DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

R. Bellis
ISC
S. Cheshire
Apple Inc.
J. Dickinson
S. Dickinson
Sinodun
A. Mankin
Salesforce
T. Pusateri
Unaffiliated

October 31, 2016

DNS Session Signaling
draft-ietf-dnsop-session-signal-01

Abstract

The EDNS(0) Extension Mechanism for DNS is explicitly defined to only have "per-message" semantics. This document defines a new Session Signaling Opcode used to carry persistent "per-session" type-length-values (TLVs), and defines an initial set of TLVs used to manage session timeouts and termination.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

The use of transports other than UDP for DNS is being increasingly specified, for example, DNS over TCP [RFC1035][RFC7766] and DNS over TLS [RFC7858]. Such transports can offer persistent, long-lived sessions and therefore when using them for transporting DNS messages it is of benefit to have a mechanism that can establish parameters associated with those sessions, such as timeouts. In such situations it is also advantageous to support server initiated messages.

The EDNS(0) Extension Mechanism for DNS [RFC6891] is explicitly defined to only have "per-message" semantics. Whilst EDNS(0) has been used to signal at least one session related parameter (the

EDNS(0) TCP KeepAlive option [RFC7828]) the result is less than optimal due to the restrictions imposed by the EDNS(0) semantics and the lack of server initiated signalling. This document defines a new Session Signaling Opcode used to carry persistent "per-session" type-length-values (TLVs), and defines an initial set of TLVs used to manage session timeouts and termination.

With EDNS(0), multiple options may be packed into a single OPT pseudo-RR, and there is no generalized mechanism for a client to be able to tell whether a server has processed or otherwise acted upon each individual option within the combined OPT RR. The specifications for each individual option need to define how each different option is to be acknowledged, if necessary.

With Session Signaling, in contrast, there is no compelling motivation to pack multiple operations into a single message for efficiency reasons. Each Session Signaling operation is communicated in its own separate DNS message, and the transport protocol can take care of packing separate DNS messages into a single IP packet if appropriate. For example, TCP can pack multiple small DNS messages into a single TCP segment. The RCODE in each response message indicates the success or failure of the operation in question.

It should be noted that the message format for Session Signaling operations (see Section 3.3) differs from the DNS packet format used for standard queries and responses, in that it has a shorter header (four octets instead of usual twelve octets).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

The term "server" means the software with a listening socket, awaiting incoming connection requests.

The term "client" means the software which initiates a connection to the server's listening socket.

The terms "initiator" and "responder" correspond respectively to the initial sender and subsequent receiver of a Session Signaling TLV, regardless of which was the "client" and "server" in the usual DNS sense.

The term "sender" may apply to either an initiator or responder.

The term "session" in the context of this document means the exchange of DNS messages using an end-to-end transport protocol where:

- The connection between client and server is persistent and relatively long-lived (i.e., minutes or hours, rather than seconds).
- Either end of the connection may initiate messages to the other
- Messages are delivered in order

3. Protocol Details

Session Signaling messages MUST only be carried in protocols and in environments where a session may be established according to the definition above. Standard DNS over TCP [RFC1035][RFC7766], and DNS over TLS [RFC7858] are suitable protocols. DNS over plain UDP is not appropriate since it fails on the requirement for in-order message delivery, and, in the presence of NAT gateways and firewalls with short UDP timeouts, it fails to provide a persistent bi-directional communication channel unless an excessive amount of keepalive traffic is used.

Session Signaling messages relate only to the specific session in which they are being carried. Where a middle box (e.g., a DNS proxy, forwarder, or session multiplexer) is in the path the middle box MUST NOT blindly forward the message in either direction. This does not preclude the use of these messages in the presence of a NAT box that rewrites IP-layer or transport-layer headers but otherwise maintains the effect of a single session.

A client MAY attempt to initiate Session Signaling messages at any time on a connection; receiving a NOTIMP response in reply indicates that the server does not implement Session Signaling, and the client SHOULD NOT issue further Session Signaling messages on that connection.

