                       DNS Queries over HTTPS (DoH)
                       draft-ietf-doh-dns-over-https-13

Abstract

   This document defines a protocol for sending DNS queries and getting
   DNS responses over HTTPS. Each DNS query-response pair is mapped
   into an HTTP exchange.

Status of This Memo

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

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

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

   This Internet-Draft will expire on February 9, 2019.

Table of Contents

1. **Introduction**

   This document defines a specific protocol for sending DNS [RFC1035]
   queries and getting DNS responses over HTTP [RFC7540] using https
   URIs (and therefore TLS [RFC5246] security for integrity and
   confidentiality). Each DNS query-response pair is mapped into an
   HTTP exchange.

   The described approach is more than a tunnel over HTTP. It
   establishes default media formatting types for requests and responses
   but uses normal HTTP content negotiation mechanisms for selecting
   alternatives that endpoints may prefer in anticipation of serving new
   use cases. In addition to this media type negotiation, it aligns
   itself with HTTP features such as caching, redirection, proxying,
   authentication, and compression.

The integration with HTTP provides a transport suitable for both existing DNS clients and native web applications seeking access to the DNS.

Two primary uses cases were considered during this protocol's development. They included preventing on-path devices from interfering with DNS operations and allowing web applications to access DNS information via existing browser APIs in a safe way consistent with Cross Origin Resource Sharing (CORS) [CORS]. No special effort has been taken to enable or prevent application to other use cases. This document focuses on communication between DNS clients (such as operating system stub resolvers) and recursive resolvers.

2. Terminology

A server that supports this protocol is called a "DoH server" to differentiate it from a "DNS server" (one that only provides DNS service over one or more of the other transport protocols standardized for DNS). Similarly, a client that supports this protocol is called a "DoH client".

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. Protocol Requirements

[[RFC Editor: Please remove this entire section before publication.]]

The protocol described here bases its design on the following protocol requirements:

o   The protocol must use normal HTTP semantics.

o   The queries and responses must be able to be flexible enough to express every DNS query that would normally be sent in DNS over UDP (including queries and responses that use DNS extensions, but not those that require multiple responses).

o   The protocol must permit the addition of new formats for DNS queries and responses.

o   The protocol must ensure interoperability by specifying a single format for requests and responses that is mandatory to implement.

   That format must be able to support future modifications to the
   DNS protocol including the inclusion of one or more EDNS options
   (including those not yet defined).

   o The protocol must use a secure transport that meets the
      requirements for HTTPS.

3.1. Non-requirements

   o Supporting network-specific DNS64 [RFC6147]

   o Supporting other network-specific inferences from plaintext DNS
      queries

   o Supporting insecure HTTP

4. Selection of DoH Server

   Configuration, discovery, and updating of the URI Template [RFC6570]
   (see Section 5.1) is done out of band from this protocol. Note that
   configuration might be manual (such as a user typing URI Templates in
   a user interface for "options") or automatic (such as URI Templates
   being supplied in responses from DHCP or similar protocols). DoH
   Servers MAY support more than one URI Template. This allows the
   different endpoints to have different properties such as different
   authentication requirements or service level guarantees.

   A DoH client uses configuration to select the URI, and thus the DoH
   server, that is to be used for resolution. [RFC2818] defines how
   HTTPS verifies the DoH server's identity.

   A DoH client MUST NOT use a different URI simply because it was
   discovered outside of the client's configuration, or because a server
   offers an unsolicited response that appears to be a valid answer to a
   DNS query. This specification does not extend DNS resolution
   privileges to URIs that are not recognized by the DoH client as
   configured URIs. Such scenarios may create additional operational,
   tracking, and security hazards that require limitations for safe
   usage. A future specification may support this use case.

5. The HTTP Exchange

5.1. The HTTP Request

   A DoH client encodes a single DNS query into an HTTP request using
   either the HTTP GET or POST method and the other requirements of this
   section. The DoH server defines the URI used by the request through
   the use of a URI Template.

The URI Template defined in this document is processed without any variables when the HTTP method is POST. When the HTTP method is GET the single variable "dns" is defined as the content of the DNS request (as described in Section 7), encoded with base64url [RFC4648].

