

Extension mechanisms for DNS (EDNS0)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Abstract

The Domain Name System's wire protocol includes a number of fixed fields whose range has been or soon will be exhausted and does not allow clients to advertise their capabilities to servers. This document describes backward compatible mechanisms for allowing the protocol to grow.

1 - Rationale and Scope

1.1. DNS (see [[RFC1035](#)]) specifies a Message Format and within such messages there are standard formats for encoding options, errors, and name compression. The maximum allowable size of a DNS Message is fixed. Many of DNS's protocol limits are too small for uses which are or which are desired to become common. There is no way for implementations to advertise their capabilities.

1.2. Existing clients will not know how to interpret the protocol extensions detailed here. In practice, these clients will be upgraded when they have need of a new feature, and only new features will make use of the extensions. We must however take account of client behaviour in the face of extra fields, and design a fallback scheme for interoperability with these clients.

2 - Affected Protocol Elements

2.1. The DNS Message Header's (see [RFC1035 4.1.1]) second full 16-bit word is divided into a 4-bit OPCODE, a 4-bit RCODE, and a number of 1-bit flags. The original reserved Z bits have been allocated to various purposes, and most of the RCODE values are now in use. More flags and more possible RCODEs are needed.

2.2. The first two bits of a wire format domain label are used to denote the type of the label. [RFC1035 4.1.4] allocates two of the four possible types and reserves the other two. Proposals for use of the remaining types far outnumber those available. More label types are needed.

2.3. DNS Messages are limited to 512 octets in size when sent over UDP. While the minimum maximum reassembly buffer size still allows a limit of **512 octets of UDP payload, most of the hosts now connected to the Internet are able to reassemble larger datagrams.** Some mechanism must be created to allow requestors to advertise larger buffer sizes to responders.

3 - Extended Label Types

3.1. The ``0 1'' label type will now indicate an extended label type, whose value is encoded in the lower six bits of the first octet of a label. All subsequently developed label types should be encoded using an extended label type.

3.2. The ``1 1 1 1 1 1'' extended label type will be reserved for future expansion of the extended label type code space.

4 - OPT pseudo-RR

4.1. One OPT pseudo-RR can be added to the additional data section of either a request or a response. An OPT is called a pseudo-RR because it pertains to a particular transport level message and not to any actual DNS data. OPT RRs shall never be cached, forwarded, or stored in or loaded from master files. The quantity of OPT pseudo-RRs per message

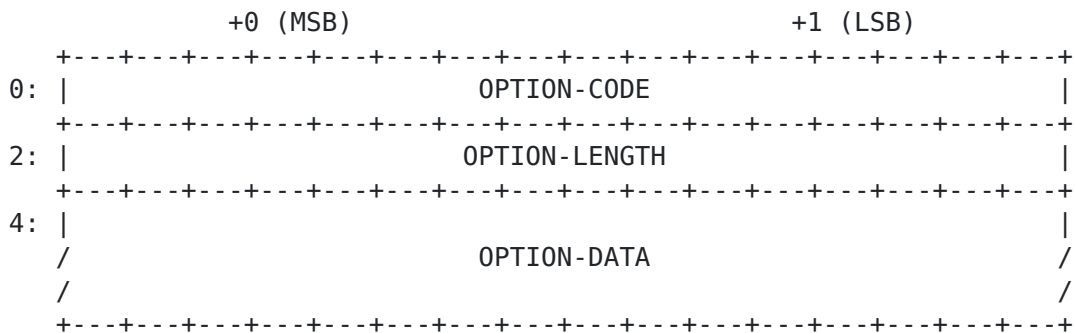
shall be either zero or one, but not greater.

4.2. An OPT RR has a fixed part and a variable set of options expressed as {attribute, value} pairs. The fixed part holds some DNS meta data and also a small collection of new protocol elements which we expect to be so popular that it would be a waste of wire space to encode them as {attribute, value} pairs.

4.3. The fixed part of an OPT RR is structured as follows:

Field Name	Field Type	Description
NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	sender's UDP payload size
TTL	u_int32_t	extended RCODE and flags
RDLEN	u_int16_t	describes RDATA
RDATA	octet stream	{attribute,value} pairs

4.4. The variable part of an OPT RR is encoded in its RDATA and is structured as zero or more of the following:



OPTION-CODE (Assigned by IANA.)

OPTION-LENGTH Size (in octets) of OPTION-DATA.

OPTION-DATA Varies per OPTION-CODE.

4.5. The sender's UDP payload size (which OPT stores in the RR CLASS field) is the number of octets of the largest UDP payload that can be reassembled and delivered in the sender's network stack. Note that path MTU, with or without fragmentation, may be smaller than this.

4.5.1. Note that a 512-octet UDP payload requires a 576-octet IP reassembly buffer. Choosing 1280 on an Ethernet connected requestor would be reasonable. The consequence of choosing too large a value may be an ICMP message from an intermediate gateway, or even a silent drop of the response message.

