Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 2, 2012

J. Gersch
Secure64 SW Corp
D. Massey
Colorado State University
E. Osterweil
Verisign
May 1, 2012

Reverse DNS Naming Convention for CIDR Address Blocks
draft-gersch-dnsop-revdns-cidr-02.txt

Abstract

The reverse DNS naming method is used to specify a complete IP address. At present there is no standard way for the reverse DNS to handle address ranges. As an example, there is no formal mechanism to define a reverse DNS name for the block of addresses specified by the IPv4 prefix 129.82.0.0/16. Defining such a reverse DNS naming convention would be useful for a number of applications. This draft proposes a naming convention for encoding CIDR address blocks into the reverse DNS namespace.

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 November 2, 2012.

Copyright Notice

Table of Contents

1. Introduction

This draft proposes a common naming convention for entering CIDR prefixes into the Reverse DNS.

The Reverse DNS provides a naming convention for both IPv4 and IPv6 addresses. At this time, the most common use of the reverse-DNS is to associate an IP address with a PTR resource record that identifies the corresponding host name. For example, IP address 129.82.138.2 is encoded as 2.138.82.129.in-addr.arpa and a PTR resource record identifies the host name as alpha.netsec.colostate.edu. The Reverse DNS would be more expressive if we had a formal convention for encoding and returning information associated with a network address range, not just a unique IP address. For example, one would like to store and resolve resource records associated with a prefix range such as 129.82.128/17.

Given such a capability, a variety of new applications and services would be enabled. For example, one might store geographic locations associated with a prefix, prefix ownership information, and a variety of other data. This list of possible applications is not intended to be complete, but instead suggest some of the possibilities. This draft proposes a naming convention for the prefix name and argues that a process for naming a prefix should be consistent across applications.

1.1. Aligning the DNS and IP Hierarchies

A key observation is that both the DNS names and IP addresses are part of a hierarchical tree structure and any naming convention should respect and align with these tree structures.

In the DNS hierarchical tree structure 128.82.129.in-addr.apra is logically below 82.129.in-addr.apra, which is logically below 129.in-addr.arpa. Other "flat" approaches to naming, such as Distributed Hash Tables, have been proposed, but the DNS tree structure remains a

powerful abstraction. It forms the basis for the operation of DNS; caching, delegation, DNSSEC signing, and so forth all benefit from the DNS tree structure.

IP addresses also have a logical tree structure where 129.82.128.0/24 is subprefix (logically below) 129.82.0.0/16 which is a subprefix of 129.0.0.0/8. The reverse DNS aligns with the structure; 128.82.129.in-addr.arpa is logically below 82.129.in-addr.arpa which is logically below 129.in-addr.arpa. This alignment between the DNS hierarchy and the IP address hierarchy serves both systems well and allows one to easily encode prefixes that fall on an octet boundary (e.g. IPv4 prefixes whose mask length is a multiple of 8).

The challenge is to preserve this alignment even when even when CIDR prefixes do not fall on octet boundaries. For example, 129.82.128.0/19 is a subprefix of 129.82.128.0/18. The DNS name for 129.82.128.0/19 should be logically below the DNS name for 129.82.128.0/18. This document introduces a naming convention for CIDR prefixes that preserves this alignment.

1.2. Purpose

In order to enable these applications, one must map an IPv4 or IPv6 prefix into a reverse-DNS name. There are various subtleties, advantages and disadvantages that emerge when trying to define a naming convention. Today, zone administrators can use their own individual approaches to encode a prefix in the reverse DNS. This requires no DNS protocol changes and no modifications to resolvers, caches, or authoritative servers. The emergence of different encoding standards complicates (but does not prevent) the design of systems that would make use of these resource records. The aim of this work is to introduce a standard convention.

1.3. Terminology

The following terms are used throughout out the document:

Reverse DNS:
: We use the term Reverse DNS to refer to the domains in-addr.arpa and ip6.arpa.

