao, S., Kruegel, C., and G. Vigna, "Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates", ACM 2018 Applied Networking Research Workshop, DOI 10.1145/3232755.3232859, July 2018, <https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_06A-4_Borgolte_paper.pdf>.

[DikeBreaks] Moura, G., Heidemann, J., Mueller, M., Schmidt, R., and M. Davids, "When the Dike Breaks: Dissecting DNS Defenses During DDos", ACM 2018 Internet Measurement Conference, DOI 10.1145/3278532.3278534, October 2018, <https://www.isi.edu/~johnh/PAPERS/Moura18b.pdf>.

[DITL] "DITL Traces and Analysis | DNS-OARC", n.d., <https://www.dns-oarc.net/oarc/data/ditl>.

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

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

Authors' Addresses

David C Lawrence
Oracle

Email: tale@dd.org

Warren "Ace" Kumari
Google
1600 Amphitheatre Parkway
Mountain View CA 94043
USA

Email: warren@kumari.net

Puneet Sood
Google

Email: puneets@google.com

DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2019

L. Lhotka
P. Spacek
CZ.NIC

June 27, 2019

YANG Types for DNS Classes and Resource Record Types
draft-lhotka-dnsop-iana-class-type-yang-02

Abstract

This document contains the initial revision of the YANG module iana-dns-class-rr-type that contains derived types reflecting two IANA registries: DNS CLASSes and Resource Record (RR) TYPEs. These YANG types are intended as a minimum basis for future data modeling work.

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 December 29, 2019.

Table of Contents

1. Introduction

YANG [RFC7950] has become a de facto standard as a language for modeling configuration and state data, as well as specifying management operations and asynchronous notifications. It is reasonable to expect that the approach based on utilizing such data models along with standard management protocols such as NETCONF and RESTCONF can be effectively used in DNS operations, too. In fact, several efforts are currently underway that attempt to use NETCONF or RESTCONF for configuring and managing

- authoritative servers
- resolvers
- zone data.

While it is possible to use the management protocols mentioned above with ad hoc or proprietary data models, their real potential can be realized only if there is a (completely or partly) unified data model supported by multiple DNS software implementations. Operators can then, for instance, run several different DNS servers in parallel, and use a common configuration and management interface and data for all of them. Also, it becomes considerably easier to migrate to another implementation.

Based on the previous experience from the IETF Routing Area, it is to be expected that the development of unified data models for DNS will be a lengthy and complicated process that will require active cooperation and compromises from the vendors and developers of major DNS server platforms. Nevertheless, it is likely that any DNS-related data modeling effort will need to use various DNS parameters and enumerations that are specified in several IANA registries. For use with YANG, these parameters and enumerations have to be

translated into corresponding YANG types or other structures. Such translations should be straightforward and relatively uncontroversial.

This document is a first step in translating DNS-related IANA registries to YANG. It contains the initial revision of the YANG module "iana-dns-class-rr-type" that defines derived types for the common parameters of DNS resource records (RR): class and type. These YANG types, "dns-class" and "rr-type", reflect the IANA registries "DNS CLASSes" and "Resource Record (RR) TYPEs" [IANA-DNS-PARAMETERS].

It is worth emphasizing that the role of the DNSOP Working Group is only in preparing and publishing this initial revision of the YANG module. Subsequently, whenever a new class or RR type is added to the above registries, IANA will also update the iana-dns-class-rr-type YANG module, following the instructions in Section 4 below.

2. YANG Design Considerations

The IANA document "Domain Name System (DNS) Parameters" [IANA-DNS-PARAMETERS] contains altogether thirteen registries. The YANG module iana-dns-class-rr-type defines derived types corresponding to only two of the registries that are essential for data models involving zone data, namely "DNS CLASSes" and "Resource Record (RR) TYPEs". It is expected that the remaining registries in [IANA-DNS-PARAMETERS], as well as other DNS-related IANA registries, will be analogically reflected in future YANG modules as necessary. This way, an appropriate combination of YANG modules can be chosen depending on which YANG types are needed for a given data modeling purpose.

[RFC3597] introduced the option of specifying a class or type via its assigned decimal number, as an alternative to the mnemonic name. For example, the "IN" class can be equivalently written as "CLASS1", and "AAAA" type as "TYPE28".