A server SHOULD NOT initiate Session Signaling messages until a client-initiated Session Signaling message is received first, unless in an environment where it is known in advance by other means that the client supports Session Signaling. This requirement is to ensure that the clients that do not support Session Signaling do not receive unsolicited inbound Session Signaling messages that they would not know how to handle.

Clients and servers SHOULD silently ignore unrecognized messages (both requests and responses) over the connection. This allows for backwards compatibility with future enhancements.

3.1. Session Lifecycle

A session begins when a client makes a new connection to a server.

If a client makes a connection and then fails to send any DNS messages, then after 30 seconds the server SHOULD abort the connection with a TCP RST.

The client may perform as many DNS operations as it wishes on the newly created connection. Operations SHOULD be pipelined (i.e., the client doesn't need wait for a reply before sending the next message). The server MUST act on messages in the order they are received, but responses to those messages MAY be sent out of order, if appropriate.

When a server implementing this specification receives a new connection from a client, it MUST begin by internally assigning an initial idle timeout of 30 seconds to that connection. At both servers and clients, the generation or reception of any complete DNS message, including DNS requests, responses, updates, or Session Signaling messages, resets the idle timer for that connection [RFC7766].

If, at any time during the life of the connection, half the idle-timeout value (i.e., 15 seconds by default) elapses without any DNS messages being sent or received on a connection, then the connection is considered stale and the client MUST take action. When this happens the client MUST either send at least one new message to reset the idle timer - such as a Session Signaling Idle Timeout message (see Section 4), or any other valid DNS message - or close the connection.

If a client disconnects from the network abruptly, without cleanly closing its connection, the server learns of this after failing to receive further traffic from that client. If, at any time during the life of the connection, the full idle-timeout value (i.e., 30 seconds by default) elapses without any DNS messages being sent or received on a connection, then the connection is considered delinquent and the server SHOULD forcibly terminate the connection. For sessions over TCP (or over TLS over TCP), to avoid the burden of having a connection in TIME-WAIT state, instead of closing the connection gracefully with a TCP FIN the server SHOULD abort the connection with a TCP RST (or equivalent for other protocols). (In the BSD Sockets API this is achieved by setting the SO_LINGER option to zero before closing the socket.)

If the client wishes to keep an idle connection open for longer than the default duration without having to send traffic every 15 seconds,

then it uses the Session Signaling Idle Timeout message to request a longer idle timeout, as described in Section 4.

3.1.1. Client-Initiated Termination

A client is not required to wait until half of the idle-timeout value before closing a connection. A client SHOULD close a connection at any time, at the client's discretion, if it determines that, at that time, it has no current or reasonably anticipated imminent future need for the connection.

Upon receiving an error response from the server, a client SHOULD NOT automatically close the connection. An error relating to one particular operation on a connection does not necessarily imply that all other operations on that connection have also failed, or that future operations will fail. The client should assume that the server will make its own decision about whether or not to close the connection, based on the server's determination of whether the error condition pertains to this particular operation, or would also apply to any subsequent operations. If the server does not close the connection then the client SHOULD continue to use that connection for subsequent operations.

3.1.2. Server-Initiated Termination

After sending an error response to a client, the server MAY close the connection, or may allow the connection to remain open. For error conditions that only affect the single operation in question, the server SHOULD return an error response to the client and leave the connection open for further operations. For error conditions that are likely to make all operations unsuccessful in the immediate future, the server SHOULD return an error response to the client and then close the connection by sending a Terminate Session request message, as described in Section 5.

There may be rare cases where a server is overloaded and wishes to shed load. If the server handles this by simply closing connections, the likely behaviour of clients is to detect this as a network failure, and reconnect.

To avoid this reconnection implosion, in this situation the server also sends a Terminate Session request message, with an RCODE of SERVFAIL, to inform the client of the overload situation.

After sending a Terminate Session request message, the server MUST NOT send any further messages on that connection.

