

Network Working Group  
Internet-Draft  
Updates: [7706](#) (if approved)  
Intended status: Informational  
Expires: January 6, 2020

W. Kumari  
Google  
P. Hoffman  
ICANN  
July 5, 2019

**Running a Root Server Local to a Resolver**  
**draft-ietf-dnsop-7706bis-04**

**Abstract**

Some DNS recursive resolvers have longer-than-desired round-trip times to the closest DNS root server. Some DNS recursive resolver operators want to prevent snooping of requests sent to DNS root servers by third parties. Such resolvers can greatly decrease the round-trip time and prevent observation of requests by running a copy of the full root zone on the same server, such as on a loopback address. This document shows how to start and maintain such a copy of the root zone that does not pose a threat to other users of the DNS, at the cost of adding some operational fragility for the operator.

This draft will update [RFC 7706](#). See [Section 1.1](#) for a list of topics that will be added in the update.

[ Ed note: Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication.]

[ This document is being collaborated on in Github at: <https://github.com/wkumari/draft-kh-dnsop-7706bis>. The most recent version of the document, open issues, and so on should all be available there. The authors gratefully accept pull requests. ]

**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 <https://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 January 6, 2020.

## Copyright Notice

Copyright (c) 2019 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 (<https://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.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Updates from <a href="#">RFC 7706</a> . . . . .	<a href="#">4</a>
<a href="#">1.2.</a>	Requirements Notation . . . . .	<a href="#">5</a>
<a href="#">2.</a>	Requirements . . . . .	<a href="#">5</a>
<a href="#">3.</a>	Operation of the Root Zone on the Local Server . . . . .	<a href="#">5</a>
<a href="#">4.</a>	Using the Root Zone Server on the Same Host . . . . .	<a href="#">7</a>
<a href="#">5.</a>	Security Considerations . . . . .	<a href="#">7</a>
<a href="#">6.</a>	References . . . . .	<a href="#">7</a>
<a href="#">6.1.</a>	Normative References . . . . .	<a href="#">7</a>
<a href="#">6.2.</a>	Informative References . . . . .	<a href="#">8</a>
<a href="#">Appendix A.</a>	Current Sources of the Root Zone . . . . .	<a href="#">8</a>
<a href="#">A.1.</a>	Root Zone Services . . . . .	<a href="#">9</a>
<a href="#">Appendix B.</a>	Example Configurations of Common Implementations . . . . .	<a href="#">9</a>
<a href="#">B.1.</a>	Example Configuration: BIND 9.12 . . . . .	<a href="#">9</a>
<a href="#">B.2.</a>	Example Configuration: Unbound 1.8 . . . . .	<a href="#">11</a>
<a href="#">B.3.</a>	Example Configuration: BIND 9.14 . . . . .	<a href="#">12</a>
<a href="#">B.4.</a>	Example Configuration: Unbound 1.9 . . . . .	<a href="#">12</a>
<a href="#">B.5.</a>	Example Configuration: Knot Resolver . . . . .	<a href="#">13</a>
<a href="#">B.6.</a>	Example Configuration: Microsoft Windows Server 2012 . . . . .	<a href="#">13</a>
	Acknowledgements . . . . .	<a href="#">14</a>
	Authors' Addresses . . . . .	<a href="#">14</a>



## **1. Introduction**

DNS recursive resolvers have to provide answers to all queries from their customers, even those for domain names that do not exist. For each queried name that has a top-level domain (TLD) that is not in the recursive resolver's cache, the resolver must send a query to a root server to get the information for that TLD, or to find out that the TLD does not exist. Research shows that the vast majority of queries going to the root are for names that do not exist in the root zone because negative answers are sometimes cached for a much shorter period of time.

Many of the queries from recursive resolvers to root servers get answers that are referrals to other servers. Malicious third parties might be able to observe that traffic on the network between the recursive resolver and root servers.

The primary goals of this design are to provide more reliable answers for queries to the root zone during network attacks, and to prevent queries and responses from being visible on the network. This design will probably have little effect on getting faster responses to stub resolver for good queries on TLDs, because the TTL for most TLDs is usually long-lived (on the order of a day or two) and is thus usually already in the cache of the recursive resolver; the same is true for the TTL for negative answers from the root servers. (Although the primary goal of the design is for serving the root zone, the method can be used for any zone.)

