Appsawg
Internet-Draft
Intended status: Informational
Expires: June 6, 2014

D. Worley
Ariadne

December 3, 2013

ROID: An Experimental Protocol for Name-Based Information Retrieval
draft-worley-roid-00

Abstract

Information-centric networking (ICN) is an approach to evolve the Internet infrastructure to access data by unique names. ROID ("Resolver for OIDs") is an experimental protocol to support simple implementation of ICN. ROID defines a simple model for mapping data names -- in the form of URNs of the "oid" scheme -- into URLs which can be used to access the data objects. This experimental version of ROID provides the resolution information via DNS records that can be served by a standard DNS server (named a/k/a Bind). This structure is designed for easy deployment of ROID in existing environments in order to support proof-of-concept implementations of ICN.

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 June 6, 2014.

Table of Contents

1. Introduction

The core concept of "information-centric networking" is that information objects can be retrieved by providing a name for the object rather than providing a location from which the object is to be retrieved. As described by the IRTF:

> Information-centric networking (ICN) is an approach to evolve the Internet infrastructure to directly support this use by introducing uniquely named data as a core Internet principle. Data becomes independent from location, application, storage, and means of transportation, enabling in-network caching and replication. The expected benefits are improved efficiency, better scalability with respect to information/bandwidth demand and better robustness in challenging communication scenarios. These concepts are known under different terms, including but not limited to: Network of Information (NetInf), Named Data Networking (NDN) and Publish/Subscribe Networking.[IRTF-ICNRG]

ROID ("Resolver for OIDs") is an experimental protocol to support simple implementation of ICN. ROID defines a simple model for mapping data names -- in the form of URNs of the "oid" scheme -- into URLs which can be used to access the data objects.

This experimental version of ROID provides the resolution information via DNS records that can be served by a standard DNS server. That is, it maps the ROID resolution information model into DNS resource records, exploiting particular behaviors of DNS servers. This structure is designed to make deployment of ROID simple in existing environments in order to support proof-of-concept implementations of ICN.

2. Object Identifiers

Object identifiers (OIDs) are a uniform, extensible set of names with a particularly simple structure. Each object identifier is a finite sequence of non-negative integers. A typical example is written "2.999.28.143". OIDs are hierarchically organized, so 2.999.28.143 is considered a child of 2.999.28, which is in turn considered a child of 2.999.

Like DNS names, the OID hierarchy is a hierarchy of delegated authority. An OID can be assigned to a individual or organization, which is then authoritative for allocating meanings to OIDs that are descendant from that OID. Recursively, the OID owner can delegate ownership of a descendant OID to another individual or organization.

For example, the OID 1.3.6.1.4 has been delegated to IANA. The OIDs that are children of 1.3.6.1.4.1 are delegated to organizations under the "Private Enterprise Number" registry of IANA. The OID 1.3.6.1.4.1.14490 is delegated to Ariadne Internet Services, Inc. Ariadne has reserved the children of 1.3.6.1.4.1.14490.27 for delegation to entities that want to experiment with using ROID. 1.3.6.1.4.1.14490.27.78 could be assigned to such an entity, which would then assign meanings to its descendant OIDs.

(The OID 2.999 is delegated for use in examples, in the same way that example.com is reserved for use in examples.)

OIDs can be represented as URNs using the "oid" schema.[RFC3061] For example, the OID 1.3.6.1.4.1.14490.5.1.6910 (which has been defined by Ariadne to name RFC 6910) has the corresponding URN <urn:oid:1.3.6.1.4.1.14490.5.1.6910>.

3. ROID Resolution Information Model

The ROID information model is based on a subset of the OID universe, with certain OIDs naming objects. These OIDs are called "name OIDs".

The other OIDs that are within the ROID model are called "group OIDs". All ancestor OIDs of name OIDs or group OIDs are group OIDs, and thus are not name OIDs and do not name objects. But a group OID

is not required to have name OID descendants. Thus, all name OIDs are leaf nodes within the tree of OIDs within the ROID model, but some group OIDs may also be leaf nodes. (Presumably they will be provided with descendant name OIDs at some later time.)