A Terminate Session request message MUST NOT be initiated by a client. If a server receives a Terminate Session request message this is an error and the server MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

Upon receipt of a Terminate Session request from the server, the client MUST make note of the reconnect delay for this server, and then immediately close the connection. This is to place the burden of TCP's TIME-WAIT state on the client.

After sending the Terminate Session request the server SHOULD allow the client five seconds to close the connection, and if the client has not closed the connection after five seconds then the server SHOULD abort the connection with a TCP RST (or equivalent for other protocols). (In the BSD Sockets API this is achieved by setting the SO_LINGER option to zero before closing the socket.)

In the case where the server is canceling some, but not all, of the existing operations on a connection, with a REFUSED (5) RCODE (perhaps because it has been reconfigured and is no longer authoritative for those names), the RECONNECT DELAY MAY be zero, indicating that the client SHOULD immediately attempt to re-establish its operations. It is likely that some of the new attempts will be successful and some will not.

In the case where a server is terminating a large number of connections at once (e.g., if the system is restarting) and the server doesn't want to be inundated with a flood of simultaneous retries, it SHOULD send different RECONNECT delay values to each client. These adjustments MAY be selected randomly, pseudorandomly, or deterministically (e.g., incrementing the time value by one tenth of a second for each successive client, yielding a post-restart reconnection rate of ten clients per second).

Apart from the cases described above, a server MUST NOT close a connection with a client, except in extraordinary error conditions. Closing the connection is the client's responsibility, to be done at the client's discretion, when it so chooses. A server only closes a connection under exceptional circumstances, such as when the server application software or underlying operating system is restarting, the server application terminated unexpectedly (perhaps due to a bug that makes it crash), or the server is undergoing maintenance procedures. When possible, a server SHOULD send a Terminate Session message informing the client of the reason for the connection being closed, and allow the client five seconds to receive it before the server resorts to forcibly aborting the connection.

After a connection is closed by the server, the client SHOULD try to reconnect, to that server, or to another suitable server, if more than one is available. If reconnecting to the same server, the client MUST respect the indicated delay before attempting to reconnect.

If a server is low on resources it MAY simply terminate a client connection with a TCP RST. However, the likely behaviour of the client may be simply to reconnect immediately, putting more burden on the server. Therefore, a server SHOULD instead choose to shed client load by sending a Terminate Session message, as described above. Upon reception of the Termination TLV the client is expected to close the connection, and if it does not then the server will abort the connection five seconds later.

3.2. Connection Sharing

A client that supports Session Signaling SHOULD NOT make multiple connections to the same DNS server.

A single server may support multiple services, including DNS Updates [RFC2136], DNS Push Notifications [I-D.ietf-dnssd-push], and other services, for one or more DNS zones. When a client discovers that the target server for several different operations is the same target hostname and port, the client SHOULD use a single shared connection for all those operations. A client SHOULD NOT open multiple connections to the same target host and port just because the names being operated on are different or happen to fall within different zones. This is to reduce unnecessary connection load on the DNS server.

For the purposes here, the determination of "same server" is made by comparing the target hostname and port of the desired DNS server, not the IP address(es) that the target hostname resolves to. The hostname and port of the desired DNS server in question may be obtained via manual configuration, may be learned automatically from "_dns-update-tls._tcp.<zone>" or "_dns-push-tls._tcp.<zone>" SRV records, or may be learned by other means used by other protocols.

If two different target hostnames happen to resolve to the same IP address(es), then the client SHOULD NOT recognize these as the "same server" for the purposes of using a single shared connection to that server. If an administrator wishes to use a single server for multiple zones and/or multiple roles (e.g., both DNS Updates and DNS Push Notifications), and wishes to have clients use a single shared connection for operations on that server, then the administrator MUST specify the same target server hostname for all the desired zones and/or roles, either in the appropriate manual configuration data

(for clients that are configured manully) or in the appropriate SRV records (for clients that learn configuration from the network).

