DNS Push Notifications
draft-ietf-dnssd-push-22

Abstract

   The Domain Name System (DNS) was designed to return matching records
   efficiently for queries for data that are relatively static. When
   those records change frequently, DNS is still efficient at returning
   the updated results when polled, as long as the polling rate is not
   too high. But there exists no mechanism for a client to be
   asynchronously notified when these changes occur. This document
   defines a mechanism for a client to be notified of such changes to
   DNS records, called DNS Push Notifications.

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.

Copyright Notice

Table of Contents

1. Introduction

   Domain Name System (DNS) records may be updated using DNS Update
   [RFC2136]. Other mechanisms such as a Discovery Proxy [DisProx] can
   also generate changes to a DNS zone. This document specifies a
   protocol for DNS clients to subscribe to receive asynchronous
   notifications of changes to RRSets of interest. It is immediately
   relevant in the case of DNS Service Discovery [RFC6763] but is not
   limited to that use case, and provides a general DNS mechanism for
   DNS record change notifications. Familiarity with the DNS protocol
   and DNS packet formats is assumed [RFC1034] [RFC1035] [RFC6895].

1.1. Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
   "OPTIONAL" in this document are to be interpreted as described in
   BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
   capitals, as shown here. These words may also appear in this
   document in lower case as plain English words, absent their normative
   meanings.

2. Motivation

   As the domain name system continues to adapt to new uses and changes
   in deployment, polling has the potential to burden DNS servers at
   many levels throughout the network. Other network protocols have
   successfully deployed a publish/subscribe model following the
   Observer design pattern [obs]. XMPP Publish-Subscribe [XEP0060] and
   Atom [RFC4287] are examples. While DNS servers are generally highly
   tuned and capable of a high rate of query/response traffic, adding a
   publish/subscribe model for tracking changes to DNS records can
   deliver more timely notification of changes with reduced CPU usage
   and lower network traffic.

   Multicast DNS [RFC6762] implementations always listen on a well known
   link-local IP multicast group, and record changes are sent to that
   multicast group address for all group members to receive. Therefore,
   Multicast DNS already has asynchronous change notification
   capability. However, when DNS Service Discovery [RFC6763] is used
   across a wide area network using Unicast DNS (possibly facilitated
   via a Discovery Proxy [DisProx]) it would be beneficial to have an
   equivalent capability for Unicast DNS, to allow clients to learn
   about DNS record changes in a timely manner without polling.

   The DNS Long-Lived Queries (LLQ) mechanism [LLQ] is an existing
   deployed solution to provide asynchronous change notifications, used
   by Apple's Back to My Mac [RFC6281] service introduced in Mac OS X
   10.5 Leopard in 2007. Back to My Mac was designed in an era when the
   data center operations staff asserted that it was impossible for a
   server to handle large numbers of mostly-idle TCP connections, so LLQ
   was defined as a UDP-based protocol, effectively replicating much of
   TCP's connection state management logic in user space, and creating
   its own poor imitations of existing TCP features like the three-way
   handshake, flow control, and reliability.

   This document builds on experience gained with the LLQ protocol, with
   an improved design. Instead of using UDP, this specification uses
   DNS Stateful Operations (DSO) [RFC8490] running over TLS over TCP,
   and therefore doesn't need to reinvent existing TCP functionality.
   Using TCP also gives long-lived low-traffic connections better
   longevity through NAT gateways without depending on the gateway to
   support NAT Port Mapping Protocol (NAT-PMP) [RFC6886] or Port Control
   Protocol (PCP) [RFC6887], or resorting to excessive keepalive
   traffic.

3. Overview

A DNS Push Notification client subscribes for Push Notifications for
a particular RRSet by connecting to the appropriate Push Notification
server for that RRSet, and sending DSO message(s) indicating the
RRSet(s) of interest. When the client loses interest in receiving
further updates to these records, it unsubscribes.

The DNS Push Notification server for a DNS zone is any server capable
of generating the correct change notifications for a name. It may be
a primary, secondary, or stealth name server [RFC7719].
Consequently, the "_dns-push-tls._tcp.<zone>" SRV record for a zone
MAY reference the same target host and port as that zone's
"_dns-update-tls._tcp.<zone>" SRV record. When the same target host
and port is offered for both DNS Updates and DNS Push Notifications,
a client MAY use a single TCP connection to that server for both DNS
Updates and DNS Push Notification Subscriptions.

Supporting DNS Updates and DNS Push Notifications on the same server
is OPTIONAL. A DNS Push Notification server is not required to
support DNS Update.

DNS Updates and DNS Push Notifications may be handled on different
ports on the same target host, in which case they are not considered
to be the "same server" for the purposes of this specification, and
communications with these two ports are handled independently.

Standard DNS Queries MAY be sent over a DNS Push Notification (i.e.,
DSO) session. For any zone for which the server is authoritative, it
MUST respond authoritatively for queries on names falling within that
zone (e.g., the <zone> in the "_dns-push-tls._tcp.<zone>" SRV record)
both for normal DNS queries and for DNS Push Notification
subscriptions. For names for which the server is acting as a
recursive resolver, e.g. when the server is the local recursive
resolver, for any query for which it supports DNS Push Notification
subscriptions, it MUST also support standard queries.

DNS Push Notifications impose less load on the responding server than
rapid polling would, but Push Notifications do still have a cost, so
DNS Push Notification clients MUST NOT recklessly create an excessive
number of Push Notification subscriptions. Specifically:

(a) A subscription should only be active when there is a valid reason
to need live data (for example, an on-screen display is currently
showing the results to the user) and the subscription SHOULD be
cancelled as soon as the need for that data ends (for example, when
the user dismisses that display). In the case of a device like a
smartphone which, after some period of inactivity, goes to sleep or

otherwise darkens its screen, it should cancel its subscriptions when
darkening the screen (since the user cannot see any changes in the
display anyway) and reinstate its subscriptions when re-awakening
from display sleep.

(b) A DNS Push Notification client SHOULD NOT routinely keep a DNS
Push Notification subscription active 24 hours a day, 7 days a week,
just to keep a list in memory up to date so that if the user does
choose to bring up an on-screen display of that data, it can be
displayed really fast. DNS Push Notifications are designed to be
fast enough that there is no need to pre-load a "warm" list in memory
just in case it might be needed later.

Generally, as described in the DNS Stateful Operations specification
[RFC8490], a client must not keep a session to a server open
indefinitely if it has no subscriptions (or other operations) active
on that session. A client MAY close a session as soon as it becomes
idle, and then if needed in the future, open a new session when
required. Alternatively, a client MAY speculatively keep an idle
session open for some time, subject to the constraint that it MUST
NOT keep a session open that has been idle for more than the
session's idle timeout (15 seconds by default) [RFC8490].

4. Transport

Other DNS operations like DNS Update [RFC2136] MAY use either User
Datagram Protocol (UDP) [RFC0768] or Transmission Control Protocol
(TCP) [RFC0793] as the transport protocol, in keeping with the
historical precedent that DNS queries must first be sent over UDP
[RFC1123]. This requirement to use UDP has subsequently been relaxed
[RFC7766].

In keeping with the more recent precedent, DNS Push Notification is
defined only for TCP. DNS Push Notification clients MUST use DNS
Stateful Operations [RFC8490] running over TLS over TCP [RFC7858].

Connection setup over TCP ensures return reachability and alleviates
concerns of state overload at the server which is a potential problem
with connectionless protocols using spoofed source addresses. All
subscribers are guaranteed to be reachable by the server by virtue of
the TCP three-way handshake. Flooding attacks are possible with any
protocol, and a benefit of TCP is that there are already established
industry best practices to guard against SYN flooding and similar
attacks [SYN] [RFC4953].

Use of TCP also allows DNS Push Notifications to take advantage of
current and future developments in TCP, such as Multipath TCP (MPTCP)

[RFC6824], TCP Fast Open (TFO) [RFC7413], Tail Loss Probe (TLP)
[I-D.dukkipati-tcpm-tcp-loss-probe], and so on.

Transport Layer Security (TLS) [RFC8446] is well understood and
deployed across many protocols running over TCP. It is designed to
prevent eavesdropping, tampering, and message forgery. TLS is
REQUIRED for every connection between a client subscriber and server
in this protocol specification. Additional security measures such as
client authentication during TLS negotiation MAY also be employed to
increase the trust relationship between client and server.

5. State Considerations

Each DNS Push Notification server is capable of handling some finite
number of Push Notification subscriptions. This number will vary
from server to server and is based on physical machine
characteristics, network bandwidth, and operating system resource
allocation. After a client establishes a session to a DNS server,
each subscription is individually accepted or rejected. Servers may
employ various techniques to limit subscriptions to a manageable
level. Correspondingly, the client is free to establish simultaneous
sessions to alternate DNS servers that support DNS Push Notifications
for the zone and distribute subscriptions at the client's discretion.
In this way, both clients and servers can react to resource
constraints.

