

DNS Extensions Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 11, 2011

S. Crocker  
Shinkuro Inc.  
S. Rose  
NIST  
October 8, 2010

Signaling Cryptographic Algorithm Understanding in DNSSEC  
draft-crocker-dnssec-algo-signal-07

Abstract

The DNS Security Extensions (DNSSEC) were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. These digital signatures can be generated using different algorithms. This draft sets out to specify a way for validating end-system resolvers to signal to a server which cryptographic algorithms they support.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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>.

This Internet-Draft will expire on April 11, 2011.

## Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Signaling Algorithm Understood (AU) Using EDNS . . . . .	3
3. Client Considerations . . . . .	4
3.1. Recommendations for Stub Clients . . . . .	5
3.2. Recursive Cache Considerations . . . . .	5
4. Intermediate Middlebox Considerations . . . . .	5
5. Server Considerations . . . . .	5
6. Traffic Analysis Considerations . . . . .	6
7. IANA Considerations . . . . .	6
8. Security Considerations . . . . .	6
9. Normative References . . . . .	6

## 1. Introduction

The DNS Security Extensions (DNSSEC) [RFC4033], [RFC4034] and [RFC4035] were developed to provide origin authentication and integrity protection for DNS data by using digital signatures. Each digital signature RR (RRSIG) contains an algorithm code number. These algorithm codes tell validators which cryptographic algorithm was used to generate the digital signature. Authentication across delegation boundaries is maintained by storing a hash of a subzone's key in the parent zone stored in a Delegation Signer (DS) RR. These DS RRs contain a second code number to identify the hash algorithm used to construct the DS RR.

This draft sets out to specify a way for validating end-system resolvers to tell a server which cryptographic and/or hash algorithms they support in a DNS query. This is done using the EDNS attribute values in the OPT meta-RR [RFC2671].

This proposed EDNS option serves to measure the acceptance and use of new digital signing and hash algorithms. This algorithm signaling option can be used by zone administrators as a gauge to measure the successful deployment of code that implements a newly deployed digital signature or hash algorithm used with DNSSEC. A zone administrator may be able to determine when to stop serving the old algorithm when the server sees that all or almost all of its clients signal that they are able to accept the new algorithm.

This draft does not seek to include another process for including new algorithms for use with DNSSEC (see . It also does not address the question of which algorithms are to be included in any official list of mandatory or recommended cryptographic algorithms for use with DNSSEC. Rather, this document specifies a means by which a client query can signal a set of algorithms it implements.

## 2. Signaling Algorithm Understood (AU) Using EDNS

The EDNS0 specification outlined in [RFC2671] defines a way to include new options using a standardized mechanism. These options are contained in the RDATA of the OPT meta-RR. This document defines a new EDNS0 option for a client to signal which algorithms the client supports.



it wishes to receive DNSSEC RRs in the response.

Note that when including the PRIVATEDNS (253) and/or the PRIVATEOID (254) codes, the client only indicates that it understands one or more private algorithms but does not indicate which algorithms.

### 3.1. Recommendations for Stub Clients

Typically, stub resolvers rely on an upstream recursive server (or cache) to provide a response; any algorithm support on the stub resolver's side could be overruled by the upstream recursive server. The AU EDNS option is NOT RECOMMENDED for non-validating stub clients.

The exception to the above is that validating stub resolvers which set the CD bit in queries MAY set the AU option. In the most common scenario, the validating stub indicates that it wishes to perform its own validation (via the CD bit) and may therefore wish to indicate which cryptographic algorithm(s) it supports.

### 3.2. Recursive Cache Considerations

DNSSEC validating recursive caches MAY set the AU option on any outgoing query from the cache when performing recursion on behalf of a non-DNSSEC aware stub client. If the stub indicates it is DNSSEC-aware, but does not set the AU option in the query, the DNSSEC validating recursive cache SHOULD NOT set the AU option to avoid conflicts.

Forwarders that do not do validation or caching SHOULD copy the AU option seen in received queries as they represent the wishes of the validating downstream resolver that issued the original query.

## 4. Intermediate Middlebox Considerations

Intermediate middleboxes SHOULD copy the AU option seen in queries from end system resolvers. If the system is validating, it SHOULD also check for the presence of the CD bit in the query. If present, the intermediate middlebox SHOULD copy the AU option as seen in the query. If not set or if the DNSSEC-OK bit is not set, then the validating intermediate middlebox MAY chose to ignore the AU option in the query and MAY include its own preference as the AU option.

## 5. Server Considerations

When an authoritative server sees the AU option in the OPT meta-RR in a request the normal algorithm for servicing requests is followed.

If the AU option is present but the DNSSEC-OK bit is not set, then the authoritative server ignores the ALG-CODE list and does not include any additional DNSSEC RRs in the response.

## 6. Traffic Analysis Considerations

Zone administrators that are planning or are in the process of completing a cryptographic algorithm rollover operation should monitor DNS query traffic and record the values of the AU option in queries. This monitoring can measure the deployment of client code that implements (and signals) certain algorithms. Exactly how to capture DNS traffic and measure new algorithm adoption is beyond the scope of this document.

Zone administrators can use this data to set plans for starting an algorithm rollover and when older algorithms can be phased out without disrupting the majority of clients. In order to keep this disruption to a minimum, zone administrators should wait to complete an algorithm rollover until a large majority of clients signal that they understand the new algorithm. Note that clients that do not implement the AU option may be older implementations which would also not implement any newly deployed algorithm.

## 7. IANA Considerations

The algorithm codes used to identify DNSSEC algorithms has already been established by IANA. This document does not seek to alter that registry in any way.

This draft seeks to update the "DNS EDNS0 Options" registry by adding the AU option and referencing this document. The code for the option should be TBD.

## 8. Security Considerations

This document specifies a way for a client to signal its digital signature algorithm preference to a cache or server. It is not meant to be a discussion on algorithm superiority. The signal is an optional code contained in the OPT meta-RR used with EDNS0. The goal of this option is to signal new algorithm uptake in client code to allow zone administrators to know when it is possible to complete an algorithm rollover in a DNSSEC signed zone.

## 9. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

#### Authors' Addresses

Steve Crocker  
Shinkuro Inc.  
5110 Edgemoor Lane  
Bethesda, MD 20814  
USA

EMail: [steve@shinkuro.com](mailto:steve@shinkuro.com)

Scott Rose  
NIST  
100 Bureau Dr.  
Gaithersburg, MD 20899  
USA

Phone: +1-301-975-8439  
EMail: [scottr.nist@gmail.com](mailto:scottr.nist@gmail.com)





Network Working Group  
Internet-Draft  
Updates: 4033, 4034, 4035, 5155  
(if approved)  
Intended status: Standards Track  
Expires: April 1, 2013

S. Weiler  
SPARTA, Inc.  
D. Blacka  
Verisign, Inc.  
September 28, 2012

Clarifications and Implementation Notes for DNSSEC  
draft-ietf-dnsext-dnssec-bis-updates-20

Abstract

This document is a collection of technical clarifications to the DNSSEC document set. It is meant to serve as a resource to implementors as well as a repository of DNSSEC errata.

This document updates the core DNSSEC documents (RFC4033, RFC4034, and RFC4035) as well as the NSEC3 specification (RFC5155). It also defines NSEC3 and SHA-2 as core parts of the DNSSEC specification.

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 April 1, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction and Terminology . . . . .	4
1.1. Structure of this Document . . . . .	4
1.2. Terminology . . . . .	4
2. Important Additions to DNSSEC . . . . .	4
2.1. NSEC3 Support . . . . .	4
2.2. SHA-2 Support . . . . .	5
3. Scaling Concerns . . . . .	5
3.1. Implement a BAD cache . . . . .	5
4. Security Concerns . . . . .	5
4.1. Clarifications on Non-Existence Proofs . . . . .	5
4.2. Validating Responses to an ANY Query . . . . .	6
4.3. Check for CNAME . . . . .	6
4.4. Insecure Delegation Proofs . . . . .	7
5. Interoperability Concerns . . . . .	7
5.1. Errors in Canonical Form Type Code List . . . . .	7
5.2. Unknown DS Message Digest Algorithms . . . . .	7
5.3. Private Algorithms . . . . .	8
5.4. Caution About Local Policy and Multiple RRSIGs . . . . .	9
5.5. Key Tag Calculation . . . . .	9
5.6. Setting the DO Bit on Replies . . . . .	9
5.7. Setting the AD Bit on Queries . . . . .	9
5.8. Setting the AD Bit on Replies . . . . .	10
5.9. Always set the CD bit on Queries . . . . .	10
5.10. Nested Trust Anchors . . . . .	10
5.11. Mandatory Algorithm Rules . . . . .	11
5.12. Ignore Extra Signatures From Unknown Keys . . . . .	12
6. Minor Corrections and Clarifications . . . . .	12
6.1. Finding Zone Cuts . . . . .	12
6.2. Clarifications on DNSKEY Usage . . . . .	12
6.3. Errors in Examples . . . . .	13
6.4. Errors in RFC 5155 . . . . .	13
7. IANA Considerations . . . . .	13
8. Security Considerations . . . . .	13
9. References . . . . .	14
9.1. Normative References . . . . .	14
9.2. Informative References . . . . .	15
Appendix A. Acknowledgments . . . . .	15
Appendix B. Discussion of Setting the CD Bit . . . . .	16
Appendix C. Discussion of Trust Anchor Preference Options . . . . .	19
C.1. Closest Encloser . . . . .	19
C.2. Accept Any Success . . . . .	20
C.3. Preference Based on Source . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction and Terminology

This document lists some additions, clarifications and corrections to the core DNSSEC specification, as originally described in [RFC4033], [RFC4034], and [RFC4035], and later amended by [RFC5155]. (See section Section 2 for more recent additions to that core document set.)

It is intended to serve as a resource for implementors and as a repository of items that need to be addressed when advancing the DNSSEC documents along the Standards Track.

### 1.1. Structure of this Document

The clarifications and changes to DNSSEC are sorted according to their importance, starting with ones which could, if ignored, lead to security problems and progressing down to clarifications that are expected to have little operational impact.

### 1.2. 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 [RFC2119].

## 2. Important Additions to DNSSEC

This section lists some documents that are now considered core DNSSEC protocol documents in addition to those originally specified in Section 10 of [RFC4033].

### 2.1. NSEC3 Support

[RFC5155] describes the use and behavior of the NSEC3 and NSEC3PARAM records for hashed denial of existence. Validator implementations are strongly encouraged to include support for NSEC3 because a number of highly visible zones use it. Validators that do not support validation of responses using NSEC3 will be hampered in validating large portions of the DNS space.

[RFC5155] is now considered part of the DNS Security Document Family as described by [RFC4033], Section 10.

Note that the algorithm identifiers defined in RFC5155 (DSA-NSEC3-SHA1 and RSASHA1-NSEC3-SHA1) and RFC5702 (RSASHA256 and RSASHA512) signal that a zone might be using NSEC3, rather than NSEC. The zone

may be using either and validators supporting these algorithms MUST support both NSEC3 and NSEC responses.