Future specifications for new media types MUST define the variables used for URI Template processing with this protocol.

DoH servers MUST implement both the POST and GET methods.

When using the POST method the DNS query is included as the message body of the HTTP request and the Content-Type request header field indicates the media type of the message. POST-ed requests are smaller than their GET equivalents.

Using the GET method is friendlier to many HTTP cache implementations.

The DoH client SHOULD include an HTTP "Accept" request header field to indicate what type of content can be understood in response. Irrespective of the value of the Accept request header field, the client MUST be prepared to process "application/dns-message" (as described in Section 7) responses but MAY also process any other type it receives.

In order to maximize cache friendliness, DoH clients using media formats that include DNS ID, such as application/dns-message, SHOULD use a DNS ID of 0 in every DNS request. HTTP correlates the request and response, thus eliminating the need for the ID in a media type such as application/dns-message. The use of a varying DNS ID can cause semantically equivalent DNS queries to be cached separately.

DoH clients can use HTTP/2 padding and compression in the same way that other HTTP/2 clients use (or don't use) them.

5.1.1. HTTP Request Examples

These examples use HTTP/2 style formatting from [RFC7540].

These examples use a DoH service with a URI Template of "https://dnsserver.example.net/dns-query{?dns}" to resolve IN A records.

The requests are represented as application/dns-message typed bodies.

The first example request uses GET to request www.example.com

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query?dns=AAABAAABAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAAQAB
accept = application/dns-message
```

The same DNS query for www.example.com, using the POST method would be:

```
:method = POST
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/dns-message
content-type = application/dns-message
content-length = 33
```

```
<33 bytes represented by the following hex encoding>
00 00 01 00 00 01 00 00  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
01
```

In this example, the 33 bytes are the DNS message in DNS wire format.

Finally, a GET based query for a.62characterlabel-makes-base64url-distinct-from-standard-base64.example.com is shown as an example to emphasize that the encoding alphabet of base64url is different than regular base64 and that padding is omitted.

The DNS query, expressed in DNS wire format, is 94 bytes represented by the following:

```
00 00 01 00 00 01 00 00  00 00 00 00 01 61 3e 36
32 63 68 61 72 61 63 74  65 72 6c 61 62 65 6c 2d
6d 61 6b 65 73 2d 62 61  73 65 36 34 75 72 6c 2d
64 69 73 74 69 6e 63 74  2d 66 72 6f 6d 2d 73 74
61 6e 64 61 72 64 2d 62  61 73 65 36 34 07 65 78
61 6d 70 6c 65 03 63 6f  6d 00 00 01 00 01
```

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query? (no space or CR)
        dns=AAABAAABAAAAAAAAWE-NjJjaGFyYWN0ZXJsYWJl (no space or CR)
        bC1tYWtlcy1iYXNlNjR1cmwtZGlzdGluY3QtZnJvbS1z (no space or CR)
        dGFuZGFyZC1iYXNlNjQHZXhhbXBsZQNjb20AAAEAAQ
accept = application/dns-message
```

5.2. The HTTP Response

The only response type defined in this document is "application/dns-
message", but it is possible that other response formats will be
defined in the future. A DoH server MUST be able to process
application/dns-message request messages.

Different response media types will provide more or less information
from a DNS response. For example, one response type might include
information from the DNS header bytes while another might omit it.
The amount and type of information that a media type gives is solely
up to the format, and not defined in this protocol.

Each DNS request-response pair is matched to one HTTP exchange. The
responses may be processed and transported in any order using HTTP's
multi-streaming functionality ([RFC7540] Section 5).

Section 6.1 discusses the relationship between DNS and HTTP response
caching.

5.2.1. Handling DNS and HTTP Errors

DNS response codes indicate either success or failure for the DNS
query. A successful HTTP response with a 2xx status code ([RFC7231]
Section 6.3) can be used for any valid DNS response, regardless of
the DNS response code. For example, a successful 2xx HTTP status
code is used even with a DNS message whose DNS response code
indicates failure, such as SERVFAIL or NXDOMAIN.