6. Protocol Operation

The DNS Push Notification protocol is a session-oriented protocol,
and makes use of DNS Stateful Operations (DSO) [RFC8490].

For details of the DSO message format refer to the DNS Stateful Oper-
ations specification [RFC8490]. Those details are not repeated here.

DNS Push Notification clients and servers MUST support DSO. A single
server can support DNS Queries, DNS Updates, and DNS Push
Notifications (using DSO) on the same TCP port.

A DNS Push Notification exchange begins with the client discovering
the appropriate server, using the procedure described in Section 6.1,
and then making a TLS/TCP connection to it.

A typical DNS Push Notification client will immediately issue a DSO
Keepalive operation to request a session timeout and/or keepalive
interval longer than the the 15-second default values, but this is
not required. A DNS Push Notification client MAY issue other
requests on the session first, and only issue a DSO Keepalive
operation later if it determines that to be necessary. Sending
either a DSO Keepalive operation or a Push Notification subscription
over the TLS/TCP connection to the server signals the client's
support of DSO and serves to establish a DSO session.

In accordance with the current set of active subscriptions, the
server sends relevant asynchronous Push Notifications to the client.
Note that a client MUST be prepared to receive (and silently ignore)
Push Notifications for subscriptions it has previously removed, since
there is no way to prevent the situation where a Push Notification is
in flight from server to client while the client's UNSUBSCRIBE
message cancelling that subscription is simultaneously in flight from
client to server.

6.1. Discovery

   The first step in establishing a DNS Push Notification subscription
   is to discover an appropriate DNS server that supports DNS Push
   Notifications for the desired zone.

   The client begins by opening a DSO Session to its normal configured
   DNS recursive resolver and requesting a Push Notification
   subscription. This connection is made to TCP port 853, the default
   port for DNS-over-TLS [RFC7858]. If the request for a Push
   Notification subscription is successful, and the recursive resolver
   doesn't already have an active subscription for that name, type, and
   class, then the recursive resolver will make a corresponding Push
   Notification subscription on the client's behalf. Results received
   are relayed to the client. This is closely analogous to how a client
   sends a normal DNS query to its configured DNS recursive resolver
   which, if it doesn't already have appropriate answer(s) in its cache,
   issues an upstream query to satisfy the request.

   In many contexts, the recursive resolver will be able to handle Push
   Notifications for all names that the client may need to follow. Use
   of VPN tunnels and split-view DNS can create some additional
   complexity in the client software here; the techniques to handle VPN
   tunnels and split-view DNS for DNS Push Notifications are the same as
   those already used to handle this for normal DNS queries.

   If the recursive resolver does not support DNS over TLS, or does
   support DNS over TLS but is not listening on TCP port 853, or does
   support DNS over TLS on TCP port 853 but does not support DSO on that
   port, then the DSO Session session establishment will fail [RFC8490].

   If the recursive resolver does support DSO but not Push Notification
   subscriptions, then it will return the DSO error code, DSOTYPENI
   (11).

   In some cases, the recursive resolver may support DSO and Push
   Notification subscriptions, but may not be able to subscribe for Push
   Notifications for a particular name. In this case, the recursive
   resolver should return SERVFAIL to the client. This includes being
   unable to establish a connection to the zone's DNS Push Notification
   server or establishing a connection but receiving a non success
   response code. In some cases, where the client has a pre-established
   trust relationship with the owner of the zone (that is not handled
   via the usual mechanisms for VPN software) the client may handle
   these failures by contacting the zone's DNS Push server directly.

   In any of the cases described above where the client fails to
   establish a DNS Push Notification subscription via its configured

recursive resolver, the client should proceed to discover the
appropriate server for direct communication. The client MUST also
determine which TCP port on the server is listening for connections,
which need not be (and often is not) the typical TCP port 53 used for
conventional DNS, or TCP port 853 used for DNS over TLS.

The discovery algorithm described here is an iterative algorithm,
which starts with the full name of the record to which the client
wishes to subscribe. Successive SOA queries are then issued,
trimming one label each time, until the closest enclosing
authoritative server is discovered. There is also an optimization to
enable the client to take a "short cut" directly to the SOA record of
the closest enclosing authoritative server in many cases.

1. The client begins the discovery by sending a DNS query to its
    local resolver, with record type SOA [RFC1035] for the record
    name to which it wishes to subscribe. As an example, suppose the
    client wishes to subscribe to PTR records with the name
    _ipp._tcp.foo.example.com (to discover Internet Printing Protocol
    (IPP) printers [RFC8010] [RFC8011] being advertised at
    "foo.example.com"). The client begins by sending an SOA query
    for _ipp._tcp.foo.example.com to the local recursive resolver.
    The goal is to determine the server authoritative for the name
    _ipp._tcp.foo.example.com. The closest enclosing DNS zone
    containing the name _ipp._tcp.foo.example.com could be
    example.com, or foo.example.com, or _tcp.foo.example.com, or even
    _ipp._tcp.foo.example.com. The client does not know in advance
    where the closest enclosing zone cut occurs, which is why it uses
    the iterative procedure described here to discover this
    information.

2. If the requested SOA record exists, it will be returned in the
    Answer section with a NOERROR response code, and the client has
    succeeded in discovering the information it needs.
    (This language is not placing any new requirements on DNS
    recursive resolvers. This text merely describes the existing
    operation of the DNS protocol [RFC1034] [RFC1035].)

3. If the requested SOA record does not exist, the client will get
    back a NOERROR/NODATA response or an NXDOMAIN/Name Error
    response. In either case, the local resolver would normally
    include the SOA record for the closest enclosing zone of the
    requested name in the Authority Section. If the SOA record is
    received in the Authority Section, then the client has succeeded
    in discovering the information it needs.
    (This language is not placing any new requirements on DNS
    recursive resolvers. This text merely describes the existing

operation of the DNS protocol regarding negative responses
[RFC2308].)

4. If the client receives a response containing no SOA record, then
    it proceeds with the iterative approach. The client strips the
    leading label from the current query name and if the resulting
    name has at least one label in it, the client sends an SOA query
    for that new name, and processing continues at step 2 above,
    repeating the iterative search until either an SOA is received,
    or the query name consists of a single label, i.e., a Top Level
    Domain (TLD). In the case of a single-label TLD, this is a
    network configuration error which should not happen and the
    client gives up. The client may retry the operation at a later
    time, of the client's choosing, such after a change in network
    attachment.

5. Once the SOA is known (either by virtue of being seen in the
    Answer Section, or in the Authority Section), the client sends a
    DNS query with type SRV [RFC2782] for the record name
    "_dns-push-tls._tcp.<zone>", where <zone> is the owner name of
    the discovered SOA record.

6. If the zone in question is set up to offer DNS Push Notifications
    then this SRV record MUST exist. (If this SRV record does not
    exist then the zone is not correctly configured for DNS Push
    Notifications as specified in this document.) The SRV "target"
    contains the name of the server providing DNS Push Notifications
    for the zone. The port number on which to contact the server is
    in the SRV record "port" field. The address(es) of the target
    host MAY be included in the Additional Section, however, the
    address records SHOULD be authenticated before use as described
    below in Section 7.2 and in the specification for using DANE TLSA
    Records with SRV Records [RFC7673], if applicable.

7. More than one SRV record may be returned. In this case, the
    "priority" and "weight" values in the returned SRV records are
    used to determine the order in which to contact the servers for
    subscription requests. As described in the SRV specification
    [RFC2782], the server with the lowest "priority" is first
    contacted. If more than one server has the same "priority", the
    "weight" indicates the weighted probability that the client
    should contact that server. Higher weights have higher
    probabilities of being selected. If a server is not willing to
    accept a subscription request, or is not reachable within a
    reasonable time, as determined by the client, then a subsequent
    server is to be contacted.

Each time a client makes a new DNS Push Notification subscription session, it SHOULD repeat the discovery process in order to determine the preferred DNS server for subscriptions at that time. However, the client device MUST respect the DNS TTL values on records it receives, and store them in its local cache with this lifetime. This means that, as long as the DNS TTL values on the authoritative records were set to reasonable values, repeated application of this discovery process can be completed nearly instantaneously by the client, using only locally-stored cached data.

6.2. DNS Push Notification SUBSCRIBE

After connecting, and requesting a longer idle timeout and/or
keepalive interval if necessary, a DNS Push Notification client
then indicates its desire to receive DNS Push Notifications for
a given domain name by sending a SUBSCRIBE request to the server.
A SUBSCRIBE request is encoded in a DSO message [RFC8490].
This specification defines a primary DSO TLV for DNS Push
Notification SUBSCRIBE Requests (tentatively DSO Type Code 0x40).