Authority delegation points in the set of ROID OIDs are the same OIDs within the ROID OID universe that are authority delegation points in the universe of object identifiers. If authority for an object identifier and its descendants is delegated to an entity, and that entity further delegates authority for the same OID to another entity, the ROID model can only represent the ultimate delegate for a single OID. (Of course, this does not restrict the model's representation of further delegation of a descendant OID.)

The ROID information model is organized into records. Each record contains the following data elements:

OID This is the ROID OID about which the record provides information. The record is said to be attached to this OID.

type This the the record type, which is taken from a fixed enumeration, which is described below. The type determines which OIDs may have this type of record attached, and what strings are valid data fields for this type of record.

data The data field is a string of Unicode characters. Its meaning and which strings are valid values depends on the record type.

The following record types describe delegation authority. They can be attached to group OIDs or name OIDs. If they are attached to a group OID, they apply to the group OID and all descendant OIDs, excluding any descendant OIDs that have their own OWN record, and the descendant OIDs thereof.

<OID> OWN "<string>"
("owner") This record provides the human-readable name and contact information (as a Unicode string) of the entity that is the owner of the OID. An example is, '1.3.6.1.4.1.14490 OWN "Ariadne Internet Services, Inc., Waltham, MA, USA"'.

<OID> OUR "<URL-string>"
("owner URL") This record provides a URL which provides a way to contact the owner. There can be more than one of these records for an OID. The OID must have an OWN record An example is, '1.3.6.1.4.1.14490 OUR "mailto:oid@ariadne.com"'.

The following record types provide information about the object named by a name OID.

<OID> URL "<URL-string>"
("URL") This record provides a URL which provides a way to access the object named by the OID. There can be more than one of these records for an OID.

<OID> DES "<string>"
("description") This record is a text string that provides a human-readable description of the object named by the OID. There can be more than one of these records for an OID.

<OID> DUR "<URL-string>"
("description URL") This record provides a URL which can be used to retrieve a human-readable description of the object named by the OID. There can be more than one of these records for an OID.

The following record types provide relocation information.

<OID> MVP "<target-OID>"

<OID> MVT "<target-OID>"

("moved permanently" and "moved temporarily") These record types provide relocation information. Both records indicate that <OID> and any descendant OIDs should be resolved by replacing the initial portion that is <OID> with <target-OID>. An example is: If there is a record '1.3.6.1.4.14490.8 MVP 2.999.2342', then information regarding OID 1.3.6.1.4.1.14490.8.5.1.6910 should be sought at 2.999.2342.5.1.6910.

The MVP record type means "moved permanently". This means that <target-OID> is expected to be valid longer into the future than <OID> and so any reference to an object that is to be stored should be translated to use the <target-OID> prefix.

The second record type means "moved temporarily". This has the opposite implication, that <OID> is expected to be valid further into the future than <target-OID>, and so no such translation should be done.

4. Representation in DNS Records

In the experimental version of ROID, ROID information is stored in the DNS hierarchy. ROID records are represented by DNS resource records as described below. All representing DNS records have class "IN" (Internet).

ROID authority delegation points are not represented explicitly in DNS. DNS zone delegations are allowed at any point in the DNS

hierarchy, although it is expected that DNS zone delegation points will be the same as ROID authority delegation points. This correspondence allows an entity that has authority over a particular subset of the ROID space to provide the corresponding DNS service using DNS servers that it controls.

ROID stores information about OIDs in DNS resource records. Each OID is translated into a domain name by reversing the list of integers, and then appending the labels ".OID.ARPA". Resource records for this domain name represent the ROID data. An example is that the ROID records for the OID 2.999.2342.5.1.6910 are represented by DNS records for the domain name 6910.1.5.2342.999.2.OID.ARPA.

DNS/ROID delegation points are represented by NS records which map an OID delegation point to the DNS servers that are authoritative for that OID and its descendant OIDs:

```
14490.1.4.1.6.3.1.OID.ARPA.  NS NS1.ARIADNE.COM.
14490.1.4.1.6.3.1.OID.ARPA.  NS NS2.ARIADNE.COM.
```

ROID records for an OID that is a delegation point are always mapped to DNS records served by the delegated servers rather than the delegating servers, as is usual in DNS. (The DNS zone file will also need to contain A records for the servers, per the usual rules for DNS.)