HTTP responses with non-successful HTTP status codes do not contain
replies to the original DNS question in the HTTP request. DoH
clients need to use the same semantic processing of non-successful
HTTP status codes as other HTTP clients. This might mean that the
DoH client retries the query with the same DoH server, such as
authorization failures (HTTP status code 401 [RFC7235] Section 3.1).
It could also mean that the DoH client retries with a different DoH
server, such as for unsupported media types (HTTP status code 415,
[RFC7231] Section 6.5.13), or where the server cannot generate a
representation suitable for the client (HTTP status code 406,
[RFC7231] Section 6.5.6), and so on.

5.2.2. HTTP Response Example

This is an example response for a query for the IN AAAA records for
"www.example.com" with recursion turned on. The response bears one
answer record with an address of 2001:db8:abcd:12:1:2:3:4 and a TTL
of 3709 seconds.

```
:status = 200
content-type = application/dns-message
content-length = 61
cache-control = max-age=3709

<61 bytes represented by the following hex encoding>
00 00 81 80 00 01 00 01  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 1c 00
01 c0 0c 00 1c 00 01 00  00 0e 7d 00 10 20 01 0d
b8 ab cd 00 12 00 01 00  02 00 03 00 04
```

6. HTTP Integration

This protocol MUST be used with the https scheme URI [RFC7230].

Section 9 and Section 10 discuss additional considerations for the integration with HTTP.

6.1. Cache Interaction

A DoH exchange can pass through a hierarchy of caches that include both HTTP- and DNS-specific caches. These caches may exist between the DoH server and client, or on the DoH client itself. HTTP caches are by design generic; that is, they do not understand this protocol. Even if a DoH client has modified its cache implementation to be aware of DoH semantics, it does not follow that all upstream caches (for example, inline proxies, server-side gateways and Content Delivery Networks) will be.

As a result, DoH servers need to carefully consider the HTTP caching metadata they send in response to GET requests (POST requests are not cacheable unless specific response header fields are sent; this is not widely implemented, and not advised for DoH).

In particular, DoH servers SHOULD assign an explicit freshness lifetime ([RFC7234] Section 4.2) so that the DoH client is more likely to use fresh DNS data. This requirement is due to HTTP caches being able to assign their own heuristic freshness (such as that described in [RFC7234] Section 4.2.2), which would take control of the cache contents out of the hands of the DoH server.

The assigned freshness lifetime of a DoH HTTP response MUST be less than or equal to the smallest TTL in the Answer section of the DNS response. A freshness lifetime equal to the smallest TTL in the Answer section is RECOMMENDED. For example, if a HTTP response carries three RRsets with TTLs of 30, 600, and 300, the HTTP freshness lifetime should be 30 seconds (which could be specified as

"Cache-Control: max-age=30"). This requirement helps assure that
none of the RRsets contained in a DNS response are served stale from
an HTTP cache.

If the DNS response has no records in the Answer section, and the DNS
response has an SOA record in the Authority section, the response
freshness lifetime MUST NOT be greater than the MINIMUM field from
that SOA record (see [RFC2308]).

The stale-while-revalidate and stale-if-error Cache-Control
directives ([RFC5861]) could be well-suited to a DoH implementation
when allowed by server policy. Those mechanisms allow a client, at
the server's discretion, to reuse a cache entry that is no longer
fresh. In such a case, the client reuses all of a cached entry, or
none of it.

DoH servers also need to consider caching when generating responses
that are not globally valid. For instance, if a DoH server
customizes a response based on the client's identity, it would not
want to allow global reuse of that response. This could be
accomplished through a variety of HTTP techniques such as a Cache-
Control max-age of 0, or by using the Vary response header field
([RFC7231] Section 7.1.4) to establish a secondary cache key
([RFC7234] Section 4.1).

DoH clients MUST account for the Age response header field's value
([RFC7234]) when calculating the DNS TTL of a response. For example,
if an RRset is received with a DNS TTL of 600, but the Age header
field indicates that the response has been cached for 250 seconds,
the remaining lifetime of the RRset is 350 seconds. This requirement
applies to both DoH client HTTP caches and DoH client DNS caches.

