Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2017

T. Pusateri
Seeking affiliation
S. Cheshire
Apple Inc.

June 17, 2017

DNS Push Notifications
draft-ietf-dnssd-push-11

Abstract

The Domain Name System (DNS) was designed to return matching records efficiently for queries for data that is relatively static. When those records change frequently, DNS is still efficient at returning the updated results when polled. 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 http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on December 19, 2017.

Table of Contents

1. Introduction

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

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 to state changes 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 result in 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 new services and updates are sent 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) [I-D.sekar-dns-llq] mechanism is an existing deployed solution to provide asynchronous change notifications, used by Apple's Back to My Mac Service [RFC6281]. Back to My Mac was designed in an era when the data centre operations staff asserted that it was impossible for a server to handle large numbers of mostly-idle TCP connections, so LLQ had to 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 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 resorting to excessive keepalive traffic. Instead of inventing a new vocabulary of messages to communicate DNS zone changes as LLQ did, this specification adopts the syntax and semantics of DNS Update messages [RFC2136].

3. Overview

The existing DNS Update protocol [RFC2136] provides a mechanism for clients to add or delete individual resource records (RRs) or entire resource record sets (RRSets) on the zone's server.

This specification adopts a simplified subset of these existing syntax and semantics, and uses them for DNS Push Notification messages going in the opposite direction, from server to client, to communicate changes to a zone. The client subscribes for Push Notifications by connecting to the server and sending DNS message(s) indicating the RRSet(s) of interest. When the client loses interest in updates to these records, it unsubscribes.

The DNS Push Notification server for a zone is any server capable of generating the correct change notifications for a name. It may be a master, slave, 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 Queries.

Supporting DNS Updates and DNS Push Notifications on the same server is OPTIONAL. A DNS Push Notification server does NOT also have 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 connection, provided that these are queries for names falling within the server's zone (the <zone> in the "_dns-push-tls._tcp.<zone>" SRV record). The RD (Recursion Desired) bit MUST be zero. If a query is received with the RD bit set, matching records for names falling within the server's zones should be returned with the RA (Recursion Available) bit clear. If the query is for a name not in the server's zone, an error with RCODE NOTAUTH (Not Authoritative) should be returned.

DNS Push Notification clients are NOT required to implement DNS Update Prerequisite processing. Prerequisites are used to perform tentative atomic test-and-set type operations when a client updates records on a server, and that concept has no applicability when it comes to an authoritative server informing a client of changes to DNS records.

This DNS Push Notification specification includes support for DNS classes, for completeness. However, in practice, it is anticipated that for the foreseeable future the only DNS class in use will be DNS class "IN", as is the reality today with existing DNS servers and clients. A DNS Push Notification server MAY choose to implement only DNS class "IN". If messages are received for a class other than "IN", and that class is not supported, an error with RCODE NOTIMPL (Not Implemented) should be returned.

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. 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). Implementations MAY want to implement idle timeouts, so that if the user ceases interacting with the device, the display showing the result of the DNS Push Notification subscription is automatically dismissed after a certain period of inactivity. For example, if a user presses the "Print" button on their smartphone, and then leaves the phone showing the printer discovery screen until the phone goes to sleep, then the printer discovery screen should be automatically dismissed as the device goes to sleep. If the user does still intend to print, this will require them to press the "Print" button again when they wake their phone up.

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

4. Transport

Implementations of 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 TLS over TCP, see RFC 7858 [RFC7858].

Connection setup over TCP ensures return reachability and alleviates concerns of state overload at the server through anonymous subscriptions. 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 [IPJ.9-4-TCPSYN] [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) [RFC5246] is well understood and deployed across many protocols running over TCP. It is designed to prevent eavesdropping, tampering, or 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 connection 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 connections to alternate DNS servers that support DNS Push Notifications for the zone and distribute subscriptions at its discretion. In this way, both clients and servers can react to resource constraints. Token bucket rate limiting schemes are also effective in providing fairness by a server across numerous client requests.

6. Protocol Operation

The DNS Push Notification protocol is a session-oriented protocol, and makes use of DNS Session Signaling [SessSig].

For details of the DNS Session Signaling message format refer to the DNS Session Signaling specification [SessSig]. Those details are not repeated here.

DNS Push Notification clients and servers MUST support DNS Session Signaling, but the server SHOULD NOT issue any DNS Session Signaling operations until after the client has first initiated a DNS Session Signaling operation of its own. A single server can support DNS Queries, DNS Updates, and DNS Push Notifications (using DNS Session Signaling) on the same TCP port, and until the client has sent at least one DNS Session Signaling operation the server does not know what kind of client has connected to it. Once the client has indicated willingness to use DNS Session Signaling operations by sending one of its own, either side of the connection may then initiate further Session Signaling operations at any time.

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 DNS Session Signaling Keepalive operation to request a session timeout or keepalive interval longer than the the 15-second defaults, but this is not required. A DNS Push Notification client MAY issue other requests on the connection first, and only issue a DNS Session Signaling Keepalive operation later if it determines that to be necessary.