MVP and MVT ROID records are represented by artificial NS records which give the target OID's domain name, prefixed by the label "MVP" or "MVT", respectively. These NS records do not have corresponding A records.

```
1.3.6.1.4.14490.8 MVP 2.999.2342
```

is represented by

```
8.14490.4.1.6.3.1.OID.ARPA.  NS MVP.2342.9.0.OID.ARPA
```

This peculiar way of using NS records is to exploit the behaviors of DNS servers to speed up ROID resolution operations.

A ROID record of any other type type is represented by a TXT record containing two text fields. The first text field is the ROID information type (represented as three upper-case letters), and the second text field is the ROID information data. Note that text fields in DNS TXT records are actually binary strings of up to 255 octets.[RFC1035] URLs are stored in ASCII (or equivalently UTF-8) encoding. <string>s are stored in UTF-8 encoding.

1.3.6.1.4.1.14490 OUR "mailto:oid@ariadne.com"

is represented by

14490.1.4.1.6.3.1.OID.ARPA. TXT "OUR" "mailto:oid@ariadne.com"

5. URN Resolution Process

ROID resolution proceeds much like ordinary DNS resolution. In the simplest case, to resolve an OID URN into access URLs, a DNS query is made for TXT records for the corresponding domain name, ignoring all TXT records whose first data fields is not "URL".

For example, to resolve urn:oid:1.3.6.1.4.1.14490.5.1.6910, a query is made for TXT records for the domain name 6910.1.5.14490.1.4.1.6.3.1.OID.ARPA. That query returns these records:

6910.1.5.14490.1.4.1.6.3.1.OID.ARPA. TXT "URL" "http://tools.ietf.org/html/rfc6910" 6910.1.5.14490.1.4.1.6.3.1.OID.ARPA. TXT "DES" "RFC 6910"

From the first TXT record the access URL http://tools.ietf.org/html/rfc6910 is extracted.

The DNS server that is queried may not have data for the requested domain name, but rather only have delegation that points to another server providing data for a subordinate DNS zone. (ROID resolution queries to DNS servers must be non-recursive, as described below, so the resolver cannot depend on the server making a query to the delegate server.) In that case, the server will return the NS records for the delegation and A records for the delegate servers. The resolver must then recursively query the delegate servers for the desired records.

If the OID is a descendant of a group OID for which there is a relocation record, the query of the server will return the artificial NS record that represents the relocation record. For example, if we are resolving 1.3.6.1.4.1.14490.21.2.6910, and there is a relocation

1.3.6.1.4.1.14490.21.2 MVT 1.3.1.6.1.4.1.14490.5.1

then queries to DNS servers for 6910.2.21.14490.1.4.1.6.3.1.OID.ARPA will eventually return

2.21.14490.1.4.1.6.3.1.OID.ARPA. NS MVT.1.5.14490.1.4.1.6.3.1.OID.ARPA.

Based on this relocation, the resolver updates the domain name it is searching for to 6910.1.5.14490.1.4.1.6.3.1.OID.ARPA. In principle, the resolver needs to begin querying for this new domain name at a root server, although caching previous query results will usually allow the resolver to begin querying for a new domain name at a delegate server.

Because the NS records that represent relocation records do not name actual DNS servers, the server that is queried must be prevented from attempting to recursively follow the NS record. For this reason, all queries from the resolver to a DNS server must be specified as non-recursive.

The values of DES and DUR records can be located in the same way as the values of URL records.

The values of OWN and OUR records that apply to an OID require more work to obtain because those records may be attached to an OID that is ancestral to the OID in question. The resolver must successively query for TXT records for the domain names of the OID, the OID's parent, it's parent's parent, etc. until an OWN record is found. (This process will likely be greatly accelerated by caching query results.) Once an OWN record is located, any OUR records will be attached to the same OID as the OWN record is (by the structure of the ROID information model).