## 2.2. SHA-2 Support

[RFC4509] describes the use of SHA-256 as a digest algorithm in Delegation Signer (DS) RRs. [RFC5702] describes the use of the RSASHA256 and RSASHA512 algorithms in DNSKEY and RRSIG RRs. Validator implementations are strongly encouraged to include support for these algorithms for DS, DNSKEY, and RRSIG records.

Both [RFC4509] and [RFC5702] are now considered part of the DNS Security Document Family as described by [RFC4033], Section 10.

## 3. Scaling Concerns

### 3.1. Implement a BAD cache

Section 4.7 of RFC4035 permits security-aware resolvers to implement a BAD cache. That guidance has changed: security-aware resolvers SHOULD implement a BAD cache as described in RFC4035.

This change in guidance is based on operational experience with DNSSEC administrative errors leading to significant increases in DNS traffic, with an accompanying realization that such events are more likely and more damaging than originally supposed. An example of one such event is documented in "Roll Over and Die" [Huston].

## 4. Security Concerns

This section provides clarifications that, if overlooked, could lead to security issues.

### 4.1. Clarifications on Non-Existence Proofs

[RFC4035] Section 5.4 under-specifies the algorithm for checking non-existence proofs. In particular, the algorithm as presented would allow a validator to interpret an NSEC or NSEC3 RR from an ancestor zone as proving the non-existence of an RR in a child zone.

An "ancestor delegation" NSEC RR (or NSEC3 RR) is one with:

- o the NS bit set,
- o the SOA bit clear, and

- o a signer field that is shorter than the owner name of the NSEC RR, or the original owner name for the NSEC3 RR.

Ancestor delegation NSEC or NSEC3 RRs MUST NOT be used to assume non-existence of any RRs below that zone cut, which include all RRs at that (original) owner name other than DS RRs, and all RRs below that owner name regardless of type.

Similarly, the algorithm would also allow an NSEC RR at the same owner name as a DNAME RR, or an NSEC3 RR at the same original owner name as a DNAME, to prove the non-existence of names beneath that DNAME. An NSEC or NSEC3 RR with the DNAME bit set MUST NOT be used to assume the non-existence of any subdomain of that NSEC/NSEC3 RR's (original) owner name.

#### 4.2. Validating Responses to an ANY Query

[RFC4035] does not address how to validate responses when QTYPE=\*. As described in Section 6.2.2 of [RFC1034], a proper response to QTYPE=\* may include a subset of the RRsets at a given name. That is, it is not necessary to include all RRsets at the QNAME in the response.

When validating a response to QTYPE=\*, all received RRsets that match QNAME and QCLASS MUST be validated. If any of those RRsets fail validation, the answer is considered Bogus. If there are no RRsets matching QNAME and QCLASS, that fact MUST be validated according to the rules in [RFC4035] Section 5.4 (as clarified in this document). To be clear, a validator must not expect to receive all records at the QNAME in response to QTYPE=\*.

#### 4.3. Check for CNAME

Section 5 of [RFC4035] says nothing explicit about validating responses based on (or that should be based on) CNAMEs. When validating a NOERROR/NODATA response, validators MUST check the CNAME bit in the matching NSEC or NSEC3 RR's type bitmap in addition to the bit for the query type.

Without this check, an attacker could successfully transform a positive CNAME response into a NOERROR/NODATA response by (e.g.) simply stripping the CNAME RRset from the response. A naive validator would then note that the QTYPE was not present in the matching NSEC/NSEC3 RR, but fail to notice that the CNAME bit was set, and thus the response should have been a positive CNAME response.

#### 4.4. Insecure Delegation Proofs

[RFC4035] Section 5.2 specifies that a validator, when proving a delegation is not secure, needs to check for the absence of the DS and SOA bits in the NSEC (or NSEC3) type bitmap. The validator also MUST check for the presence of the NS bit in the matching NSEC (or NSEC3) RR (proving that there is, indeed, a delegation), or alternately make sure that the delegation is covered by an NSEC3 RR with the Opt-Out flag set.

Without this check, an attacker could reuse an NSEC or NSEC3 RR matching a non-delegation name to spoof an unsigned delegation at that name. This would claim that an existing signed RRset (or set of signed RRsets) is below an unsigned delegation, thus not signed and vulnerable to further attack.

### 5. Interoperability Concerns

#### 5.1. Errors in Canonical Form Type Code List

When canonicalizing DNS names (for both ordering and signing), DNS names in the RDATA section of NSEC resource records are not downcased. DNS names in the RDATA section of RRSIG resource records are downcased.

The guidance in the above paragraph differs from what has been published before but is consistent with current common practice. [RFC4034] Section 6.2 item 3 says that names in both of these RR types should be downcased. The earlier [RFC3755] says that they should not. Current practice follows neither document fully.

Section 6.2 of RFC4034 also erroneously lists HINFO as a record that needs downcasing, and twice at that. Since HINFO records contain no domain names, they are not subject to downcasing.

#### 5.2. Unknown DS Message Digest Algorithms

Section 5.2 of [RFC4035] includes rules for how to handle delegations to zones that are signed with entirely unsupported public key algorithms, as indicated by the key algorithms shown in those zone's DS RRsets. It does not explicitly address how to handle DS records that use unsupported message digest algorithms. In brief, DS records using unknown or unsupported message digest algorithms MUST be treated the same way as DS records referring to DNSKEY RRs of unknown or unsupported public key algorithms.

The existing text says:

If the validator does not support any of the algorithms listed in an authenticated DS RRset, then the resolver has no supported authentication path leading from the parent to the child. The resolver should treat this case as it would the case of an authenticated NSEC RRset proving that no DS RRset exists, as described above.

In other words, when determining the security status of a zone, a validator disregards any authenticated DS records that specify unknown or unsupported DNSKEY algorithms. If none are left, the zone is treated as if it were unsigned.

This document modifies the above text to additionally disregard authenticated DS records using unknown or unsupported message digest algorithms.

### 5.3. Private Algorithms

As discussed above, Section 5.2 of [RFC4035] requires that validators make decisions about the security status of zones based on the public key algorithms shown in the DS records for those zones. In the case of private algorithms, as described in [RFC4034] Appendix A.1.1, the eight-bit algorithm field in the DS RR is not conclusive about what algorithm(s) is actually in use.

If no private algorithms appear in the DS RRset, or if any supported algorithm appears in the DS RRset, no special processing is needed. Furthermore, if the validator implementation does not support any private algorithms, or only supports private algorithms using an algorithm number not present in the DS RRset, no special processing is needed.

In the remaining cases, the security status of the zone depends on whether or not the resolver supports any of the private algorithms in use (provided that these DS records use supported message digest algorithms, as discussed in Section 5.2 of this document). In these cases, the resolver MUST retrieve the corresponding DNSKEY for each private algorithm DS record and examine the public key field to determine the algorithm in use. The security-aware resolver MUST ensure that the hash of the DNSKEY RR's owner name and RDATA matches the digest in the DS RR as described in Section 5.2 of [RFC4035], authenticating the DNSKEY. If all of the retrieved and authenticated DNSKEY RRs use unknown or unsupported private algorithms, then the zone is treated as if it were unsigned.

Note that if none of the private algorithm DS RRs can be securely matched to DNSKEY RRs and no other DS establishes that the zone is secure, the referral should be considered Bogus data as discussed in



[RFC4035].

This clarification facilitates the broader use of private algorithms, as suggested by [RFC4955].

#### 5.4. Caution About Local Policy and Multiple RRSIGs

When multiple RRSIGs cover a given RRset, [RFC4035] Section 5.3.3 suggests that "the local resolver security policy determines whether the resolver also has to test these RRSIG RRs and how to resolve conflicts if these RRSIG RRs lead to differing results."

This document specifies that a resolver SHOULD accept any valid RRSIG as sufficient, and only determine that an RRset is Bogus if all RRSIGs fail validation.

If a resolver adopts a more restrictive policy, there's a danger that properly-signed data might unnecessarily fail validation due to cache timing issues. Furthermore, certain zone management techniques, like the Double Signature Zone-signing Key Rollover method described in section 4.2.1.2 of [RFC4641], will not work reliably. Such a resolver is also vulnerable to malicious insertion of gibberish signatures.

#### 5.5. Key Tag Calculation

[RFC4034] Appendix B.1 incorrectly defines the Key Tag field calculation for algorithm 1. It correctly says that the Key Tag is the most significant 16 of the least significant 24 bits of the public key modulus. However, [RFC4034] then goes on to incorrectly say that this is 4th to last and 3rd to last octets of the public key modulus. It is, in fact, the 3rd to last and 2nd to last octets.

#### 5.6. Setting the DO Bit on Replies

As stated in Section 3 of [RFC3225], the DO bit of the query MUST be copied in the response. However, in order to interoperate with implementations that ignore this rule on sending, resolvers MUST ignore the DO bit in responses.

#### 5.7. Setting the AD Bit on Queries

The semantics of the AD bit in the query were previously undefined. Section 4.6 of [RFC4035] instructed resolvers to always clear the AD bit when composing queries.

This document defines setting the AD bit in a query as a signal indicating that the requester understands and is interested in the

value of the AD bit in the response. This allows a requestor to indicate that it understands the AD bit without also requesting DNSSEC data via the DO bit.

#### 5.8. Setting the AD Bit on Replies

Section 3.2.3 of [RFC4035] describes under which conditions a validating resolver should set or clear the AD bit in a response. In order to interoperate with legacy stub resolvers and middleboxes that neither understand nor ignore the AD bit, validating resolvers SHOULD only set the AD bit when a response both meets the conditions listed in RFC 4035, section 3.2.3, and the request contained either a set DO bit or a set AD bit.

#### 5.9. Always set the CD bit on Queries

When processing a request with the CD bit set, a resolver SHOULD attempt to return all response data, even data that has failed DNSSEC validation. RFC4035 section 3.2.2 requires a resolver processing a request with the CD bit set to set the CD bit on its upstream queries.

This document further specifies that validating resolvers SHOULD set the CD bit on every upstream query. This is regardless of whether the CD bit was set on the incoming query or whether it has a trust anchor at or above the QNAME.

[RFC4035] is ambiguous about what to do when a cached response was obtained with the CD bit unset, a case that only arises when the resolver chooses not to set the CD bit on all upstream queries, as specified above. In the typical case, no new query is required, nor does the cache need to track the state of the CD bit used to make a given query. The problem arises when the cached response is a server failure (RCODE 2), which may indicate that the requested data failed DNSSEC validation at an upstream validating resolver. ([RFC2308] permits caching of server failures for up to five minutes.) In these cases, a new query with the CD bit set is required.

Appendix B discusses more of the logic behind the recommendation presented in this section.

#### 5.10. Nested Trust Anchors

A DNSSEC validator may be configured such that, for a given response, more than one trust anchor could be used to validate the chain of trust to the response zone. For example, imagine a validator configured with trust anchors for "example." and "zone.example." When the validator is asked to validate a response to

"www.sub.zone.example.", either trust anchor could apply.