Accordingly, the derived types "dns-class" and "rr-type" are defined in the YANG module as a union of two member types:

- 16-bit decimal integer ("uint16")
- mnemonic name, represented by the enumeration type "dns-class-name" and "rr-type-name", respectively.

As unassigned and reserved class and types values are not included in the mnemonic name enumerations, they can be used only via their decimal codes.

3. YANG Module

RFC Editor: In this section, replace all occurrences of "XXXX" with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

<CODE BEGINS> file "ietf-iana-dns-class-rr-type@2019-06-27.yang"

```
module iana-dns-class-rr-type {

  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type";

  prefix dnsct;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "        Internet Assigned Numbers Authority

     Postal: ICANN
             4676 Admiralty Way, Suite 330
             Marina del Rey, CA 90292

     Tel:    +1 310 823 9358

     <mailto:iana@iana.org>";

  description
    "This YANG module translates IANA registries 'DNS CLASSes' and
     'Resource Record (RR) TYPEs' to YANG derived types.

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

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";
```

```
  reference
    "IANA 'Domain Name System (DNS) Parameters' registry
     https://www.iana.org/assignments/dns-parameters";

  revision 2019-06-27 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Types for DNS Classes and Resource Record
       Types";
  }

  /* Typedefs */

  typedef dns-class-name {
    type enumeration {
      enum IN {
        value "1";
        description
          "Internet";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum CH {
        value "3";
        description
          "Chaos";
        reference
          "Moon, D., 'Chaosnet', A. I. Memo 628, MIT Artificial
           Intelligence Laboratory, June 1981";
      }
      enum HS {
        value "4";
        description
          "Hesiod";
        reference
          "Dyer, S. and Hsu, F, 'Hesiod', Project Athena Technical
           Plan - Name Service, April 1987";
      }
      enum NONE {
        value "254";
        description
          "QCLASS NONE";
        reference
          "RFC 2136: Dynamic Updates in the Domain Name System (DNS
           UPDATE)";
      }
```

```
        enum ANY {
          value "255";
          description
            "QCLASS * (ANY)";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
      }
      description
        "This enumeration type defines mnemonic names and corresponding
         numeric values of DNS classes.";
      reference
        "RFC 6895: Domain Name System (DNS) IANA Considerations";
    }

    typedef dns-class {
      type union {
        type uint16;
        type dns-class-name;
      }
      description
        "This type allows for referring to a DNS class using either the
         assigned mnemonic name or numeric value.";
    }

    typedef rr-type-name {
      type enumeration {
        enum A {
          value "1";
          description
            "A host address.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum NS {
          value "2";
          description
            "An authoritative name server.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MD {
          value "3";
          status "obsolete";
          description
```

```
            "A mail destination (obsolete - use MX).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MF {
          value "4";
          status "obsolete";
          description
            "A mail forwarder (obsolete - use MX).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum CNAME {
          value "5";
          description
            "The canonical name for an alias.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum SOA {
          value "6";
          description
            "Start of a zone of authority.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MB {
          value "7";
          description
            "A mailbox domain name (experimental).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MG {
          value "8";
          description
            "A mail group member (experimental).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MR {
          value "9";
```

```
        description
          "A mail rename domain name (experimental).";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum NULL {
        value "10";
        description
          "A null RR (experimental).";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum WKS {
        value "11";
        description
          "A well known service description.";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum PTR {
        value "12";
        description
          "A domain name pointer.";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum HINFO {
        value "13";
        description
          "Host information.";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum MINFO {
        value "14";
        description
          "Mailbox or mail list information.";
        reference
          "RFC 1035: Domain Names - Implementation and
           Specification";
      }
      enum MX {
        value "15";
```

```
          description
            "Mail exchange.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum TXT {
          value "16";
          description
            "Text strings.";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum RP {
          value "17";
          description
            "Responsible person.";
          reference
            "RFC 1183: New DNS RR Definitions";
        }
        enum AFSDB {
          value "18";