DSO messages with the SUBSCRIBE TLV as the Primary TLV are permitted
in early data, provided that the precautions described in Section 7.3
are followed.

The entity that initiates a SUBSCRIBE request is by definition the
client. A server MUST NOT send a SUBSCRIBE request over an existing
session from a client. If a server does send a SUBSCRIBE request
over a DSO session initiated by a client, this is a fatal error and
the client should immediately abort the connection with a TLS
close_notify alert (see Section 6.1 of the TLS 1.3 specification
[RFC8446]). After sending the TLS close_notify alert the client MUST
gracefully close the underlying connection using a TCP FIN, so that
the TLS close_notify is reliably delivered. The mechanisms for
gracefully closing a TCP connection with a TCP FIN vary depending on
the networking API. For example, in the BSD Sockets API, sending a
TCP FIN is achieved by calling "shutdown(s,SHUT_WR)" and keeping the
socket open until all remaining data has been read from it.

6.2.1. SUBSCRIBE Request

A SUBSCRIBE request begins with the standard DSO 12-byte header
[RFC8490], followed by the SUBSCRIBE primary TLV. A SUBSCRIBE
request message is illustrated in Figure 1.

The MESSAGE ID field MUST be set to a unique value, that the client
is not using for any other active operation on this DSO session. For
the purposes here, a MESSAGE ID is in use on this session if the
client has used it in a request for which it has not yet received a
response, or if the client has used it for a subscription which it
has not yet cancelled using UNSUBSCRIBE. In the SUBSCRIBE response
the server MUST echo back the MESSAGE ID value unchanged.

The other header fields MUST be set as described in the DSO spec-
ification [RFC8490]. The DNS OPCODE field contains the OPCODE value
for DNS Stateful Operations (6). The four count fields MUST be zero,
and the corresponding four sections MUST be empty (i.e., absent).

The DSO-TYPE is SUBSCRIBE (tentatively 0x40).

The DSO-LENGTH is the length of the DSO-DATA that follows, which
specifies the name, type, and class of the record(s) being sought.

```
                           1 1 1 1 1 1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
   |                  MESSAGE ID                   |   \
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
   |QR| OPCODE(6) |       Z         |    RCODE    |    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
   |            QDCOUNT (MUST BE ZERO)            |    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > HEADER
   |            ANCOUNT (MUST BE ZERO)            |    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
   |            NSCOUNT (MUST BE ZERO)            |    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
   |            ARCOUNT (MUST BE ZERO)            |   /
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
   |     DSO-TYPE = SUBSCRIBE (tentatively 0x40)  |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
   |   DSO-LENGTH (number of octets in DSO-DATA)  |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
   |                                              |   \
   \                    NAME                      \    |
   \                                              \    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > DSO-DATA
   |                    TYPE                      |    |
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
   |                    CLASS                     |   /
   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
```
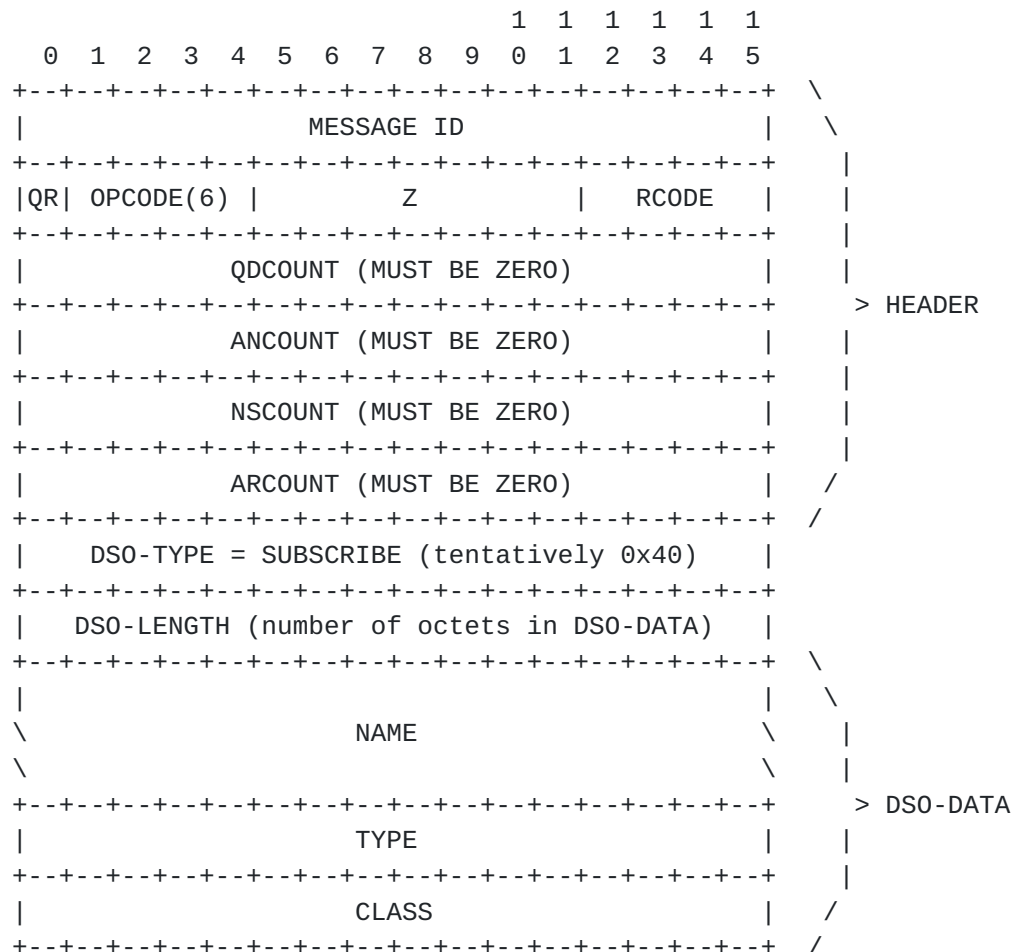
Figure 1: SUBSCRIBE Request

The DSO-DATA for a SUBSCRIBE request MUST contain exactly one NAME,
TYPE, and CLASS. Since SUBSCRIBE requests are sent over TCP,
multiple SUBSCRIBE DSO request messages can be concatenated in a
single TCP stream and packed efficiently into TCP segments.

If accepted, the subscription will stay in effect until the client
cancels the subscription using UNSUBSCRIBE or until the DSO session
between the client and the server is closed.

SUBSCRIBE requests on a given session MUST be unique. A client MUST
NOT send a SUBSCRIBE message that duplicates the NAME, TYPE and CLASS
of an existing active subscription on that DSO session. For the
purpose of this matching, the established DNS case-insensitivity for
US-ASCII letters applies (e.g., "example.com" and "Example.com" are
the same). If a server receives such a duplicate SUBSCRIBE message

   this is a fatal error and the server MUST immediately terminate the
   connection with a TLS close_notify alert followed by a TCP FIN.

   DNS wildcarding is not supported. That is, a wildcard ("*") in a
   SUBSCRIBE message matches only a literal wildcard character ("*") in
   the zone, and nothing else.

   Aliasing is not supported. That is, a CNAME in a SUBSCRIBE message
   matches only a literal CNAME record in the zone, and not to any
   records referenced by the owner name.

   A client may SUBSCRIBE to records that are unknown to the server at
   the time of the request (providing that the name falls within one of
   the zone(s) the server is responsible for) and this is not an error.
   The server MUST NOT return NXDOMAIN in this case. The server MUST
   accept these requests and send Push Notifications if and when
   matching records are found in the future.

   If neither TYPE nor CLASS are ANY (255) then this is a specific
   subscription to changes for the given NAME, TYPE and CLASS. If one
   or both of TYPE or CLASS are ANY (255) then this subscription matches
   any type and/or any class, as appropriate.

   NOTE: A little-known quirk of DNS is that in DNS QUERY requests,
   QTYPE and QCLASS 255 mean "ANY" not "ALL". They indicate that the
   server should respond with ANY matching records of its choosing, not
   necessarily ALL matching records. This can lead to some surprising
   and unexpected results, where a query returns some valid answers but
   not all of them, and makes QTYPE=ANY queries less useful than people
   sometimes imagine.

   When used in conjunction with SUBSCRIBE, TYPE and CLASS 255 should be
   interpreted to mean "ALL", not "ANY". After accepting a subscription
   where one or both of TYPE or CLASS are 255, the server MUST send Push
   Notification Updates for ALL record changes that match the
   subscription, not just some of them.

6.2.2. **SUBSCRIBE Response**

   Each SUBSCRIBE request generates exactly one SUBSCRIBE response from
   the server.

   A SUBSCRIBE response begins with the standard DSO 12-byte header
   [RFC8490]. The QR bit in the header is set indicating it is a
   response. The header MAY be followed by one or more optional TLVs,
   such as a Retry Delay TLV.