When presented with this situation, DNSSEC validators have a choice of which trust anchor(s) to use. Which to use is a matter of implementation choice. Appendix C discusses several possible algorithms.

It is possible and advisable to expose the choice of policy as a configuration option. As a default, it is suggested that validators implement the "Accept Any Success" policy described in Appendix C.2 while exposing other policies as configuration options.

The "Accept Any Success" policy is to try all applicable trust anchors until one gives a validation result of Secure, in which case the final validation result is Secure. If and only if all applicable trust anchors give a result of Insecure, the final validation result is Insecure. If one or more trust anchors lead to a Bogus result and there is no Secure result, then the final validation result is Bogus.

#### 5.11. Mandatory Algorithm Rules

The last paragraph of RFC4035 Section 2.2 includes rules describing which algorithms must be used to sign a zone. Since these rules have been confusing, they are restated using different language here:

The DS RRset and DNSKEY RRset are used to signal which algorithms are used to sign a zone. The presence of an algorithm in either a zone's DS or DNSKEY RRset signals that that algorithm is used to sign the entire zone.

A signed zone MUST include a DNSKEY for each algorithm present in the zone's DS RRset and expected trust anchors for the zone. The zone MUST also be signed with each algorithm (though not each key) present in the DNSKEY RRset. It is possible to add algorithms at the DNSKEY that aren't in the DS record, but not vice-versa. If more than one key of the same algorithm is in the DNSKEY RRset, it is sufficient to sign each RRset with any subset of these DNSKEYs. It is acceptable to sign some RRsets with one subset of keys (or key) and other RRsets with a different subset, so long as at least one DNSKEY of each algorithm is used to sign each RRset. Likewise, if there are DS records for multiple keys of the same algorithm, any subset of those may appear in the DNSKEY RRset.

This requirement applies to servers, not validators. Validators SHOULD accept any single valid path. They SHOULD NOT insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work. A validator MAY have a configuration option to perform a signature completeness

test to support troubleshooting.

#### 5.12. Ignore Extra Signatures From Unknown Keys

Validating resolvers MUST disregard RRSIGs in a zone that do not (currently) have a corresponding DNSKEY in the zone. Similarly, a validating resolver MUST disregard RRSIGs with algorithm types that don't exist in the DNSKEY RRset.

Good key rollover and algorithm rollover practices, as discussed in RFC4641 and its successor documents and as suggested by the rules in the previous section, may require that such RRSIGs be present in a zone.

### 6. Minor Corrections and Clarifications

#### 6.1. Finding Zone Cuts

Appendix C.8 of [RFC4035] discusses sending DS queries to the servers for a parent zone but does not state how to find those servers. Specific instructions can be found in Section 4.2 of [RFC4035].

#### 6.2. Clarifications on DNSKEY Usage

It is possible to use different DNSKEYs to sign different subsets of a zone, constrained only by the rules in Section 5.11. It is even possible to use a different DNSKEY for each RRset in a zone, subject only to practical limits on the size of the DNSKEY RRset and the above rules. However, be aware that there is no way to tell resolvers what a particular DNSKEY is supposed to be used for -- any DNSKEY in the zone's signed DNSKEY RRset may be used to authenticate any RRset in the zone. For example, if a weaker or less trusted DNSKEY is being used to authenticate NSEC RRsets or all dynamically updated records, that same DNSKEY can also be used to sign any other RRsets from the zone.

Furthermore, note that the SEP bit setting has no effect on how a DNSKEY may be used -- the validation process is specifically prohibited from using that bit by [RFC4034] section 2.1.2. It is possible to use a DNSKEY without the SEP bit set as the sole secure entry point to the zone, yet use a DNSKEY with the SEP bit set to sign all RRsets in the zone (other than the DNSKEY RRset). It is also possible to use a single DNSKEY, with or without the SEP bit set, to sign the entire zone, including the DNSKEY RRset itself.

### 6.3. Errors in Examples

The text in [RFC4035] Section C.1 refers to the examples in B.1 as "x.w.example.com" while B.1 uses "x.w.example". This is painfully obvious in the second paragraph where it states that the RRSIG labels field value of 3 indicates that the answer was not the result of wildcard expansion. This is true for "x.w.example" but not for "x.w.example.com", which of course has a label count of 4 (antithetically, a label count of 3 would imply the answer was the result of a wildcard expansion).

The first paragraph of [RFC4035] Section C.6 also has a minor error: the reference to "a.z.w.w.example" should instead be "a.z.w.example", as in the previous line.

### 6.4. Errors in RFC 5155

A NSEC3 record that matches an Empty Non-Terminal effectively has no type associated with it. This NSEC3 record has an empty type bit map. Section 3.2.1 of [RFC5155] contains the statement:

Blocks with no types present MUST NOT be included.

However, the same section contains a regular expression:

Type Bit Maps Field = ( Window Block # | Bitmap Length | Bitmap )+

The plus sign in the regular expression indicates that there is one or more of the preceding element. This means that there must be at least one window block. If this window block has no types, it contradicts with the first statement. Therefore, the correct text in RFC 5155 3.2.1 should be:

Type Bit Maps Field = ( Window Block # | Bitmap Length | Bitmap )\*

## 7. IANA Considerations

This document specifies no IANA Actions.

## 8. Security Considerations

This document adds SHA-2 and NSEC3 support to the core DNSSEC protocol. Security considerations for those features are discussed in the documents defining them. Additionally, this document addresses some ambiguities and omissions in the core DNSSEC documents that, if not recognized and addressed in implementations, could lead

to security failures. In particular, the validation algorithm clarifications in Section 4 are critical for preserving the security properties DNSSEC offers. Furthermore, failure to address some of the interoperability concerns in Section 5 could limit the ability to later change or expand DNSSEC, including adding new algorithms.

The recommendation in Section 5.9 to always set the CD bit has security implications. By setting the CD bit, a resolver will not benefit from more stringent validation rules or a more complete set of trust anchors at an upstream validator.

## 9. References

### 9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, December 2001.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.
- [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, May 2006.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.
- [RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, October 2009.

## 9.2. Informative References

- [Huston] Michaelson, G., Wallstrom, P., Arends, R., and G. Huston, "Roll Over and Die?", February 2010.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.
- [RFC3755] Weiler, S., "Legacy Resolver Compatibility for Delegation Signer (DS)", RFC 3755, May 2004.
- [RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, September 2006.
- [RFC4955] Blacka, D., "DNS Security (DNSSEC) Experiments", RFC 4955, July 2007.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC 5011, September 2007.
- [RFC5074] Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074, November 2007.

## Appendix A. Acknowledgments

The editors would like to thank Rob Austein for his previous work as an editor of this document.

The editors are extremely grateful to those who, in addition to finding errors and omissions in the DNSSEC document set, have provided text suitable for inclusion in this document.

The lack of specificity about handling private algorithms, as described in Section 5.3, and the lack of specificity in handling ANY queries, as described in Section 4.2, were discovered by David Blacka.

The error in algorithm 1 key tag calculation, as described in Section 5.5, was found by Abhijit Hayatnagarkar. Donald Eastlake contributed text for Section 5.5.

The bug relating to delegation NSEC RR's in Section 4.1 was found by Roy Badami. Roy Arends found the related problem with DNAME.

The errors in the [RFC4035] examples were found by Roy Arends, who also contributed text for Section 6.3 of this document.

Text on the mandatory algorithm rules was derived from suggestions by Matthijs Mekking and Ed Lewis.

The CD bit logic was analyzed in depth by David Blacka, Olafur Gudmundsson, Mike St. Johns, and Andrew Sullivan.

The editors would like to thank Alfred Hoenes, Ed Lewis, Danny Mayer, Olafur Gudmundsson, Suzanne Woolf, Rickard Bellgrim, Mike St. Johns, Mark Andrews, Wouter Wijngaards, Matthijs Mekking, Andrew Sullivan, Jeremy Reed, Paul Hoffman, Mohan Parthasarathy, Florian Weimer, Warren Kumari and Scott Rose for their contributions to this document.

## Appendix B. Discussion of Setting the CD Bit

[RFC4035] may be read as relying on the implicit assumption that there is at most one validating system between the stub resolver and the authoritative server for a given zone. It is entirely possible, however, for more than one validator to exist between a stub resolver and an authoritative server. If these different validators have disjoint trust anchors configured, then it is possible that each would be able to validate some portion of the DNS tree but neither is able to validate all of it. Accordingly, it might be argued that it is desirable not to set the CD bit on upstream queries, because that allows for maximal validation.

In section Section 5.9 of this document, it is recommended to set the CD bit on an upstream query even when the incoming query arrives with CD=0. This is for two reasons: it encourages a more predictable validation experience as only one validator is always doing the validation, and it ensures that all DNSSEC data that exists may be available from the local cache should a query with CD=1 arrive.

As a matter of policy, it is possible to set the CD bit differently than suggested in Section 5.9. A different choice will, of course, not always yield the benefits listed above. It is beyond the scope of this document to outline all of the considerations and counter considerations for all possible policies. Nevertheless, it is possible to describe three approaches and their underlying philosophy of operation. These are laid out in the tables below.

The table that describes each model has five columns. The first column indicates the value of the CD bit that the resolver receives (for instance, on the name server side in an iterative resolver, or as local policy or from the API in the case of a stub). The second column indicates whether the query needs to be forwarded for resolution (F) or can be satisfied from a local cache (C). The third



column is a line number, so that it can be referred to later in the table. The fourth column indicates any relevant conditions at the resolver: whether the resolver has a covering trust anchor and so on. If there are no parameters here, the column is empty. The fifth and final column indicates what action the resolver takes.

The tables differentiate between "cached data" and "cached RCODE=2". This is a shorthand; the point is that one has to treat RCODE=2 (server failure) as special, because it might indicate a validation failure somewhere upstream. The distinction is really between "cached RCODE=2" and "cached everything else".

The tables are probably easiest to think of in terms of describing what happens when a stub resolver sends a query to an intermediate resolver, but they are perfectly general and can be applied to any validating resolver.

Model 1: "always set"

This model is so named because the validating resolver sets the CD bit on queries it makes regardless of whether it has a covering trust anchor for the query. The general philosophy represented by this table is that only one resolver should be responsible for validation irrespective of the possibility that an upstream resolver may be present with trust anchors that cover different or additional QNAMEs. It is the model recommended in Section 5.9 of this document.

CD	F/C	line	conditions	action
1	F	A1		Set CD=1 on upstream query
0	F	A2		Set CD=1 on upstream query
1	C	A3		Return the cache contents (data or RCODE=2)
0	C	A4	no covering TA	Return cache contents (data or RCODE=2)
0	C	A5	covering TA	Validate cached result and return it.

Model 2: "never set when receiving CD=0"

This model is so named because it sets CD=0 on upstream queries for all received CD=0 queries even if it has a covering trust anchor. The general philosophy represented by this table is that more than one resolver may take responsibility for validating a QNAME and that a validation failure for a QNAME by any resolver in the chain is a validation failure for the query. Using this model is NOT RECOMMENDED.