6. OID Allocations

An entity can obtain an OID allocation in many ways, including:

National standards organization National standards organizations can delegate OIDs. For example, the American National Standards Institute is the United States' member body in ISO, and provides an OID registration service.[ANSI-OID] ISO member body OIDs are children of OID 1.2. ANSI's OID is 1.2.840, and the OIDs it delegates are children of 1.2.840.1.

IANA private enterprise numbers IANA assigns "private enterprise numbers" to any entity that requests one.[PEN-request] The OIDs IANA delegates are children of 1.3.6.1.4.1.[PEN-registry]

Assigned by Ariadne Ariadne has reserved an OID subtree for assignments to entities experimenting with ROID. This subtree is rooted at 1.3.6.1.4.1.14490.26. Entities can be assigned an OID by contacting <mailto:oid@ariadne.com>.

Based on telephone numbers If an entity has been assigned an E.164 telephone number, Ariadne has delegated to the entity the OID

which is 1.3.6.1.4.1.14490.27.<E.164-number>. An example is, the assignee of the E.164 number +1 617 555 1212 is delegated the OID 1.3.6.1.4.1.14490.27.16175551212.

7. Informative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC3061] Mealling, M., "A URN Namespace of Object Identifiers", RFC 3061, February 2001.

[IRTF-ICNRG] Internet Research Task Force, "Information-Centric Networking Research Group (ICNRG)", June 2013, <http://irtf.org/icnrg>.

[ANSI-OID] American National Standards Institute, "Organization Name Registration", August 2013, <http://www.ansi.org/other_services/registration_programs/reg_org.aspx?menuid=10>.

[PEN-request] Internet Assigned Numbers Authority, "Private Enterprise Number (PEN) Request Template", August 2013, <http://pen.iana.org/pen/PenApplication.page>.

[PEN-registry] Internet Assigned Numbers Authority, "PRIVATE ENTERPRISE NUMBERS", August 2013, <http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

Appendix A. Sample Resolver and Testing Framework

This appendix contains sample code for a simple resolver and a testing framework that exercises it. The ROID information records that it contains are:

1.3.6.1.4.1.14490.5.1.6910 URL "http://tools.ietf.org/html/rfc6910"

1.3.6.1.4.1.14490.21.1 MVP 1.3.6.1.4.1.14490.21.2

1.3.6.1.4.1.14490.21.2 MVT 1.3.6.1.4.1.14490.5.1

The test script resolves the URN <urn:oid:1.3.6.1.4.1.14490.5.1.6910>.

The test system executes two copies of named (a/k/a Bind), one listening on 127.0.0.200:12345 functioning as a root server, and one listening on 127.0.0.201:12345 functioning as the server for the zone 14490.1.4.1.6.3.1.OID.ARPA. On my test system both loopback addresses function out-of-the-box, but in order to allow named to use them, they must be explicitly activated with the commands:

```
ip address add 127.0.0.200 dev lo

ip address add 127.0.0.201 dev lo
```

