

Derivation of DNS Name Predecessor and Successor

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes two methods for deriving the canonically-ordered predecessor and successor of a DNS name. These methods may be used for dynamic NSEC resource record synthesis, enabling security-aware name servers to provide authenticated denial of existence without disclosing other owner names in a DNSSEC secured zone.

Table of Contents

1. Introduction	2
2. Notational Conventions	3
3. Derivations	3
3.1. Absolute Method	3
3.1.1. Derivation of DNS Name Predecessor	3
3.1.2. Derivation of DNS Name Successor	4
3.2. Modified Method	4
3.2.1. Derivation of DNS Name Predecessor	5
3.2.2. Derivation of DNS Name Successor	6
4. Notes	6
4.1. Test for Existence	6
4.2. Case Considerations	7
4.3. Choice of Range	7
4.4. Wild Card Considerations	8
4.5. Possible Modifications	8
4.5.1. Restriction of Effective Maximum DNS Name Length	8
4.5.2. Use of Modified Method with Zones Containing	

SRV RRs	8
5. Examples	9
5.1. Examples of Immediate Predecessors Using Absolute Method ..	10
5.2. Examples of Immediate Successors Using Absolute Method	14
5.3. Examples of Predecessors Using Modified Method	19
5.4. Examples of Successors Using Modified Method	20
6. Security Considerations	21
7. Acknowledgements	21
8. References	21
8.1. Normative References	21
8.2. Informative References	22

1. Introduction

One of the proposals for avoiding the exposure of zone information during the deployment DNSSEC is dynamic NSEC resource record (RR) synthesis. This technique is described in [DNSSEC-TRANS] and [RFC4470], and involves the generation of NSEC RRs that just span the query name for non-existent owner names. In order to do this, the DNS names that would occur just prior to and just following a given query name must be calculated in real time, as maintaining a list of all possible owner names that might occur in a zone would be impracticable.

Section 6.1 of [RFC4034] defines canonical DNS name order. This document does not amend or modify this definition. However, the derivation of immediate predecessor and successor, although trivial, is non-obvious. Accordingly, several methods are described here as an aid to implementors and a reference to other interested parties.

This document describes two methods:

1. An "absolute method", which returns the immediate predecessor or successor of a domain name such that no valid DNS name could exist between that DNS name and the predecessor or successor.
2. A "modified method", which returns a predecessor and successor that are more economical in size and computation. This method is restricted to use with zones consisting exclusively of owner names that contain no more than one label more than the owner name of the apex, where the longest possible owner name (i.e., one with a maximum length left-most label) would not exceed the maximum DNS name length. This is, however, the type of zone for which the technique of online signing is most likely to be used.

2. Notational Conventions

The following notational conventions are used in this document for economy of expression:

N: An unspecified DNS name.

P(N): Immediate predecessor to N (absolute method).

S(N): Immediate successor to N (absolute method).

P'(N): Predecessor to N (modified method).

S'(N): Successor to N (modified method).

3. Derivations

These derivations assume that all uppercase US-ASCII letters in N have already been replaced by their corresponding lowercase equivalents. Unless otherwise specified, processing stops after the first step in which a condition is met.

The derivations make reference to maximum label length and maximum DNS name length; these are defined in Section 3.1 of [RFC1034] to be 63 and 255 octets, respectively.

3.1. Absolute Method

3.1.1. Derivation of DNS Name Predecessor

To derive P(N):

1. If N is the same as the owner name of the zone apex, prepend N repeatedly with labels of the maximum length possible consisting of octets of the maximum sort value (e.g., 0xff) until N is the maximum length possible; otherwise proceed to the next step.
2. If the least significant (left-most) label of N consists of a single octet of the minimum sort value (e.g., 0x00), remove that label; otherwise proceed to the next step.
3. If the least significant (right-most) octet in the least significant (left-most) label of N is the minimum sort value, remove the least significant octet and proceed to step 5.
4. Decrement the value of the least significant (right-most) octet of the least significant (left-most) label, skipping any values that correspond to uppercase US-ASCII letters, and then append

the least significant (left-most) label with as many octets as possible of the maximum sort value. Proceed to the next step.

5. Prepend N repeatedly with labels of as long a length as possible consisting of octets of the maximum sort value until N is the maximum length possible.