          description
            "AFS data base location.";
          reference
            "- RFC 1183: New DNS RR Definitions

             - RFC 5864: DNS SRV Resource Records for AFS";
        }
        enum X25 {
          value "19";
          description
            "X.25 PSDN address.";
          reference
            "RFC 1183: New DNS RR Definitions";
        }
        enum ISDN {
          value "20";
          description
            "ISDN address.";
          reference
            "RFC 1183: New DNS RR Definitions";
        }
        enum RT {
          value "21";
          description
            "Route through.";
```

```
          reference
            "RFC 1183: New DNS RR Definitions";
        }
        enum NSAP {
          value "22";
          description
            "NSAP address, NSAP style A record.";
          reference
            "RFC 1706: DNS NSAP Resource Records";
        }
        enum NSAP-PTR {
          value "23";
          description
            "Domain name pointer, NSAP style.";
          reference
            "- RFC 1348: DNS NSAP RRs

             - RFC 1637: DNS NSAP Resource Records

             - RFC 1706: DNS NSAP Resource Records";
        }
        enum SIG {
          value "24";
          description
            "Security signature.";
          reference
            "- RFC 4034: Resource Records for the DNS Security
               Extensions

             - RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)

             - RFC 2535: Domain Name System Security Extensions

             - RFC 2536: DSA KEYs and SIGs in the Domain Name System
               (DNS)

             - RFC 2537: RSA/MD5 KEYs and SIGs in the Domain Name
               System (DNS)

             - RFC 2931: DNS Request and Transaction Signatures
               (SIG(0)s)

             - RFC 3110: RSA/SHA-1 SIGs and RSA KEYs in the Domain Name
               System (DNS)

             - RFC 3008: Domain Name System Security (DNSSEC) Signing
               Authority";
```

```
        }
        enum KEY {
          value "25";
          description
            "Security key.";
          reference
            "- RFC 4034: Resource Records for the DNS Security
               Extensions

             - RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)

             - RFC 2535: Domain Name System Security Extensions

             - RFC 2536: DSA KEYs and SIGs in the Domain Name System
               (DNS)

             - RFC 2537: RSA/MD5 KEYs and SIGs in the Domain Name
               System (DNS)

             - RFC 2539: Storage of Diffie-Hellman Keys in the Domain
               Name System (DNS)

             - RFC 3008: Domain Name System Security (DNSSEC) Signing
               Authority

             - RFC 3110: RSA/SHA-1 SIGs and RSA KEYs in the Domain Name
               System (DNS)";
        }
        enum PX {
          value "26";
          description
            "X.400 mail mapping information.";
          reference
            "RFC 2163: Using the Internet DNS to Distribute MIXER
             Conformant Global Address Mapping (MCGAM)";
        }
        enum GPOS {
          value "27";
          description
            "Geographical position.";
          reference
            "RFC 1712: DNS Encoding of Geographical Location";
        }
        enum AAAA {
          value "28";
          description
            "IPv6 address.";
```

```
          reference
            "RFC 3596: DNS Extensions to Support IP Version 6";
        }
        enum LOC {
          value "29";
          description
            "Location information.";
          reference
            "RFC 1876: A Means for Expressing Location Information in
             the Domain Name System";
        }
        enum NXT {
          value "30";
          status "obsolete";
          description
            "Next domain (obsolete).";
          reference
            "- RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)

             - RFC 2535: Domain Name System Security Extensions";
        }
        enum EID {
          value "31";
          description
            "Endpoint identifier.";
        }
        enum NIMLOC {
          value "32";
          description
            "Nimrod locator.";
        }
        enum SRV {
          value "33";
          description
            "Server selection.";
          reference
            "RFC 2782: A DNS RR for specifying the location of services
             (DNS SRV)";
        }
        enum ATMA {
          value "34";
          description
            "ATM address.";
          reference
            "ATM Forum Technical Committee, 'ATM Name System V2.0',
             AF-DANS-0152.00, July 2000";
        }
```

```
        enum NAPTR {
          value "35";
          description
            "Naming authority pointer.";
          reference
            "- RFC 2915: The Naming Authority Pointer (NAPTR) DNS
               Resource Record

             - RFC 2168: Resolution of Uniform Resource Identifiers
               using the Domain Name System

             - RFC 3403: Dynamic Delegation Discovery System (DDDS)
               Part Three: The Domain Name System (DNS) Database";
        }
        enum KX {
          value "36";
          description
            "Key exchanger.";
          reference
            "RFC 2230: Key Exchange Delegation Record for the DNS";
        }
        enum CERT {
          value "37";
          description
            "Certificate.";
          reference
            "RFC 4398: Storing Certificates in the Domain Name System
             (DNS)";
        }
        enum A6 {
          value "38";
          status "obsolete";