   The MESSAGE ID field MUST echo the value given in the MESSAGE ID
   field of the SUBSCRIBE request. This is how the client knows which
   request is being responded to.

   A SUBSCRIBE response message MUST NOT include a SUBSCRIBE TLV. If a
   client receives a SUBSCRIBE response message containing a SUBSCRIBE
   TLV then the response message is processed but the SUBSCRIBE TLV MUST
   be silently ignored.

```
                                1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
    |                  MESSAGE ID                   |   \
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |QR| OPCODE(6) |         Z           |   RCODE   |   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |             QDCOUNT (MUST BE ZERO)            |   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > HEADER
    |             ANCOUNT (MUST BE ZERO)            |   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |             NSCOUNT (MUST BE ZERO)            |   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |             ARCOUNT (MUST BE ZERO)            |   /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
```
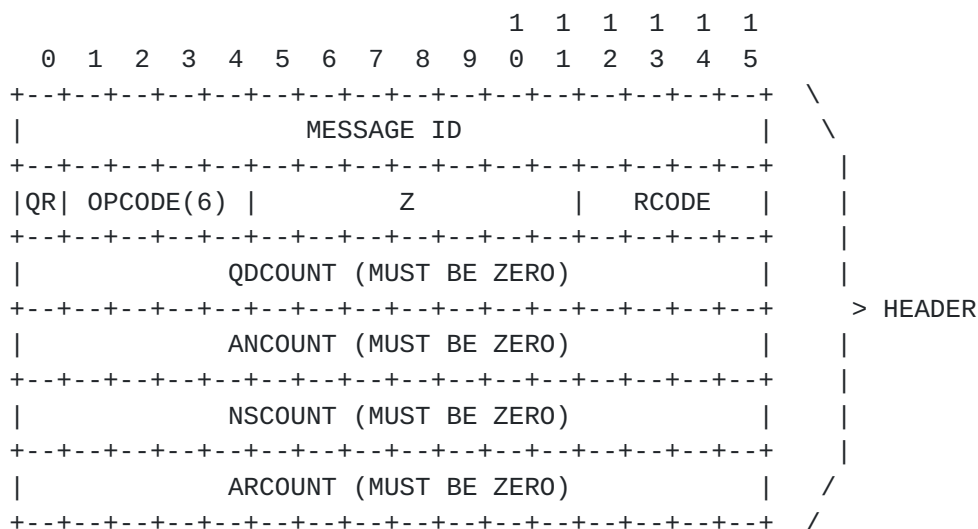
                    Figure 2: SUBSCRIBE Response Message

In the SUBSCRIBE response the RCODE indicates whether or not the
subscription was accepted. Supported RCODEs are as follows:

```
+-----------+-------+-----------------------------------------------+
| Mnemonic  | Value | Description                                   |
+-----------+-------+-----------------------------------------------+
| NOERROR   |   0   | SUBSCRIBE successful.                         |
| FORMERR   |   1   | Server failed to process request due to a     |
|           |       | malformed request.                            |
| SERVFAIL  |   2   | Server failed to process request due to a     |
|           |       | problem with the server.                      |
| NOTIMP    |   4   | Server does not implement DSO.                |
| REFUSED   |   5   | Server refuses to process request for policy  |
|           |       | or security reasons.                          |
| NOTAUTH   |   9   | Server is not authoritative for the requested |
|           |       | name.                                         |
| DSOTYPENI |  11   | SUBSCRIBE operation not supported.            |
+-----------+-------+-----------------------------------------------+
```

Table 1: SUBSCRIBE Response codes

This document specifies only these RCODE values for SUBSCRIBE
Responses. Servers sending SUBSCRIBE Responses SHOULD use one of
these values. Note that NXDOMAIN is not a valid RCODE in response to
a SUBSCRIBE Request. However, future circumstances may create
situations where other RCODE values are appropriate in SUBSCRIBE
Responses, so clients MUST be prepared to accept SUBSCRIBE Responses
with any other RCODE value.

If the server sends a nonzero RCODE in the SUBSCRIBE response, that
means:

a. the client is (at least partially) misconfigured,
b. the server resources are exhausted, or
c. there is some other unknown failure on the server.

In any case, the client shouldn't retry the subscription to this
server right away. If multiple SRV records were returned as
described in Section 6.1, Paragraph 7, a subsequent server can be
tried immediately.

If the client has other successful subscriptions to this server,
these subscriptions remain even though additional subscriptions may
be refused. Neither the client nor the server are required to close
the connection, although, either end may choose to do so.

If the server sends a nonzero RCODE then it SHOULD append a Retry
Delay TLV [RFC8490] to the response specifying a delay before the

client attempts this operation again. Recommended values for the
delay for different RCODE values are given below. These recommended
values apply both to the default values a server should place in the
Retry Delay TLV, and the default values a client should assume if the
server provides no Retry Delay TLV.

   For RCODE = 1 (FORMERR) the delay may be any value selected by the
   implementer. A value of five minutes is RECOMMENDED, to reduce
   the risk of high load from defective clients.

   For RCODE = 2 (SERVFAIL) the delay should be chosen according to
   the level of server overload and the anticipated duration of that
   overload. By default, a value of one minute is RECOMMENDED. If a
   more serious server failure occurs, the delay may be longer in
   accordance with the specific problem encountered.

   For RCODE = 4 (NOTIMP), which occurs on a server that doesn't
   implement DNS Stateful Operations [RFC8490], it is unlikely that
   the server will begin supporting DSO in the next few minutes, so
   the retry delay SHOULD be one hour. Note that in such a case, a
   server that doesn't implement DSO is unlikely to place a Retry
   Delay TLV in its response, so this recommended value in particular
   applies to what a client should assume by default.

   For RCODE = 5 (REFUSED), which occurs on a server that implements
   DNS Push Notifications, but is currently configured to disallow
   DNS Push Notifications, the retry delay may be any value selected
   by the implementer and/or configured by the operator.

   If the server being queried is listed in a
   "_dns-push-tls._tcp.<zone>" SRV record for the zone, then this is
   a misconfiguration, since this server is being advertised as
   supporting DNS Push Notifications for this zone, but the server
   itself is not currently configured to perform that task. Since it
   is possible that the misconfiguration may be repaired at any time,
   the retry delay should not be set too high. By default, a value
   of 5 minutes is RECOMMENDED.

   For RCODE = 9 (NOTAUTH), which occurs on a server that implements
   DNS Push Notifications, but is not configured to be authoritative
   for the requested name, the retry delay may be any value selected
   by the implementer and/or configured by the operator.

   If the server being queried is listed in a
   "_dns-push-tls._tcp.<zone>" SRV record for the zone, then this is
   a misconfiguration, since this server is being advertised as
   supporting DNS Push Notifications for this zone, but the server
   itself is not currently configured to perform that task. Since it

is possible that the misconfiguration may be repaired at any time,
the retry delay should not be set too high. By default, a value
of 5 minutes is RECOMMENDED.

For RCODE = 11 (DSOTYPENI), which occurs on a server that
implements DSO but doesn't implement DNS Push Notifications, it is
unlikely that the server will begin supporting DNS Push
Notifications in the next few minutes, so the retry delay SHOULD
be one hour.

For other RCODE values, the retry delay should be set by the
server as appropriate for that error condition. By default, a
value of 5 minutes is RECOMMENDED.

For RCODE = 9 (NOTAUTH), the time delay applies to requests for other
names falling within the same zone. Requests for names falling
within other zones are not subject to the delay. For all other
RCODEs the time delay applies to all subsequent requests to this
server.

After sending an error response the server MAY allow the session to
remain open, or MAY send a DNS Push Notification Retry Delay
Operation TLV instructing the client to close the session, as
described in the DSO specification [RFC8490]. Clients MUST correctly
handle both cases.

6.3. DNS Push Notification Updates

Once a subscription has been successfully established, the server
generates PUSH messages to send to the client as appropriate. In the
case that the answer set was already non-empty at the moment the
subscription was established, an initial PUSH message will be sent
immediately following the SUBSCRIBE Response. Subsequent changes to
the answer set are then communicated to the client in subsequent PUSH
messages.

6.3.1. PUSH Message

A PUSH unidirectional message begins with the standard DSO 12-byte
header [RFC8490], followed by the PUSH primary TLV. A PUSH message
is illustrated in Figure 3.

In accordance with the definition of DSO unidirectional messages, the
MESSAGE ID field MUST be zero. There is no client response to a PUSH
message.

The other header fields MUST be set as described in the DSO spec-
ification [RFC8490]. The DNS OPCODE field contains the OPCODE value
for DNS Stateful Operations (6). The four count fields MUST be zero,
and the corresponding four sections MUST be empty (i.e., absent).

The DSO-TYPE is PUSH (tentatively 0x41).

The DSO-LENGTH is the length of the DSO-DATA that follows, which
specifies the changes being communicated.