3.1.2. Derivation of DNS Name Successor

To derive S(N):

1. If N is two or more octets shorter than the maximum DNS name length, prepend N with a label containing a single octet of the minimum sort value (e.g., 0x00); otherwise proceed to the next step.
2. If N is one octet shorter than the maximum DNS name length and the least significant (left-most) label is one or more octets shorter than the maximum label length, append an octet of the minimum sort value to the least significant label; otherwise proceed to the next step.
3. Increment the value of the least significant (right-most) octet in the least significant (left-most) label that is less than the maximum sort value (e.g., 0xff), skipping any values that correspond to uppercase US-ASCII letters, and then remove any octets to the right of that one. If all octets in the label are the maximum sort value, then proceed to the next step.
4. Remove the least significant (left-most) label. Unless N is now the same as the owner name of the zone apex (this will occur only if N was the maximum possible name in canonical DNS name order, and thus has wrapped to the owner name of zone apex), repeat starting at step 2.

3.2. Modified Method

This method is for use with zones consisting only of single-label owner names where an owner name consisting of label of maximum length would not result in a DNS name that exceeded the maximum DNS name length. This method is computationally simpler and returns values that are more economical in size than the absolute method. It differs from the absolute method detailed above in the following ways:

1. Step 1 of the derivation P(N) has been omitted as the existence of the owner name of the zone apex never requires denial.

2. A new step 1 has been introduced that removes unnecessary labels.
3. Step 4 of the derivation $P(N)$ has been omitted as it is only necessary for zones containing owner names consisting of more than one label. This omission generally results in a significant reduction of the length of derived predecessors.
4. Step 1 of the derivation $S(N)$ had been omitted as it is only necessary for zones containing owner names consisting of more than one label. This omission results in a tiny reduction of the length of derived successors, and maintains consistency with the modification of step 4 of the derivation $P(N)$ described above.
5. Steps 2 and 4 of the derivation $S(N)$ have been modified to eliminate checks for maximum DNS name length, as it is an assumption of this method that no DNS name in the zone can exceed the maximum DNS name length.

3.2.1. Derivation of DNS Name Predecessor

To derive $P'(N)$:

1. If N is two or more labels longer than the owner name of the apex, repeatedly remove the least significant (left-most) label until N is only one label longer than the owner name of the apex; otherwise proceed to the next step.
2. If the least significant (left-most) label of N consists of a single octet of the minimum sort value (e.g., 0x00), remove that label; otherwise proceed to the next step. (If this condition is met, $P'(N)$ is the owner name of the apex.)
3. If the least significant (right-most) octet in the least significant (left-most) label of N is the minimum sort value, remove the least significant octet.
4. Decrement the value of the least significant (right-most) octet, skipping any values that correspond to uppercase US-ASCII letters, and then append the label with as many octets as possible of the maximum sort value.

3.2.2. Derivation of DNS Name Successor

To derive $S'(N)$:

1. If N is two or more labels longer than the owner name of the apex, repeatedly remove the least significant (left-most) label until N is only one label longer than the owner name of the apex. Proceed to the next step.
2. If the least significant (left-most) label of N is one or more octets shorter than the maximum label length, append an octet of the minimum sort value to the least significant label; otherwise proceed to the next step.
3. Increment the value of the least significant (right-most) octet in the least significant (left-most) label that is less than the maximum sort value (e.g., 0xff), skipping any values that correspond to uppercase US-ASCII letters, and then remove any octets to the right of that one. If all octets in the label are the maximum sort value, then proceed to the next step.
4. Remove the least significant (left-most) label. (This will occur only if the least significant label is the maximum label length and consists entirely of octets of the maximum sort value, and thus has wrapped to the owner name of the zone apex.)

4. Notes

4.1. Test for Existence

Before using the result of $P(N)$ or $P'(N)$ as the owner name of an NSEC RR in a DNS response, a name server should test to see whether the name exists. If it does, either a standard non-synthesised NSEC RR should be used, or the synthesised NSEC RR should reflect the RRset types that exist at the NSEC RR's owner name in the Type Bit Map field as specified by Section 4.1.2 of [RFC4034]. Implementors will likely find it simpler to use a non-synthesised NSEC RR. For further details, see Section 2 of [RFC4470].

4.2. Case Considerations

Section 3.5 of [RFC1034] specifies that "while upper and lower case letters are allowed in names, no significance is attached to the case". Additionally, Section 6.1 of [RFC4034] states that when determining canonical DNS name order, "uppercase US-ASCII letters are treated as if they were lowercase US-ASCII letters". Consequently, values corresponding to US-ASCII uppercase letters must be skipped when decrementing and incrementing octets in the derivations described in Section 3.