          description
            "IPv6 address (obsolete - use AAAA).";
          reference
            "- RFC 3226: DNSSEC and IPv6 A6 Aware Server/Resolver
               Message Size Requirements

             - RFC 2874: DNS Extensions to Support IPv6 Address
               Aggregation and Renumbering

             - RFC 6563: Moving A6 to Historic Status";
        }
        enum DNAME {
          value "39";
          description
            "DNAME.";
          reference
```

```
            "- RFC 2672: Non-Terminal DNS Name Redirection

             - RFC 6672: DNAME Redirection in the DNS";
          }
          enum SINK {
            value "40";
            description
              "Kitchen sink.";
          }
          enum OPT {
            value "41";
            description
              "OPT pseudo-RR.";
            reference
              "- RFC 6891: Extension Mechanisms for DNS (EDNS(0))

               - RFC 3225: Indicating Resolver Support of DNSSEC";
          }
          enum APL {
            value "42";
            description
              "Address prefix list.";
            reference
              "RFC 3123: A DNS RR Type for Lists of Address Prefixes (APL
               RR)";
          }
          enum DS {
            value "43";
            description
              "Delegation signer.";
            reference
              "- RFC 4034: Resource Records for the DNS Security
                 Extensions

               - RFC 3658: Delegation Signer (DS) Resource Record (RR)";
          }
          enum SSHFP {
            value "44";
            description
              "SSH key fingerprint.";
            reference
              "RFC 4255: Using DNS to Securely Publish Secure Shell (SSH)
               Key Fingerprints";
          }
          enum IPSECKEY {
            value "45";
            description
              "IPSec key.";
```

```
          reference
            "RFC 4025: A Method for Storing IPsec Keying Material in
             DNS";
        }
        enum RRSIG {
          value "46";
          description
            "RR signature.";
          reference
            "- RFC 4034: Resource Records for the DNS Security
               Extensions

             - RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)";
        }
        enum NSEC {
          value "47";
          description
            "NSEC resource record.";
          reference
            "- RFC 4034: Resource Records for the DNS Security
               Extensions

             - RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)";
        }
        enum DNSKEY {
          value "48";
          description
            "DNSKEY resource record.";
          reference
            "- RFC 4034: Resource Records for the DNS Security
               Extensions

             - RFC 3755: Legacy Resolver Compatibility for Delegation
               Signer (DS)";
        }
        enum DHCID {
          value "49";
          description
            "DHCID resource record.";
          reference
            "RFC 4701: A DNS Resource Record (RR) for Encoding Dynamic
             Host Configuration Protocol (DHCP) Information (DHCID
             RR)";
        }
        enum NSEC3 {
          value "50";
```

```
      description
        "NSEC3 resource record.";
      reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated
         Denial of Existence";
    }
    enum NSEC3PARAM {
      value "51";
      description
        "NSEC3PARAM resource record.";
      reference
        "RFC 5155: DNS Security (DNSSEC) Hashed Authenticated
         Denial of Existence";
    }
    enum TLSA {
      value "52";
      description
        "TLSA resource record.";
      reference
        "RFC 6698: The DNS-Based Authentication of Named Entities
         (DANE) Transport Layer Security (TLS) Protocol: TLSA";
    }
    enum SMIMEA {
      value "53";
      description
        "S/MIME cert association";
      reference
        "RFC 8162: Using Secure DNS to Associate Certificates with
         Domain Names for S/MIME";
    }
    enum HIP {
      value "55";
      description
        "Host identity protocol.";
      reference
        "RFC 5205: Host Identity Protocol (HIP) Domain Name System
         (DNS) Extension";
    }
    enum NINFO {
      value "56";
      description
        "NINFO resource record.";
    }
    enum RKEY {
      value "57";
      description
        "RKEY resource record.";
    }
```

```
        enum TALINK {
          value "58";
          description
            "Trust anchor LINK.";
        }
        enum CDS {
          value "59";
          description
            "Child DS.";
          reference
            "RFC 7344: Automating DNSSEC Delegation Trust
             Maintenance";
        }
        enum CDNSKEY {
          value "60";
          description
            "DNSKEY(s) the child wants reflected in DS.";
          reference
            "RFC 7344: Automating DNSSEC Delegation Trust
             Maintenance";
        }
        enum OPENPGPKEY {
          value "61";
          description
            "OpenPGP key.";
          reference
            "RFC 7929: DNS-Based Authentication of Named Entities
             (DANE) Bindings for OpenPGP";
        }
        enum CSYNC {
          value "62";
          description
            "Child-to-parent synchronization.";
          reference
            "RFC 7477: Child-to-Parent Synchronization in DNS";
        }
        enum SPF {
          value "99";
          description
            "SPF (sender policy framework) resource record.";
          reference
            "RFC 7208: Sender Policy Framework (SPF) for Authorizing