Figure 1: ROID-resolver: Sample resolver

```
#! /bin/perl

# Resolver for ROID, version 1.

# Usage:
#    ROID-resolver urn:oid:xxx.xxx.xxx
# Options:
#    -c|canonical   print the canonical form of the URN
#    -d|debug       increment the debugging level
#    -n|nameserver  supply the address of a nameserver to use
#    -p|port        port of nameserver to query
#
# Outputs the object access URLs from the ROID URL records.

use strict;

# Process the options.

# True if the canonical form is to be printed.
my($canonical) = 0;
# The debugging level.
my($debug) = 0;
# The root DNS domain for resolution.
# Will never have a leading and trailing dot (after processing below).
my($root_domain) = 'oid.arpa';
# The nameservers the user specifies.
my(@nameservers);
# The port of the nameserver to query.
# (Unfortunately, you can't put ":port" on a nameserver specification.)
my($nameserver_port);

use Getopt::Long;
GetOptions(
    'c|canonical' => \$canonical,
    'd|debug+' => \$debug,
```

```
       'n|nameserver=s' => \@nameservers,
       'p|port=i' => \$nameserver_port,
       );

   # Ensure $root_domain does not have a leading dot.
   $root_domain =~ s%^\.%%;
   # Ensure $root_domain does not have a trailing dot.
   $root_domain =~ s%\.$%%;
   print STDERR "  \$root_domain = '$root_domain'\n" if $debug >= 1;

   # Set 127.0.0.1 as the only nameserver if the user did not provide any.
   @nameservers = ('127.0.0.1') if $#nameservers < 0;
   print STDERR "  \@nameservers = (",
       (@nameservers ? "'" . join("' ,'", @nameservers) . "'" : ""),
       ")\n"
       if $debug >= 1;

   print STDERR "  \$nameserver_port = '$nameserver_port'\n" if $debug >= 1;

   # Get the argument.
   if ($#ARGV != 0) {
       print STDERR <<EOF;
   Usage:
      $0 [options] urn:oid:xxx.xxx.xxx

   Outputs the object access URLs from the ROID URL records.

       d|debug               increment debugging level
       n|nameserver=string   provide IP address of a nameserver to use
                             (default is 127.0.0.1)
       p|port=number         port number of nameservers to query (default is
53)

   EOF
        exit 1;
   }
   my($urn) = $ARGV[0];
   print STDERR "  \$urn = '$urn'\n" if $debug >= 1;

   # Check that it's an oid URN.
   # The scheme (urn) is case-insensitive per RFC 3986.
   # The namespace ID (oid) is case-insensitive per RFC 2141.
   my($oid);
   unless (($oid) = $urn =~ m%^urn:oid:((\d|[1-9]\d+)(\.(\d|[1-9]\d+))*)$%i) {
       print STDERR "Argument '$urn' is not an OID URN or is syntactically
invalid.\n";
       exit 1;
   }
   print STDERR "  URN passes\n" if $debug >= 2;
```

```
   print STDERR "  \$oid = '$oid'\n" if $debug >= 1;
```

```
   # Construct the domain name we want to look up.
   my($domain_name) =
       join('.', reverse(split(/\./, $oid))) . '.' . $root_domain . '.';
   print STDERR "  \$domain_name = '$domain_name'\n" if $debug >= 1;

   # Set up the DNS resolver.
   use Net::DNS::Resolver;
   my($resolver) = Net::DNS::Resolver->new(
       recurse     => 0,
       debug       => ($debug >= 2),
       port        => $nameserver_port,
       );

   # Set up for the loop.
   # $domain_name and $current_domain_name always end in '.'.
   my($current_domain_name) = $domain_name;
   my(@current_nameservers) = @nameservers;
   # Initialize the canonical domain name.
   # This records the domain name that represents 'best' version of the
   # OID.  It records all permanent redirections *before* any temporary
   # redirections.
   my($canonical_domain_name) = $domain_name;
   my($temporary_redirection_seen) = 0;

   LOOKUP: {
       print STDERR "  Looking up '$current_domain_name' at ",
           join(' ', @current_nameservers), "\n"
       if $debug >= 1;

       # Set the nameservers.
       $resolver->nameservers(@current_nameservers);

       # Fetch the DNS records for the domain name.
       my($answer) = $resolver->send($current_domain_name, 'ANY');
       print STDERR "  send() ", (defined($answer) ? "succeeded" : "failed"),
"\n"
       if $debug >= 2;

       # Check the answer.

       # If $answer is undefined, no answer was received from the DNS server.