However, server implementers and operators should be aware that even when the same target hostname is correctly used, this connection sharing may not be possible in all cases. A single client device may be home to multiple independent client software instances that don't coordinate with each other, so a DNS server MUST be prepared to accept multiple connections from different source ports on the same client IP address. This is undesirable from an efficiency standpoint, but it may be unavoidable in some situations, so a DNS server MUST be prepared to accept multiple connections from the same client IP address.

Independent client devices behind the same NAT gateway will also typically appear to the DNS server to be different source ports on the same client IP address.

3.3. Message Format

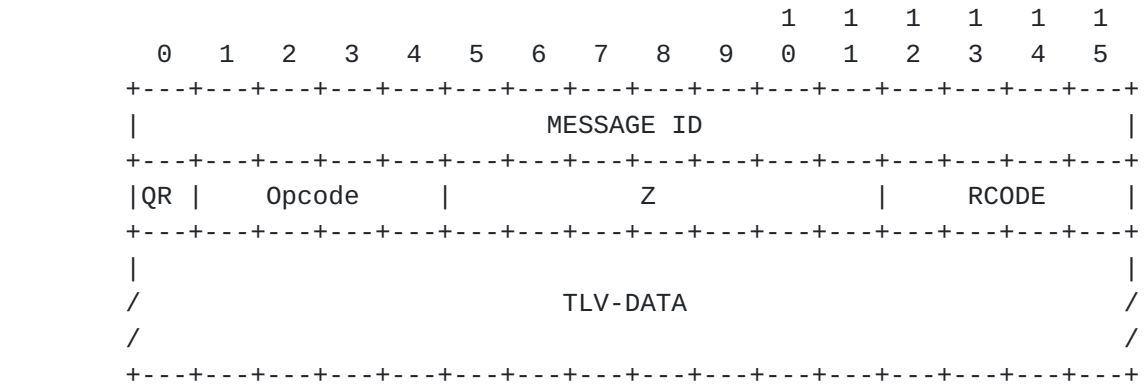
A Session Signaling message begins with the first 4 octets of the standard DNS message header [RFC1035], with the Opcode field set to the Session Signaling Opcode. A Session Signaling message does not contain the QDCOUNT, ANCOUNT, NSCOUNT and ARCOUNT fields fields used in standard DNS queries and responses. This 4-octet header is followed by a single Session Signaling operation TLV. The operation TLV may be followed by one or more modifier TLVs, such as the Terminate TLV (0), which, in error responses, indicates the time interval during which the client SHOULD NOT re-attempt a failed operation. Future specifications may define additional modifier TLVs that may be used in addition to the operation TLV. A Session Signaling message MUST contain exactly one operation TLV.

Since Session Signaling messages contain no ARCOUNT field, there is, by design, no way to add an EDNS(0) option to a Session Signaling message. If functionality provided by current or future EDNS(0) options is desired for Session Signaling messages, a Session Signaling operation TLV or modifier TLV needs to be defined to carry the necessary information.

Similarly there is, by design, no way to add a TSIG record to a Session Signaling message. If this capability becomes necessary in the future a Session Signaling modifier TLV needs to be defined to perform this function.

Note however that, while Session Signaling _messages_ cannot include EDNS(0) or TSIG records, a Session Signaling _session_ is typically used to carry a whole series of DNS messages of different kinds,

including Session Signaling messages, and other DNS message types like Query [RFC1034][RFC1035] and Update [RFC2136], and those messages can carry EDNS(0) and TSIG records. This specification explicitly prohibits use of the EDNS(0) TCP Keepalive Option [RFC7828] in messages sent on a Session Signaling session (because it duplicates the functionality provided by the Session Signaling Idle Timeout TLV), but messages may contain other EDNS(0) options where appropriate.



The MESSAGE ID, QR, Opcode and RCODE fields have their usual meanings [RFC1035].

In a request message (QR=0) the RCODE is generally set to zero on transmission, and silently ignored on reception, except where specified otherwise (for example, the Terminate Session operation, where the RCODE indicates the reason for termination).