The DSO-DATA contains one or more change notifications. A PUSH
Message MUST contain at least one change notification. If a PUSH
Message is received that contains no change notifications, this is a
fatal error, and the receiver MUST immediately terminate the
connection with a TLS close_notify alert followed by a TCP FIN.

The change notification records are formatted similarly to how DNS
Resource Records are conventionally expressed in DNS messages, as
illustrated in Figure 3, and are interpreted as described below.

The TTL field holds an unsigned 32-bit integer [RFC2181]. If the TTL
is in the range 0 to 2,147,483,647 seconds ($2^{31} - 1$, or 0x7FFFFFFF),
then a new DNS Resource Record with the given name, type, class and
RDATA is added. A TTL of 0 means that this record should be retained
for as long as the subscription is active, and should be discarded
immediately the moment the subscription is cancelled.

   If the TTL has the value 0xFFFFFFFF, then the DNS Resource Record
   with the given name, type, class and RDATA is removed.

   If the TTL has the value 0xFFFFFFFE, then this is a 'collective'
   remove notification. For collective remove notifications RDLEN MUST
   be zero and consequently the RDATA MUST be empty. If a change
   notification is received where TTL = 0xFFFFFFFE and RDLEN is not
   zero, this is a fatal error, and the receiver MUST immediately
   terminate the connection with a TLS close_notify alert followed by a
   TCP FIN. There are three types of collective remove notification:

   For collective remove notifications, if CLASS is 255 (ANY), then for
   the given name this deletes all records of all types in all classes.
   In this case TYPE MUST be set to zero on transmission, and MUST be
   silently ignored on reception.

   For collective remove notifications, if CLASS is not 255 (ANY) and
   TYPE is 255 (ANY) then for the given name this deletes all records of
   all types in the specified class.

   For collective remove notifications, if CLASS is not 255 (ANY) and
   TYPE is not 255 (ANY) then for the given name this deletes all
   records of the specified type in the specified class.

   Summary of change notification types:

      Delete all RRsets from a name, in all classes
      TTL=0xFFFFFFFE, RDLENGTH=0, CLASS=255 (ANY)

      Delete all RRsets from a name, in given class:
      TTL=0xFFFFFFFE, RDLENGTH=0, CLASS specifies class, TYPE=255 (ANY)

      Delete specified RRset from a name, in given class:
      TTL=0xFFFFFFFE, RDLENGTH=0
      CLASS and TYPE specify the RRset being deleted

      Delete an individual RR from a name:
      TTL=0xFFFFFFFF
      CLASS, TYPE, RDLENGTH and RDATA specify the RR being deleted.

      Add individual RR to a name
      TTL>=0
      CLASS, TYPE, RDLENGTH, RDATA and TTL specify the RR being added.

   Note that it is valid for the RDATA of an added or removed DNS
   Resource Record to be empty (zero length). For example, an Address
   Prefix List Resource Record [RFC3123] may have empty RDATA.
   Therefore, a change notification with RDLEN=0 does not automatically

indicate a remove notification. If RDLEN=0 and TTL is the in the
range 0 - 0x7FFFFFFF, this change notification signals the addition
of a record with the given name, type, class, and empty RDATA. If
RDLEN=0 and TTL = 0xFFFFFFFF, this change notification signals the
removal specifically of that single record with the given name, type,
class, and empty RDATA.

If the TTL is any value other than 0xFFFFFFFF, 0xFFFFFFFE, or a value
in the range 0 - 0x7FFFFFFF, then the receiver SHOULD silently ignore
this particular change notification record. The connection is not
terminated and other valid change notification records within this
PUSH message are processed as usual.

For efficiency, when generating a PUSH message, a server SHOULD
include as many change notifications as it has immediately available
to send, rather than sending each change notification as a separate
DSO message. Once it has exhausted the list of change notifications
immediately available to send, a server SHOULD then send the PUSH
message immediately, rather than waiting to see if additional change
notifications become available.

For efficiency, when generating a PUSH message, a server SHOULD use
standard DNS name compression, with offsets relative to the beginning
of the DNS message [RFC1035]. When multiple change notifications in
a single PUSH message have the same owner name, this name compression
can yield significant savings. Name compression should be performed
as specified in Section 18.14 of the Multicast DNS specification
[RFC6762], namely, owner names should always be compressed, and names
appearing within RDATA should be compressed for only the RR types
listed below:

   NS, CNAME, PTR, DNAME, SOA, MX, AFSDB, RT, KX, RP, PX, SRV, NSEC

Servers may generate PUSH messages up to a maximum DNS message length
of 16,382 bytes, counting from the start of the DSO 12-byte header.
Including the two-byte length prefix that is used to frame DNS over a
byte stream like TLS, this makes a total of 16,384 bytes. Servers
MUST NOT generate PUSH messages larger than this. Where the
immediately available change notifications are sufficient to exceed a
DNS message length of 16,382 bytes, the change notifications MUST be
communicated in separate PUSH messages of up to 16,382 bytes each.
DNS name compression becomes less effective for messages larger than
16,384 bytes, so little efficiency benefit is gained by sending
messages larger than this.

If a client receives a PUSH message with a DNS message length larger
than 16,382 bytes, the this is a fatal error, and the receiver MUST

immediately terminate the connection with a TLS close_notify alert
followed by a TCP FIN.

```
                                1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
    |              MESSAGE ID (MUST BE ZERO)        |   \
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |QR| OPCODE(6) |        Z        |    RCODE     |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |            QDCOUNT (MUST BE ZERO)             |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > HEADER
    |            ANCOUNT (MUST BE ZERO)             |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |            NSCOUNT (MUST BE ZERO)             |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |            ARCOUNT (MUST BE ZERO)             |   /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
    |       DSO-TYPE = PUSH (tentatively 0x41)      |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |   DSO-LENGTH (number of octets in DSO-DATA)   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
    \                    NAME                      \   \
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |                    TYPE                       |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |                    CLASS                      |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |                    TTL                        |    |
    |      (32-bit unsigned big-endian integer)     |   > DSO-DATA
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    |                    RDLEN                      |    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    \          RDATA (sized as necessary)          \    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
    :      NAME, TYPE, CLASS, TTL, RDLEN, RDATA     :    |
    :             Repeated As Necessary            :   /
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
```

Figure 3: PUSH Message

When processing the records received in a PUSH Message, the receiving
client MUST validate that the records being added or deleted
correspond with at least one currently active subscription on that
session. Specifically, the record name MUST match the name given in
a SUBSCRIBE request, subject to the usual established DNS case-
insensitivity for US-ASCII letters. If the TYPE in the SUBSCRIBE
request was not ANY (255) then the TYPE of the record must match the

TYPE given in the SUBSCRIBE request. If the CLASS in the SUBSCRIBE
request was not ANY (255) then the CLASS of the record must match the
CLASS given in the SUBSCRIBE request. If a matching active
subscription on that session is not found, then that individual
record addition/deletion is silently ignored. Processing of other
additions and deletions in this message is not affected. The DSO
session is not closed. This is to allow for the unavoidable race
condition where a client sends an outbound UNSUBSCRIBE while inbound
PUSH messages for that subscription from the server are still in
flight.

In the case where a single change affects more than one active
subscription, only one PUSH message is sent. For example, a PUSH
message adding a given record may match both a SUBSCRIBE request with
the same TYPE and a different SUBSCRIBE request with TYPE=ANY (255).
It is not the case that two PUSH messages are sent because the new
record matches two active subscriptions.

The server SHOULD encode change notifications in the most efficient
manner possible. For example, when three AAAA records are deleted
from a given name, and no other AAAA records exist for that name, the
server SHOULD send a "delete an RRset from a name" PUSH message, not
three separate "delete an individual RR from a name" PUSH messages.
Similarly, when both an SRV and a TXT record are deleted from a given
name, and no other records of any kind exist for that name, the
server SHOULD send a "delete all RRsets from a name" PUSH message,
not two separate "delete an RRset from a name" PUSH messages.

A server SHOULD combine multiple change notifications in a single
PUSH message when possible, even if those change notifications apply
to different subscriptions. Conceptually, a PUSH message is a
session-level mechanism, not a subscription-level mechanism.

The TTL of an added record is stored by the client. While the
subscription is active, the TTL is not decremented, because a change
to the TTL would produce a new update. For as long as a relevant
subscription remains active, the client SHOULD assume that when a
record goes away the server will notify it of that fact.
Consequently, a client does not have to poll to verify that the
record is still there. Once a subscription is cancelled
(individually, or as a result of the DSO session being closed) record
aging for records covered by the subscription resumes and records are
removed from the local cache when their TTL reaches zero.