Once the connection is made, the client may then add and remove Push Notification subscriptions. 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.

The exchange between client and server terminates when either end closes the TCP connection with a TCP FIN or RST.

6.1. Discovery

The first step in DNS Push Notification subscription is to discover an appropriate DNS server that supports DNS Push Notifications for the desired zone. 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 [RFC7858].

1. The client begins the discovery by sending a DNS query to its local resolver, with record type SOA [RFC1035] for the record to which it wishes to subscribe. As an example, if it wishes to subscribe to PTR records with the name _printers._tcp.foo.example.com, it sends an SOA query for _printers._tcp.foo.example.com. The goal is to determine the authoritative server for foo.example.com.

2. If the SOA record exists as exactly specified in the query, it is expected to be returned in the Answer section with a NOERROR response code. If the exact SOA record does not exist, the client may get back a NOERROR/NODATA response or it may get back a NXDOMAIN/Name Error response. This depends on the resolver implementation and whether the domain exists. The client is looking for an SOA record to be returned in either the Answer section or the Authority section with a NOERROR response code. If the client receives an NXDOMAIN/Name Error response code, it should strip the leading label from the query name and if the resulting name has at least one label in it, the client should send a new SOA query, repeating this until a NOERROR response code is received or the query name is empty. In the case of an empty name, the client may retry the operation at a later time, of the client's choosing, such after a change in network attachment.

3. In the example above, if an SOA record query is sent for _printers._tcp.foo.example.com and an NXDOMAIN/Name Error is returned with an SOA record in the Authority section for foo.example.com, the client should strip the leading label and query an SOA record for _tcp.foo.example.com. If a NOERROR/ NODATA response is received with an SOA record in the Authority section for foo.example.com, this is sufficent. If an NXDOMAIN/ Name Error response is received, the client should again strip the leading label and query an SOA record for foo.example.com. If the foo.example.com domain exists, this should result in a NOERROR response with the SOA record in the Answer section. If the domain foo.example.com does not exist, the response will likely be an NXDOMAIN/Name Error with an SOA record for

example.com in the Authority section. This means the subdomain foo.example.com has not been properly delegated by example.com.

4. If a NOERROR/NODATA response is received but contains no SOA in the Authority section, the client could try stripping the leading label and issuing another SOA query. Additional information about negative responses can be found in Section 2 of [RFC2308].

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. For implementors of this specification, an authoritative answer for that SRV record, and only such an answer, will determine whether the zone supports DNS Push Notifications.

7. If the SRV record does exist, 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 [RFC7673].

8. 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 reachable or is not willing to accept a subscription request, then a subsequent server is to be contacted.

Each time a client makes a new DNS Push Notification subscription connection, 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 over the established TLS connection to the server. A SUBSCRIBE request is encoded in a DNS Session Signaling [SessSig] message. This specification defines a DNS Session Signaling TLV for DNS Push Notification SUBSCRIBE Requests/Responses (tentatively Session Signaling Type Code 0x40).

The entity that initiates a SUBSCRIBE request is by definition the client. A server should not send a SUBSCRIBE request over an existing connection from a client. If a server does send a SUBSCRIBE request over the connection initiated by a client, it is an error and the client should acknowledge the request with the error response RCODE NOTAUTH (Not Authoritative).

6.2.1. SUBSCRIBE Request

A SUBSCRIBE request message begins with the standard DNS Session Signaling 12-byte header [SessSig], followed by the SUBSCRIBE TLV. A SUBSCRIBE request message is illustrated below:

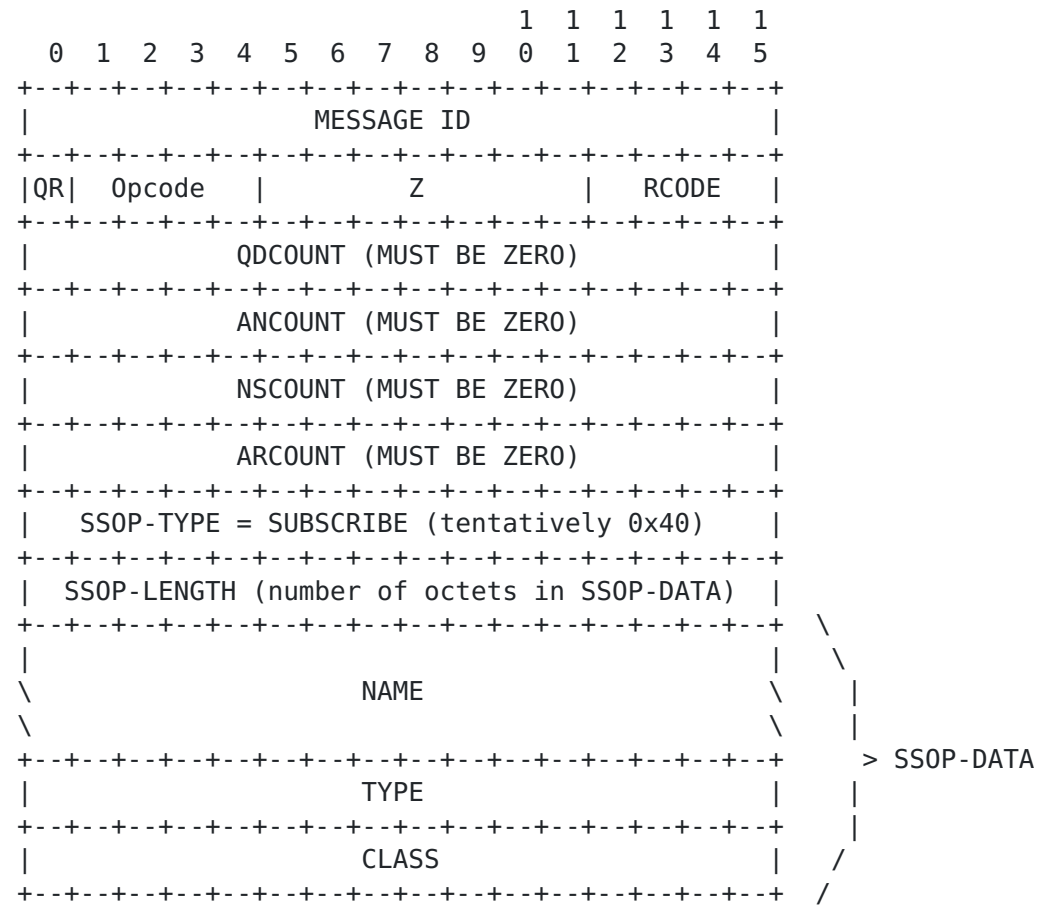

Figure 1

The MESSAGE ID field MUST be set to a unique value, that the client is not using for any other active operation on this connection. For the purposes here, a MESSAGE ID is in use on this connection 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 DNS Session Signaling specification [SessSig]. The DNS Opcode is the Session Signaling Opcode (tentatively 6). The four count fields MUST be

empty, and the corresponding four sections MUST be empty (i.e., absent).

The SSOP-TYPE is SUBSCRIBE (tentatively 0x40). The SSOP-LENGTH is the length of the SSOP-DATA that follows, which specifies the name, type, and class of the record(s) being sought.

The SSOP-DATA for a SUBSCRIBE request MUST contain exactly one question. The SSOP-DATA for a SUBSCRIBE request has no QDCOUNT field to specify more than one question. Since SUBSCRIBE requests are sent over TCP, multiple SUBSCRIBE 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 connection between the client and the server is closed.

SUBSCRIBE requests on a given connection 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 TLS/TCP connection. For the purpose of this matching, the established DNS case-insensitivity for US-ASCII letters applies (e.g., "foo.com" and "Foo.com" are the same). If a server receives such a duplicate SUBSCRIBE message this is an error and the server MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

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 nothing else.

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 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 message begins with the standard DNS Session Signaling 12-byte header [SessSig], possibly followed by one or more optional Modifier TLVs, such as a Retry Delay Modifier TLV.

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

A SUBSCRIBE response message MUST NOT contain a Session Signaling Operation TLV. The Session Signaling Operation TLV is NOT copied from the SUBSCRIBE request.

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.
NXDOMAIN	3	NOT APPLICABLE. DNS Push Notification servers MUST NOT return NXDOMAIN errors in response to SUBSCRIBE requests.
NOTIMP	4	Server does not recognize DNS Session Signaling Opcode.
REFUSED	5	Server refuses to process request for policy or security reasons.
NOTAUTH	9	Server is not authoritative for the requested name.
SSOPNOTIMP	11	SUBSCRIBE operation not supported.

SUBSCRIBE Response codes

This document specifies only these RCODE values for SUBSCRIBE Responses. Servers sending SUBSCRIBE Responses SHOULD use one of these values. 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 RCODE value.

If the server sends a nonzero RCODE in the SUBSCRIBE response, either the client is (at least partially) misconfigured, the server resources are exhausted, or there is some other unknown failure on the server. In any case, the client shouldn't retry the subscription right away. Either end can terminate the connection, but the client may want to try this subscription again or it may have other successful subscriptions that it doesn't want to abandon. If the server sends a nonzero RCODE then it SHOULD append a Retry Delay Modifier TLV [SessSig] 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:

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 Session Signaling [SessSig], it is unlikely that the server will begin supporting DNS Session Signaling in the next few minutes, so the retry delay SHOULD be one hour.