The Z bits are currently unused, and in both requests and responses the Z bits SHOULD be set to zero (0) on transmission and silently ignored on reception, unless a future document specifies otherwise.

3.4. Message Handling

On a connection between a client and server that support Session Signaling, once the client has sent at least one Session Signaling message (or it is known in advance by other means that the client supports Session Signaling) either end may unilaterally send Session Signaling messages at any time, and therefore either client or server may be the initiator of a message. The initiator MUST set the value of the QR bit in the DNS header to zero (0), and the responder MUST set it to one (1).

Every Session Signaling request message (QR=0) MUST elicit a response (QR=1), which MUST have the same MESSAGE ID in the DNS message header as in the corresponding request.

An initiator MUST NOT reuse a MESSAGE ID that is already in use for an outstanding request, unless specified otherwise by the relevant specification for the Session Signaling TLV in question. At the very least, this means that a MESSAGE ID MUST NOT be reused for a particular SSOP-TYPE while the initiator is waiting for a response to a previous request with the same SSOP-TYPE, unless specified otherwise by the relevant specification for the Session Signaling TLV in question. (For a long-lived operation, such as a DNS Push Notification subscription [I-D.ietf-dnssd-push] the MESSAGE ID for the operation MUST NOT be reused for a new subscription as long as the existing subscription is active.)

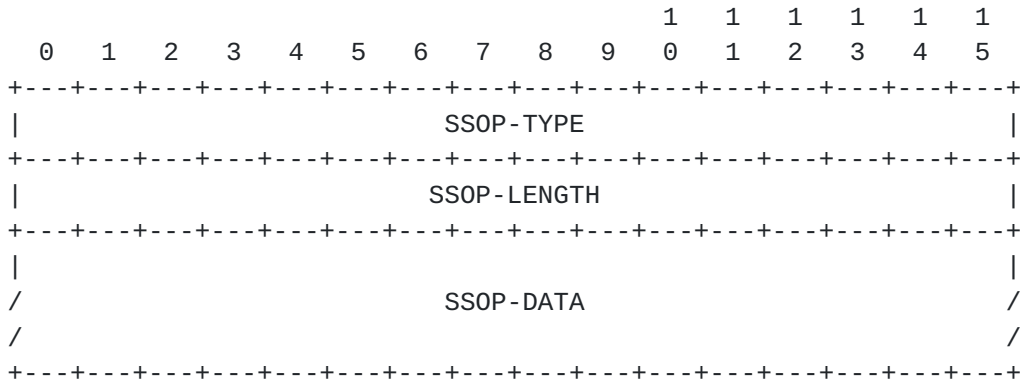
The namespaces of 16-bit MESSAGE IDs are disjoint in each direction. For example, it is _not_ an error for both client and server to send a request message with the same ID. In effect, the 16-bit MESSAGE ID combined with the identity of the initiator (client or server) serves as a 17-bit unique identifier for a particular operation on a session.

If a client or server receives a response (QR=1) where the MESSAGE ID does not match any of its outstanding operations, this is a fatal error and it MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

The RCODE value in a response may be one of the following values:

Code	Mnemonic	Description
0	NOERROR	TLV processed successfully
1	FORMERR	TLV format error
4	NOTIMP	Session Signaling not supported
5	REFUSED	TLV declined for policy reasons
11	SSOPNOTIMP	Session Signaling operation Type Code not supported

3.5. TLV Format



SSOP-TYPE: A 16 bit field in network order giving the type of the current Session Signaling TLV per the IANA DNS Session Signaling Type Codes Registry.

SSOP-LENGTH: A 16 bit field in network order giving the size in octets of SSOP-DATA.

SSOP-DATA: Type-code specific.