CD	F/C	line	conditions	action
=====				
1	F	N1		Set CD=1 on upstream query
0	F	N2		Set CD=0 on upstream query
1	C	N3	cached data	Return cached data
1	C	N4	cached RCODE=2	Treat as line N1
0	C	N5	no covering TA	Return cache contents (data or RCODE=2)
0	C	N6	covering TA & cached data was generated with CD=1	Treat as line N2
0	C	N7	covering TA & cached data was generated with CD=0	Validate and return

## Model 3: "sometimes set"

This model is so named because it sets the CD bit on upstream queries triggered by received CD=0 queries based on whether the validator has a trust anchor configured that covers the query. If there is no covering trust anchor, the resolver clears the CD bit in the upstream query. If there is a covering trust anchor, the resolver sets CD=1 and performs validation itself. The general philosophy represented by this table is that a resolver should try and validate QNAMEs for which it has trust anchors and should not preclude validation by other resolvers for QNAMEs for which it does not have covering trust anchors. Using this model is NOT RECOMMENDED.

CD	F/C	line	conditions	action
1	F	S1		Set CD=1 on upstream query
0	F	S2	covering TA	Set CD=1 on upstream query
0	F	S3	no covering TA	Set CD=0 on upstream query
1	C	S4	cached data	Return cached data
1	C	S5	cached RCODE=2	Treat as line S1
0	C	S6	cached data was generated with CD=0	Return cache contents
0	C	S7	cached data was generated with CD=1 & covering TA	Validate & return cache contents
0	C	S8	cached RCODE=2	Return cache contents
0	C	S9	cached data was generated with CD=1 & no covering TA	Treat as line S3

## Appendix C. Discussion of Trust Anchor Preference Options

This section presents several different policies for validating resolvers to use when they have a choice of trust anchors available for validating a given answer.

## C.1. Closest Encloser

One policy is to choose the trust anchor closest to the QNAME of the response. For example, consider a validator configured with trust anchors for "example." and "zone.example." When asked to validate a response for "www.sub.zone.example.", a validator using the "Closest

Encloser" policy would choose the "zone.example." trust anchor.

This policy has the advantage of allowing the operator to trivially override a parent zone's trust anchor with one that the operator can validate in a stronger way, perhaps because the resolver operator is affiliated with the zone in question. This policy also minimizes the number of public key operations needed, which is of benefit in resource-constrained environments.

This policy has the disadvantage of giving the user some unexpected and unnecessary validation failures when sub-zone trust anchors are neglected. As a concrete example, consider a validator that configured a trust anchor for "zone.example." in 2009 and one for "example." in 2011. In 2012, "zone.example." rolls its KSK and updates its DS records, but the validator operator doesn't update its trust anchor. With the "closest encloser" policy, the validator gets validation failures.

### C.2. Accept Any Success

Another policy is to try all applicable trust anchors until one gives a validation result of Secure, in which case the final validation result is Secure. If and only if all applicable trust anchors give a result of Insecure, the final validation result is Insecure. If one or more trust anchors lead to a Bogus result and there is no Secure result, then the final validation result is Bogus.

This has the advantage of causing the fewest validation failures, which may deliver a better user experience. If one trust anchor is out of date (as in our above example), the user may still be able to get a Secure validation result (and see DNS responses).

This policy has the disadvantage of making the validator subject to the compromise of the weakest of these trust anchors while making it relatively painless to keep old trust anchors configured in perpetuity.

### C.3. Preference Based on Source

When the trust anchors have come from different sources (e.g. automated updates ([RFC5011]), one or more DLV registries ([RFC5074]), and manually configured), a validator may wish to choose between them based on the perceived reliability of those sources. The order of precedence might be exposed as a configuration option.

For example, a validator might choose to prefer trust anchors found in a DLV registry over those manually configured on the theory that the manually configured ones will not be as aggressively maintained.

Conversely, a validator might choose to prefer manually configured trust anchors over those obtained from a DLV registry on the theory that the manually configured ones have been more carefully authenticated.

Or the validator might do something more complex: prefer a sub-set of manually configured trust anchors (based on a configuration option), then trust anchors that have been updated using the RFC5011 mechanism, then trust anchors from one DLV registry, then trust anchors from a different DLV registry, then the rest of the manually configured trust anchors.

#### Authors' Addresses

Samuel Weiler  
SPARTA, Inc.  
7110 Samuel Morse Drive  
Columbia, Maryland 21046  
US

Email: [weiler@tislabs.com](mailto:weiler@tislabs.com)

David Blacka  
Verisign, Inc.  
12061 Bluemont Way  
Reston, VA 20190  
US

Email: [davidb@verisign.com](mailto:davidb@verisign.com)



DNSEXT Working Group  
Internet-Draft  
Obsoletes: 2671, 2673  
(if approved)

Intended status: Standards Track  
Expires: July 4, 2013

J. Damas  
Bond Internet Systems  
M. Graff

P. Vixie  
Internet Systems Consortium  
December 31, 2012

Extension Mechanisms for DNS (EDNS(0))  
draft-ietf-dnsext-rfc2671bis-edns0-10

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 requestors to advertise their capabilities to responders. This document describes backward compatible mechanisms for allowing the protocol to grow.

This document updates the EDNS(0) specification (and obsoletes RFC 2671) based on feedback from deployment experience in several implementations. It also obsoletes RFC 2673 ("Binary Labels in the Domain Name System") and adds considerations on the use of extended labels in the DNS.

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 July 4, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.



## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	4
3. EDNS Support Requirement . . . . .	5
4. DNS Message changes . . . . .	5
4.1. Message Header . . . . .	5
4.2. Label Types . . . . .	5
4.3. UDP Message Size . . . . .	5
5. Extended Label Types . . . . .	6
6. The OPT pseudo-RR . . . . .	6
6.1. OPT Record Definition . . . . .	6
6.1.1. Basic elements . . . . .	6
6.1.2. Wire Format . . . . .	7
6.1.3. OPT Record TTL Field Use . . . . .	8
6.1.4. Flags . . . . .	9
6.2. Behaviour . . . . .	9
6.2.1. Cache behaviour . . . . .	9
6.2.2. Fallback . . . . .	9
6.2.3. Requestor's Payload Size . . . . .	10
6.2.4. Responder's Payload Size . . . . .	10
6.2.5. Payload Size Selection . . . . .	11
6.2.6. Support in MiddleBoxes . . . . .	11
7. Transport Considerations . . . . .	12
8. Security Considerations . . . . .	12
9. IANA Considerations . . . . .	13
9.1. OPT Option Code Allocation Procedure . . . . .	14
10. References . . . . .	14
10.1. Normative References . . . . .	14
10.2. Informative References . . . . .	14
Appendix A. Document Editing History . . . . .	15
A.1. Changes since RFC2671 . . . . .	15
A.2. Changes since -02 . . . . .	15
Authors' Addresses . . . . .	15

## 1. Introduction

DNS [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 over UDP not using the extensions described in this document is limited to 512 bytes. Many of DNS's protocol limits, such as the maximum message size over UDP, are too small to efficiently support the additional information that can be conveyed in the DNS (e.g. several IPv6 addresses or DNSSEC signatures). Finally, RFC 1035 does not define any way for implementations to advertise their capabilities to any of the other actors they interact with.

[RFC2671] added an extension mechanism to DNS. This mechanism is widely supported and a number of new DNS uses and protocol extensions depend on the presence of these extensions. This memo refines that specification and obsoletes [RFC2671].

Unextended agents will not know how to interpret the protocol extensions defined in [RFC2671] and restated here. Extended agents need to be prepared for handling the interactions with unextended clients in the face of new protocol elements, and fall back gracefully to unextended DNS.

EDNS is a hop-by-hop extension to DNS. This means the use of EDNS is negotiated between each pair of hosts in a DNS resolution process. For instance the stub resolver communicating with the recursive resolver or the recursive resolver communicating with an authoritative server.

[RFC2671] specified extended label types. The only such label proposed was in [RFC2673] for a label type called "Bitstring Labels." For various reasons introducing a new label type was found to be extremely difficult, and [RFC2673] was moved to Experimental. This document deprecates Extended Labels, and therefore Binary Labels, obsoleting [RFC2673].

## 2. Terminology

"Requestor" is the side which sends a request. "Responder" is an authoritative, recursive resolver, or other DNS component which responds to questions. Other terminology is used as per its use in the references.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

### 3. EDNS Support Requirement

EDNS provides a mechanism to improve the scalability of DNS as its uses get more diverse on the Internet. It does this by enabling the use of UDP transport for DNS messages with sizes beyond the limits specified in RFC 1035 as well as providing extra data space for additional flags and return codes (RCODEs). However, implementation experiencing indicates that adding new RCODEs should be avoided due to the difficulty in upgrading the installed base. Flags SHOULD be used only when necessary for DNS resolution to function.

For many uses, a EDNS Option Code may be preferred.

With time, some applications of DNS have made EDNS a requirement for their deployment. For instance, DNSSEC uses the additional flag space introduced in EDNS to signal the request to include DNSSEC data in a DNS response.

Given the increase in DNS response sizes when including larger data items such as AAAA Records, DNSSEC information (e.g. RRSIG or DNSKEY) or large TXT Records, the additional UDP payload capabilities provided by EDNS can help improve the scalability of the DNS by avoiding widespread use of TCP for DNS transport.

### 4. DNS Message changes

#### 4.1. Message Header

The DNS Message Header's second full 16-bit word is divided into a 4-bit OPCODE, a 4-bit RCODE, and a number of 1-bit flags (see , section 4.1.1 [RFC1035]). Some of these were marked for future use, and most these have since been allocated. Also, most of the RCODE values are now in use. The OPT pseudo-RR specified below contains extensions to the RCODE bit field as well as additional flag bits.

#### 4.2. Label Types

The first two bits of a wire format domain label are used to denote the type of the label. [RFC1035] allocates two of the four possible types and reserves the other two. More label types were defined in [RFC2671]. This document obsoletes the use of the 2-bit combination defined by [RFC2671] to identify extended label types.

#### 4.3. UDP Message Size

Traditional DNS Messages are limited to 512 octets in size when sent over UDP [RFC1035]. Fitting the increasing amounts of data that can

be transported in DNS in this 512-byte limit is becoming more difficult. For instance, inclusion of DNSSEC records frequently requires a much larger response than a 512 byte message can hold.

EDNS(0) is intended to provide support for transporting these larger packet sizes while continuing to use UDP. It specifies a way to advertise additional features such as larger response size capability, which is intended to help avoid truncated UDP responses which then cause retry over TCP.

## 5. Extended Label Types

The first octet in the on-the-wire representation of a DNS label specifies the label type; the basic DNS specification [RFC1035] dedicates the two most significant bits of that octet for this purpose.

[RFC2671] defined DNS label type 0b01 for use as an indication for Extended Label Types. A specific Extended Label Type was selected by the 6 least significant bits of the first octet. Thus, Extended Label Types were indicated by the values 64-127 (0b01xxxxxx) in the first octet of the label.