Prefix:

A prefix refers to IPv4 or IPv6 address range specified by a network portion and mask length, as described in [RFC4632]. For example, 129.82.0.0/16 and 129.82.128/18 are examples of IPv4 prefixes.

Octet Boundary:
An IPv4 prefix falls on an octet boundary if its mask length is a multiple 8. For example, 129.82.0.0/16 is on an octet boundary while 129.82.128/18 does not fall on octet boundary. Prefixes that are on octet boundary naturally map to the reverse DNS. Prefixes that are not on octet boundary are more complex and the main challenge for any naming convention.

2. Conventions Used In This Document

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 [RFC2119].

3. Design Requirements

A naming convention to specify CIDR address blocks in the reverse-DNS has several design goals:

1. Autonomy: The owner of a reverse-DNS zone file associated with a CIDR address block should be able to act independently from any other organization in order to create or modify data records within the DNS zone.

2. Coverage Authority: With the exception of data that has been sub-delegated to a child zone, the reverse DNS zone must be authoritative for all sub-prefixes below the covering prefix. Any query for a sub-prefix must be answered with a data record or NXDOMAIN specifying this zone as the authority.

3. Allow Delegation: It must allow the zone owner to delegate smaller address blocks to a child zone which will be independently managed.

4. Conformance: It should align with naming conventions and delegation structures already in use by the RIR's for IN-ADDR.ARPA and IP6.ARPA.

5. Simplicity: The naming structure should be understandable, or at a minimum, able to be easily constructed by software provisioning tools and utilities such as DIG.

4. Related Work

The process of mapping CIDR addresses into the reverse-DNS name space is difficult because the prefix length of an IPv4 CIDR address is an arbitrary number from 0 to 32. These numbers do not necessarily align with an IPv4 octet.

4.1. Naming via RFC 1101

The problem of associating records with network names dates back to at least [RFC1101]. This work coincides with some of the early development of DNS and discusses issues regarding hosts.txt files. The RFC observation makes a key observation that one should provide "mappings for subnets, even when nested".

The approach taken here clearly states how to map an IPv4 prefix that is on an octet boundary. The RFC maps "10.IN-ADDR.ARPA for class A net 10, 2.128.IN-ADDR.ARPA for class B net 128.2, etc." This is essentially the same as the approach proposed here, although we append an "m" label (discussed later in this document).

[RFC1101] also mentions more specific subnets, but the details are limited. We believe the approach proposed here builds on the best ideas from this RFC and expands on the details of how the naming convention would work.

4.2. CIDR Naming via RFC 2317

Since CIDR address no longer align with octet boundaries, the CIDR specification in [RFC4632] notes that there is "some increase in work for those who maintain parts of the IN-ADDR.ARPA zone." [RFC2317] is offered as a technique to populate IN-ADDR.ARPA. The intent of this work is to encode IPv4 addresses and the approach is designed to "address spaces covering fewer than 256 addresses."

Suppose organization A owns 129.82.138.0/30. This address space covers four IPv4 addresses; namely 129.82.138.0, 129.82.138.1, 129.82.138.2 and 129.82.138.3. Giving organization A control of the reverse zone "138.82.129.in-addr.arpa." would allow Organization A to enter PTR resource records for each of its 4 addresses. However, it also gives organization A the ability to enter PTR resource records for 252 other IP addresses from 129.82.138.4 to 129.82.138.255. These addresses are managed by other organizations. Sharing the 138.82.129.in-addr.arpa between multiple organization is not practical and creating a separate zone for each IP address (e.g. creating the zone 0.138.82.129.in-addr.arpa) is very high overhead to store a single PTR record.

[RFC2317] addresses this problem by creating CNAME records in

138.82.129.in-addr.arpa zone. Organization A administers a zone named 0/32.138.129.in-addr.arpa. CNAME records in the 138.82.129.in-addr.arpa zone point to entries in Organization A's 0/32.138.82.129.in-addr.arpa zone. For example, 1.138.82.129.in-addr.arpa. is a CNAME pointing to 1.0/32.138.82.129.in-addr.arpa. A full description is found in [RFC2317].