DoH clients can request an uncached copy of a response by using the
"no-cache" request cache control directive ([RFC7234],
Section 5.2.1.4) and similar controls. Note that some caches might
not honor these directives, either due to configuration or
interaction with traditional DNS caches that do not have such a
mechanism.

HTTP conditional requests ([RFC7232]) may be of limited value to DoH,
as revalidation provides only a bandwidth benefit and DNS
transactions are normally latency bound. Furthermore, the HTTP
response header fields that enable revalidation (such as "Last-
Modified" and "Etag") are often fairly large when compared to the
overall DNS response size, and have a variable nature that creates
constant pressure on the HTTP/2 compression dictionary [RFC7541].
Other types of DNS data, such as zone transfers, may be larger and
benefit more from revalidation.

6.2. HTTP/2

HTTP/2 [RFC7540] is the minimum RECOMMENDED version of HTTP for use
with DoH.

The messages in classic UDP-based DNS [RFC1035] are inherently
unordered and have low overhead. A competitive HTTP transport needs
to support reordering, parallelism, priority, and header compression
to achieve similar performance. Those features were introduced to
HTTP in HTTP/2 [RFC7540]. Earlier versions of HTTP are capable of
conveying the semantic requirements of DoH but may result in very
poor performance.

6.3. Server Push

Before using DoH response data for DNS resolution, the client MUST
establish that the HTTP request URI can be used for the DoH query.
For HTTP requests initiated by the DoH client, this is implicit in
the selection of URI. For HTTP server push ([RFC7540] Section 8.2)
extra care must be taken to ensure that the pushed URI is one that
the client would have directed the same query to if the client had
initiated the request.

6.4. Content Negotiation

In order to maximize interoperability, DoH clients and DoH servers
MUST support the "application/dns-message" media type. Other media
types MAY be used as defined by HTTP Content Negotiation ([RFC7231]
Section 3.4). Those media types MUST be flexible enough to express
every DNS query that would normally be sent in DNS over UDP
(including queries and responses that use DNS extensions, but not
those that require multiple responses).

7. Definition of the application/dns-message media type

The data payload for the application/dns-message media type is a
single message of the DNS on-the-wire format defined in Section 4.2.1
of [RFC1035], which in turn refers to the full wire format defined in
Section 4.1 of that RFC.

Although [RFC1035] says "Messages carried by UDP are restricted to
512 bytes", that was later updated by [RFC6891]. This media type
restricts the maximum size of the DNS message to 65535 bytes.

Note that the wire format used in this media type is different than
the wire format used in [RFC7858] (which uses the format defined in
Section 4.2.2 of [RFC1035] that includes two length bytes).

   DoH clients using this media type MAY have one or more EDNS options
   [RFC6891] in the request. DoH servers using this media type MUST
   ignore the value given for the EDNS UDP payload size in DNS requests.

   When using the GET method, the data payload for this media type MUST
   be encoded with base64url [RFC4648] and then provided as a variable
   named "dns" to the URI Template expansion. Padding characters for
   base64url MUST NOT be included.

   When using the POST method, the data payload for this media type MUST
   NOT be encoded and is used directly as the HTTP message body.

8. IANA Considerations

8.1. Registration of application/dns-message Media Type

To: ietf-types@iana.org
Subject: Registration of MIME media type
         application/dns-message

MIME media type name: application

MIME subtype name: dns-message

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: This is a binary format. The contents are a
DNS message as defined in RFC 1035. The format used here is for DNS
over UDP, which is the format defined in the diagrams in RFC 1035.

Security considerations: The security considerations for carrying
this data are the same for carrying DNS without encryption.

Interoperability considerations: None.

Published specification: This document.

Applications that use this media type:
  Systems that want to exchange full DNS messages.

Additional information:

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person & email address to contact for further information:
    Paul Hoffman, paul.hoffman@icann.org

Intended usage: COMMON

Restrictions on usage: n/a

Author: Paul Hoffman, paul.hoffman@icann.org

Change controller: IESG

9. Privacy Considerations

[RFC7626] discusses DNS privacy considerations in both "On the wire"
(Section 2.4), and "In the server" (Section 2.5) contexts. This is
also a useful framing for DoH's privacy considerations.