Extended Label Types are extremely difficult to deploy due to lack of support in clients and intermediate gateways as described in [RFC3363] which moved [RFC2673] to experimental status, and [RFC3364], which describes the pros and cons. As such proposals that contemplate extended labels SHOULD weigh this deployment cost against the possibility of implementing functionality in other ways.

Finally, implementations MUST NOT generate or pass Binary Labels in their communications as they are now deprecated.

## 6. The OPT pseudo-RR

### 6.1. OPT Record Definition

#### 6.1.1. Basic elements

An OPT pseudo-RR (sometimes called a meta-RR) MAY be added to the additional data section of a request.

The OPT RR has RR type 41.

If present in requests, compliant responders MUST include an OPT record in their respective responses.

An OPT record does not carry any DNS data. It is used only to contain control information pertaining to the question and answer sequence of a specific transaction. OPT RRs MUST NOT be cached, forwarded, or stored in or loaded from master files.

The OPT RR MAY be placed anywhere within the additional data section. When an OPT RR MUST is included within any DNS message only ONE OPT RR can be present. If a query message with more than one OPT RR is received, a FORMERR (RCODE=1) MUST be returned. The placement flexibility for the OPT RR does not override the need for the TSIG or SIG(0) RRs to be the last in the additional section whenever they are present.

#### 6.1.2. Wire Format

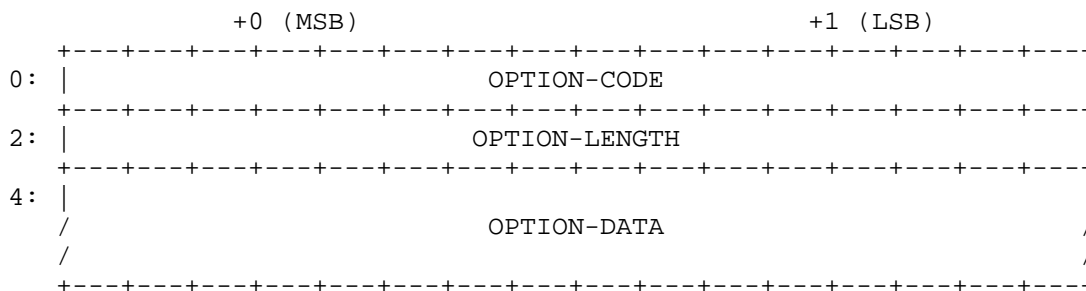
An OPT RR has a fixed part and a variable set of options expressed as {attribute, value} pairs. The fixed part holds some DNS meta data and also a small collection of basic extension elements which we expect to be so popular that it would be a waste of wire space to encode them as {attribute, value}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

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

Field Name	Field Type	Description
NAME	domain name	MUST be 0 (root domain)
TYPE	u_int16_t	OPT (41)
CLASS	u_int16_t	requestor's UDP payload size
TTL	u_int32_t	extended RCODE and flags
RDLEN	u_int16_t	length of all RDATA
RDATA	octet stream	{attribute,value} pairs

OPT RR Format

The variable part of an OPT RR may contain zero or more options in the RDATA. Each option MUST be treated as a bit field. Each option is encoded as:

**OPTION-CODE**

Assigned by the Expert Review process as defined by the dnsect working group and the IESG.

**OPTION-LENGTH**

Size (in octets) of OPTION-DATA.

**OPTION-DATA**

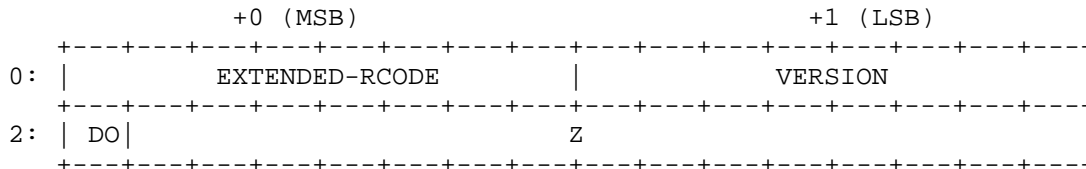
Varies per OPTION-CODE. MUST be treated as a bit field.

The order of appearance of option tuples is not defined. If one option modifies the behavior of another or multiple options are related to one another in some way, they have the same effect regardless of ordering in the RDATA wire encoding.

Any OPTION-CODE values not understood by a responder or requestor MUST be ignored. Specifications of such options might wish to include some kind of signaled acknowledgement. For example, an option specification might say that if a responder sees and supports option XYZ, it MUST include option XYZ in its response.

**6.1.3. OPT Record TTL Field Use**

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 (together with the 4 bits defined in [RFC1035]). Note that EXTENDED-RCODE value 0 indicates that an unextended RCODE is in use (values 0 through 15).

**VERSION**

Indicates the implementation level of the setter. 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 MAY ideally be a run time configuration option. If a responder does not implement the VERSION level of the request, then it MUST respond with RCODE=BADVERS. All responses MUST be limited in format to the VERSION level of the request, but the VERSION of each response SHOULD 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 and including RCODE=BADVERS.

**6.1.4. Flags****DO**

DNSSEC OK bit as defined by [RFC3225].

**Z**

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

**6.2. Behaviour****6.2.1. Cache behaviour**

The OPT record MUST NOT be cached.

**6.2.2. Fallback**

If a requestor detects that the remote end does not support EDNS(0), it MAY issue queries without an OPT record. It MAY cache this knowledge for a brief time in order to avoid fallback delays in the future. However, if DNSSEC or any future option using EDNS is required, no fallback should be performed as they are only signalled through EDNS. If an implementation detects that some servers for the zone support EDNS(0) while others would force the use of TCP to fetch

all data, preference MAY be given to those which support EDNS(0). Implementers SHOULD analyse this choice and the impact on both endpoints.

#### 6.2.3. Requestor's Payload Size

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

Values lower than 512 MUST be treated as equal to 512.

The requestor SHOULD place a value in this field that it can actually receive. For example, if a requestor sits behind a firewall which will block fragmented IP packets, a requestor SHOULD NOT choose a value which will cause fragmentation. Doing so will prevent large responses from being received, and can cause fallback to occur. This knowledge may be auto-detected by the implementation or provided by a human administrator.

Note that a 512-octet UDP payload requires a 576-octet IP reassembly buffer. Choosing between 1280 and 1410 bytes for IP (v4 or v6) over Ethernet would be reasonable.

Bigger values SHOULD be considered by implementers to be used where fragmentation is not a concern. Implementations SHOULD use their largest configured or implemented values as a starting point in an EDNS transaction in the absence of previous knowledge about the destination server.

Choosing a very large value will guarantee fragmentation at the IP layer, and may prevent answers from being received due to a single fragment loss or misconfigured firewalls.

The requestor's maximum payload size can change over time. It MUST NOT be cached for use beyond the transaction in which it is advertised.

#### 6.2.4. Responder's Payload Size

The responder's maximum payload size can change over time, but can be reasonably expected to remain constant between two closely spaced sequential transactions; for example, an arbitrary QUERY used as a probe 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.

#### 6.2.5. Payload Size Selection

Due to transaction overhead, it is not recommended to advertise an architectural limit as a maximum UDP payload size. Even on system stacks capable of reassembling 64KB datagrams, memory usage at low levels in the system will be a concern. A good compromise may be the use of a EDNS maximum payload size of 4096 octets as a starting point.

A requestor MAY choose to implement a fallback to smaller advertised sizes to work around firewall or other network limitations. A requestor SHOULD choose to use a fallback mechanism which begins with a large size, such as 4096. If that fails, a fallback around the 1280-1410 byte range SHOULD be tried, as it has a reasonable chance to fit within a single Ethernet frame. Failing that, a requestor MAY choose a 512 byte packet, which with large answers may cause a TCP retry.

Values of less than 512 bytes MUST be treated as equal to 512 bytes.

#### 6.2.6. Support in MiddleBoxes

In a network that carries DNS traffic there could be active equipment other than that participating directly in the DNS resolution process (stub and caching resolvers, authoritative servers) that affect the transmission of DNS messages (e.g. firewalls, load balancers, proxies, etc) referred to here as MiddleBoxes.

Conformant MiddleBoxes MUST NOT limit DNS messages over UDP to 512 bytes.

MiddleBoxes which simply forward requests to a recursive resolver MUST NOT modify and MUST NOT delete the OPT record contents in either direction.

MiddleBoxes which have additional functionality, such as answering queries or acting as intelligent forwarders, SHOULD be able to process the OPT record and act based on its contents. These boxes MUST consider the incoming request and any outgoing requests as separate transactions if the characteristics of the messages are different.

A more in depth discussion of this type of equipment and other considerations regarding their interaction with DNS traffic is found

in [RFC5625]

## 7. Transport Considerations

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.

Lack of presence of an OPT record in a request MUST be taken as an indication that the requestor does not implement any part of this specification and that the responder MUST NOT include an OPT record in its response.

Extended agents MUST be prepared for handling the interactions with unextended clients in the face of new protocol elements, and fall back gracefully to unextended DNS when needed.

Responders which choose not to implement the protocol extensions defined in this document MUST respond with a return code (RCODE) of FORMERR to messages containing an OPT RR in the additional section and MUST NOT include an OPT record in the response.

If there is a problem with processing the OPT record itself, such as an option value that is badly formatted or includes out of range values, a FORMERR MUST be returned. If this occurs the response MUST include an OPT record. This is intended to allow the requestor to distinguish between servers which do not implement EDNS and format errors within EDNS.

The minimal response MUST be the DNS header, question section, and an OPT record. This MUST also occur when an truncated response (using the DNS header's TC bit) is returned.

## 8. Security Considerations

Requestor-side specification of the maximum buffer size may open a 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.

Announcing very large UDP buffer sizes may result in dropping by middleboxes (see Section 6.2.6). This could cause retransmissions with no hope of success. Some devices have been found to reject fragmented UDP packets.

Announcing too small UDP buffer sizes may result in fallback to TCP with a corresponding load impact on DNS servers. This is especially important with DNSSEC, where answers are much larger.

## 9. IANA Considerations

The IANA has assigned RR type code 41 for OPT.

[RFC2671] specified a number of IANA sub-registries within "DOMAIN NAME SYSTEM PARAMETERS:"

- o DNS EDNS(0) Options
- o EDNS Version Number
- o EDNS Header Flags

Additionally, two entries were generated in existing registries:

EDNS Extended Label Type in the DNS Label Types Registry

Bad OPT Version in the DNS RCODES registry

IANA is advised to update references to [RFC2671] in these entries and sub-registries to this document.

[RFC2671] created the "EDNS Extended Label Type Registry". We request that this registry be closed.

This document assigns option code 65535 in the "EDNS Option Codes" registry to "Reserved for future expansion."

[RFC2671] expands the RCODE space from 4 bits to 12 bits. This allows more than the 16 distinct RCODE values allowed in [RFC1035]. IETF Standards Action is required to add a new RCODE.

This document assigns EDNS Extended RCODE 16 to "BADVERS" in the DNS RCODES registry.