This document describes a method for the operator of a recursive resolver to have a complete root zone locally, and to hide these queries from outsiders. The basic idea is to create an up-to-date root zone server on the same host as the recursive server, and use that server when the recursive resolver looks up root information. The recursive resolver validates all responses from the root server on the same host, just as it would all responses from a remote root server.

This design explicitly only allows the new root zone server to be run on the same server as the recursive resolver, in order to prevent the server from serving authoritative answers to any other system. Specifically, the root server on the local system **MUST** be configured to only answer queries from the resolvers on the same host, and **MUST NOT** answer queries from any other resolver.

At the time that [RFC 7706](#) was published, it was considered controversial: there was not consensus on whether this was a "best practice". In fact, many people felt that it is an excessively risky practice because it introduced a new operational piece to local DNS



operations where there was not one before. Since then, the DNS operational community has largely shifted to believing that local serving of the root zone for an individual resolver is a reasonable practice. The advantages listed above do not come free: if this new system does not work correctly, users can get bad data, or the entire recursive resolution system might fail in ways that are hard to diagnose.

This design uses authoritative name server software running on the same machine as the recursive resolver. Thus, recursive resolver software such as BIND or modern versions of common open source recursive resolver software do not need to add new functionality, but other recursive resolver software might need to be able to talk to an authoritative server running on the same host.

A different approach to solving some of the problems discussed in this document is described in [[RFC8198](#)].

### **1.1. Updates from [RFC 7706](#)**

[RFC 7706](#) explicitly required that the root server instance be run on the loopback interface of the host running the validating resolver. However, [RFC 7706](#) also had examples of how to set up common software that did not use the loopback interface. Thus, this document loosens the restriction on the interface but keeps the requirement that only systems running on that single host be able to query that root server instance.

Removed the prohibition on distribution of recursive DNS servers including configurations for this design because some already do, and others have expressed an interest in doing so.

Added the idea that a recursive resolver using this design might switch to using the normal (remote) root servers if the local root server fails.

Refreshed the list of where one can get copies of the root zone.

Added examples of other resolvers and updated the existing examples.

[ This section will list all the changes from [RFC 7706](#). For this draft, it is also the list of changes that we will make in future versions of the draft. ]

[ Make the use cases explicit. Be clearer that a real use case is folks who are worried that root server unavailability due to DDoS against them is a reason some people would use the mechanisms here. ]



[ Describe how slaving the root zone from root zone servers does not fully remove the reliance on the root servers being available. ]

[ Other new topics might go here. ]

### **1.2. Requirements Notation**

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 [\[RFC2119\]](#).

## **2. Requirements**

In order to implement the mechanism described in this document:

- o The system MUST be able to validate a zone with DNSSEC [\[RFC4033\]](#).
- o The system MUST have an up-to-date copy of the key used to sign the DNS root.
- o The system MUST be able to retrieve a copy of the entire root zone (including all DNSSEC-related records).
- o The system MUST be able to run an authoritative server for the root zone on the same host. The root server instance MUST only respond to queries from the same host. One way to assure not responding to queries from other hosts is to make the address of the authoritative server one of the loopback addresses (that is, an address in the range 127/8 for IPv4 or ::1 in IPv6).

A corollary of the above list is that authoritative data in the root zone used on the local authoritative server MUST be identical to the same data in the root zone for the DNS. It is possible to change the unsigned data (the glue records) in the copy of the root zone, but such changes could cause problems for the recursive server that accesses the local root zone, and therefore any changes to the glue records SHOULD NOT be made.

## **3. Operation of the Root Zone on the Local Server**

The operation of an authoritative server for the root in the system described here can be done separately from the operation of the recursive resolver, or it might be part of the configuration of the recursive resolver system.

The steps to set up the root zone are:





1. Retrieve a copy of the root zone. (See [Appendix A](#) for some current locations of sources.)
2. Start the authoritative server with the root zone on an address on the host that is not in use. For IPv4, this could be 127.0.0.1, but if that address is in use, any address in 127/8 is acceptable. For IPv6, this would be ::1. It can also be a publicly-visible address on the host, but only if the authoritative server software allows restricting the addresses that can access the authoritative server, and the software is configured to only allow access from addresses on this single host.