9.1. On The Wire

DoH encrypts DNS traffic and requires authentication of the server.
This mitigates both passive surveillance [RFC7258] and active attacks
that attempt to divert DNS traffic to rogue servers ([RFC7626]
Section 2.5.1). DNS over TLS [RFC7858] provides similar protections,
while direct UDP and TCP based transports are vulnerable to this
class of attack.

Additionally, the use of the HTTPS default port 443 and the ability
to mix DoH traffic with other HTTPS traffic on the same connection
can deter unprivileged on-path devices from interfering with DNS
operations and make DNS traffic analysis more difficult.

9.2. In The Server

The DNS wire format [RFC1035] contains no client identifiers;
however, various transports of DNS queries and responses do provide
data that can be used to correlate requests. HTTPS presents new
considerations for correlation, such as explicit HTTP cookies and
implicit fingerprinting of the unique set and ordering of HTTP
request header fields.

A DoH implementation is built on IP, TCP, TLS, and HTTP. Each layer
contains one or more common features that can be used to correlate
queries to the same identity. DNS transports will generally carry
the same privacy properties of the layers used to implement them.
For example, the properties of IP, TCP, and TLS apply to DNS over TLS
implementations.

The privacy considerations of using the HTTPS layer in DoH are
incremental to those of DNS over TLS. DoH is not known to introduce
new concerns beyond those associated with HTTPS.

At the IP level, the client address provides obvious correlation
information. This can be mitigated by use of a NAT, proxy, VPN, or
simple address rotation over time. It may be aggravated by use of a
DNS server that can correlate real-time addressing information with
other personal identifiers, such as when a DNS server and DHCP server
are operated by the same entity.

DNS implementations that use one TCP connection for multiple DNS
requests directly group those requests. Long-lived connections have
better performance behaviors than short-lived connections, but group
more requests. TCP-based solutions may also seek performance through
the use of TCP Fast Open [RFC7413]. The cookies used in TCP Fast
Open allow servers to correlate TCP sessions.

TLS-based implementations often achieve better handshake performance
through the use of some form of session resumption mechanism such as
session tickets [RFC5077]. Session resumption creates trivial
mechanisms for a server to correlate TLS connections together.

HTTP's feature set can also be used for identification and tracking
in a number of different ways. For example, authentication request
header fields explicitly identify profiles in use, and HTTP Cookies
are designed as an explicit state-tracking mechanism between the
client and serving site and often are used as an authentication
mechanism.

Additionally, the User-Agent and Accept-Language request header
fields often convey specific information about the client version or
locale. This facilitates content negotiation and operational work-
arounds for implementation bugs. Request header fields that control
caching can expose state information about a subset of the client's
history. Mixing DoH requests with other HTTP requests on the same
connection also provides an opportunity for richer data correlation.

The DoH protocol design allows applications to fully leverage the
HTTP ecosystem, including features that are not enumerated here.
Utilizing the full set of HTTP features enables DoH to be more than
an HTTP tunnel, but at the cost of opening up implementations to the
full set of privacy considerations of HTTP.

Implementations of DoH clients and servers need to consider the
benefit and privacy impact of these features, and their deployment
context, when deciding whether or not to enable them.
Implementations are advised to expose the minimal set of data needed
to achieve the desired feature set.

Determining whether or not a DoH implementation requires HTTP cookie
[RFC6265] support is particularly important because HTTP cookies are
the primary state tracking mechanism in HTTP. HTTP Cookies SHOULD
NOT be accepted by DOH clients unless they are explicitly required by
a use case.

10. Security Considerations

Running DNS over HTTPS relies on the security of the underlying HTTP transport. This mitigates classic amplification attacks for UDP-based DNS. Implementations utilizing HTTP/2 benefit from the TLS profile defined in [RFC7540] Section 9.2.

Session-level encryption has well-known weaknesses with respect to traffic analysis which might be particularly acute when dealing with DNS queries. HTTP/2 provides further advice about the use of compression ([RFC7540] Section 10.6) and padding ([RFC7540] Section 10.7). DoH Servers can also add DNS padding [RFC7830] if the DoH client requests it in the DNS query. An experimental effort to offer guidance on choosing the padding length can be found in [I-D.ietf-dprive-padding-policy].