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.
This is a misconfiguration, since this server is listed in a "_dns-push-tls._tcp.<zone>" SRV record, but the server itself is not currently configured to support DNS Push Notifications. 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.
This is a misconfiguration, since this server is listed in a "_dns-push-tls._tcp.<zone>" SRV record, but the server itself is not currently configured to support DNS Push Notifications for that zone. 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 (DNS Push SUBSCRIBE operation not supported), which occurs on a server that 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 connection to remain open, or MAY send a DNS Push Notification Retry Delay Operation TLV asserting the client close the TCP connection, as described in the DNS Session Signaling specification [SessSig]. 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 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 message begins with the standard DNS Session Signaling 12-byte header [SessSig], followed by the PUSH TLV. A PUSH message is illustrated below:

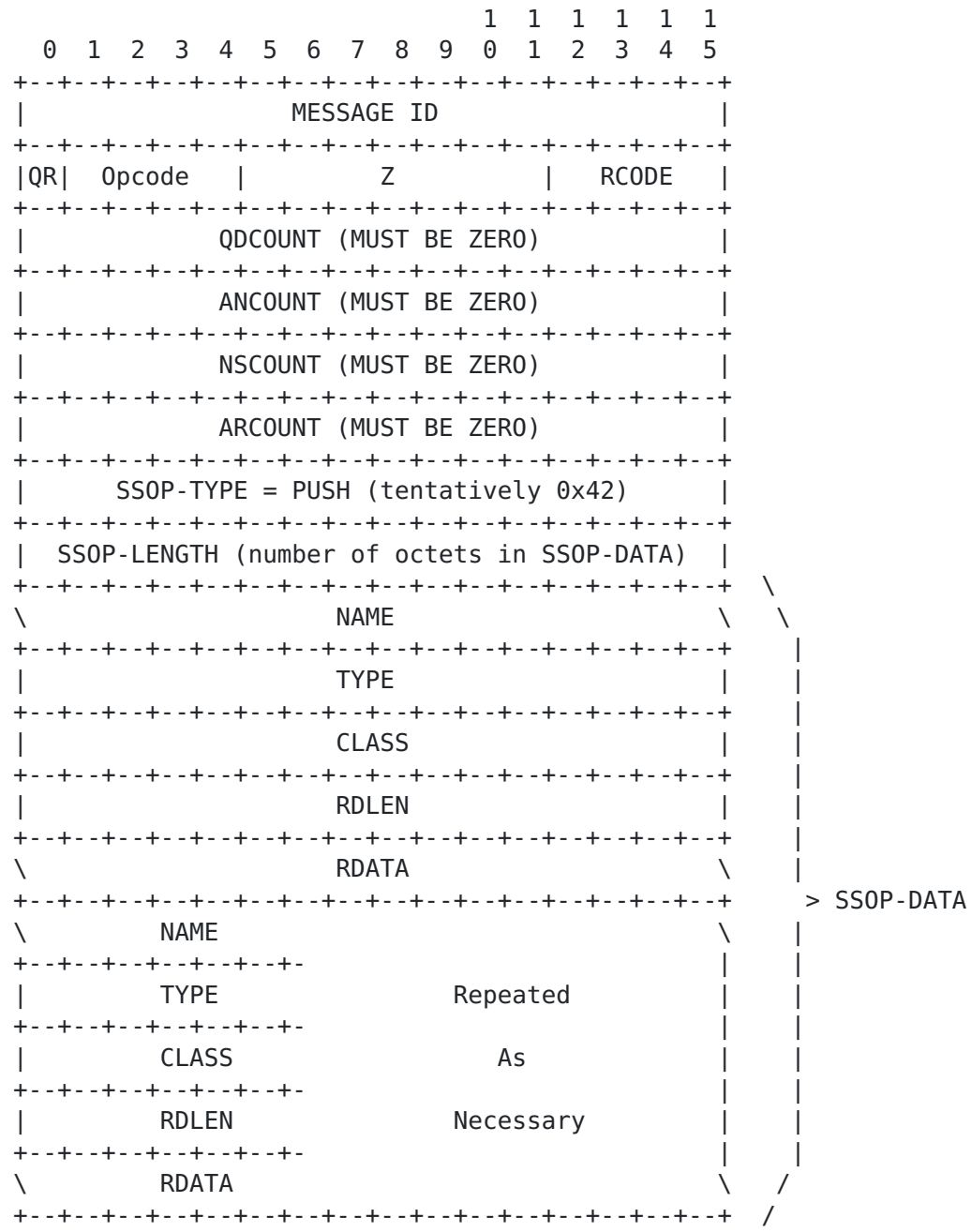

Figure 2

The MESSAGE ID field MUST be set to a unique value, that the server is not currently using for any other active outgoing request that it has sent on this connection. The MESSAGE ID in the outgoing PUSH message is selected by the server and has no relationship to the MESSAGE ID in any of the client subscriptions it may relate to. In the PUSH response the client MUST echo back the MESSAGE ID value unchanged.

The other header fields MUST be set as described in the DNS Session Signaling specification [SessSig]. The DNS Opcode is the Session Signaling Opcode (tentatively 6). The four count fields MUST be empty, and the corresponding four sections MUST be empty (i.e., absent).

The SSOP-TYPE is PUSH (tentatively 0x41). The SSOP-LENGTH is the length of the SSOP-DATA that follows, which specifies the changes being communicated.