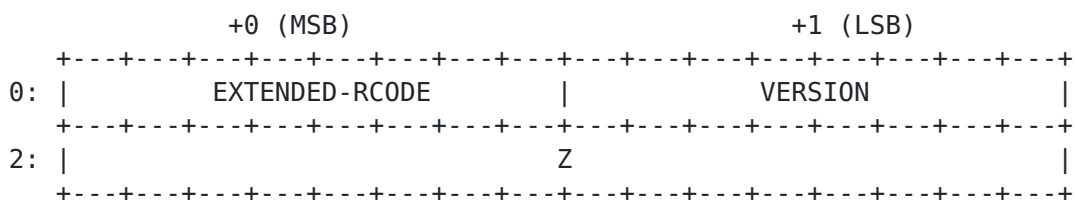
4.5.2. Both requestors and responders are advised to take account of the path's discovered MTU (if already known) when considering message sizes.

4.5.3. The requestor's maximum payload size can change over time, and should therefore not be cached for use beyond the transaction in which it is advertised.

4.5.4. The responder's maximum payload size can change over time, but can be reasonably expected to remain constant between two sequential transactions; for example, a meaningless QUERY to discover a responder's maximum UDP payload size, followed immediately by an UPDATE which takes advantage of this size. (This is considered preferable to the outright use of TCP for oversized requests, if there is any reason to suspect that the responder implements EDNS, and if a request will not fit in the default 512 payload size limit.)

4.5.5. Due to transaction overhead, it is unwise to advertise an architectural limit as a maximum UDP payload size. Just because your stack can reassemble 64KB datagrams, don't assume that you want to spend more than about 4KB of state memory per ongoing transaction.

4.6. The extended RCODE and flags (which OPT stores in the RR TTL field) are structured as follows:



EXTENDED-RCODE Forms upper 8 bits of extended 12-bit RCODE. Note that EXTENDED-RCODE value "0" indicates that an unextended RCODE is in use (values "0" through "15").

VERSION Indicates the implementation level of whoever sets it. Full conformance with this specification is indicated by version ``0.''. Requestors are encouraged to set this to

the lowest implemented level capable of expressing a transaction, to minimize the responder and network load of discovering the greatest common implementation level between requestor and responder. A requestor's version numbering strategy should ideally be a run time configuration option.

If a responder does not implement the VERSION level of the request, then it answers with RCODE=BADVERS. All responses will be limited in format to the VERSION level of the request, but the VERSION of each response will be the highest implementation level of the responder. In this way a requestor will learn the implementation level of a responder as a side effect of every response, including error responses, including RCODE=BADVERS.

Z Set to zero by senders and ignored by receivers, unless modified in a subsequent specification.

5 - Transport Considerations

5.1. The presence of an OPT pseudo-RR in a request should be taken as an indication that the requestor fully implements the given version of EDNS, and can correctly understand any response that conforms to that feature's specification.

5.2. Lack of use of these features in a request must be taken as an indication that the requestor does not implement any part of this specification and that the responder may make no use of any protocol extension described here in its response.

5.3. Responders who do not understand these protocol extensions are expected to send a response with RCODE NOTIMPL, FORMERR, or SERVFAIL. Therefore use of extensions should be ``probed'' such that a responder who isn't known to support them be allowed a retry with no extensions if it responds with such an RCODE. If a responder's capability level is cached by a requestor, a new probe should be sent periodically to test for changes to responder capability.

6 - Security Considerations

Requestor-side specification of the maximum buffer size may open a new DNS denial of service attack if responders can be made to send messages which are too large for intermediate gateways to forward, thus leading to potential ICMP storms between gateways and responders.

7 - IANA Considerations

IANA is hereby requested to assign an RR type code for OPT.

It is the recommendation of this document and its working group that IANA create a registry for EDNS Extended Label Types, for EDNS Option Codes, and for EDNS Version Numbers.

This document assigns label type 0b01xxxxxx as "EDNS Extended Label Type." We request that IANA record this assignment.

This document assigns extended label type 0bxx111111 as "Reserved for future extended label types." We request that IANA record this assignment.

This document assigns option code 65535 to "Reserved for future expansion."

This document expands the RCODE space from 4 bits to 12 bits. This will allow IANA to assign more than the 16 distinct RCODE values allowed in [\[RFC1035\]](#).

This document assigns EDNS Extended RCODE "16" to "BADVERS".

IESG approval should be required to create new entries in the EDNS Extended Label Type or EDNS Version Number registries, while any published RFC (including Informational, Experimental, or BCP) should be grounds for allocation of an EDNS Option Code.

8 - Acknowledgements

Paul Mockapetris, Mark Andrews, Robert Elz, Don Lewis, Bob Halley, Donald Eastlake, Rob Austein, Matt Crawford, Randy Bush, and Thomas Narten were each instrumental in creating and refining this specification.

9 - References

[RFC1035] P. Mockapetris, ``Domain Names - Implementation and Specification,`` [RFC 1035](#), USC/Information Sciences Institute, November 1987.

10 - Author's Address

Paul Vixie
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
+1 650 779 7001
<vixie@isc.org>

Expires December 1999

[Page 7]