The HTTPS connection provides transport security for the interaction between the DoH server and client, but does not provide the response integrity of DNS data provided by DNSSEC. DNSSEC and DoH are independent and fully compatible protocols, each solving different problems. The use of one does not diminish the need nor the usefulness of the other. It is the choice of a client to either perform full DNSSEC validation of answers or to trust the DoH server to do DNSSEC validation and inspect the AD (Authentic Data) bit in the returned message to determine whether an answer was authentic or not. As noted in Section 5.2, different response media types will provide more or less information from a DNS response so this choice may be affected by the response media type.

Section 6.1 describes the interaction of this protocol with HTTP caching. An adversary that can control the cache used by the client can affect that client's view of the DNS. This is no different than the security implications of HTTP caching for other protocols that use HTTP.

In the absence of DNSSEC information, a DoH server can give a client invalid data in response to a DNS query. Section 4 disallows the use of DoH DNS responses that do not originate from configured servers. This prohibition does not guarantee protection against invalid data, but it does reduce the risk.

11. Operational Considerations

Local policy considerations and similar factors mean different DNS servers may provide different results to the same query, for instance in split DNS configurations [RFC6950]. It logically follows that the server that is queried can influence the end result. Therefore a client's choice of DNS server may affect the responses it gets to its

queries. For example, in the case of DNS64 [RFC6147], the choice
could affect whether IPv6/IPv4 translation will work at all.

The HTTPS channel used by this specification establishes secure two-
party communication between the DoH client and the DoH server.
Filtering or inspection systems that rely on unsecured transport of
DNS will not function in a DNS over HTTPS environment.

Some HTTPS client implementations perform real time third-party
checks of the revocation status of the certificates being used by
TLS. If this check is done as part of the DoH server connection
procedure and the check itself requires DNS resolution to connect to
the third party a deadlock can occur. The use of OCSP [RFC6960]
servers or AIA for CRL fetching ([RFC5280] Section 4.2.2.1) are
examples of how this deadlock can happen. To mitigate the
possibility of deadlock, DoH servers SHOULD NOT rely on DNS-based
references to external resources in the TLS handshake. For OCSP, the
server can bundle the certificate status as part of the handshake
using a mechanism appropriate to the version of TLS, such as using
[RFC6066] Section 8 for TLS version 1.2. AIA deadlocks can be
avoided by providing intermediate certificates that might otherwise
be obtained through additional requests. Note that these deadlocks
also need to be considered for server that a DoH server might
redirect to.

A DoH client may face a similar bootstrapping problem when the HTTP
request needs to resolve the hostname portion of the DNS URI. Just
as the address of a traditional DNS nameserver cannot be originally
determined from that same server, a DoH client cannot use its DoH
server to initially resolve the server's host name into an address.
Alternative strategies a client might employ include making the
initial resolution part of the configuration, IP-based URIs and
corresponding IP-based certificates for HTTPS, or resolving the DNS
API server's hostname via traditional DNS or another DoH server while
still authenticating the resulting connection via HTTPS.

HTTP [RFC7230] is a stateless application-level protocol and
therefore DoH implementations do not provide stateful ordering
guarantees between different requests. DoH cannot be used as a
transport for other protocols that require strict ordering.

A DoH server is allowed to answer queries with any valid DNS
response. For example, a valid DNS response might have the TC
(truncation) bit set in the DNS header to indicate that the server
was not able to retrieve a full answer for the query but is providing
the best answer it could get. A DoH server can reply to queries with
an HTTP error for queries that it cannot fulfill. In this same

example, a DoH server could use an HTTP error instead of a non-error
response that has the TC bit set.

Many extensions to DNS, using [RFC6891], have been defined over the
years. Extensions that are specific to the choice of transport, such
as [RFC7828], are not applicable to DoH.

12. References

12.1. Normative References

[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>.

[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>.