The SSOP-DATA contains one or more Update records. A PUSH Message MUST contain at least one Update record. If a PUSH Message is received that contains no Update records, this is a fatal error, and the receiver MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols). The Update records are formatted in the customary way for Resource Records in DNS messages with the stipulation that DNS name compression is not permitted in DNS Session Signaling TLVs. Update records in a PUSH Message are interpreted according to the same rules as for DNS Update [RFC2136] messages, namely:

Delete all RRsets from a name:
TTL=0, CLASS=ANY, RDLENGTH=0, TYPE=ANY.

Delete an RRset from a name:
TTL=0, CLASS=ANY, RDLENGTH=0;
TYPE specifies the RRset being deleted.

Delete an individual RR from a name:
TTL=0, CLASS=NONE;
TYPE, RDLENGTH and RDATA specifies the RR being deleted.

Add to an RRset:
TTL, CLASS, TYPE, RDLENGTH and RDATA specifies the RR being added.

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 connection. Specifically, the record name MUST match the name given in the 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 connection 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 TCP connection 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. 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 connection-level mechanism, not a subscription-level mechanism.

Reception of a PUSH message by a client generates a PUSH response back to the server.

The TTL of an added record is stored by the client and decremented as time passes, with the caveat that for as long as a relevant subscription is active, the TTL does not decrement below 1 second. 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 TCP connection being closed) record aging resumes and records are removed from the local cache when their TTL reaches zero.

6.3.2. PUSH Response

Each PUSH message generates exactly one PUSH response from the receiver.

A PUSH response message begins with the standard DNS Session Signaling 12-byte header [SessSig], possibly followed by one or more optional Modifier TLVs, such as a Retry Delay Modifier TLV.

The MESSAGE ID field MUST echo the value given in the ID field of the PUSH message.

A PUSH response message MUST NOT contain a Session Signaling Operation TLV. The Session Signaling Operation TLV is NOT copied from the PUSH message.

In a PUSH response the RCODE MUST be zero. Receiving a PUSH response with a nonzero RCODE is a fatal error, and the receiver MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

6.4. DNS Push Notification UNSUBSCRIBE

To cancel an individual subscription without closing the entire connection, the client sends an UNSUBSCRIBE message over the established TCP connection to the server. The UNSUBSCRIBE message is encoded in a DNS Session Signaling [SessSig] message. This specification defines a DNS Session Signaling TLV for DNS Push Notification UNSUBSCRIBE Requests/Responses (tentatively Session Signaling Type Code 0x42).

A server MUST NOT initiate an UNSUBSCRIBE request. If a server does send a UNSUBSCRIBE request over the connection initiated by a client, it is an error and the client should acknowledge the request with the error response RCODE NOTAUTH (Not Authoritative).

6.4.1. UNSUBSCRIBE Request

An UNSUBSCRIBE request message begins with the standard DNS Session Signaling 12-byte header [SessSig], followed by the UNSUBSCRIBE TLV.

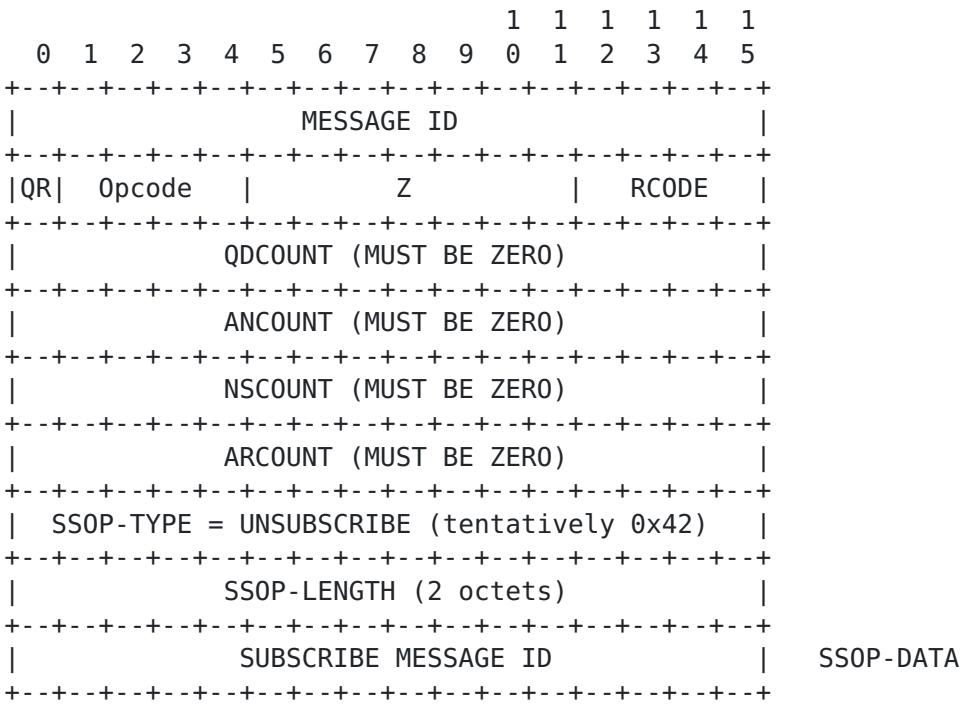

Figure 3

In the UNSUBSCRIBE TLV the SSOP-TYPE is UNSUBSCRIBE (tentatively 0x42). The SSOP-LENGTH is 2 octets.