6.4. DNS Push Notification UNSUBSCRIBE

To cancel an individual subscription without closing the entire DSO
session, the client sends an UNSUBSCRIBE message over the established
DSO session to the server. The UNSUBSCRIBE message is encoded as a
DSO unidirectional message [RFC8490]. This specification defines a
primary unidirectional DSO TLV for DNS Push Notification UNSUBSCRIBE
Messages (tentatively DSO Type Code 0x42).

A server MUST NOT initiate an UNSUBSCRIBE message. If a server does
send an UNSUBSCRIBE message over a DSO session initiated by a client,
this is a fatal error and the client should immediately abort the
connection with a TLS close_notify alert followed by a TCP FIN.

6.4.1. UNSUBSCRIBE Message

An UNSUBSCRIBE unidirectional message begins with the standard DSO
12-byte header [RFC8490], followed by the UNSUBSCRIBE primary TLV.
An UNSUBSCRIBE message is illustrated in Figure 4.

In accordance with the definition of DSO unidirectional messages, the
MESSAGE ID field MUST be zero. There is no server response to an
UNSUBSCRIBE message.

The other header fields MUST be set as described in the DSO spec-
ification [RFC8490]. The DNS OPCODE field contains the OPCODE value
for DNS Stateful Operations (6). The four count fields MUST be zero,
and the corresponding four sections MUST be empty (i.e., absent).

The DSO-TYPE is UNSUBSCRIBE (tentatively 0x42).

The DSO-LENGTH field contains the value 2, the length of the 2-octet
MESSAGE ID contained in the DSO-DATA.

The DSO-DATA contains the value given in the MESSAGE ID field of an
active SUBSCRIBE request. This is how the server knows which
SUBSCRIBE request is being cancelled. After receipt of the
UNSUBSCRIBE message, the SUBSCRIBE request is no longer active.

It is allowable for the client to issue an UNSUBSCRIBE message for a
previous SUBSCRIBE request for which the client has not yet received
a SUBSCRIBE response. This is to allow for the case where a client
starts and stops a subscription in less than the round-trip time to
the server. The client is NOT required to wait for the SUBSCRIBE
response before issuing the UNSUBSCRIBE message.

Consequently, it is possible for a server to receive an UNSUBSCRIBE
message that does not match any currently active subscription. This

can occur when a client sends a SUBSCRIBE request, which subsequently
fails and returns an error code, but the client sent an UNSUBSCRIBE
message before it became aware that the SUBSCRIBE request had failed.
Because of this, servers MUST silently ignore UNSUBSCRIBE messages
that do not match any currently active subscription.

```
                              1 1 1 1 1 1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
  |            MESSAGE ID (MUST BE ZERO)          |   \
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
  |QR| OPCODE(6) |        Z         |   RCODE    |    |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
  |            QDCOUNT (MUST BE ZERO)            |    |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > HEADER
  |            ANCOUNT (MUST BE ZERO)            |    |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
  |            NSCOUNT (MUST BE ZERO)            |    |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
  |            ARCOUNT (MUST BE ZERO)            |   /
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
  |   DSO-TYPE = UNSUBSCRIBE (tentatively 0x42)  |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  |              DSO-LENGTH (2)                  |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
  |              SUBSCRIBE MESSAGE ID            |   > DSO-DATA
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
```
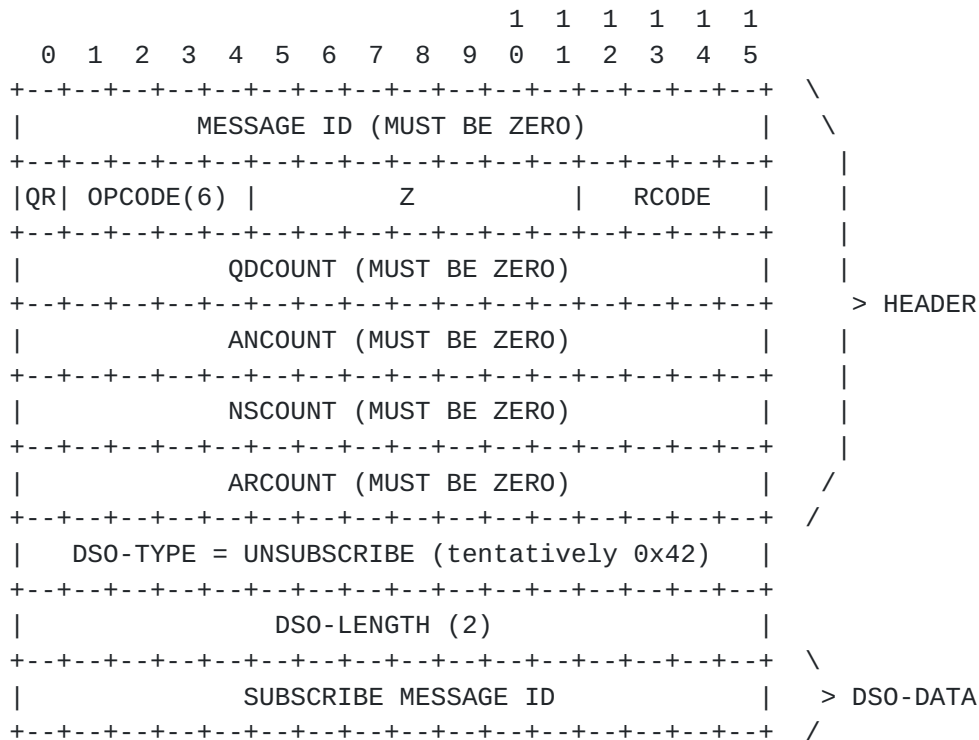
Figure 4: UNSUBSCRIBE Message

6.5. **DNS Push Notification RECONFIRM**

   Sometimes, particularly when used with a Discovery Proxy [DisProx], a
   DNS Zone may contain stale data. When a client encounters data that
   it believes may be stale (e.g., an SRV record referencing a target
   host+port that is not responding to connection requests) the client
   can send a RECONFIRM message to ask the server to re-verify that the
   data is still valid. For a Discovery Proxy, this causes it to issue
   new Multicast DNS queries to ascertain whether the target device is
   still present. How the Discovery Proxy causes these new Multicast
   DNS queries to be issued depends on the details of the underlying
   Multicast DNS implementation being used. For example, a Discovery
   Proxy built on Apple's dns_sd.h API responds to a DNS Push
   Notification RECONFIRM message by calling the underlying API's
   DNSServiceReconfirmRecord() routine.

   For other types of DNS server, the RECONFIRM operation is currently
   undefined, and SHOULD result in a NOERROR response, but otherwise
   need not cause any action to occur.

   Frequent use of RECONFIRM operations may be a sign of network
   unreliability, or some kind of misconfiguration, so RECONFIRM
   operations MAY be logged or otherwise communicated to a human
   administrator to assist in detecting, and remedying, such network
   problems.

   If, after receiving a valid RECONFIRM message, the server determines
   that the disputed records are in fact no longer valid, then
   subsequent DNS PUSH Messages will be generated to inform interested
   clients. Thus, one client discovering that a previously-advertised
   device (like a network printer) is no longer present has the side
   effect of informing all other interested clients that the device in
   question is now gone.

6.5.1. RECONFIRM Message

A RECONFIRM unidirectional message begins with the standard DSO
12-byte header [RFC8490], followed by the RECONFIRM primary TLV.
A RECONFIRM message is illustrated in Figure 5.

In accordance with the definition of DSO unidirectional messages, the
MESSAGE ID field MUST be zero. There is no server response to a
RECONFIRM message.

The other header fields MUST be set as described in the DSO spec-
ification [RFC8490]. The DNS OPCODE field contains the OPCODE value
for DNS Stateful Operations (6). The four count fields MUST be zero,
and the corresponding four sections MUST be empty (i.e., absent).

The DSO-TYPE is RECONFIRM (tentatively 0x43).