The contents of the root zone MUST be refreshed using the timers from the SOA record in the root zone, as described in [[RFC1035](#)]. This inherently means that the contents of the local root zone will likely be a little behind those of the global root servers because those servers are updated when triggered by NOTIFY messages.

If the contents of the root zone cannot be refreshed before the expire time in the SOA, the local root server MUST return a SERVFAIL error response for all queries sent to it until the zone can be successfully be set up again. Because this would cause a recursive resolver on the same host that is relying on this root server to also fail, a resolver might be configured to immediately switch to using other (non-local) root servers if the resolver receives a SERVFAIL response from a local root server.

In the event that refreshing the contents of the root zone fails, the results can be disastrous. For example, sometimes all the NS records for a TLD are changed in a short period of time (such as 2 days); if the refreshing of the local root zone is broken during that time, the recursive resolver will have bad data for the entire TLD zone.

An administrator using the procedure in this document SHOULD have an automated method to check that the contents of the local root zone are being refreshed; this might be part of the resolver software. One way to do this is to have a separate process that periodically checks the SOA of the root zone from the local root zone and makes sure that it is changing. At the time that this document is published, the SOA for the root zone is the digital representation of the current date with a two-digit counter appended, and the SOA is changed every day even if the contents of the root zone are unchanged. For example, the SOA of the root zone on January 2, 2018 was 2018010201. A process can use this fact to create a check for the contents of the local root zone (using a program not specified in this document).



#### **4. Using the Root Zone Server on the Same Host**

A recursive resolver that wants to use a root zone server operating as described in [Section 3](#) simply specifies the local address as the place to look when it is looking for information from the root. All responses from the root server MUST be validated using DNSSEC.

Note that using this simplistic configuration will cause the recursive resolver to fail if the local root zone server fails. A more robust configuration would cause the resolver to start using the normal remote root servers when the local root server fails (such as if it does not respond or gives SERVFAIL responses).

See [Appendix B](#) for more discussion of this for specific software.

To test the proper operation of the recursive resolver with the local root server, use a DNS client to send a query for the SOA of the root to the recursive server. Make sure the response that comes back has the AA bit in the message header set to 0.

#### **5. Security Considerations**

A system that does not follow the DNSSEC-related requirements given in [Section 2](#) can be fooled into giving bad responses in the same way as any recursive resolver that does not do DNSSEC validation on responses from a remote root server. Anyone deploying the method described in this document should be familiar with the operational benefits and costs of deploying DNSSEC [[RFC4033](#)].

As stated in [Section 1](#), this design explicitly only allows the new root zone server to be run on the same host, answering queries only from resolvers on that host, in order to prevent the server from serving authoritative answers to any system other than the recursive resolver. This has the security property of limiting damage to any other system that might try to rely on an altered copy of the root.

#### **6. References**

##### **6.1. Normative References**

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.



[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.

## **6.2. Informative References**

[Manning2013]

Manning, W., "Client Based Naming", 2013, <[http://www.sfc.wide.ad.jp/dissertation/bill\\_e.html](http://www.sfc.wide.ad.jp/dissertation/bill_e.html)>.

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", [RFC 8198](#), DOI 10.17487/RFC8198, July 2017, <<https://www.rfc-editor.org/info/rfc8198>>.

## **Appendix A. Current Sources of the Root Zone**

The root zone can be retrieved from anywhere as long as it comes with all the DNSSEC records needed for validation. Currently, one can get the root zone from ICANN by zone transfer (AXFR) over TCP from DNS servers at `xfr.lax.dns.icann.org` and `xfr.cjr.dns.icann.org`. One can also get the root zone from ICANN as a text file over HTTP (not HTTPS) at <<http://http.1.root-servers.org/root.txt>>.

Currently, the root can also be retrieved by AXFR over TCP from the following root server operators:

- o `b.root-servers.net`
- o `c.root-servers.net`
- o `d.root-servers.net`
- o `f.root-servers.net`
- o `g.root-servers.net`
- o `k.root-servers.net`

It is crucial to note that none of the above services are guaranteed to be available. It is possible that ICANN or some of the root server operators will turn off the AXFR capability on the servers listed above. Using AXFR over TCP to addresses that are likely to be anycast (as the ones above are) may conceivably have transfer problems due to anycast, but current practice shows that to be unlikely.