This approach was not intended to encode IP address for address spaces smaller than a "/24". It was not intended for encoding prefixes. It does not specify how one might encode a prefix and it is not trivial to extend this approach to CIDR prefixes. In particular, the design requirements of Coverage Authority, Allowing Delegation, and arguably Simplicity are not easily met by extending the RFC to included prefixes.

4.3. Prior Work on CIDR Names for Routing

Over a decade ago, [I-D.bates-bgp4-nlri-orig-verif] proposed to use the reverse DNS to verify the origin AS associated with a prefix. This requires both a naming convention for converting the name into a prefix and additional resource record types for storing origin information, along with recommendations on their use.

Our focus in this draft is on the naming convention. Draft [I-D.bates-bgp4-nlri-orig-verif] as well as other subsequent work on BGP security, extends [RFC2317] style names to encode a prefix. For example, the draft proposes to encode the prefix 10.1.128/20 as the DNS name 128/20.1.10.bgp.in-addr.arpa.

In [I-D.bates-bgp4-nlri-orig-verif], the DNS hierarchy and the IP address hierarchy diverge and the approach fails to meet the Coverage Authority requirement. To see this, consider the prefixes 10.1.128/20 and 10.1.128/21. in CIDR terminology, 10.1.128/21 is covered by 10.1.128/20, but this relationship is not captured in the DNS hierarchy. 10.1.128/21 is encoded as 128/21.1.10.bgp.in-addr.arpa and thus 10.1.128/20 and 10.1.128/21 are siblings in the DNS tree structure.

This can be overcome by introducing a large number of CNAME records; one for every potential subprefix. We instead provide an approach where the CIDR hierarchy and DNS hierarchy align.

5. Reverse DNS CIDR Name Specification

The naming method described in this section is based on the well-known technique of ANDing a bit-mask with the low-order octet of an IP address. The binary result is then broken up into individual sub-names using the "." separator. The result looks like an ENUM or IPv6 reverse-DNS address; that is, a string of chained empty non-terminal sub-names.

This name-chaining creates the desired effect of being able to allow a DNS zone delegation at any point in the chain. The naming scheme allows the creation of two /17's from a /16, two /18's from a /17, and so on.

5.1. IPv4 Address Block Naming

The CIDR to Reverse-DNS naming convention works as follows:

1. Remove any octets that are not significant. An octet is significant if it includes any part of the network address. An octet is not significant if all bits correspond to the host portion of the address. For example, 129.82.0.0/16 --> 129.82 and 129.82.160.0/19 --> 129.82.160

2. Calculate N where N = prefix_length mod 8. If N equals 0, invert the address and insert a "m" label as the first label to indicate this a prefix name and append in-addr.arpa to the end, per the usual reverse-DNS method; 129.82 --> m.82.129.in-addr.arpa.

3. If N is not equal 0, the prefix is not on an octet boundary and we perform the following name construction:

   A. Truncate the name to remove the least significant octet. Add a "m" label to this domain name to indicate "mask".

   B. Convert the least significant octet to binary, separating each bit into its own label (with a "." character).

   C. Truncate the binary labels to the N significant labels that correspond to the given prefix_length.

   D. Reverse the string and add ".in-addr.arpa."

Several examples illustrate this algorithm. These examples show the conversion to binary, followed by the truncation, followed by the name reversal.

```
129.82.0.0/16   --> m.82.129.in-addr.arpa. (at octet boundary)
```



```
129.82.64.0/18      --> 129.82.m.0.1.0.0.0.0.0.0
                --> 129.82.m.0.1 (N = 18 mod 8 = 2)
                --> 1.0.m.82.129.in-addr.arpa.

129.82.64.0/20  --> 129.82.m.0.1.0.0.0.0.0.0
                    --> 129.82.m.0.1.0.0  (N = 20 mod 8 = 4)
                --> 0.0.1.0.m.82.129.in-addr.arpa.

    129.82.160.0/20 --> 129.82.m.1.0.1.0.0.0.0.0
                    --> 129.82.m.1.0.1.0 (N = 20 mod 8 = 4)
                    --> 0.1.0.1.m.82.129.in-addr.arpa.

    129.82.160.0/23 --> 129.82.m.1.0.1.0.0.0.0.0
                    --> 129.82.m.1.0.1.0.0.0.0 (N = 23 mod 8 = 7)
                    --> 0.0.0.0.1.0.1.m.82.129.in-addr.arpa.

    15.192.0.0/12   --> 15.192.m.1.1.0.0.0.0.0.0
                    --> 15.192.m.1.1.0.0    (N = 12 mod 8 = 4)
                    --> 0.0.1.1.m.15.in-addr.arpa.
```