             Use of Domains in Email, Version 1";
        }
        enum UINFO {
          value "100";
          description
            "IANA-reserved.";
```

```
        }
        enum UID {
          value "101";
          description
            "IANA-reserved.";
        }
        enum GID {
          value "102";
          description
            "IANA-reserved.";
        }
        enum UNSPEC {
          value "103";
          description
            "IANA-reserved.";
        }
        enum NID {
          value "104";
          description
            "Node identifier.";
          reference
            "RFC 6742: DNS Resource Records for the Identifier-Locator
             Network Protocol (ILNP)";
        }
        enum L32 {
          value "105";
          description
            "L32 resource record.";
          reference
            "RFC 6742: DNS Resource Records for the Identifier-Locator
             Network Protocol (ILNP)";
        }
        enum L64 {
          value "106";
          description
            "L64 resource record.";
          reference
            "RFC 6742: DNS Resource Records for the Identifier-Locator
             Network Protocol (ILNP)";
        }
        enum LP {
          value "107";
          description
            "LP resource record.";
          reference
            "RFC 6742: DNS Resource Records for the Identifier-Locator
             Network Protocol (ILNP)";
        }
```

```
        enum EUI48 {
          value "108";
          description
            "An EUI-48 address.";
          reference
            "RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses
             in the DNS";
        }
        enum EUI64 {
          value "109";
          description
            "An EUI-64 address.";
          reference
            "RFC 7043: Resource Records for EUI-48 and EUI-64 Addresses
             in the DNS";
        }
        enum TKEY {
          value "249";
          description
            "Transaction key.";
          reference
            "RFC 2930: Secret Key Establishment for DNS (TKEY RR)";
        }
        enum TSIG {
          value "250";
          description
            "Transaction signature.";
          reference
            "RFC 2845: Secret Key Transaction Authentication for DNS
             (TSIG)";
        }
        enum IXFR {
          value "251";
          description
            "Incremental transfer.";
          reference
            "RFC 1995: Incremental Zone Transfer in DNS";
        }
        enum AXFR {
          value "252";
          description
            "Transfer of an entire zone.";
          reference
            "- RFC 1035: Domain Names - Implementation and
               Specification

             - RFC 5936: DNS Zone Transfer Protocol (AXFR)";
        }
```

```
        enum MAILB {
          value "253";
          description
            "Mailbox-related RRs (MB, MG or MR).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum MAILA {
          value "254";
          status "obsolete";
          description
            "Mail agent RRs (obsolete - see MX).";
          reference
            "RFC 1035: Domain Names - Implementation and
             Specification";
        }
        enum * {
          value "255";
          description
            "A request for all records the server/cache has
             available.";
          reference
            "- RFC 1035: Domain Names - Implementation and
               Specification

             - RFC 6895: Domain Name System (DNS) IANA
               Considerations";
        }
        enum URI {
          value "256";
          description
            "URI resource record.";
          reference
            "RFC 7553: The Uniform Resource Identifier (URI) DNS
             Resource Record";
        }
        enum CAA {
          value "257";
          description
            "Certification authority authorization.";
          reference
            "RFC 6844: DNS Certification Authority Authorization (CAA)
             Resource Record";
        }
        enum AVC {
          value "258";
          description
```

```
            "Application visibility and control.";
          }
          enum DOA {
            value "259";
            description
              "Digital object architecture";
            reference
              "draft-durand-doa-over-dns: DOA over DNS";
          }
          enum AMTRELAY {
            value "260";
            description
              "Automatic multicast tunneling relay.";
            reference
              "J. Holland: DNS Reverse IP AMT Discovery.
               draft-ietf-mboned-driad-amt-discovery.";
          }
          enum TA {
            value "32768";
            description
              "DNSSEC trust authorities.";
          }
          enum DLV {
            value "32769";
            description
              "DNSSEC lookaside validation.";
            reference
              "RFC 4431: The DNSSEC Lookaside Validation (DLV) DNS
               Resource Record";
          }
        }
        description
          "This enumeration type defines mnemonic names and corresponding
           numeric values of DNS resource record types.";
        reference
          "- RFC 6895: Domain Name System (DNS) IANA Considerations

           - RFC 1035: Domain Names - Implementation and Specification";
      }

      typedef rr-type {
        type union {
          type uint16;
          type rr-type-name;
        }
        description
          "This type allows for referring to a DNS resource record type
           using either the assigned mnemonic name or numeric value.";
```

```
    }
  }

  <CODE ENDS>
```