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

[RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
            (TLS) Protocol Version 1.2", RFC 5246,
            DOI 10.17487/RFC5246, August 2008,
            <https://www.rfc-editor.org/info/rfc5246>.

[RFC6265]   Barth, A., "HTTP State Management Mechanism", RFC 6265,
            DOI 10.17487/RFC6265, April 2011,
            <https://www.rfc-editor.org/info/rfc6265>.

[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>.

[RFC7230]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
            Protocol (HTTP/1.1): Message Syntax and Routing",
            RFC 7230, DOI 10.17487/RFC7230, June 2014,
            <https://www.rfc-editor.org/info/rfc7230>.

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

   [RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Conditional Requests", RFC 7232,
              DOI 10.17487/RFC7232, June 2014,
              <https://www.rfc-editor.org/info/rfc7232>.

   [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke,
              Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching",
              RFC 7234, DOI 10.17487/RFC7234, June 2014,
              <https://www.rfc-editor.org/info/rfc7234>.

   [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Authentication", RFC 7235,
              DOI 10.17487/RFC7235, June 2014,
              <https://www.rfc-editor.org/info/rfc7235>.

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

   [RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for
              HTTP/2", RFC 7541, DOI 10.17487/RFC7541, May 2015,
              <https://www.rfc-editor.org/info/rfc7541>.

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

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

12.2. Informative References

   [CORS]   "Cross-Origin Resource Sharing", n.d.,
              <https://fetch.spec.whatwg.org/#http-cors-protocol>.

   [I-D.ietf-dprive-padding-policy]
              Mayrhofer, A., "Padding Policy for EDNS(0)", draft-ietf-
              dprive-padding-policy-06 (work in progress), July 2018.

   [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818,
              DOI 10.17487/RFC2818, May 2000,
              <https://www.rfc-editor.org/info/rfc2818>.

   [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig,
              "Transport Layer Security (TLS) Session Resumption without
              Server-Side State", RFC 5077, DOI 10.17487/RFC5077,
              January 2008, <https://www.rfc-editor.org/info/rfc5077>.

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

   [RFC5861] Nottingham, M., "HTTP Cache-Control Extensions for Stale
              Content", RFC 5861, DOI 10.17487/RFC5861, May 2010,
              <https://www.rfc-editor.org/info/rfc5861>.

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

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

   [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>.

   [RFC6950] Peterson, J., Kolkman, O., Tschofenig, H., and B. Aboba,
              "Architectural Considerations on Application Features in
              the DNS", RFC 6950, DOI 10.17487/RFC6950, October 2013,
              <https://www.rfc-editor.org/info/rfc6950>.

   [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A.,
              Galperin, S., and C. Adams, "X.509 Internet Public Key
              Infrastructure Online Certificate Status Protocol - OCSP",
              RFC 6960, DOI 10.17487/RFC6960, June 2013,
              <https://www.rfc-editor.org/info/rfc6960>.

   [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an
              Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May
              2014, <https://www.rfc-editor.org/info/rfc7258>.

   [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP
              Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014,
              <https://www.rfc-editor.org/info/rfc7413>.

   [RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The
              edns-tcp-keepalive EDNS0 Option", RFC 7828,
              DOI 10.17487/RFC7828, April 2016,
              <https://www.rfc-editor.org/info/rfc7828>.

   [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830,
              DOI 10.17487/RFC7830, May 2016,
              <https://www.rfc-editor.org/info/rfc7830>.

   [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>.

Acknowledgments

Previous Work on DNS over HTTP or in Other Formats

   The following is an incomplete list of earlier work that related to
   DNS over HTTP/1 or representing DNS data in other formats.

   The list includes links to the tools.ietf.org site (because these
   documents are all expired) and web sites of software.

   o https://tools.ietf.org/html/draft-mohan-dns-query-xml

   o https://tools.ietf.org/html/draft-daley-dnsxml

   o https://tools.ietf.org/html/draft-dulaunoy-dnsop-passive-dns-cof

   o https://tools.ietf.org/html/draft-bortzmeyer-dns-json

o   https://www.nlnetlabs.nl/projects/dnssec-trigger/

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Patrick McManus
Mozilla

Email: mcmanus@ducksong.com