To repeat the requirement from earlier in this document: if the contents of the zone cannot be refreshed before the expire time, the server MUST return a SERVFAIL error response for all queries until the zone can be successfully be set up again.

### **A.1. Root Zone Services**

At the time that this document is published, there is one root zone service that is active, and one that has been announced as in the planning stages. This section describes all known active services.

LocalRoot (<<https://localroot.isi.edu/>>) is an experimental service that embodies many of the ideas in this document. It distributes the root zone by AXFR, and also offers DNS NOTIFY messages when the LocalRoot system sees that the root zone has changed.

## **Appendix B. Example Configurations of Common Implementations**

This section shows fragments of configurations for some popular recursive server software that is believed to correctly implement the requirements given in this document. The examples have been updated since the publication of [RFC 7706](#).

The IPv4 and IPv6 addresses in this section were checked recently by testing for AXFR over TCP from each address for the known single-letter names in the root-servers.net zone.

### **B.1. Example Configuration: BIND 9.12**

BIND 9.12 acts both as a recursive resolver and an authoritative server. Because of this, there is "fate-sharing" between the two servers in the following configuration. That is, if the root server dies, it is likely that all of BIND is dead.

Note that a future version of BIND will support a much more robust method for creating a local mirror of the root or other zones; see [Appendix B.3](#).

Using this configuration, queries for information in the root zone are returned with the AA bit not set.

When slaving a zone, BIND 9.12 will treat zone data differently if the zone is slaved into a separate view (or a separate instance of the software) versus slaved into the same view or instance that is also performing the recursion.

Validation: When using separate views or separate instances, the DS records in the slaved zone will be validated as the zone data is





accessed by the recursive server. When using the same view, this validation does not occur for the slaved zone.

Caching: When using separate views or instances, the recursive server will cache all of the queries for the slaved zone, just as it would using the traditional "root hints" method. Thus, as the zone in the other view or instance is refreshed or updated, changed information will not appear in the recursive server until the TTL of the old record times out. Currently, the TTL for DS and delegation NS records is two days. When using the same view, all zone data in the recursive server will be updated as soon as it receives its copy of the zone.



```
view root {
    match-destinations { 127.12.12.12; };
    zone "." {
        type slave;
        file "rootzone.db";
        notify no;
        masters {
            199.9.14.201;      # b.root-servers.net
            192.33.4.12;       # c.root-servers.net
            199.7.91.13;       # d.root-servers.net
            192.5.5.241;       # f.root-servers.net
            192.112.36.4;      # g.root-servers.net
            193.0.14.129;      # k.root-servers.net
            192.0.47.132;      # xfr.cjr.dns.icann.org
            192.0.32.132;      # xfr.lax.dns.icann.org
            2001:500:200::b;    # b.root-servers.net
            2001:500:2::c;     # c.root-servers.net
            2001:500:2d::d;    # d.root-servers.net
            2001:500:2f::f;    # f.root-servers.net
            2001:500:12::d0d;  # g.root-servers.net
            2001:7fd::1;      # k.root-servers.net
            2620:0:2830:202::132; # xfr.cjr.dns.icann.org
            2620:0:2d0:202::132; # xfr.lax.dns.icann.org
        };
    };
};

view recursive {
    dnssec-validation auto;
    allow-recursion { any; };
    recursion yes;
    zone "." {
        type static-stub;
        server-addresses { 127.12.12.12; };
    };
};
```

## **B.2. Example Configuration: Unbound 1.8**

Similar to BIND, Unbound starting with version 1.8 can act both as a recursive resolver and an authoritative server.



```
auth-zone:
  name: "."
  master: 199.9.14.201      # b.root-servers.net
  master: 192.33.4.12       # c.root-servers.net
  master: 199.7.91.13       # d.root-servers.net
  master: 192.5.5.241       # f.root-servers.net
  master: 192.112.36.4      # g.root-servers.net
  master: 193.0.14.129      # k.root-servers.net
  master: 192.0.47.132      # xfr.cjr.dns.icann.org
  master: 192.0.32.132      # xfr.lax.dns.icann.org
  master: 2001:500:200::b    # b.root-servers.net
  master: 2001:500:2::c     # c.root-servers.net
  master: 2001:500:2d::d    # d.root-servers.net
  master: 2001:500:2f::f    # f.root-servers.net
  master: 2001:500:12::d0d  # g.root-servers.net
  master: 2001:7fd::1       # k.root-servers.net
  master: 2620:0:2830:202::132 # xfr.cjr.dns.icann.org
  master: 2620:0:2d0:202::132 # xfr.lax.dns.icann.org
  fallback-enabled: yes
  for-downstream: no
  for-upstream: yes
```