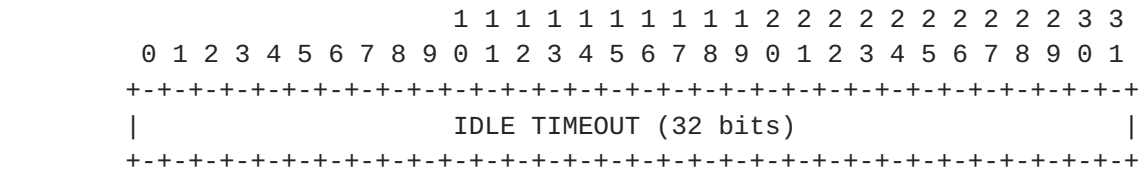
4. Idle Timeout TLV

The Idle Timeout TLV (1) is be used by a client to reset a connection's idle timer, and at the same time to request what the idle timeout should be from this point forward in the connection.

Once the client has sent at least one Session Signaling message (or it is known in advance by other means that the client supports Session Signaling) the Idle Timeout TLV also MAY be initiated by a server, to unilaterally inform the client of a new idle timeout this point forward in this connection.

It is not required that the Idle Timeout TLV be used in every session. While many Session Signaling operations (such as DNS Push Notifications [I-D.ietf-dnssd-push]) will be used in conjunction with a long-lived connection, not all Session Signaling operations require a long-lived connection, and in some cases the default 30-second timeout may be perfectly appropriate.

The SSOP-DATA for the the Idle Timeout TLV is as follows:



IDLE TIMEOUT: the idle timeout for the current session, specified as a 32 bit word in network order in units of milliseconds. This is the timeout at which the server will forcibly terminate the connection with a TCP RST (or equivalent for other protocols); after half this interval the client MUST take action to either preserve the connection, or close it if it is no longer needed.

In a client-initiated Session Signaling Idle Timeout message, the IDLE TIMEOUT contains the client's requested value for the idle timeout. In a server response to a client-initiated message, the IDLE TIMEOUT contains the server's chosen value for the idle timeout, which the client MUST respect. This is modeled after the DHCP protocol, where the client requests a certain lease lifetime using DHCP option 51 [RFC2132], but the server is the ultimate authority for deciding what lease lifetime is actually granted.

In a server-initiated Session Signaling Idle Timeout message, the IDLE TIMEOUT unilaterally informs the client of the new idle timeout this point forward in this connection. In a client response to a server-initiated message, there is no SSOP-DATA. SSOP-LENGTH is zero. The RCODE MUST be zero.

<< SC: Do we even need a client response to this server-initiated message? The response conveys no information. On the other hand, it may simplify the specification if we say that _all_ request messages elicit exactly one response message. Please weigh in with opinions. We need to decide this. Currently draft-ietf-dnssd-push says that push notification messages from server to client do not elicit any response message from the client. We need to decide if this is allowed. >>

Note that the lower the IDLE TIMEOUT value, the higher the load on client and server. For example, an IDLE TIMEOUT value of 200ms would result in a continuous stream of at least ten messages per second, in both directions, to keep the connection alive. And, in this example, a single packet loss and retransmission could introduce a momentary pause in the stream of messages, long enough to cause the server to overzealously abort the connection.

Because of this concern, the server MUST NOT send a Idle Timeout message (either a response to a client-initiated request, or a server-initiated message) with an IDLE TIMEOUT value less than ten seconds. If a client receives an Idle Timeout message specifying an IDLE TIMEOUT value less than ten seconds this is an error and the client MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

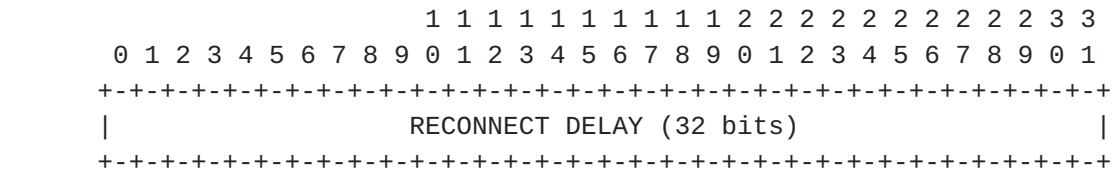
The Idle Timeout TLV (1) has similar intent to the EDNS(0) TCP Keepalive Option [RFC7828]. A client/server pair that supports Session Signaling MUST NOT use the EDNS(0) TCP KeepAlive option within any message on a connection once bi-directional Session Signaling support has been confirmed. Once bi-directional Session Signaling support has been confirmed, if either client or server receives a DNS message over the session that contains an EDNS(0) TCP KeepAlive option, this is an error and the receiver of the EDNS(0) TCP KeepAlive option MUST immediately terminate the connection with a TCP RST (or equivalent for other protocols).