The SSOP-DATA contains the MESSAGE ID field of the value given in the ID field of an active SUBSCRIBE request. This is how the server knows which SUBSCRIBE request is being cancelled. After receipt of the UNSUBSCRIBE request, the SUBSCRIBE request is no longer active. If a server receives an UNSUBSCRIBE message where the MESSAGE ID does not match the ID of an active SUBSCRIBE request the server MUST return a response containing RCODE = 3 (NXDOMAIN).

It is allowable for the client to issue an UNSUBSCRIBE request 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 request. A consequence of this is that if the client issues an UNSUBSCRIBE request for an as-yet unacknowledged SUBSCRIBE request, and the SUBSCRIBE request is

subsequently unsuccessful for some reason, then when the UNSUBSCRIBE request is eventually processed it will be an UNSUBSCRIBE request for a nonexistent subscription, which will result NXDOMAIN response.

6.4.2. UNSUBSCRIBE Response

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

An UNSUBSCRIBE response message begins with the standard DNS Session Signaling 12-byte header [SessSig], possibly followed by one or more optional Modifier TLVs, such as a Retry Delay Modifier TLV.

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

An UNSUBSCRIBE response message MUST NOT contain a Session Signaling Operation TLV. The Session Signaling Operation TLV is NOT copied from the UNSUBSCRIBE request.

In the UNSUBSCRIBE response the RCODE indicates whether or not the unsubscribe request was successful. Supported RCODEs are as follows:

Mnemonic	Value	Description
NOERROR	0	UNSUBSCRIBE successful.
FORMERR	1	Server failed to process request due to a malformed request.
NXDOMAIN	3	Specified subscription does not exist.
NOTIMP	4	Server does not recognize DNS Session Signaling Opcode.
SSOPNOTIMP	11	UNSUBSCRIBE operation not supported.

UNSUBSCRIBE Response codes

This document specifies only these RCODE values for UNSUBSCRIBE Responses. Servers sending UNSUBSCRIBE Responses SHOULD use one of these values. However, future circumstances may create situations where other RCODE values are appropriate in UNSUBSCRIBE Responses, so clients MUST be prepared to accept UNSUBSCRIBE Responses with any RCODE value.

Having being successfully revoked with a correctly-formatted UNSUBSCRIBE message (resulting in a response with RCODE NOERROR) the previously referenced subscription is no longer active and the server MAY discard the state associated with it immediately, or later, at the server's discretion.

Nonzero RCODE values signal some kind of error.

RCODE value FORMERR indicates a message format error.

RCODE value NXDOMAIN indicates a MESSAGE ID that does not correspond to any active subscription.

RCODE values NOTIMP and SSOPNOTIMP should not occur in practice.

A server would only generate NOTIMP if it did not support Session Signaling, and if the server does not support Session Signaling then it should not be possible for a client to have an active subscription to cancel.

Similarly, a server would only generate SSOPNOTIMP if it did not support Push Notifications, and if the server does not support Push Notifications then it should not be possible for a client to have an active subscription to cancel.

Nonzero RCODE values other than NXDOMAIN indicate a serious problem with the client. After sending an error response other than NXDOMAIN, the server SHOULD send a DNS Session Signaling Retry Delay Operation TLV and then close the TCP connection, as described in the DNS Session Signaling specification [SessSig].

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 believe 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 request 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 requests to ascertain whether the target device is still present. 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 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 request, 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 Request

A RECONFIRM request message begins with the standard DNS Session Signaling 12-byte header [SessSig], followed by the RECONFIRM TLV. A RECONFIRM request message is illustrated below:

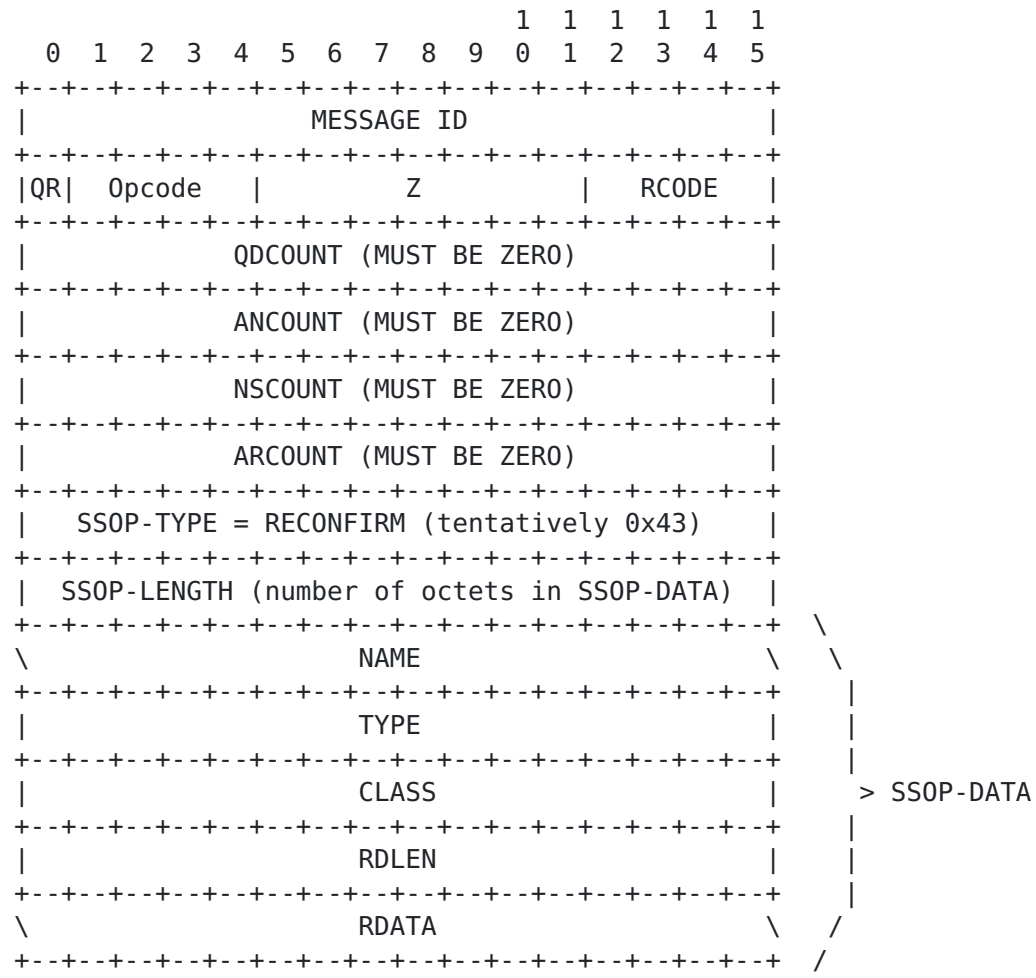

Figure 4

The MESSAGE ID field MUST be set to a unique value, that the client is not using for any other active operation on this connection. For the purposes here, a MESSAGE ID is in use on this connection 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 RECONFIRM response the server MUST echo back the MESSAGE ID value unchanged.

The other header fields MUST be set as described in the DNS Session Signaling specification [SessSig]. The DNS Opcode is the Session

Signaling Opcode (tentatively 6). The four count fields MUST be empty, and the corresponding four sections MUST be empty (i.e., absent).

The SSOP-TYPE is RECONFIRM (tentatively 0x43). The SSOP-LENGTH is the length of the data that follows, which specifies the name, type, class, and content of the record being disputed.

The SSOP-DATA for a RECONFIRM request MUST contain exactly one record. The SSOP-DATA for a RECONFIRM request has no count field to specify more than one record. Since RECONFIRM requests are sent over TCP, multiple RECONFIRM request 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.

6.5.2. RECONFIRM Response

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

A RECONFIRM response message begins with the standard DNS Session Signaling 12-byte header [SessSig], possibly followed by one or more optional Modifier TLVs, such as a Retry Delay Modifier TLV.

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

A RECONFIRM response message MUST NOT contain a Session Signaling Operation TLV. The Session Signaling Operation TLV is NOT copied from the RECONFIRM request.

In the RECONFIRM response the RCODE confirms receipt of the reconfirmation request. Supported RCODEs are as follows:

Mnemonic	Value	Description
NOERROR	0	RECONFIRM accepted.
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.
NXDOMAIN	3	NOT APPLICABLE. DNS Push Notification servers MUST NOT return NXDOMAIN errors in response to RECONFIRM requests.
NOTIMP	4	Server does not recognize DNS Session Signaling Opcode.
REFUSED	5	Server refuses to process request for policy or security reasons.
NOTAUTH	9	Server is not authoritative for the requested name.
SSOPNOTIMP	11	RECONFIRM operation not supported.

RECONFIRM Response codes

This document specifies only these RCODE values for RECONFIRM Responses. Servers sending RECONFIRM Responses SHOULD use one of these values. However, future circumstances may create situations where other RCODE values are appropriate in RECONFIRM Responses, so clients MUST be prepared to accept RECONFIRM Responses with any RCODE value.

Nonzero RCODE values signal some kind of error.

RCODE value FORMERR indicates a message format error, for example TYPE or CLASS being ANY (255).

RCODE value SERVFAIL indicates that the server has exhausted its resources or other serious problem occurred.

RCODE values NOTIMP indicates that the server does not support Session Signaling, and Session Signaling is required for RECONFIRM requests.

RCODE value REFUSED indicates that the server supports RECONFIRM requests but is currently not configured to accept them from this client.

RCODE value NOTAUTH indicates that the server is not authoritative for the requested name, and can do nothing to remedy the apparent error. Note that there may be future cases in which a server is able to pass on the RECONFIRM request to the ultimate source of the information, and in these cases the server should return NOERROR.

RCODE value SSOPNOTIMP indicates that the server does not support RECONFIRM requests.