The conversion from a reverse-DNS name back to CIDR is simple. First calculate the prefix length from the name using the formula:

```
plen = 8*(count of full octets) + (count of binary digits)
```

Then reverse the string, add up the values of the binary digits to build a final octet, then append a "/" and the prefix length.

Examples:

```
1.0.m.82.129.in-addr.arpa  --> 129.82.64.0/18
(example has 2 octets + 2 binary digits, so mask length = 18)

0.0.1.0.m.82.129.in-addr.arpa --> 129.82.64.0/20
```

(example has 2 octets + 4 binary digits, so mask length = 20)

0.0.0.1.0.1.m.129.in-addr.arpa--> 129.160.0/14
(example has 1 octet + 6 binary digits, so mask length = 14)

5.2. IPv6 Address Block Naming

The IPv6 naming convention is similar, with the exception that 4-bit nibble boundaries are used instead of octets, the mod calculation is based on 4 instead of 8, and "ip6.arpa" is used as the suffix.

Examples:

2607:fa88::/32 --> m.8.8.a.f.7.0.6.2.ip6.arpa
(on nibble boundary)

2607:fa88:8000::/33 --> 2.6.0.7.f.a.8.8.m.1.0.0.0
--> 2.6.0.7.d.a.8.8.m.1 (33 mod 4 = 1)
--> 1.m.8.8.a.f.7.0.6.2.ip6.arpa

2607:fa88:e000::/35 --> 2.6.0.7.f.a.8.8.m.1.1.1.0
--> 2.6.0.7.d.a.8.8.m.1.1.1(35 mod 4 = 3)
--> 1.1.1.m.8.8.a.f.7.0.6.2.ip6.arpa

6. Additional Considerations

This draft proposes a naming convention for IPv4 and IPv6 prefixes. With the introduction of a such a convention, a number of new possibilities are enabled and a number of issues have been raised. In this section, we summarize some of the main discussions. Though these are not directly part of the naming convention, they do help to review issues that may help application designers make better use of prefix names and help operators manage reverse zones. We first discuss how the naming convention interacts with the current octet (IPv4) or nibble (IPv6) based reverse DNS tree structure and then turn to the problem of prefix enumeration and find the longest match for a prefix.

6.1. Legacy Behavior at Octet Boundaries

The existing reverse DNS structure is aligned on octet boundaries for IPv4 and nibble boundaries for IPv6. The naming convention introduced here adds to the existing reverse DNS tree; it does not change the existing structure. This is a deliberate choice not to reinvent the reverse DNS but rather to enhance the existing structure. The naming convention proposed here builds on the existing reverse DNS structure and thus inherits both advantages and disadvantages from the existing system.

One disadvantage is the existing octet boundary based tree for IPv4 (and nibble based tree for IPv6). To understand why this is a disadvantage, suppose organization A owns the address space 129.82.0.0/16. Organization A allocates the address space 129.82.128.0/17 to Organization B. In typical operations, Organization A would delegate 128 zones to Organization B; the zones 128.82.129.in-addr.arpa, 129.82.129.in-addr.arpa, ..., 255.82.129.in-addr.arpa. Because the reverse DNS had no notion of CIDR prefixes, all 128 delegations are need to give organization B full control over its PTR records.