[RFC2671] called for the recording of assignment of extended label types 0bxx111111 as "Reserved for future extended label types". This request implied a request to open a new registry for Extended Label Types but due to the possibility of ambiguity new text registrations were instead made within the general "DNS Label Types" registry which also registers entries originally defined by [RFC1035].

This document requests IANA to close registration of further Extended

Label Types in the "DNS Label Types" Registry.

IETF Standards Action is required for assignments of new EDNS(0) flags. Flags SHOULD be used only when necessary for DNS resolution to function. For many uses, a EDNS Option Code may be preferred.

IETF Standards Action is required to create new entries in the EDNS Version Number registry. Within the EDNS Option Code space Expert Review is required for allocation of an EDNS Option Code. IANA is requested to keep a registry for the EDNS Option Code space.

#### 9.1. OPT Option Code Allocation Procedure

OPT Option Codes are assigned by expert review.

Assignment of Option Codes should be liberal, but duplicate functionality is to be avoided.

### 10. References

#### 10.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.
- [RFC3225] Conrad, D., "Indicating Resolver Support of DNSSEC", RFC 3225, December 2001.

#### 10.2. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2673] Crawford, M., "Binary Labels in the Domain Name System", RFC 2673, August 1999.
- [RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, August 2002.
- [RFC3364] Austein, R., "Tradeoffs in Domain Name System (DNS) Support for Internet Protocol version 6 (IPv6)", RFC 3364, August 2002.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines",  
BCP 152, RFC 5625, August 2009.

## Appendix A. Document Editing History

Following is a list of high-level changes made to the original RFC2671.

### A.1. Changes since RFC2671

- o Support for the OPT record is now mandatory.
- o Extended label types obsoleted and the registry is closed.
- o The bitstring label type, which was already moved from draft to experimental, is requested to be moved to historical.
- o Changes in how EDNS buffer sizes are selected, with recommendations on how to select them.
- o Front material (IPR notice and such) was updated to current requirements.

### A.2. Changes since -02

- o Specified the method for allocation of constants.
- o Cleaned up a lot of wording, along with quite a bit of document structure changes.

## Authors' Addresses

Joao Damas  
Bond Internet Systems  
Av Albufera 14  
S.S. Reyes, Madrid 28701  
ES

Phone: +1 650.423.1312  
Email: joao@bondis.org

Michael Graff

Email: explorer@flame.org

Paul Vixie  
Internet Systems Consortium  
950 Charter Street  
Redwood City, California 94063  
US

Phone: +1 650.423.1301  
Email: vixie@isc.org



DNS Extensions Working Group  
Internet-Draft  
Obsoletes: 2672 (if approved)  
Updates: 3363,4294 (if approved)  
Intended status: Standards Track  
Expires: October 21, 2012

S. Rose  
NIST  
W. Wijngaards  
NLnet Labs  
April 19, 2012

DNAME Redirection in the DNS  
draft-ietf-dnsext-rfc2672bis-dname-26

## Abstract

The DNAME record provides redirection for a sub-tree of the domain name tree in the DNS system. That is, all names that end with a particular suffix are redirected to another part of the DNS. This is a revision to the original specification in RFC 2672 (which this document obsoletes) as well as updating RFC 3363 and RFC 4294 to align with this revision.

## 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 RFC 2119 [RFC2119].

## 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 October 21, 2012.

## Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.



This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	5
2. The DNAME Resource Record . . . . .	5
2.1. Format . . . . .	5
2.2. The DNAME Substitution . . . . .	6
2.3. DNAME Owner Name Matching the QNAME . . . . .	8
2.4. Names Next to and Below a DNAME Record . . . . .	8
2.5. Compression of the DNAME record. . . . .	8
3. Processing . . . . .	9
3.1. CNAME synthesis . . . . .	9
3.2. Server algorithm . . . . .	10
3.3. Wildcards . . . . .	12
3.4. Acceptance and Intermediate Storage . . . . .	12
3.4.1. Resolver Algorithm . . . . .	12
4. DNAME Discussions in Other Documents . . . . .	13
5. Other Issues with DNAME . . . . .	15
5.1. Canonical hostnames cannot be below DNAME owners . . . . .	15
5.2. Dynamic Update and DNAME . . . . .	15
5.3. DNSSEC and DNAME . . . . .	15
5.3.1. Signed DNAME, Unsigned Synthesized CNAME . . . . .	15
5.3.2. DNAME Bit in NSEC Type Map . . . . .	16
5.3.3. DNAME Chains as Strong as the Weakest Link . . . . .	16
5.3.4. Validators Must Understand DNAME . . . . .	16
5.3.4.1. DNAME in Bitmap Causes Invalid Name Error . . . . .	16
5.3.4.2. Valid Name Error Response Involving DNAME in Bitmap . . . . .	17
5.3.4.3. Response With Synthesized CNAME . . . . .	17
6. Examples of DNAME Use in a Zone . . . . .	17
6.1. Organizational Renaming . . . . .	17
6.2. Classless Delegation of Shorter Prefixes . . . . .	18
6.3. Network Renumbering Support . . . . .	18
7. IANA Considerations . . . . .	19
8. Security Considerations . . . . .	19
9. Acknowledgments . . . . .	20
10. References . . . . .	20
10.1. Normative References . . . . .	20
10.2. Informative References . . . . .	21

Appendix A. Changes from RFC 2672 . . . . .	21
A.1. Changes to Server Behavior . . . . .	21
A.2. Changes to Client Behavior . . . . .	22

## 1. Introduction

DNAME is a DNS Resource Record type originally defined in RFC 2672 [RFC2672]. DNAME provides redirection from a part of the DNS name tree to another part of the DNS name tree.

The DNAME RR and the CNAME RR [RFC1034] cause a lookup to (potentially) return data corresponding to a domain name different from the queried domain name. The difference between the two resource records is that the CNAME RR directs the lookup of data at its owner to another single name, a DNAME RR directs lookups for data at descendants of its owner's name to corresponding names under a different (single) node of the tree.

Take for example, looking through a zone (see RFC 1034 [RFC1034], section 4.3.2, step 3) for the domain name "foo.example.com" and a DNAME resource record is found at "example.com" indicating that all queries under "example.com" be directed to "example.net". The lookup process will return to step 1 with the new query name of "foo.example.net". Had the query name been "www.foo.example.com" the new query name would be "www.foo.example.net".

This document is a revision of the original specification of DNAME in RFC 2672 [RFC2672]. DNAME was conceived to help with the problem of maintaining address-to-name mappings in a context of network renumbering. With a careful set-up, a renumbering event in the network causes no change to the authoritative server that has the address-to-name mappings. Examples in practice are classless reverse address space delegations.

Another usage of DNAME lies in aliasing of name spaces. For example, a zone administrator may want sub-trees of the DNS to contain the same information. Examples include punycode [RFC3492] alternates for domain spaces.

This revision of the DNAME specification does not change the wire format or the handling of DNAME Resource Records. Discussion is added on problems that may be encountered when using DNAME.

## 2. The DNAME Resource Record

### 2.1. Format

The DNAME RR has mnemonic DNAME and type code 39 (decimal). It is CLASS-insensitive.

Its RDATA is comprised of a single field, <target>, which contains a fully qualified domain name that MUST be sent in uncompressed form [RFC1035], [RFC3597]. The <target> field MUST be present. The presentation format of <target> is that of a domain name [RFC1035].

```
<owner> <ttl> <class> DNAME <target>
```

The effect of the DNAME RR is the substitution of the record's <target> for its owner name, as a suffix of a domain name. This substitution is to be applied for all names below the owner name of the DNAME RR. This substitution has to be applied for every DNAME RR found in the resolution process, which allows fairly lengthy valid chains of DNAME RRs.

Details of the substitution process, methods to avoid conflicting resource records, and rules for specific corner cases are given in the following subsections.

## 2.2. The DNAME Substitution

When following RFC 1034 [RFC1034], section 4.3.2's algorithm's third step, "start matching down, label by label, in the zone" and a node is found to own a DNAME resource record a DNAME substitution occurs. The name being sought may be the original query name or a name that is the result of a CNAME resource record being followed or a previously encountered DNAME. As in the case when finding a CNAME resource record or NS resource record set, the processing of a DNAME will happen prior to finding the desired domain name.

A DNAME substitution is performed by replacing the suffix labels of the name being sought matching the owner name of the DNAME resource record with the string of labels in the RDATA field. The matching labels end with the root label in all cases. Only whole labels are replaced. See the table of examples for common cases and corner cases.

In the table below, the QNAME refers to the query name. The owner is the DNAME owner domain name, and the target refers to the target of the DNAME record. The result is the resulting name after performing the DNAME substitution on the query name. "no match" means that the query did not match the DNAME and thus no substitution is performed and a possible error message is returned (if no other result is possible). Thus every line contains one example substitution. In the examples below, 'cyc' and 'shortloop' contain loops.

QNAME	owner	DNAME	target	result
com.	example.com.		example.net.	<no match>
example.com.	example.com.		example.net.	[0]
a.example.com.	example.com.		example.net.	a.example.net.
a.b.example.com.	example.com.		example.net.	a.b.example.net.
ab.example.com.	b.example.com.		example.net.	<no match>
foo.example.com.	example.com.		example.net.	foo.example.net.
a.x.example.com.	x.example.com.		example.net.	a.example.net.
a.example.com.	example.com.		y.example.net.	a.y.example.net.
cyc.example.com.	example.com.		example.com.	cyc.example.com.
cyc.example.com.	example.com.		c.example.com.	cyc.c.example.com.
shortloop.x.x.	x.		.	shortloop.x.
shortloop.x.	x.		.	shortloop.

[0] The result depends on the QTYPE. If the QTYPE = DNAME, then the result is "example.com." else "<no match>"

Table 1. DNAME Substitution Examples.

It is possible for DNAMEs to form loops, just as CNAMEs can form loops. DNAMEs and CNAMEs can chain together to form loops. A single corner case DNAME can form a loop. Resolvers and servers should be cautious in devoting resources to a query, but be aware that fairly long chains of DNAMEs may be valid. Zone content administrators should take care to insure that there are no loops that could occur when using DNAME or DNAME/CNAME redirection.

The domain name can get too long during substitution. For example, suppose the target name of the DNAME RR is 250 octets in length (multiple labels), if an incoming QNAME that has a first label over 5 octets in length, the result would be a name over 255 octets. If this occurs the server returns an RCODE of YXDOMAIN [RFC2136]. The DNAME record and its signature (if the zone is signed) are included in the answer as proof for the YXDOMAIN (value 6) RCODE.

### 2.3. DNAME Owner Name Matching the QNAME

Unlike a CNAME RR, a DNAME RR redirects DNS names subordinate to its owner name; the owner name of a DNAME is not redirected itself. The domain name that owns a DNAME record is allowed to have other resource record types at that domain name, except DNAMEs, CNAMEs or other types that have restrictions on what they can co-exist with. When there is a match of the QTYPE to a type (or types) also owned by the owner name the response is sourced from the owner name. E.g., a QTYPE of ANY would return the (available) types at the owner name, not the target name.