The following pseudo-code is illustrative:

Decrement the value of an octet:

```
if (octet == '[')          // '[' is just after uppercase 'Z'
    octet = '@';          // '@' is just prior to uppercase 'A'
else
    octet--;
```

Increment the value of an octet:

```
if (octet == '@')          // '@' is just prior to uppercase 'A'
    octet = '[';          // '[' is just after uppercase 'Z'
else
    octet++;
```

4.3. Choice of Range

[RFC2181] makes the clarification that "any binary string whatever can be used as the label of any resource record". Consequently, the minimum sort value may be set as 0x00 and the maximum sort value as 0xff, and the range of possible values will be any DNS name that contains octets of any value other than those corresponding to uppercase US-ASCII letters.

However, if all owner names in a zone are in the letter-digit-hyphen, or LDH, format specified in [RFC1034], it may be desirable to restrict the range of possible values to DNS names containing only LDH values. This has the effect of

1. making the output of tools such as 'dig' and 'nslookup' less subject to confusion,
2. minimising the impact that NSEC RRs containing DNS names with non-LDH values (or non-printable values) might have on faulty DNS resolver implementations, and

3. preventing the possibility of results that are wildcard DNS names (see Section 4.4).

This may be accomplished by using a minimum sort value of 0x1f (US-ASCII character '-') and a maximum sort value of 0x7a (US-ASCII character lowercase 'z'), and then skipping non-LDH, non-lowercase values when incrementing or decrementing octets.

4.4. Wild Card Considerations

Neither derivation avoids the possibility that the result may be a DNS name containing a wildcard label, i.e., a label containing a single octet with the value 0x2a (US-ASCII character '*'). With additional tests, wildcard DNS names may be explicitly avoided; alternatively, if the range of octet values can be restricted to those corresponding to letter-digit-hyphen, or LDH, characters (see Section 4.3), such DNS names will not occur.

Note that it is improbable that a result that is a wildcard DNS name will occur unintentionally; even if one does occur either as the owner name of, or in the RDATA of an NSEC RR, it is treated as a literal DNS name with no special meaning.

4.5. Possible Modifications

4.5.1. Restriction of Effective Maximum DNS Name Length

[RFC1034] specifies that "the total number of octets that represent a name (i.e., the sum of all label octets and label lengths) is limited to 255", including the null (zero-length) label that represents the root. For the purpose of deriving predecessors and successors during NSEC RR synthesis, the maximum DNS name length may be effectively restricted to the length of the longest DNS name in the zone. This will minimise the size of responses containing synthesised NSEC RRs but, especially in the case of the modified method, may result in some additional computational complexity.

Note that this modification will have the effect of revealing information about the longest name in the zone. Moreover, when the contents of the zone changes, e.g., during dynamic updates and zone transfers, care must be taken to ensure that the effective maximum DNS name length agrees with the new contents.

4.5.2. Use of Modified Method with Zones Containing SRV RRs

Normally, the modified method cannot be used in zones that contain Service Record (SRV) RRs [RFC2782], as SRV RRs have owner names that contain multiple labels. However, the use of SRV RRs can be

accommodated by various techniques. There are at least four possible ways to do this:

1. Use conventional NSEC RRs for the region of the zone that contains first-level labels beginning with the underscore ('_') character. For the purposes of generating these NSEC RRs, the existence of (possibly fictional) ownernames '9{63}' and 'a' could be assumed, providing a lower and upper bound for this region. Then all queries where the QNAME does not exist but contains a first-level label beginning with an underscore could be handled using the normal DNSSEC protocol.

This approach would make it possible to enumerate all DNS names in the zone containing a first-level label beginning with underscore, including all SRV RRs, but this may be of less a concern to the zone administrator than incurring the overhead of the absolute method or of the following variants of the modified method.

2. The absolute method could be used for synthesising NSEC RRs for all queries where the QNAME contains a leading underscore. However, this re-introduces the susceptibility of the absolute method to denial of service activity, as an attacker could send queries for an effectively inexhaustible supply of domain names beginning with a leading underscore.
3. A variant of the modified method could be used for synthesising NSEC RRs for all queries where the QNAME contains a leading underscore. This variant would assume that all predecessors and successors to queries where the QNAME contains a leading underscore may consist of two labels rather than only one. This introduces a little additional complexity without incurring the full increase in response size and computational complexity as the absolute method.
4. Finally, a variant of the modified method that assumes that all owner names in the zone consist of one or two labels could be used. However, this negates much of the reduction in response size of the modified method and may be nearly as computationally complex as the absolute method.

5. Examples

In the following examples,

the owner name of the zone apex is "example.com.",

the range of octet values is 0x00 - 0xff excluding values corresponding to uppercase US-ASCII letters, and

non-printable octet values are expressed as three-digit decimal numbers preceded by a backslash (as specified in Section 5.1 of [RFC1035]).