4. IANA Considerations

RFC Editor: In this section, replace all occurrences of "XXXX" with the actual RFC number (and remove this note).

This document defines the initial version of the IANA-maintained iana-dns-class-rr-type YANG module.

The iana-dns-class-rr-type YANG module is intended to reflect the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries in [IANA-DNS-PARAMETERS].

IANA has added this new note to the "iana-dns-class-rr-type YANG Module" registry:

> Classes and types of DNS resource records must not be directly added to the iana-dns-class-rr-type YANG module. They must instead be added to the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries, respectively.

When a new DNS class or RR type is added to the "DNS CLASSes" or "Resource Record (RR) TYPEs" registry, a new "enum" statement must be added to the "dns-class-name" or "rr-type-name" type, respectively. The assigned name defined by the "enum" statement is the same as the mnemonic name of the new class or type. The following substatements to the "enum" statement should be defined:

"value": Use the decimal value from the registry.

"status": Include only if a class or type registration has been deprecated (use the value "deprecated") or obsoleted (use the value "obsolete").

"description": Replicate the corresponding information from the registry, namely the full name of the new DNS class, or the meaning of the new RR type, if any.

"reference": Replicate the reference from the registry, if any, and add the title of the document, if applicable.

Unassigned or reserved values are not included in the "dns-class-name" and "rr-type-name" enumeration types.

Each time the iana-dns-class-rr-type YANG module is updated, a new "revision" statement must be added before the existing "revision" statements.

IANA has added this new note to the "DNS CLASSes" and "Resource Record (RR) TYPEs" registries:

> When this registry is modified, the YANG module iana-dns-class-rr-type must be updated as defined in RFC XXXX.

The "Reference" text in the "DNS CLASSes" registry has been updated as follows:

OLD:
[RFC6895]

NEW:
[RFC6895][RFCXXXX]

The "Reference" text in the "Resource Record (RR) TYPEs" registry has been updated as follows:

OLD:
[RFC6895][RFC1035]

NEW:
[RFC6895][RFC1035][RFCXXXX]

4.1. URI Registrations

This document registers a URI in the "IETF XML Registry" [RFC3688]. The following registration has been made:

URI: urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

4.2. YANG Module Registrations

This document registers a YANG module in the "YANG Module Names" registry [RFC6020]. The following registration has been made:

```
name:      iana-dns-class-rr-type
namespace: urn:ietf:params:xml:ns:yang:iana-dns-class-rr-type
prefix:    dnsct
reference: RFC XXXX
```

5. Security Considerations

This documents translates two IANA registries into YANG data types and otherwise introduces no technology or protocol. Consequently, there are no security issues to be considered for this document.