DNAME RRs MUST NOT appear at the same owner name as an NS RR unless the owner name is the zone apex as this would constitute data below a zone cut.

If a DNAME record is present at the zone apex, there is still a need to have the customary SOA and NS resource records there as well. Such a DNAME cannot be used to mirror a zone completely, as it does not mirror the zone apex.

These rules also allow DNAME records to be queried through RFC 1034 [RFC1034] compliant, DNAME-unaware caches.

### 2.4. Names Next to and Below a DNAME Record

Resource records MUST NOT exist at any sub-domain of the owner of a DNAME RR. To get the contents for names subordinate to that owner name, the DNAME redirection must be invoked and the resulting target queried. A server MAY refuse to load a zone that has data at a sub-domain of a domain name owning a DNAME RR. If the server does load the zone, those names below the DNAME RR will be occluded as described in RFC 2136 [RFC2136], section 7.18. Also a server ought to refuse to load a zone subordinate to the owner of a DNAME record in the ancestor zone. See Section 5.2 for further discussion related to dynamic update.

DNAME is a singleton type, meaning only one DNAME is allowed per name. The owner name of a DNAME can only have one DNAME RR, and no CNAME RRs can exist at that name. These rules make sure that for a single domain name only one redirection exists, and thus no confusion which one to follow. A server ought to refuse to load a zone that violates these rules.

### 2.5. Compression of the DNAME record.

The DNAME owner name can be compressed like any other owner name. The DNAME RDATA target name MUST NOT be sent out in compressed form

and MUST be downcased for DNSSEC validation.

Although the previous DNAME specification [RFC2672] (that is obsoleted by this specification) talked about signaling to allow compression of the target name, such signaling has never been specified and this document also does not specify this signaling behavior.

RFC 2672 (obsoleted by this document) stated that the EDNS version had a meaning for understanding of DNAME and DNAME target name compression. This document revises RFC 2672, in that there is no EDNS version signaling for DNAME.

### 3. Processing

#### 3.1. CNAME synthesis

When preparing a response, a server performing a DNAME substitution will in all cases include the relevant DNAME RR in the answer section. Relevant cases includes the following:

1. The DNAME is being employed as a substitution instruction.
2. The DNAME itself matches the QTYPE and the owner name matches QNAME.

When the owner name matches the QNAME and the QTYPE matches another type owned there, the DNAME is not included in the answer.

A CNAME RR with TTL equal to the corresponding DNAME RR is synthesized and included in the answer section when the DNAME is employed as a substitution instruction. The owner name of the CNAME is the QNAME of the query. The DNSSEC specification [RFC4033], [RFC4034], [RFC4035] says that the synthesized CNAME does not have to be signed. The signed DNAME has an RRSIG and a validating resolver can check the CNAME against the DNAME record and validate the signature over the DNAME RR.

Servers MUST be able to answer a query for a synthesized CNAME. Like other query types this invokes the DNAME, and then the server synthesizes the CNAME and places it into the answer section. If the server in question is a cache, the synthesized CNAME's TTL SHOULD be equal to the decremented TTL of the cached DNAME.

Resolvers MUST be able to handle a synthesized CNAME TTL of zero or equal to the TTL of the corresponding DNAME record (as some older authoritative server implementations set the TTL of synthesized CNAMEs to zero). A TTL of zero means that the CNAME can be discarded



immediately after processing the answer.

### 3.2. Server algorithm

Below is the server algorithm, which appeared in RFC 2672 Section 4.1.

1. Set or clear the value of recursion available in the response depending on whether the name server is willing to provide recursive service. If recursive service is available and requested via the RD bit in the query, go to step 5, otherwise step 2.
2. Search the available zones for the zone which is the nearest ancestor to QNAME. If such a zone is found, go to step 3, otherwise step 4.
3. Start matching down, label by label, in the zone. The matching process can terminate several ways:
  - A. If the whole of QNAME is matched, we have found the node.

If the data at the node is a CNAME, and QTYPE does not match CNAME, copy the CNAME RR into the answer section of the response, change QNAME to the canonical name in the CNAME RR, and go back to step 1.

Otherwise, copy all RRs which match QTYPE into the answer section and go to step 6.
  - B. If a match would take us out of the authoritative data, we have a referral. This happens when we encounter a node with NS RRs marking cuts along the bottom of a zone.

Copy the NS RRs for the sub-zone into the authority section of the reply. Put whatever addresses are available into the additional section, using glue RRs if the addresses are not available from authoritative data or the cache. Go to step 4.
  - C. If at some label, a match is impossible (i.e., the corresponding label does not exist), look to see whether the last label matched has a DNAME record.

If a DNAME record exists at that point, copy that record into the answer section. If substitution of its <target> for its <owner> in QNAME would overflow the legal size for a <domain-name>, set RCODE to YXDOMAIN [RFC2136] and exit; otherwise perform the substitution and continue. The server MUST synthesize a CNAME record as described above and include it in the answer section. Go back to step 1.

If there was no DNAME record, look to see if the "\*" label exists.

If the "\*" label does not exist, check whether the name we are looking for is the original QNAME in the query or a name we have followed due to a CNAME or DNAME. If the name is original, set an authoritative name error in the response and exit. Otherwise just exit.

If the "\*" label does exist, match RRs at that node against QTYPE. If any match, copy them into the answer section, but set the owner of the RR to be QNAME, and not the node with the "\*" label. If the data at the node with the "\*" label is a CNAME, and QTYPE doesn't match CNAME, copy the CNAME RR into the answer section of the response changing the owner name to the QNAME, change QNAME to the canonical name in the CNAME RR, and go back to step 1. Otherwise, Go to step 6.

4. Start matching down in the cache. If QNAME is found in the cache, copy all RRs attached to it that match QTYPE into the answer section. If QNAME is not found in the cache but a DNAME record is present at an ancestor of QNAME, copy that DNAME record into the answer section. If there was no delegation from authoritative data, look for the best one from the cache, and put it in the authority section. Go to step 6.
5. Use the local resolver or a copy of its algorithm to answer the query. Store the results, including any intermediate CNAMEs and DNAMEs, in the answer section of the response.
6. Using local data only, attempt to add other RRs which may be useful to the additional section of the query. Exit.

Note that there will be at most one ancestor with a DNAME as described in step 4 unless some zone's data is in violation of the no-descendants limitation in section 3. An implementation might take advantage of this limitation by stopping the search of step 3c or

step 4 when a DNAME record is encountered.

### 3.3. Wildcards

The use of DNAME in conjunction with wildcards is discouraged [RFC4592]. Thus records of the form "\*.example.com DNAME example.net" SHOULD NOT be used.

The interaction between the expansion of the wildcard and the redirection of the DNAME is non-deterministic. Because the processing is non-deterministic, DNSSEC validating resolvers may not be able to validate a wildcarded DNAME.

A server MAY give a warning that the behavior is unspecified if such a wildcarded DNAME is loaded. The server MAY refuse it, refuse to load the zone or refuse dynamic updates.

### 3.4. Acceptance and Intermediate Storage

Recursive caching name servers can encounter data at names below the owner name of a DNAME RR, due to a change at the authoritative server where data from before and after the change resides in the cache. This conflict situation is a transitional phase that ends when the old data times out. The caching name server can opt to store both old and new data and treat each as if the other did not exist, or drop the old data, or drop the longer domain name. In any approach, consistency returns after the older data TTL times out.

Recursive caching name servers MUST perform CNAME synthesis on behalf of clients.

If a recursive caching name server encounters a DNSSEC validated DNAME RR which contradicts information already in the cache (excluding CNAME records), it SHOULD cache the DNAME RR, but it MAY cache the CNAME record received along with it, subject to the rules for CNAME. If the DNAME RR cannot be validated via DNSSEC (i.e. not BOGUS, but not able to validate), the recursive caching server SHOULD NOT cache the DNAME RR but MAY cache the CNAME record received along with it, subject to the rules of CNAME.

#### 3.4.1. Resolver Algorithm

A resolver algorithm likewise changes to handle DNAME processing. The complete algorithm becomes:

1. See if the answer is in local information or can be synthesized from a cached DNAME, and if so return it to the client.

2. Find the best servers to ask.
  3. Send queries until one returns a response.
  4. Analyze the response, either:
    - A. If the response answers the question or contains a name error, cache the data as well as returning it back to the client.
    - B. If the response contains a better delegation to other servers, cache the delegation information, and go to step 2.
    - C. If the response shows a CNAME and that is not the answer itself, cache the CNAME, change the SNAME to the canonical name in the CNAME RR and go to step 1.
    - D. If the response shows a DNAME and that is not the answer itself, cache the DNAME (upon successful DNSSEC validation if the client is a validating resolver). If substitution of the DNAME's target name for its owner name in the SNAME would overflow the legal size for a domain name, return an implementation-dependent error to the application; otherwise perform the substitution and go to step 1.
    - E. If the response shows a server failure or other bizarre contents, delete the server from the SLIST and go back to step 3.
4. DNAME Discussions in Other Documents
- In [RFC2181], in Section 10.3., the discussion on MX and NS records touches on redirection by CNAMEs, but this also holds for DNAMEs.

Excerpt from 10.3. MX and NS records (in RFC 2181).

The domain name used as the value of a NS resource record, or part of the value of a MX resource record must not be an alias. Not only is the specification clear on this point, but using an alias in either of these positions neither works as well as might be hoped, nor well fulfills the ambition that may have led to this approach. This domain name must have as its value one or more address records. Currently those will be A records, however in the future other record types giving addressing information may be acceptable. It can also have other RRs, but never a CNAME RR.

The DNAME RR is discussed in RFC 3363, section 4, on A6 and DNAME. The opening premise of this section is demonstrably wrong, and so the conclusion based on that premise is wrong. In particular, [RFC3363] deprecates the use of DNAME in the IPv6 reverse tree, which is then carried forward as a recommendation in [RFC4294]. Based on the experience gained in the meantime, [RFC3363] is revised, dropping all constraints on having DNAME RRs in these zones. This would greatly improve the manageability of the IPv6 reverse tree. These changes are made explicit below.

In [RFC3363], the paragraph

"The issues for DNAME in the reverse mapping tree appears to be closely tied to the need to use fragmented A6 in the main tree: if one is necessary, so is the other, and if one isn't necessary, the other isn't either. Therefore, in moving RFC 2874 to experimental, the intent of this document is that use of DNAME RRs in the reverse tree be deprecated."

is updated by this document and the use of DNAME RRs in the reverse tree is no longer deprecated.

In [RFC4294], the reference to DNAME was left in as an editorial oversight. The paragraph

"Those nodes are NOT RECOMMENDED to support the experimental A6 and DNAME Resource Records [RFC3363]."

is to be replaced by

"Those nodes are NOT RECOMMENDED to support the experimental A6 Resource Record [RFC3363]."

## 5. Other Issues with DNAME

There are several issues to be aware of about the use of DNAME.