The example becomes worse if Organization B further suballocates 129.82.128/22 to Organization C. In this case, Organization C needs to be given operational control of 4 zones; 128.82.129.in-addr.arpa, 129.82.129.in-addr.arpa. 130.82.129.in-addr.arpa, and 131.82.129.in-addr.arpa. Delegating these zones to organization C requires an action by the owner of 82.129.in-addr.arpa. Note that Organization does not have a business relationship with Organization A. Organization C needs to pass information to Organization B who in turn needs to pass the information to Organization A.

The above operation is not ideal. Delegating a non-octet boundary prefix requires the delegation of multiple zones. Subdelegation can

require communication with organizations that do not have direct business relationships. But it is essential to note that this is how the reverse DNS currently operates and has successfully operated for many years. Operational techniques have been developed to manage PTR records and their respective zones. For better or worse, these practices continue to work with this naming convention.

The naming convention here does introduce additional names for prefixes. In the example above, Organization A could delegate the 1.m.82.129.in-addr.apra to Organization B. Organization B could in turn delegate 0.0.0.0.0.1.m.82.129.in-addr.arpa to Organization C. Note the naming convention has allowed delegation of prefixes to work in an efficient manner that respects business relationships. For example, Organization B can delegate the prefix 129.82.128/22 to Organization C without ever involving Organization A. Had this naming convention been in place for the original reverse DNS, much of the suboptimal behavior discussed above could have been avoided. However, the naming convention explicitly chooses to enhance the

existing reverse DNS tree rather than replace.

We note that the naming convention uses the letter "m" to indicate a transition from octet/nibble numbering to binary numbering for the remainder of the name. Nothing restricts a DNS administrator from creating a name in which the sequence of binary digits extends past the next octet or nibble boundary. Applications may actually find this to be a useful capability. Nevertheless, this document defines a naming convention where each prefix maps to a unique name, as described in section 5. We therefore add the restriction that any application looking for records associated with a prefix MUST check standard naming convention (e.g. m.0.82.129.in-addr.arpa at an octet boundary) and if the desired records are found, the application MUST prefer these records over any records found at a non-standard encoding.

6.2. The Naming Convention and Zone Structures

The naming convention does not impose any semantics on zone structure. As with any DNS name, a resolver need not be aware of how the zone cuts are structured and no specific requirements are added for zone management. For example, some sites may choose to delegate at a subprefix boundary while others maintain one large zone. Names can make use of CNAME and DNAME records if the zone administrator so desires. This is simply a naming convention and does not change any existing resolver or server behavior.

6.3. Separation of Prefix Data and PTR Records

Some organization may want to separate the administration of prefix related data (geolocations, prefix ownership, and so forth) from the management of traditional PTR records. Note that all prefix related data is stored at a name that includes the "m" label. This "m" label could be used as delegation point to separate the administration of prefix data from the administration of PTR records.

To illustrate this, suppose the owner of 129.82.128/17 would like one to keep the management of prefix related data distinct from the

management of their PTR records. Note that for all prefixes with a mask length between 17 and 23 are part of the zone 1.m.82.129.in-addr.arpa. This zone can simply be delegated to the group managing prefix related data while the group managing PTR records continues to be responsible for the zones 128.82.129.in-addr.arpa to 255.82.129.in-addr.arpa.

If prefix data is also to be stored at mask lengths ranging between 24 and 32, then m.128.82.129.in-addr.arpa to m.255.82.129.in-addr.arpa can also be delegated to the group managing prefix data. In this sense, an organization can keep a complete separation between groups managing prefix data and groups managing PTR records.

During the discussion of the draft, some organizations expressed a desire to achieve this type of separation in operational practice. In particular, groups associated with routing and prefix management might manage the prefix related records while other groups associated with DHCP and IP address management currently manage the PTR records. This example simply illustrates these groups can be kept distinct if an organization so desires. As with any DNS deployment, an organization makes its own decisions on where to make zone cuts and how to manage their own delegation.