The DSO-LENGTH is the length of the data that follows, which
specifies the name, type, class, and content of the record being
disputed.

```
                                 1 1 1 1 1 1
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
      |                 MESSAGE ID                    |   \
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      |QR| OPCODE(6) |        Z          |   RCODE    |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      |             QDCOUNT (MUST BE ZERO)            |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > HEADER
      |             ANCOUNT (MUST BE ZERO)            |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      |             NSCOUNT (MUST BE ZERO)            |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      |             ARCOUNT (MUST BE ZERO)            |   /
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
      |     DSO-TYPE = RECONFIRM (tentatively 0x43)   |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      |   DSO-LENGTH (number of octets in DSO-DATA)   |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  \
      \                  NAME                         \   \
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      |                  TYPE                         |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   > DSO-DATA
      |                  CLASS                        |    |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+    |
      \                  RDATA                        \   /
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+  /
```
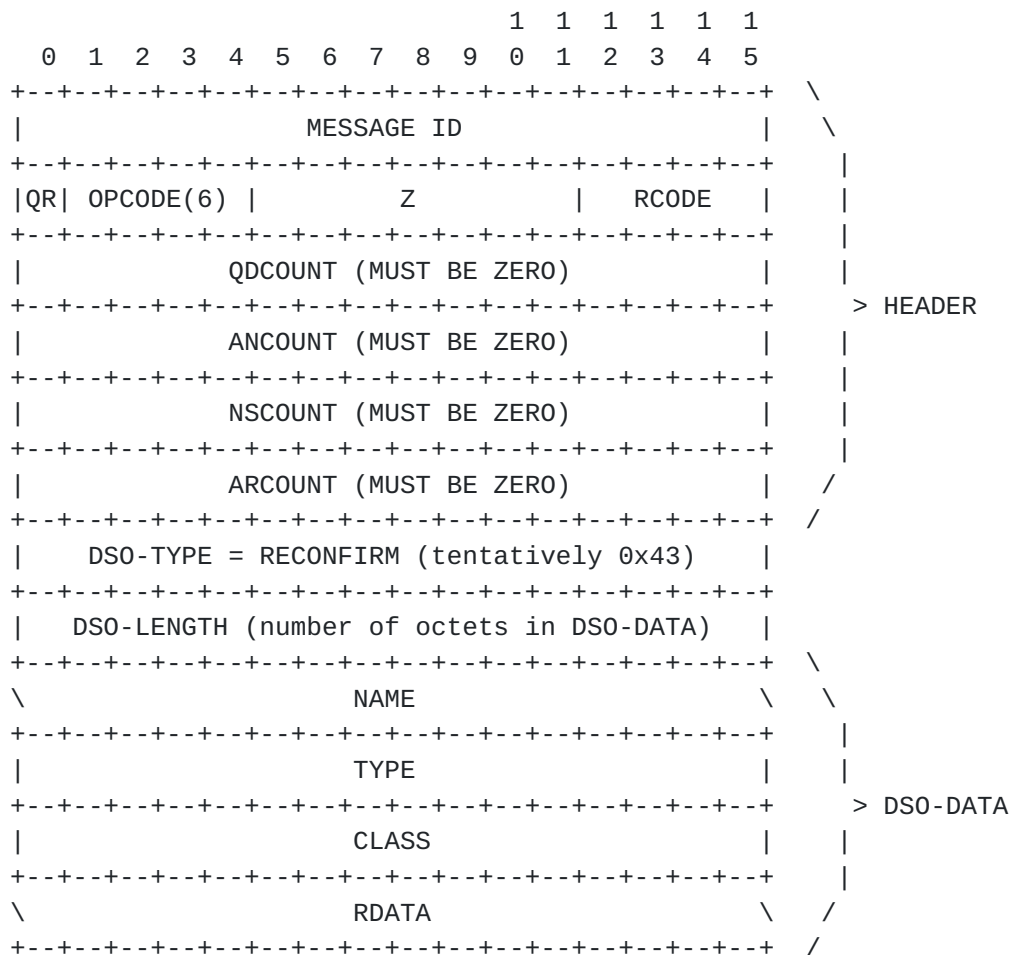
Figure 5: RECONFIRM Message

The DSO-DATA for a RECONFIRM message MUST contain exactly one record.
The DSO-DATA for a RECONFIRM message has no count field to specify
more than one record. Since RECONFIRM messages are sent over TCP,
multiple RECONFIRM messages can be concatenated in a single TCP
stream and packed efficiently into TCP segments.

TYPE MUST NOT be the value ANY (255) and CLASS MUST NOT be the value
ANY (255).

DNS wildcarding is not supported. That is, a wildcard ("*") in a
RECONFIRM message matches only a literal wildcard character ("*") in
the zone, and nothing else.

Aliasing is not supported. That is, a CNAME in a RECONFIRM message
matches only a literal CNAME record in the zone, and nothing else.

6.6. DNS Stateful Operations TLV Context Summary

   This document defines four new DSO TLVs. As suggested in Section 8.2
   of the DNS Stateful Operations specification [RFC8490], the valid
   contexts of these new TLV types are summarized below.

   The client TLV contexts are:

   C-P: Client request message, primary TLV
   C-U: Client unidirectional message, primary TLV
   C-A: Client request or unidirectional message, additional TLV
   CRP: Response back to client, primary TLV
   CRA: Response back to client, additional TLV

```
            +-------------+-----+-----+-----+-----+-----+
            |    TLV Type | C-P | C-U | C-A | CRP | CRA |
            +-------------+-----+-----+-----+-----+-----+
            |   SUBSCRIBE |  X  |     |     |     |     |
            |        PUSH |     |     |     |     |     |
            | UNSUBSCRIBE |     |  X  |     |     |     |
            |   RECONFIRM |     |  X  |     |     |     |
            +-------------+-----+-----+-----+-----+-----+
```

              Table 2: DSO TLV Client Context Summary

   The server TLV contexts are:

   S-P: Server request message, primary TLV
   S-U: Server unidirectional message, primary TLV
   S-A: Server request or unidirectional message, additional TLV
   SRP: Response back to server, primary TLV
   SRA: Response back to server, additional TLV

```
            +-------------+-----+-----+-----+-----+-----+
            |    TLV Type | S-P | S-U | S-A | SRP | SRA |
            +-------------+-----+-----+-----+-----+-----+
            |   SUBSCRIBE |     |     |     |     |     |
            |        PUSH |     |  X  |     |     |     |
            | UNSUBSCRIBE |     |     |     |     |     |
            |   RECONFIRM |     |     |     |     |     |
            +-------------+-----+-----+-----+-----+-----+
```

              Table 3: DSO TLV Server Context Summary

6.7. **Client-Initiated Termination**

   An individual subscription is terminated by sending an UNSUBSCRIBE
   TLV for that specific subscription, or all subscriptions can be
   cancelled at once by the client closing the DSO session. When a
   client terminates an individual subscription (via UNSUBSCRIBE) or all
   subscriptions on that DSO session (by ending the session) it is
   signaling to the server that it is longer interested in receiving
   those particular updates. It is informing the server that the server
   may release any state information it has been keeping with regards to
   these particular subscriptions.

   After terminating its last subscription on a session via UNSUBSCRIBE,
   a client MAY close the session immediately, or it may keep it open if
   it anticipates performing further operations on that session in the
   future. If a client wishes to keep an idle session open, it MUST
   respect the maximum idle time required by the server [RFC8490].

   If a client plans to terminate one or more subscriptions on a session
   and doesn't intend to keep that session open, then as an efficiency
   optimization it MAY instead choose to simply close the session, which
   implicitly terminates all subscriptions on that session. This may
   occur because the client computer is being shut down, is going to
   sleep, the application requiring the subscriptions has terminated, or
   simply because the last active subscription on that session has been
   cancelled.

   When closing a session, a client should perform an orderly close of
   the TLS session in order to allow for future TLS session resumption
   with the server (if available). See Section 7.4 below. Typical APIs
   will provide a session close method that will send a TLS close_notify
   alert. This instructs the recipient that the sender will not send
   any more data over the session.

   If the session is forcibly closed at the TCP level by sending a RST
   from either end of the connection, data may be lost and TLS session
   resumption of this session will not be possible.

7. Security Considerations

The Strict Privacy Usage Profile for DNS over TLS is REQUIRED for DNS Push Notifications [RFC8310]. Cleartext connections for DNS Push Notifications are not permissible. Since this is a new protocol, transition mechanisms from the Opportunistic Privacy profile are unnecessary.

Also, see Section 9 of the DNS over (D)TLS Usage Profiles document [RFC8310] for additional recommendations for various versions of TLS usage.

As a consequence of requiring TLS, client certificate authentication and verification may also be enforced by the server for stronger client-server security or end-to-end security. However, recommendations for security in particular deployment scenarios are outside the scope of this document.

DNSSEC is RECOMMENDED for the authentication of DNS Push Notification servers. TLS alone does not provide complete security. TLS certificate verification can provide reasonable assurance that the client is really talking to the server associated with the desired host name, but since the desired host name is learned via a DNS SRV query, if the SRV query is subverted then the client may have a secure connection to a rogue server. DNSSEC can provided added confidence that the SRV query has not been subverted.

7.1. Security Services

It is the goal of using TLS to provide the following security services:

Confidentiality: All application-layer communication is encrypted with the goal that no party should be able to decrypt it except the intended receiver.

Data integrity protection: Any changes made to the communication in transit are detectable by the receiver.

Authentication: An end-point of the TLS communication is authenticated as the intended entity to communicate with.

Anti-replay protection: TLS provides for the detection of and prevention against messages sent previously over a TLS connection (such as DNS Push Notifications). Prior messages cannot be re-sent at a later time as a form of a man-in-the-middle attack.