### **[B.3.](#) Example Configuration: BIND 9.14**

BIND 9.14 (which, at the time of publication of this document is a future release) can set up a local mirror of the root zone with a small configuration option:

```
zone "." {
    type mirror;
};
```

The simple "type mirror" configuration for the root zone works for the root zone because a default list of primary servers for the IANA root zone is built into BIND 9.14. In order to set up mirroring of any other zone, an explicit list of primary servers needs to be provided.

See the documentation for BIND 9.14 (when it is released) for more detail about how to use this simplified configuration

### **[B.4.](#) Example Configuration: Unbound 1.9**

Recent versions of Unbound have a "auth-zone" feature that allows local mirroring of the root zone. Configuration looks like:



```
auth-zone:
  name: "."
  master: "b.root-servers.net"
  master: "c.root-servers.net"
  master: "d.root-servers.net"
  master: "f.root-servers.net"
  master: "g.root-servers.net"
  master: "k.root-servers.net"
  fallback-enabled: yes
  for-downstream: no
  for-upstream: yes
  zonefile: "root.zone"
```

### **B.5. Example Configuration: Knot Resolver**

Knot Resolver uses its "prefill" module to load the root zone information. This is described at <<https://knot-resolver.readthedocs.io/en/stable/modules.html#root-on-loopback-rfc-7706>>.

### **B.6. Example Configuration: Microsoft Windows Server 2012**

Windows Server 2012 contains a DNS server in the "DNS Manager" component. When activated, that component acts as a recursive server. DNS Manager can also act as an authoritative server.

Using this configuration, queries for information in the root zone are returned with the AA bit set.

The steps to configure DNS Manager to implement the requirements in this document are:

1. Launch the DNS Manager GUI. This can be done from the command line ("dnsmgmt.msc") or from the Service Manager (the "DNS" command in the "Tools" menu).
2. In the hierarchy under the server on which the service is running, right-click on the "Forward Lookup Zones", and select "New Zone". This brings up a succession of dialog boxes.
3. In the "Zone Type" dialog box, select "Secondary zone".
4. In the "Zone Name" dialog box, enter ".".
5. In the "Master DNS Servers" dialog box, enter "b.root-servers.net". The system validates that it can do a zone transfer from that server. (After this configuration is





completed, the DNS Manager will attempt to transfer from all of the root zone servers.)

6. In the "Completing the New Zone Wizard" dialog box, click "Finish".
7. Verify that the DNS Manager is acting as a recursive resolver. Right-click on the server name in the hierarchy, choosing the "Advanced" tab in the dialog box. See that "Disable recursion (also disables forwarders)" is not selected, and that "Enable DNSSEC validation for remote responses" is selected.

#### Acknowledgements

The authors fully acknowledge that running a copy of the root zone on the loopback address is not a new concept, and that we have chatted with many people about that idea over time. For example, Bill Manning described a similar solution to the problems in his doctoral dissertation in 2013 [[Manning2013](#)].

Evan Hunt contributed greatly to the logic in the requirements. Other significant contributors include Wouter Wijngaards, Tony Hain, Doug Barton, Greg Lindsay, and Akira Kato. The authors also received many offline comments about making the document clear that this is just a description of a way to operate a root zone on the same host, and not a recommendation to do so.

People who contributed to this update to [RFC 7706](#) include: Florian Obser, nusenu, Wouter Wijngaards, [[ others go here ]].

#### Authors' Addresses

Warren Kumari  
Google

Email: [Warren@kumari.net](mailto:Warren@kumari.net)

Paul Hoffman  
ICANN

Email: [paul.hoffman@icann.org](mailto:paul.hoffman@icann.org)