### 5.1. Canonical hostnames cannot be below DNAME owners

The names listed as target names of MX, NS, PTR and SRV [RFC2782] records must be canonical hostnames. This means no CNAME or DNAME redirection may be present during DNS lookup of the address records for the host. This is discussed in RFC 2181 [RFC2181], section 10.3, and RFC 1912 [RFC1912], section 2.4. For SRV see RFC 2782 [RFC2782] page 4.

The upshot of this is that although the lookup of a PTR record can involve DNAMEs, the name listed in the PTR record can not fall under a DNAME. The same holds for NS, SRV and MX records. For example, when punycode [RFC3492] alternates for a zone use DNAME then the NS, MX, SRV and PTR records that point to that zone must use names that are not aliases in their RDATA. What must be done then is to have the domain names with DNAME substitution already applied to it as the MX, NS, PTR, SRV data. These are valid canonical hostnames.

### 5.2. Dynamic Update and DNAME

DNAME records can be added, changed and removed in a zone using dynamic update transactions. Adding a DNAME RR to a zone occludes any domain names that may exist under the added DNAME.

If a dynamic update message attempts to add a DNAME with a given owner name but a CNAME is associated with that name, then the server MUST ignore the DNAME. If a DNAME is already associated with that name, then it is replaced with the new DNAME. Otherwise, add the DNAME. If a CNAME is added with a given owner name but a DNAME is associated with that name, then the CNAME MUST be ignored. This is similar behavior for dynamic updates to an owner name of a CNAME RR [RFC2136].

### 5.3. DNSSEC and DNAME

The following subsections specify the behavior of implementations that understand both DNSSEC and DNAME (synthesis).

#### 5.3.1. Signed DNAME, Unsigned Synthesized CNAME

In any response, a signed DNAME RR indicates a non-terminal redirection of the query. There might or might not be a server synthesized CNAME in the answer section; if there is, the CNAME will never be signed. For a DNSSEC validator, verification of the DNAME

RR and then checking that the CNAME was properly synthesized is sufficient proof.

#### 5.3.2. DNAME Bit in NSEC Type Map

In any negative response, the NSEC or NSEC3 [RFC5155] record type bit map SHOULD be checked to see that there was no DNAME that could have been applied. If the DNAME bit in the type bit map is set and the query name is a sub-domain of the closest encloser that is asserted, then DNAME substitution should have been done, but the substitution has not been done as specified.

#### 5.3.3. DNAME Chains as Strong as the Weakest Link

A response can contain a chain of DNAME and CNAME redirections. That chain can end in a positive answer or a negative (no name error or no data error) reply. Each step in that chain results in resource records added to the answer or authority section of the response. Only if all steps are secure can the AD bit be set for the response. If one of the steps is bogus, the result is bogus.

#### 5.3.4. Validators Must Understand DNAME

Below are examples of why DNSSEC validators MUST understand DNAME. In the examples below, SOA records, wildcard denial NSECs and other material not under discussion has been omitted or shortened.

##### 5.3.4.1. DNAME in Bitmap Causes Invalid Name Error

```
;; Header: QR AA RCODE=3(NXDOMAIN)
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096

;; Question
foo.bar.example.com. IN A
;; Authority
bar.example.com. NSEC dub.example.com. A DNAME
bar.example.com. RRSIG NSEC [valid signature]
```

If this is the received response, then only by understanding that the DNAME bit in the NSEC bitmap means that foo.bar.example.com needed to have been redirected by the DNAME, the validator can see that it is a BOGUS reply from an attacker that collated existing records from the DNS to create a confusing reply.

If the DNAME bit had not been set in the NSEC record above then the answer would have validated as a correct name error response.

#### 5.3.4.2. Valid Name Error Response Involving DNAME in Bitmap

```
;; Header: QR AA RCODE=3(NXDOMAIN)
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096

;; Question
cee.example.com. IN A
;; Authority
bar.example.com. NSEC dub.example.com. A DNAME
bar.example.com. RRSIG NSEC [valid signature]
```

This response has the same NSEC records as the example above, but with this query name (cee.example.com), the answer is validated, because 'cee' does not get redirected by the DNAME at 'bar'.

#### 5.3.4.3. Response With Synthesized CNAME

```
;; Header: QR AA RCODE=0(NOERROR)
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096

;; Question
foo.bar.example.com. IN A
;; Answer
bar.example.com. DNAME bar.example.net.
bar.example.com. RRSIG DNAME [valid signature]
foo.bar.example.com. CNAME foo.bar.example.net.
```

The response shown above has the synthesized CNAME included. However, the CNAME has no signature, since the server does not sign online. So this response cannot be trusted. It could be altered by an attacker to be foo.bar.example.com CNAME bla.bla.example. The DNAME record does have its signature included, since it does not change. The validator must verify the DNAME signature and then recursively resolve further to query for the foo.bar.example.net A record.

### 6. Examples of DNAME Use in a Zone

Below are some examples of the use of DNAME in a zone. These examples are by no means exhaustive.

#### 6.1. Organizational Renaming

If an organization with domain name FROBOZZ.EXAMPLE.NET became part of an organization with domain name ACME.EXAMPLE.COM, it might ease transition by placing information such as this in its old zone.



```
frobozz.example.net.  DNAME    frobozz-division.acme.example.com.
                     MX       10      mailhub.acme.example.com.
```

The response to an extended recursive query for `www.frobozz.example.net` would contain, in the answer section, the DNAME record shown above and the relevant RRs for `www.frobozz-division.acme.example.com`.

If an organization wants to have aliases for names, for a different spelling or language, the same example applies. Note that the MX RR at the zone apex is not redirected and has to be repeated in the target zone. Also note that the services at `mailhub` or `www.frobozz-division.acme.example.com` have to recognize and handle the aliases.

## 6.2. Classless Delegation of Shorter Prefixes

The classless scheme for `in-addr.arpa` delegation [RFC2317] can be extended to prefixes shorter than 24 bits by use of the DNAME record. For example, the prefix `192.0.8.0/22` can be delegated by the following records.

```
$ORIGIN 0.192.in-addr.arpa.
8/22    NS      ns.slash-22-holder.example.com.
8       DNAME    8.8/22
9       DNAME    9.8/22
10      DNAME    10.8/22
11      DNAME    11.8/22
```

A typical entry in the resulting reverse zone for some host with address `192.0.9.33` might be

```
$ORIGIN 8/22.0.192.in-addr.arpa.
33.9    PTR     somehost.slash-22-holder.example.com.
```

The same advisory remarks concerning the choice of the "/" character apply here as in [RFC2317] .

## 6.3. Network Renumbering Support

If IPv4 network renumbering were common, maintenance of address space delegation could be simplified by using DNAME records instead of NS records to delegate.

```
$ORIGIN new-style.in-addr.arpa.
189.190      DNAME      in-addr.example.net.

$ORIGIN in-addr.example.net.
188          DNAME      in-addr.customer.example.com.

$ORIGIN in-addr.customer.example.
1            PTR        www.customer.example.com
2            PTR        mailhub.customer.example.com.
; etc ...
```

This would allow the address space 190.189.0.0/16 assigned to the ISP "example.net" to be changed without the necessity of altering the zone data describing the use of that space by the ISP and its customers.

Renumbering IPv4 networks is currently so arduous a task that updating the DNS is only a small part of the labor, so this scheme may have a low value. But it is hoped that in IPv6 the renumbering task will be quite different and the DNAME mechanism may play a useful part.

## 7. IANA Considerations

The DNAME Resource Record type code 39 (decimal) originally has been registered by [RFC2672] in the DNS Resource Record (RR) Types registry table at <http://www.iana.org/assignments/dns-parameters>. IANA should update the DNS resource record registry to point to this document for RR type 39.

## 8. Security Considerations

DNAME redirects queries elsewhere, which may impact security based on policy and the security status of the zone with the DNAME and the redirection zone's security status. For validating resolvers, the lowest security status of the links in the chain of CNAME and DNAME redirections is applied to the result.

If a validating resolver accepts wildcarded DNAMEs, this creates security issues. Since the processing of a wildcarded DNAME is non-deterministic and the CNAME that was substituted by the server has no signature, the resolver may choose a different result than what the server meant, and consequently end up at the wrong destination. Use of wildcarded DNAMEs is discouraged in any case [RFC4592].

A validating resolver MUST understand DNAME, according to [RFC4034]. The examples in Section 5.3.4 illustrate this need.

## 9. Acknowledgments

The authors of this draft would like to acknowledge Matt Larson for beginning this effort to address the issues related to the DNAME RR type. The authors would also like to acknowledge Paul Vixie, Ed Lewis, Mark Andrews, Mike StJohns, Niall O'Reilly, Sam Weiler, Alfred Hoenes and Kevin Darcy for their review and comments on this document.

## 10. References

### 10.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.
- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, March 1998.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S.

Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, July 2006.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

## 10.2. Informative References

[RFC1912] Barr, D., "Common DNS Operational and Configuration Errors", RFC 1912, February 1996.

[RFC2672] Crawford, M., "Non-Terminal DNS Name Redirection", RFC 2672, August 1999.

[RFC3363] Bush, R., Durand, A., Fink, B., Gudmundsson, O., and T. Hain, "Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)", RFC 3363, August 2002.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC4294] Loughney, J., "IPv6 Node Requirements", RFC 4294, April 2006.

## Appendix A. Changes from RFC 2672

### A.1. Changes to Server Behavior

Major changes to server behavior from the original DNAME specification are summarized below:

- o The rules for DNAME substitution have been clarified in Section 2.
- o The EDNS option to signal DNAME understanding and compression has never been specified, and this document clarifies that there is no signaling method (Section 2.5).
- o The TTL for synthesized CNAME RR's is now set to the TTL of the DNAME, not zero (Section 3.1).
- o Caching recursive servers MUST perform CNAME synthesis on behalf of clients (Section 3.4).

- o The revised server algorithm is detailed in Section 3.2.
- o Rules for dynamic update messages adding a DNAME or CNAME RR to a zone where a CNAME or DNAME already exists is detailed in Section 5.2

#### A.2. Changes to Client Behavior

Major changes to client behavior from the original DNAME specification are summarized below:

- o Clients **MUST** be able to accept synthesized CNAME RR's with a TTL of either zero or the TTL of the DNAME RR that accompanies the CNAME RR.
- o DNSSEC aware clients **SHOULD** cache DNAME RR's and **MAY** cache synthesized CNAME RR's it receives in the same response. DNSSEC aware clients **SHOULD** also check the NSEC/NSEC3 type bitmap to verify that DNAME redirection is to be done. DNSSEC validators **MUST** understand DNAME (Section 5.3).
- o The revised client algorithm is detailed in Section 3.4.1.

#### Authors' Addresses

Scott Rose  
NIST  
100 Bureau Dr.  
Gaithersburg, MD 20899  
USA

Phone: +1-301-975-8439  
Fax: +1-301-975-6238  
EMail: scottr.nist@gmail.com

Wouter Wijngaards  
NLnet Labs  
Science Park 140  
Amsterdam 1098 XG  
The Netherlands

Phone: +31-20-888-4551  
EMail: wouter@nlnetlabs.nl