Deployment recommendations on the appropriate key lengths and cypher
suites are beyond the scope of this document. Please refer to TLS
Recommendations [RFC7525] for the best current practices. Keep in
mind that best practices only exist for a snapshot in time and
recommendations will continue to change. Updated versions or errata
may exist for these recommendations.

7.2. TLS Name Authentication

As described in Section 6.1, the client discovers the DNS Push
Notification server using an SRV lookup for the record name
"_dns-push-tls._tcp.<zone>". The server connection endpoint SHOULD
then be authenticated using DANE TLSA records for the associated SRV
record. This associates the target's name and port number with a
trusted TLS certificate [RFC7673]. This procedure uses the TLS
Server Name Indication (SNI) extension [RFC6066] to inform the server
of the name the client has authenticated through the use of TLSA
records. Therefore, if the SRV record passes DNSSEC validation and a
TLSA record matching the target name is useable, an SNI extension
must be used for the target name to ensure the client is connecting
to the server it has authenticated. If the target name does not have
a usable TLSA record, then the use of the SNI extension is optional.
See Usage Profiles for DNS over TLS and DNS over DTLS [RFC8310] for
more information on authenticating domain names.

7.3. TLS Early Data

DSO messages with the SUBSCRIBE TLV as the Primary TLV are permitted
in TLS early data. Using TLS early data can save one network round
trip, and can result in obtaining results faster. However, there are
some factors to consider before using TLS early data.

TLS Early Data is not forward secret. In cases where forward secrecy
of DNS Push Notification subscriptions is required, the client should
not use TLS Early Data.

With TLS early data there are no guarantees of non-replay between
connections. If packets are duplicated and delayed in the network,
the later arrivals could be mistaken for new subscription requests.
Generally this is not a major concern, since the amount of state
generated on the server for these spurious subscriptions is small and
short-lived, since the TCP connection will not complete the three-way
handshake. Servers MAY choose to implement rate-limiting measures
that are imposed when the server detects excessive numbers of
spurious subscription requests.

For further guidance please see Section 2.3, Section 8, and
Appendix E.5 of the TLS 1.3 specification [RFC8446].

7.4. TLS Session Resumption

TLS Session Resumption is permissible on DNS Push Notification
servers. The server may keep TLS state with Session IDs [RFC8446] or
operate in stateless mode by sending a Session Ticket [RFC5077] to
the client for it to store. However, closing the TLS connection
terminates the DSO session. When the TLS session is resumed, the DNS
Push Notification server will not have any subscription state and
will proceed as with any other new DSO session. Use of TLS Session
Resumption may allow a TLS connection to be set up more quickly, but
the client will still have to recreate any desired subscriptions.

8. IANA Considerations

This document defines a new service name to be published in the IANA
Registry Service Types [RFC6335][ST] that is only applicable for the
TCP protocol.

Name	Port	Value	Definition
DNS Push Notification Service Type	None	"_dns-push-tls._tcp"	Section 6.1

Table 4: IANA Service Type Assignments

This document also defines four new DNS Stateful Operation TLV types
to be recorded in the IANA DSO Type Code Registry.

Name	Value	Early Data	Status	Definition
SUBSCRIBE	TBA (0x40)	OK	Standards Track	Section 6.2
PUSH	TBA (0x41)	NO	Standards Track	Section 6.3
UNSUBSCRIBE	TBA (0x42)	NO	Standards Track	Section 6.4
RECONFIRM	TBA (0x43)	NO	Standards Track	Section 6.5

Table 5: IANA DSO TLV Type Code Assignments

9. Acknowledgements

   The authors would like to thank Kiren Sekar and Marc Krochmal for
   previous work completed in this field.

   This draft has been improved due to comments from Ran Atkinson, Tim
   Chown, Mark Delany, Ralph Droms, Bernie Volz, Jan Komissar, Manju
   Shankar Rao, Markus Stenberg, Dave Thaler, Soraia Zlatkovic, Sara
   Dickinson, and Andrew Sullivan. Ted Lemon provided clarifying text
   that was greatly appreciated.

10. References

10.1. Normative References

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <https://www.rfc-editor.org/info/rfc768>.

   [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
              RFC 793, DOI 10.17487/RFC0793, September 1981,
              <https://www.rfc-editor.org/info/rfc793>.

   [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>.

   [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts -
              Application and Support", STD 3, RFC 1123,
              DOI 10.17487/RFC1123, October 1989,
              <https://www.rfc-editor.org/info/rfc1123>.

   [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>.

   [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>.

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

   [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>.

   [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>.

   [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>.

   [RFC7673] Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-
              Based Authentication of Named Entities (DANE) TLSA Records
              with SRV Records", RFC 7673, DOI 10.17487/RFC7673, October
              2015, <https://www.rfc-editor.org/info/rfc7673>.

   [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and
              D. Wessels, "DNS Transport over TCP - Implementation
              Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016,
              <https://www.rfc-editor.org/info/rfc7766>.

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

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

   [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S.,
              Lemon, T., and T. Pusateri, "DNS Stateful Operations",
              RFC 8490, DOI 10.17487/RFC8490, March 2019,
              <https://www.rfc-editor.org/info/rfc8490>.

   [ST]   "Service Name and Transport Protocol Port Number
               Registry", <http://www.iana.org/assignments/
               service-names-port-numbers/>.

10.2. Informative References

   [DisProx]   Cheshire, S., "Discovery Proxy for Multicast DNS-Based
               Service Discovery", draft-ietf-dnssd-hybrid-10 (work in
               progress), March 2019.

   [I-D.dukkipati-tcpm-tcp-loss-probe]
               Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis,
               "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of
               Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work
               in progress), February 2013.

   [LLQ]   Cheshire, S. and M. Krochmal, "DNS Long-Lived Queries",
               draft-sekar-dns-llq-03 (work in progress), March 2019.

   [obs]   "Observer Pattern",
               <https://en.wikipedia.org/wiki/Observer_pattern>.

   [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>.

   [RFC3123]   Koch, P., "A DNS RR Type for Lists of Address Prefixes
               (APL RR)", RFC 3123, DOI 10.17487/RFC3123, June 2001,
               <https://www.rfc-editor.org/info/rfc3123>.

   [RFC4287]   Nottingham, M., Ed. and R. Sayre, Ed., "The Atom
               Syndication Format", RFC 4287, DOI 10.17487/RFC4287,
               December 2005, <https://www.rfc-editor.org/info/rfc4287>.

   [RFC4953]   Touch, J., "Defending TCP Against Spoofing Attacks",
               RFC 4953, DOI 10.17487/RFC4953, July 2007,
               <https://www.rfc-editor.org/info/rfc4953>.

   [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>.

   [RFC6281]   Cheshire, S., Zhu, Z., Wakikawa, R., and L. Zhang,
               "Understanding Apple's Back to My Mac (BTMM) Service",
               RFC 6281, DOI 10.17487/RFC6281, June 2011,
               <https://www.rfc-editor.org/info/rfc6281>.

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

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,
              "TCP Extensions for Multipath Operation with Multiple
              Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013,
              <https://www.rfc-editor.org/info/rfc6824>.

   [RFC6886] Cheshire, S. and M. Krochmal, "NAT Port Mapping Protocol
              (NAT-PMP)", RFC 6886, DOI 10.17487/RFC6886, April 2013,
              <https://www.rfc-editor.org/info/rfc6886>.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
              DOI 10.17487/RFC6887, April 2013,
              <https://www.rfc-editor.org/info/rfc6887>.

   [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>.

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

   [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS
              Terminology", RFC 7719, DOI 10.17487/RFC7719, December
              2015, <https://www.rfc-editor.org/info/rfc7719>.

   [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>.

   [RFC8010] Sweet, M. and I. McDonald, "Internet Printing
              Protocol/1.1: Encoding and Transport", STD 92, RFC 8010,
              DOI 10.17487/RFC8010, January 2017,
              <https://www.rfc-editor.org/info/rfc8010>.

   [RFC8011] Sweet, M. and I. McDonald, "Internet Printing
              Protocol/1.1: Model and Semantics", STD 92, RFC 8011,
              DOI 10.17487/RFC8011, January 2017,
              <https://www.rfc-editor.org/info/rfc8011>.

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

   [SYN]   Eddy, W., "Defenses Against TCP SYN Flooding Attacks", The
              Internet Protocol Journal, Cisco Systems, Volume 9,
              Number 4, December 2006.

   [XEP0060] Millard, P., Saint-Andre, P., and R. Meijer, "Publish-
              Subscribe", XSF XEP 0060, July 2010.

Authors' Addresses

   Tom Pusateri
   Unaffiliated
   Raleigh, NC 27608
   USA

   Phone: +1 919 867 1330
   Email: pusateri@bangj.com

   Stuart Cheshire
   Apple Inc.
   One Apple Park Way
   Cupertino, CA 95014
   USA

   Phone: +1 (408) 996-1010
   Email: cheshire@apple.com