       if (!defined($answer)) {
       print STDERR "Query of '$current_domain_name' failed.\n";
       exit 1;
       }

       # Dissect the answer.
       my(@answers) = $answer->answer;
       my(@authorities) = $answer->authority;
```

```
       my(@additionals) = $answer->additional;

       my($done) = 0;

       # Check for URL records.
       foreach my $a (@answers) {
       if ($a->type eq 'TXT' && $a->class eq 'IN') {
           my(@txtdata) = $a->txtdata;
           print '  @txtdata = ', join(' ', @txtdata), "\n" if $debug >= 1;
           # Get URL ROID record.
           if ($txtdata[0] eq 'URL') {
           # Print the object access URL.
           print $txtdata[1], "\n" if $debug >= 2;
           $done = 1;
           }
       }
       }
       last LOOKUP if $done;

       # Check for NS delegation records.
       foreach my $a (@authorities) {
       print "  \$a->type = '", $a->type, "', \$a->class = '", $a->class,
           "', \$current_domain_name = '$current_domain_name', \$a->name = '",
$a->name,
           "'\n"
           if $debug >= 2;
       my($name) = $a->name;
       if ($a->type eq 'NS' && $a->class eq 'IN' &&
           $current_domain_name =~ m%\b\Q$name\E\.$%) {
           @current_nameservers = () if $done == 0;
           $done = 1;
           my($server) = $a->nsdname;
           print "  \$server = $server\n" if $debug >= 2;
           # The server may be a real server (if this is an NS delegation
           # point) or a temporary or permanent redirection.  Examine
           # the server name to determine what it means.
           if (my($mode, $redirection) =
           $server =~ /^(MVT|MVP)\.(.*\.OID\.ARPA)$/i) {
           # This is a redirection.
           # Replace the redirected portion of the OID with the new OID.
           $current_domain_name =~ s%\Q$name\E\.$%$redirection.%;
           # Record permanent redirections.
           if (uc($mode) eq 'MVP') {
               $canonical_domain_name = $current_domain_name
               unless $temporary_redirection_seen;
           } else {
               $temporary_redirection_seen = 1;
           }
```

```
            if ($debug >= 2) {
```

```
               print STDERR "  \$mode = '$mode', \$temporary_redirection_seen =
$temporary_redirection_seen, \$canonical_domain_name =
'$canonical_domain_name'\n";
           }
           if ($debug >= 1) {
               print STDERR "  $name $mode redirected to $redirection\n";
           }
           # Start searching from the origin again.
           @current_nameservers = @nameservers;
           redo LOOKUP;
           } else {
           # Look for an A record for the server name.
             ADDITIONALS:
           foreach my $aa (@additionals) {
               print "  \$aa->type = '", $aa->type, "', \$aa->class = '",
               $aa->class, "', \$aa->name = '", $aa->name, "'\n"
               if $debug >= 2;
               if ($aa->type eq 'A' && $aa->class eq 'IN' &&
               $aa->name eq $server) {
               print "  \$aa->address = '", $aa->address, "'\n"
                   if $debug >= 2;
               push(@current_nameservers, $aa->address);
               last ADDITIONALS;
               }
           }
           }
       }
       }
       if ($done) {
       if ($#current_nameservers < 0) {
           print STDERR "NS redirection found but no nameserver addresses
provided.\n";
           exit 1;
       }
       print STDERR "  NS delegation to nameservers ",
               join("' ,'", @current_nameservers) . "\n"
           if $debug >= 1;
       redo LOOKUP;
       }

       # Otherwise we've hit a dead end.
       print STDERR "No usable answer for '$current_domain_name' from
nameservers.\n";
       exit 1;
   }

   # If requested, print the canonical OID URN.
   if ($canonical) {
```

```
        print STDERR "  \$canonical_domain_name = '$canonical_domain_name'\n"
        if $debug >= 2;
        # Remove the root domain.
        my($c) = $canonical_domain_name;
```

```
    $c =~ s%\.\Q$root_domain\E\.$%%;
    print 'canonical urn:oid:', join('.', reverse(split(/\./, $c))), "\n";
}
```