5.1. Examples of Immediate Predecessors Using Absolute Method

Example of a typical case:

$$P(\text{foo.example.com.}) =$$
[illegible]

or, in alternate notation:

\255{49}.\255{63}.\255{63}.fon\255{60}.example.com.

where $\{n\}$ represents the number of repetitions of an octet.

Example where least significant (left-most) label of DNS name consists of a single octet of the minimum sort value:

$$P(\backslash 000.foo.example.com.) = foo.example.com.$$

Example where least significant (right-most) octet of least significant (left-most) label has the minimum sort value:

$$P(\text{foo}\backslash000.\text{example.com.}) =$$
[illegible]

or, in alternate notation:

`\255{45}.\255{63}.\255{63}.\255{63}.foo.example.com.`

Example where DNS name contains an octet that must be decremented by skipping values corresponding to US-ASCII uppercase letters:

$$P(\text{fo}\backslash[.example.com.]) =$$
[illegible]

or, in alternate notation:

\255{49}.\255{63}.\255{63}.fo\@\255{60}.example.com.

where $\{n\}$ represents the number of repetitions of an octet.

Example where DNS name is the owner name of the zone apex, and consequently wraps to the DNS name with the maximum possible sort order in the zone:

$$P(\text{example.com.}) =$$
[illegible]

or, in alternate notation:

`\255{49}.\255{63}.\255{63}.\255{63}.example.com.`

5.2. Examples of Immediate Successors Using Absolute Method

Example of typical case:

$S(\text{foo.example.com.}) = \backslash 000.\text{foo.example.com.}$

Example where DNS name is one octet short of the maximum DNS name length:

$N =$ fooo
 .oo
 oooooooooooooooooooooo.oooooooooooooooooooooooooooooooooooo
 oo.oooooooooooooo
 ooo.example.com.

or, in alternate notation:

$\text{fo}\{47\}.\text{o}\{63\}.\text{o}\{63\}.\text{o}\{63\}.\text{example.com.}$

$S(N) =$

fooo
 $\backslash 000.$ oo
 oooooooooooooooooooooo.oooooooooooooooooooooooooooooooooooo
 oo.oooooooooooo
 ooo
 oooo.example.com.

or, in alternate notation:

$\text{fo}\{47\}\backslash 000.\text{o}\{63\}.\text{o}\{63\}.\text{o}\{63\}.\text{example.com.}$

Example where DNS name is the maximum DNS name length:

```
N = fooooooooooooooooooooooooooooooooooooooooooooooooooooo
  o.oooooooooooooooooooooooooooooooooooooooooooooooooooo
  oooooooooooooooooooooo.ooooooooooooooooooooooooooooooo
  ooooooooooooooooooooooooooooooooooooooooooooooooooooo
  ooooooooooooooooooooooooooooooooooooooooooooooooooooo
  o.example.com.
```

or, in alternate notation:

```
fo{48}.o{63}.o{63}.o{63}.example.com.
```

S(N) =

```
fooooooooooooooooooooooooooooooooooooooooooooooooooooo
p.oooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooo.oooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooo
o.example.com.
```

or, in alternate notation:

```
fo{47}p.o{63}.o{63}.o{63}.example.com.
```

Example where DNS name is the maximum DNS name length and the least significant (left-most) label has the maximum sort value:

```
N =  \255\255\255\255\255\255\255\255\255\255\255\255
      \255\255\255\255\255\255\255\255\255\255\255\255
      \255\255\255\255\255\255\255\255\255\255\255\255
      \255\255\255\255\255\255\255\255\255\255\255\255
      \255.oooooooooooooooooooooooooooooooooooooooooooo
      oooooooooooooooooooooooooooooooooooooooooooooooooo
      oooooooooooooooooooooooooooooooooooooooooooooooooo
      oooooooooooooooooooooooooooooooooooooooooooooooooo
      oooo.example.com.
```

or, in alternate notation:

```
\255{49}.o{63}.o{63}.o{63}.example.com.
```

S(N) =

```
oooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooop.oooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooo.
example.com.
```

or, in alternate notation:

```
o{62}p.o{63}.o{63}.example.com.
```


Example where DNS name is the maximum DNS name length and the eight least significant (right-most) octets of the least significant (left-most) label have the maximum sort value:

```
N = foooooooooooooooooooooooooooooooooooooooooooooooooooo\255
    \255\255\255\255\255\255\255.oooooooooooooooooooo
    ooooooooooooooooooooooooooooooooooooooooooooooooooooo.ooo
    ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooo.oooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooo.example.com.
```