6.4. Prefix Enumeration

This document introduces a convention for naming IPv4 and IPv6 prefixes. It is not an enumeration technique. To illustrate the difference between lookup and enumeration we consider a hypothetical application that uses LOC resource records to associate geographic locations with prefixes. Note the use of the LOC record is simply to make the example concrete and the same argument applies to any type of data stored at a prefix.

An application can easily lookup the LOC resource record associated with a prefix using this naming convention. The application simply converts the prefix (IPv4 or IPv6) into a DNS name as described in the previous sections and queries for the LOC record associated with

that name. Using DNSSEC, an application can also authenticate the LOC record or provide authenticated denial of existence proving that no such LOC record exists.

A distinct question is how one might enumerate all possible prefixes that have LOC records. The naming convention does not directly provide enumeration. Applications might develop strategies for searching all possible names by variations of brute force searches, exploiting NSEC records (if used), or by adding additional record types to aid in finding related prefixes. The naming convention proposed here does not provide an explicit mechanism to enumerate all prefixes with a particular resource record type.

6.5. Finding Longest Matches

Another distinct question is how one could find the longest match for a given IP address or prefix. For example, the application might want to find the most specific prefix (longest match) that has a LOC record and covers a particular IP address. Similar to enumeration, the naming convention does not directly provide longest match. Applications might develop strategies for searching all covering prefixes using variations of brute force searches, exploiting NSEC records (if used), using NXDOMAIN queries to find zone boundaries, or by adding additional record types to aid in finding related prefixes. These techniques are application-dependent. The naming convention proposed here does not provide an explicit mechanism to find the longest matching prefix for an IP address.

The naming convention proposed here provides a way to name a prefix. Once one has this name, all the advantages (and disadvantages) of DNS apply. One can easily issue queries for the name and retrieve resource records associated with that name. For many applications, this is sufficient. Applications that require more complex prefix related functions, such as enumerating all prefixes of a given type or finding the longest prefix match, need to build this functionality into their application. The naming convention provides the necessary building blocks to achieve this, but does not dictate how a particular application will assemble the building blocks.

7. Security Considerations

This document only introduces a naming convention. Applications that make use of this naming convention may require the use of DNSSEC to validate the resource records stored at these names.

8. IANA Considerations

This document does not request any IANA action.

9. Acknowledgments

The authors would like to thank Danny McPherson (Verisign), Lixia Zhang (UCLA), and Kim Claffy (CAIDA) for their comments and suggestions. This document was aided via numerous discussions at NANOG, IETF and private meetings with ISPs, telecomm carriers, and research organizations too numerous to mention by name. Thanks to all for your comments and advice.

10. Change History

Changes from version 01 to 02

Concerns were raised at the IETF 83 meeting that the document appeared to specific to the routing application. Several other applications were mentioned. We clarified the introduction to show that the naming convention is application agnostic.

Expanded the related work discussion to include RFC 1101.

The "m" label is now added even when on an octet boundary.

Moved all other discussion into the Additional Considerations section; removing the alternate naming and replacing it with a discussion of existing delegations, adding a section on separating prefix and PTR records, added a section on enumerating prefixes and finding longest matches. All these changes reflect comments from the mailing list, IETF 83 discussions, and other comments. They do not change the naming scheme itself.

To emphasize the approach is application agnostic, the appendix examples were changed from using routing security records to LOC records. Any record type could be used, but LOC records were chosen as they were viewed as easy to understand.

Changes from version 00 to 01

Introduction added an additional subsection on aligning the DNS hierarchy with the IP address hierarchy.

Clarified step 1 of the naming algorithm on removing octets that are not significant.

Expanded and clarified the discussion of alternate name encoding for prefixes on an octet boundary.

Added Eric Osterweil as a co-author

11. References

11.1. Normative References

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

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

11.2. Informative References

[I-D.bates-bgp4-nlri-orig-verif] Bates, T., Bush, R., Li, T., and Y. Rekhter, "DNS-based NLRI origin AS verification in BGP", draft-bates-bgp4-nlri-orig-verif-00 (work in progress), January 1998.