5. Terminate TLV

The Terminate TLV (0) is be used by a server to request that a client close the connection, and not to reconnect for the indicated time interval. It is also used as a modifier on error responses, to indicate how long the client should wait before retrying that particular operation.

<< SC: Perhaps we should change the name of TLV (0) to be "retry delay" instead of "Terminate"? >>

The SSOP-DATA for the the Terminate TLV is as follows:



RECONNECT DELAY: a time value, specified as a 32 bit word in network order in units of milliseconds, within which the client MUST NOT establish a new session to the current server.

The RECOMMENDED value is 10 seconds.

In the case of a client request that returns a nonzero RCODE value, the server MAY append a Terminate TLV (0) to the response, indicating the time interval during which the client SHOULD NOT attempt this operation again.

When appended to a DNS response message for some client request, the Terminate TLV (0) is considered a modifier TLV. The indicated time interval during which the client SHOULD NOT retry applies only to the failed operation, not to the session as a whole.

When sent in a DNS request message, from server to client, the Terminate Session TLV (0) is considered an operation TLV. It applies to the session as a whole, and the client MUST close the connection, as described previously. The RCODE MUST indicate the reason for the termination. RCODE NOERROR indicates a routine shutdown. RCODE SERVFAIL indicates that the server is overloaded due to resource exhaustion. RCODE REFUSED indicates that the server has been reconfigured and is no longer able to perform one of the functions currently being performed on this connection (for example, a DNS Push Notification server could be reconfigured such that is is no longer accepting DNS Push Notification requests for one or more of the currently subscribed names).

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

6. IANA Considerations

6.1. DNS Session Signaling Opcode Registration

IANA are directed to assign a value (tentatively 6) in the DNS Opcodes Registry for the Session Signaling Opcode.

6.2. DNS Session Signaling RCODE Registration

IANA are directed to assign a value (tentatively 11) in the DNS RCODE Registry for the SSOPNOTIMP error code.

6.3. DNS Session Signaling Type Codes Registry

IANA are directed to create the DNS Session Signaling Type Codes Registry, with initial values as follows:

Type	Name	Status	Reference
0	SSOP-Terminate	Standard	RFC-TBD
1	SSOP-IdleTimeout	Standard	RFC-TBD
3 - 63	Unassigned, reserved for session management TLVs		
64 - 63487	Unassigned		
63488 - 64511	Reserved for local / experimental use		
64512 - 65535	Reserved for future expansion		

Registration of additional Session Signaling Type Codes requires publication of an appropriate IETF "Standards Action" or "IESG Approval" document [RFC5226].

7. Security Considerations

If this mechanism is to be used with DNS over TLS, then these messages are subject to the same constraints as any other DNS over TLS messages and MUST NOT be sent in the clear before the TLS session is established.

8. Acknowledgements

TBW

9. References

9.1. Normative References

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

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

[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>.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <http://www.rfc-editor.org/info/rfc2132>.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

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

[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>.

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

9.2. Informative References

[I-D.ietf-dnssd-push]
Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-08 (work in progress), July 2016.

[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>.

Authors' Addresses

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

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

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

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

John Dickinson
Sinodun Internet Technologies
Magadalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: jad@sinodun.com

Sara Dickinson
Sinodun Internet Technologies
Magadalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: sara@sinodun.com

Allison Mankin
Salesforce

Email: allison.mankin@gmail.com

Tom Pusateri
Unaffiliated

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