Figure 1: ROID-resolver: Sample resolver

Figure 2: zone.oid.arpa: Zone file for OID.ARPA.

```
$ORIGIN oid.arpa.
$TTL 86400
@     IN     SOA    ariadne.com.     worley.ariadne.com. (
                    2013120101 ; serial
                    21600      ; refresh after 6 hours
                    3600       ; retry after 1 hour
                    604800     ; expire after 1 week
                    86400 )    ; minimum TTL of 1 day

; Dummy entries needed to make Bind think this is a valid zone.
@     IN      NS    dummy200
dummy200 IN   A     127.0.0.200

; Delegate the 14490 enterprise subtree to 201.
14490.1.4.1.6.3.1 IN NS dummy201
dummy201 IN   A     127.0.0.201
```

Figure 2: zone.oid.arpa: Zone file for OID.ARPA.

Figure 3: zone.14490: Zone file for 14490.1.4.1.6.3.1.OID.ARPA.

```
$ORIGIN 14490.1.4.1.6.3.1.oid.arpa.
$TTL 86400
@     IN     SOA    ariadne.com.     worley.ariadne.com. (
                    2013120100 ; serial
                    21600      ; refresh after 6 hours
                    3600       ; retry after 1 hour
                    604800     ; expire after 1 week
                    86400 )    ; minimum TTL of 1 day

; Dummy entries needed to make Bind think this is a valid zone.
@     IN      NS    dummy201
dummy201 IN   A     127.0.0.201

; Continue lines using ( ... )
6910.1.5 IN   TXT    ( "URL" "http://tools.ietf.org/html/rfc6910" )

; Permanent redirection from 21.1 to 21.2
1.21  IN      NS    MVP.2.21
; Temporary redirection from 21.2.6910 to 5.1.6910
6910.2.21 IN  NS    MVT.6910.1.5
```

Figure 3: zone.14490: Zone file for 14490.1.4.1.6.3.1.OID.ARPA.

Figure 4: test-named.200.conf: Configuration for nameserver at 127.0.0.200

```
options {
    directory "/home/worley/oid/Information-Centric-Networking";
    # This configuration is served on *.200.
    listen-on { 127.0.0.200; };
    pid-file "/home/worley/oid/Information-Centric-Networking/named.pid";
};

# Turn off the control channel feature.
controls { };

zone "oid.arpa" {
     type master;
     file "zone.oid.arpa";
};
```

Figure 4: test-named.200.conf: Configuration for nameserver at 127.0.0.200

Figure 5: test-named.200.conf: Configuration for nameserver at 127.0.0.201

```
options {
    directory "/home/worley/oid/Information-Centric-Networking";
    # This configuration is served on *.201.
    listen-on { 127.0.0.201; };
    pid-file "/home/worley/oid/Information-Centric-Networking/named.pid";
};

# Turn off the control channel feature.
controls { };

zone "14490.1.4.1.6.3.1.oid.arpa" {
     type master;
     file "zone.14490";
};
```

Figure 5: test-named.200.conf: Configuration for nameserver at 127.0.0.201

Figure 6: test-script: Control script

```
   #! /bin/bash

   # Run the ROID test system.

   # Note that the addresses to be used and the port to be used are heavily
   # embedded in the Bind control files, and so their uses in this script
cannot
   # be turned into variables without adding code to update the Bind control
   # files.

   set -x

   # Set up additional addresses.
   # named won't use non-standard loopback addresses unless they are listed
   # as official addresses of the lo interface.  (Despite the fact that you
   # can ping the addresses and even connect to them without "ip address add".)
   # These commands have to be executed as root.
   ip address add 127.0.0.200 dev lo
   ip address add 127.0.0.201 dev lo

   # Start Bind.
   # Have to use full path to run named, because named may not be on a user's
path.
   # -c argument must be absolute path (see named manual page)
   # The -p option seems to be necessary to allow the listen-on option in the
   # configuration file to work.
   /usr/sbin/named -g -4 \
       -p 12345 \
       -c /home/worley/oid/Information-Centric-Networking/test-named.200.conf \
       >named.200.log 2>&1 &
   /usr/sbin/named -g -4 \
       -p 12345 \
       -c /home/worley/oid/Information-Centric-Networking/test-named.201.conf \
       >named.201.log 2>&1 &
   # Make sure named has time to start up.
   sleep 2

   # Test queries with Dig.
   #dig +norecurse @127.0.0.200 -p12345 14490.1.4.1.6.3.1.oid.arpa. ANY
   #dig +norecurse @127.0.0.201 -p12345 6910.1.5.14490.1.4.1.6.3.1.oid.arpa.
ANY

   # Run the resolver.
   ./ROID-resolver -d -c -n 127.0.0.200 -p 12345 urn:oid:
1.3.6.1.4.1.14490.21.1.6910

   # Terminate Bind.
   kill %1 %2
```

Figure 6: test-script: Control script

Author's Address

Dale R. Worley
Ariadne Internet Services, Inc.
738 Main St.
Waltham, MA 02451
US

Phone: +1 781 647 9199
Email: worley@ariadne.com