[I-D.gersch-grow-revdns-bgp]
Gersch, J., Massey, D., Osterweil, E., and L. Zhang, "DNS Resource Records for BGP Routing Data", draft-gersch-grow-revdns-bgp-00 (work in progress), February 2012.

[I-D.howard-isp-ip6rdns]
Howard, L. and A. Durand, "Reverse DNS in IPv6 for Internet Service Providers", draft-howard-isp-ip6rdns-04 (work in progress), September 2010.

[RFC1101] Mockapetris, P., "DNS encoding of network names and other types", RFC 1101, April 1989.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, March 1998.

Appendix A. Example Zone Files

A.1. Example 1

This example shows several DNS records added to an existing reverse-DNS zone file at octet boundary 129.82.0.0/16. The records show how LOC records could be specified in the zone file to be associated with an address block. Otherwise no other changes were made. This example has added records with LOC information pertinent to address blocks 129.82/16 and the four /18's at 129.82.0.0/18, 129.82.64.0/18, 129.82.128.0/18, and 129.82.192.0/18.

```
$TTL 3600
$ORIGIN 82.129.in-addr.arpa.

@    IN    SOA     rush.colostate.edu.  dnsadmin.colostate.edu. (
                        2012021300      ; serial number
```

```
                                900             ; refresh, 15 minutes
                                600             ; update retry, 10 minutes
                                86400           ; expiry, 1 day
                                3600            ; minimum, 1 hour
                               )

          IN    NS      dns1.colostate.edu.
          IN    NS      dns2.colostate.edu.

     m                  IN   LOC   latitude/longitude info for the /16
     ; 129.82.0.0/16

     0.0.m              IN   LOC   lat/long for North campus
     ; 129.82.0.0/18

     1.0.m              IN   LOC   lat/long for South campus
     ; 129.82.64.0/18

     0.1.m              IN   LOC   lat/long for Denver campus
     ; 129.82.128.0/18

     1.1.m              IN   LOC   lat/long for Boulder campus
     ; 129.82.192.0/18


     ;  delegations required for 256 /24 zones which contain PTR records

     1    IN  NS  dns1.colostate.edu.
          IN  NS  dns2.colostate.edu.
     2    IN  NS  dns1.colostate.edu.
          IN  NS  dns2.colostate.edu.

     ;  continuation to 255 is left out for the sake of brevity
```

A.2. Example 2

This example illustrates the creation of a new zone for 216.17.128.0/17 which is not at an octet boundary. The existing 256 zones delegated at IN-ADDR.ARPA for the range 0.17.128 through 255.17.216.in-addr.arpa remain unchanged; they contain PTR records maintained by the appropriate zone owners.

In this example we have added several records all at the same domain name with information pertinent to address block 216.17.128.0/17.

Only a single new delegation needs to be added to IN-ADDR.ARPA:

```
        1.m.17.216.in-addr.arpa  NS   ns.frii.net
```

This delegation refers to the new /17 zone and is not in conflict with any of the pre-existing /24 zones.

```
   $TTL 3600
   $ORIGIN 1.m.17.216.in-addr.arpa.

   @    IN   SOA     ns1.frii.net.  hostmaster.frii.net. (
                          2012021300      ; serial number
                          14400           ; refresh, 4 hours
                          3600            ; update retry, 1 hour
                          604800          ; expiry, 7 days
                          600             ; minimum, 10 minutes
                         )

        IN   NS      ns1.frii.net.
        IN   NS      ns2.frii.net.

   $ORIGIN 17.216.in-addr.arpa.

   1.m               LOC       lat/long for main office
   ;216.17.128.0/17


   ; no other delegations or PTR records are needed in this zone file
```

Authors' Addresses

Joe Gersch
Secure64 SW Corp
Fort Collins, CO
US

Email: joe.gersch@secure64.com

Dan Massey
Colorado State University
Fort Collins, CO
US

Email: massey@cs.colostate.edu

Eric Osterweil
Verisign
Reston, VA
US

Email: eosterweil@verisign.com