or, in alternate notation:

```
fo{40}\255{8}.o{63}.o{63}.o{63}.example.com.
```

S(N) =

```
fooooooooooooooooooooooooooooooooooooooooooooooooooooop.oooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooo.oooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooo.oooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooo.example.com.
```

or, in alternate notation:

```
fo{39}p.o{63}.o{63}.o{63}.example.com.
```

Example where DNS name is the maximum DNS name length and contains an octet that must be incremented by skipping values corresponding to US-ASCII uppercase letters:

```
N = foooooooooooooooooooooooooooooooooooooooooooooooooooo
    \@.oooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooo.oooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oo.example.com.
```

or, in alternate notation:

```
fo{47}\@.o{63}.o{63}.o{63}.example.com.
```

S(N) =

```
foooooooooooooooooooooooooooooooooooooooooooooooooooo
    \[.oooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooo.oooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oo.example.com.
```

or, in alternate notation:

```
fo{47}\[.o{63}.o{63}.o{63}.example.com.
```

Example where DNS name has the maximum possible sort order in the zone, and consequently wraps to the owner name of the zone apex:

[illegible]

or, in alternate notation:

`\255{49}.\255{63}.\255{63}.\255{63}.example.com.`

$S(N) = \text{example.com.}$

5.3. Examples of Predecessors Using Modified Method

Example of a typical case:

$$P'(\text{foo.example.com.}) =$$
[illegible]

or, in alternate notation:

fon\255{60}.example.com.

Example where DNS name contains more labels than DNS names in the zone:

```
P'(bar.foo.example.com.) = foo.example.com.
```

Example where least significant (right-most) octet of least significant (left-most) label has the minimum sort value:

```
P'(foo\000.example.com.) = foo.example.com.
```

Example where least significant (left-most) label has the minimum sort value:

```
P'(\000.example.com.) = example.com.
```

Example where DNS name is the owner name of the zone apex, and consequently wraps to the DNS name with the maximum possible sort order in the zone:

```
P'(example.com.) =
    \255\255\255\255\255\255\255\255\255\255\255\255
    \255\255\255\255\255\255\255\255\255\255\255\255
    \255\255\255\255\255\255\255\255\255\255\255\255
    \255\255\255\255\255\255\255\255\255\255\255\255
    \255\255\255\255\255\255\255\255\255\255\255\255
    \255\255\255.example.com.
```

or, in alternate notation:

```
\255{63}.example.com.
```

5.4. Examples of Successors Using Modified Method

Example of a typical case:

```
S'(foo.example.com.) = foo\000.example.com.
```

Example where DNS name contains more labels than DNS names in the zone:

```
S'(bar.foo.example.com.) = foo\000.example.com.
```

Example where least significant (left-most) label has the maximum sort value, and consequently wraps to the owner name of the zone apex:

[illegible]

or, in alternate notation:

`\255{63}.example.com.`

$S'(N) = \text{example.com.}$

6. Security Considerations

The derivation of some predecessors/successors requires the testing of more conditions than others. Consequently, the effectiveness of a denial-of-service attack may be enhanced by sending queries that require more conditions to be tested. The modified method involves the testing of fewer conditions than the absolute method and consequently is somewhat less susceptible to this exposure.

7. Acknowledgements

The authors would like to thank Sam Weiler, Olaf Kolkman, Olafur Gudmundsson, and Niall O'Reilly for their review and input.

8. References

8.1. 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.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS
 Specification", RFC 2181, July 1997.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR
 for specifying the location of services (DNS SRV)",
 RFC 2782, February 2000.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and
 S. Rose, "Resource Records for the DNS Security
 Extensions", RFC 4034, March 2005.
```

## 8.2. Informative References

- [RFC4470] Weiler, S. and J. Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing", RFC 4470, April 2006.
- [DNSSEC-TRANS] Arends, R., Koch, P., and J. Schlyter, "Evaluating DNSSEC Transition Mechanisms", Work in Progress, February 2005.

### Authors' Addresses

Geoffrey Sisson  
Nominet  
Sandford Gate  
Sandy Lane West  
Oxford  
OX4 6LB  
GB

Phone: +44 1865 332211  
EMail: geoff@nominet.org.uk

Ben Laurie  
Nominet  
17 Perryn Road  
London  
W3 7LR  
GB

Phone: +44 20 8735 0686  
EMail: ben@algroup.co.uk

## Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