6. References

6.1. Normative References

[IANA-DNS-PARAMETERS]
Internet Assigned Numbers Authority, "Domain Name System (DNS) Parameters", January 2018, <https://www.iana.org/assignments/dns-parameters>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

6.2. Informative References

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

Authors' Addresses

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Petr Spacek
CZ.NIC

Email: petr.spacek@nic.cz

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 13, 2021

A. Mayrhofer
nic.at GmbH
D. Klesev

M. Sabadello
Danube Tech GmbH

September 9, 2020

The Decentralized Identifier (DID) in the DNS
draft-mayrhofer-did-dns-04

Abstract

This document specifies the use of the URI Resource Record Type to publish Decentralized Identifiers (DIDs) in the DNS.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 13, 2021.

Table of Contents

1. Introduction

Decentralized Identifiers (DIDs) [W3C-DID] use a Uniform Resource Identifier (URI) scheme [RFC3986] to identify persons, organizations, or things in decentralized infrastructure, such as blockchains and distributed ledgers.

DIDs are structured around "methods", each method defining the syntax of the "method specific identifier" and the operations on the respective DIDs (See Section 3.2 of [W3C-DID] and [DID-METHODS]). For many methods, the method specific identifier is not human-friendly (such as hash values, referring to transactions on a blockchain). Most DIDs are therefore inherently hard to memorize for humans.

By referring to DIDs from the Domain Name System (DNS), those hard to memorize identifiers can be discovered via well known, human friendly and widely established names. This document specifies how DIDs can be published in the DNS for discovery on the base of host names and email addresses.

Since DIDs use a URI scheme ('did'), this specification leverages the existing URI DNS Resource Record Type (RRType) [RFC7553]. Records are scoped using the '_did' global underscore node name, as described in Section 3.1.

2. Terminology

"Owner name", "Priority", "Weight" and "Target" refer to the respective fields of the URI RRType, as specified in Section 4 of RFC 7553.

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

3. Use of the 'URI' RRType

DIDs use an URI scheme ('did:'), so the most suitable option to publish DIDs in the DNS is the use of the 'URI' RRType. During the development of this document, various alternatives were considered, see Section 6 for a list.

- When Decentralized Identifiers (DIDs) are published in the DNS, the 'URI' RRType MUST be used.

3.1. Owner Name Scoping, Target

[RFC8552] describes the advantages of scoping an existing RRType over the definition (and complex deployment) of a new RRType. The "URI" RRType is specifically mentioned as one example where scoping is particularly useful (and part of the design).

When DIDs are published in the DNS

- the records MUST be scoped by setting the global (highest-level) underscore name of the URI RRset to '_did' (0x5F 0x64 0x69 0x64),

- and the Target field of all records in the RRset MUST contain a URI of the 'did:' URI scheme.

3.2. Weight, Priority

The semantics of the Weight and Priority fields remain. When a client encounters a DID method it does not support, it SHOULD consider the respective URI "unreachable" for the purpose of record

selection, and proceed to the record with the next-lowest-numbered Priority, in accordance with Section 4.2 of RFC 7553.

4. Location of the Records

4.1. Host Names

In order to discover the set of DIDs associated with a Host Name, a client prepends the given Host Name with the '_did' global underscore name to create the Owner name, and then queries the resulting Query Name for the URI RRType set.

4.2. Email Addresses (Experimental)

To discover DIDs associated with email addresses, the (experimental) model from DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP [RFC7929] is used. A client prepares the email address following the procedure outlined in Section 5 in RFC7929 the form the Query Name, but in step 5 MUST use the string '_mailto._did' instead of '_openpgpkey' as the second left-most label. Subsequently, the client performs a DNS query, but MUST use the URI RRType as Query Type (rather than the OPENPGPKEY RRType described in said section).

5. Example

The following example is a URI Resource Record which refers from the host name "example.net" to a Decentralized Identifier using the 'sov' method:

```
_did.example.net.  IN URI 100 10 "did:sov:1234abcd"
```

6. Considered Alternatives

During the development of this document, the following alternatives were considered: A dedicated RRType, TXT records, an Enumservice, Well-Known URIs, direct registration in the Service Name Registry. Using the URI RRType was found to be the option with the least impact on existing specifications and highest interoperability potential. Support for URI RRTypes is widespread in DNS software, which means that implementation and deployment of the proposed protocol should be possible without any changes to underlying infrastructure.