Similarly, a server would only generate SSOPNOTIMP if it did not support Push Notifications, and if the server does not support Push Notifications then it should not be possible for a client to have an active subscription to cancel.

Nonzero RCODE values SERVFAIL, REFUSED and SSOPNOTIMP are benign from the client's point of view. The client may log them to aid in debugging, but otherwise they require no special action.

Nonzero RCODE values other than these three indicate a serious problem with the client. After sending an error response other than one of these three, the server SHOULD send a DNS Session Signaling Retry Delay Operation TLV and then close the TCP connection, as described in the DNS Session Signaling specification [SessSig].

6.6. 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 connection. When a client terminates an individual subscription (via UNSUBSCRIBE) or all subscriptions on that connection (by closing the connection) 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 connection via UNSUBSCRIBE, a client MAY close the connection immediately, or it may keep it open if it anticipates performing further operations on that connection in the future. If a client wishes to keep an idle connection open, it MUST respect the maximum idle time required by the server [SessSig].

If a client plans to terminate one or more subscriptions on a connection and doesn't intend to keep that connection open, then as an efficiency optimization it MAY instead choose to simply close the connection, which implicitly terminates all subscriptions on that connection. 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 connection has been cancelled.

When closing a connection, a client will generally do an abortive disconnect, sending a TCP RST. This immediately discards all remaining inbound and outbound data, which is appropriate if the client no longer has any interest in this data. In the BSD Sockets API, sending a TCP RST is achieved by setting the SO_LINGER option with a time of 0 seconds and then closing the socket.

If a client has performed operations on this connection that it would not want lost (like DNS updates) then the client SHOULD do an orderly disconnect, sending a TCP FIN. 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.

7. Security Considerations

The Strict Privacy Usage Profile for DNS over TLS is strongly recommended for DNS Push Notifications as defined in Authentication and (D)TLS Profile for DNS-over-(D)TLS [I-D.ietf-dprive-dtls-and-tls-profiles]. The Opportunistic Privacy Usage Profile is permissible as a way to support incremental deployment of security capabilities. Cleartext connections for DNS Push Notifications are not permissible.

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.

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 Sever 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 Authentication and (D)TLS Profile for DNS-over-(D)TLS [I-D.ietf-dprive-dtls-and-tls-profiles] for more information on authenticating domain names. Also note that a DNS Push server is an authoritative server and a DNS Push client is a standard DNS client. While the terminology in Authentication and (D)TLS Profile for DNS-over-(D)TLS [I-D.ietf-dprive-dtls-and-tls-profiles] explicitly states it does not apply to authoritative servers, it does in this case apply to DNS Push Notification clients and servers.

7.3. TLS Compression

In order to reduce the chances of compression-related attacks, TLS-level compression SHOULD be disabled when using TLS versions 1.2 and earlier. In the draft version of TLS 1.3 [I-D.ietf-tls-tls13], TLS-level compression has been removed completely.

7.4. TLS Session Resumption

TLS Session Resumption is permissible on DNS Push Notification servers. The server may keep TLS state with Session IDs [RFC5246] or operate in stateless mode by sending a Session Ticket [RFC5077] to the client for it to store. However, once the connection is closed, any existing subscriptions will be dropped. 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 connection. Use of TLS Session Resumption allows a new 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 the service name: "_dns-push-tls._tcp".
It is only applicable for the TCP protocol.
This name is to be published in the IANA Service Name Registry [RFC6335][SN].

This document defines four DNS Session Signaling TLV types: SUBSCRIBE with (tentative) value 0x40 (64), PUSH with (tentative) value 0x41

(65), UNSUBSCRIBE with (tentative) value 0x42 (66), and RECONFIRM with (tentative) value 0x43 (67).

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.

10. References

10.1. Normative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

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

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

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

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

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

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

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <http://www.rfc-editor.org/info/rfc2136>.

[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, <http://www.rfc-editor.org/info/rfc2782>.

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

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7766>.

[SessSig] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Mankin, A., and T. Pusateri, "DNS Session Signaling", draft-ietf-dnsop-session-signal-02 (work in progress), March 2017.

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

10.2. Informative References

[DisProx] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-06 (work in progress), March 2017.

[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.

[I-D.ietf-dprive-dtls-and-tls-profiles] Dickinson, S., Gillmor, D., and T. Reddy, "Usage and (D)TLS Profiles for DNS-over-(D)TLS", draft-ietf-dprive-dtls-and-tls-profiles-10 (work in progress), June 2017.

[I-D.sekar-dns-llq] Sekar, K., "DNS Long-Lived Queries", draft-sekar-dns-llq-01 (work in progress), August 2006.

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

[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, <http://www.rfc-editor.org/info/rfc2308>.

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

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6281>.

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

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://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, <http://www.rfc-editor.org/info/rfc6824>.

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

[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, <http://www.rfc-editor.org/info/rfc7525>.

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

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

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

Authors' Addresses

Tom Pusateri
Seeking affiliation
Hilton Head Island, SC
USA

Phone: +1 843 473 7394
Email: pusateri@bangj.com

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com