Furthermore, the Identifiers and Discovery Working Group of the Decentralized Identity Foundation (DIF) is considering a .well-known URL based approach to discovering DIDs from web sites.

7. Acknowledgements

Acknowledgements will be added here.

8. IANA Considerations

Per [RFC8552] IANA is requested to add the following entry to the DNS Underscore Global Scoped Entry Registry:

RR Type	_NODE NAME	REFERENCE
URI	_did	{THISRFC}

Table 1: Underscore Global Registry Entry Registration for '_did'

Note to RFC Editor: Please replace the above "{THISRFC}" text with a reference to this document's RFC number.

Note that IANA has already created a provisional URI scheme registration for the 'did:' scheme itself.

9. Security Considerations

Most of the considerations outlined in the base specification of the URI RRType (RFC7553) also apply to the DID use case - particularly the concerns around downgrade attacks when the record is not signed with the help of DNSSEC. Note that the DID resolving process itself (out of scope of this document) can provide additional security information. The "Linked Domain Service Endpoint" of a DID document can be used to back-reference to the Domain which was originally used to discover that DID. Such a "closed loop" (similar to verifying DNS reverse lookups against their corresponding forward lookups) would increase the confidence in non-DNSSEC scenarios.

Including a DID in the DNS allows for correlation of that DID with DNS information (and potentially registration information of that DNS name). Therefore DIDs which are supposed to be private SHOULD NOT be added to the DNS.

10. Changes

[Note to RFC Editors: This whole section is to be removed before publication]

10.1. draft-mayrhofer-did-dns-04

- Reworded "Alternatives"
- Added text about backreference using DID's Linked Domain Service Endpoint.

10.2. draft-mayrhofer-did-dns-03

- Updated DID spec to v1.0 document
- Minor editorial changes to make text more clear.

10.3. draft-mayrhofer-did-dns-02

- Updated attrleaf reference to RFC8552
- Changed author information for D. Klesev
- Added sentence on .well-known discovery scheme

10.4. draft-mayrhofer-did-dns-01

- email addresses further scoped with '_mailto._did'
- Changed protocol registration to attrleaf drafts
- Made clear requirements regarding use of the URI scheme
- Added privacy aspect to security considerations

10.5. draft-mayrhofer-did-dns-00

- Initial version

11. References

11.1. Normative References

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

[RFC7553] Faltstrom, P. and O. Kolkman, "The Uniform Resource Identifier (URI) DNS Resource Record", RFC 7553, DOI 10.17487/RFC7553, June 2015, <https://www.rfc-editor.org/info/rfc7553>.

[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>.

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

[W3C-DID] W3C, W3C., "Decentralized Identifiers (DIDs) v1.0", February 2020, <https://www.w3.org/TR/did-core/>.

11.2. Informative References

[DID-METHODS] W3C, W3C., "DID Method Registry", June 2018, <https://w3c-ccg.github.io/did-method-registry/>.

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

[RFC6116] Bradner, S., Conroy, L., and K. Fujiwara, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 6116, DOI 10.17487/RFC6116, March 2011, <https://www.rfc-editor.org/info/rfc6116>.

[RFC6117] Hoeneisen, B., Mayrhofer, A., and J. Livingood, "IANA Registration of Enumservices: Guide, Template, and IANA Considerations", RFC 6117, DOI 10.17487/RFC6117, March 2011, <https://www.rfc-editor.org/info/rfc6117>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC7929] Wouters, P., "DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP", RFC 7929, DOI 10.17487/RFC7929, August 2016, <https://www.rfc-editor.org/info/rfc7929>.

Authors' Addresses

Alexander Mayrhofer
nic.at GmbH
Karlsplatz 1/2/9
Vienna 1010
Austria

Email: alex.mayrhofer.ietf@gmail.com

Dimitrij Klesev

Email: dimitrij.klesev@gmail.com

Markus Sabadello
Danube Tech GmbH
Annagasse 8/1/8
Vienna 1010
Austria

Email: markus@danubetech.com