This FAQ has been static for many years now. Consider it a historic document. The contents should mostly still be valid.

Table of Contents

Why this FAQ?

Dan's documentation initially had no recipes for common problems. In the mean time he has greatly expanded his documentation and I suggest you go there first and read through all the documentation.

This FAQ is unofficial and is meant to contain other, more exotic questions, too, like questions about IPv6 and how to have a dnscache and a tinydns on the same IP.

How do I contribute to this FAQ?

Just write an email to felix-dnsfaq@fefe.de. Please don't just send questions, send answers. I am not an automated help desk for djbdns, although I do offer commercial support.

Oh, my name is Felix von Leitner, and for now, I am the sole maintainer of this FAQ. Feedback is welcome. The email address is not a link (to discourage dumb spam bots from finding it).

What is djbdns?

djbdns is a DNS package by DJ Bernstein that consists of

1. dnscache, a caching DNS resolver
2. tinydns, a DNS server
3. pickdns, a load-balancing DNS server
4. walldns, a reverse DNS wall
5. axfrdns, an implementation of an area transfer server
6. a few DNS clients and troubleshooting tools

Please note that many people use confusing terminology because of BIND, which integrates a caching DNS resolver and a DNS server into one package, making people say "DNS server" when they are really talking about a "DNS resolver".

djbdns separates the resolver and the server, which provides enhanced security and flexibility but when people try to run both on the same IP address (as is default mode of operation for BIND), it will fail silently.

How do I run the resolver and the server on the same IP?

Well, you don't. You run dnscache on your ethernet IP and tinydns on 127.0.0.1 and then tell dnscache to ask tinydns.

You may run both dnscache and tinydns on the same IP and port, because they are UDP servers. Your operating system will then forward each DNS query to **one of them**, and you should not assume a pattern to the selection. tinydns will just drop queries it can't answer, so every query that is forwarded to tinydns will be dropped.

The client will ask the same query again after a few seconds, so to the user this will look like the network is slow. If the next query is by chance again forwarded to tinydns, the timeout will appear longer. If you try to resolve data from tinydns but reach dnscache, you will get a server failure.

You have to decide now: do you need the caching resolver to be accessible by other computers? If so, bind dnscache to your external IP and tinydns to 127.0.0.1 and tell dnscache to consult tinydns for the relevant addresses. See also Split Horizon.

This setup has the drawback that dnscache will never return authoritative answers (this is a bit in DNS answer packets). This won't matter for most clients, but technically it is not correct. For example, your domain registry might complain about this. (Thanks to Brian Kifiak for pointing this out) DON'T DO THIS ON PRODUCTION NAME SERVERS! Proper resolvers (i.e. dnscache) will not accept non-authoritative answers.

The best solution is to use IP aliases, for example on Linux you can simply assign IPs to "lo:1", "lo:2" and so on.

Can I use djbdns without daemontools?

Yes.

Many people are intimidated by the daemontools setup instructions and are thus afraid to try out djbdns. DJ Bernstein does not advocate this, but you can use the different DNS servers and dnscache without svscan and supervise. This FAQ entry only says how to avoid running "svscan", i.e. you still have to install the daemontools package for the envuidgid and softlimit programs which are generally useful and used by the djbdns invocation scripts.

Just follow the setup instructions and you will have a shell script `/service/tinydns/run`. This shell script will start the service (but it does not fork into the background, so for a System V style init script you would use something like this:

```
case "$1" in
  start)        echo "Starting tinydns."
                exec /service/tinydns/run &
                echo $! > /var/run/tinydns.pid
                ;;
  stop)         echo "Shutting down tinydns."
                kill `cat /var/run/tinydns.pid`
                rm -f /var/run/tinydns.pid
                ;;
  *)            echo "Usage: $0 {start|stop}"
                exit 1
```

```
esac
exit 0
```

Tobias Oetiker has contributed an extended init script.

If you don't want a `/service` directory, replace `/service/tinydns` with the path to your tinydns installation. Replace tinydns with dnscache, walldns etc for the other services.

Please note that the daemontools really are very helpful and you should at least evaluate the /service concept for a few days before dismissing running svscan off-hand.

How does "split horizon" DNS work with tinydns?

Split horizon DNS means that you have one machine that answers internal and external DNS queries, i.e. internal hosts can look up hosts in the Internet and external hosts can look up your DNS zone.

The catch is that

1. external hosts should only see part of your DNS
2. external hosts should talk to tinydns, not dnscache (i.e. don't answer DNS queries for other people)

For split horizon DNS, you normally have two network interfaces in your host. Let's call them eth0 and eth1. eth0 is the external interface and has the IP 204.71.200.33, eth1 is the internal interface and has the IP 10.1.2.3. Your domain is called `yahoo.com`.

Now, to get the desired functionality, you need to run two copies of tinydns. One will serve the public version of your zone to the world, so obviously you use your external IP 204.71.200.33 for it.

```
# tinydns-conf tinydns dnslog /var/tinydns-public 204.71.200.33
```

To offer a caching DNS resolver to your LAN, you install a copy of dnscache to use your internal IP 10.1.2.3.

```
# dnscache-conf dnscache dnslog /var/dnscache 10.1.2.3
```

dnscache will only answer queries from 127.0.0.1 by default, so you will have to tell it otherwise:

```
# touch /var/dnscache/root/ip/10
```

This will allow dnscache queries from 10.* IPs. Since this network interface is in your LAN, outsiders should not be able to send queries to it (if you have a firewall and your setup is kosher).

The second copy of tinydns will serve the internal data of your zone to your LAN, and since both IPs are already taken, you tell it to use the loopback IP 127.0.0.1.

```
# tinydns-conf tinydns dnslog /var/tinydns-private 127.0.0.1
```

Now, all you have to do is to tell your dnscache that it should consult your local tinydns when it is asked to resolve your internal yahoo.com addresses. That's quite easy:

```
# echo 127.0.0.1 > /var/dnscache/root/servers/yahoo.com
# echo 127.0.0.1 > /var/dnscache/root/servers/10.in-addr.arpa
```

The second line makes sure that reverse lookups will also be forwarded to your internal tinydns.

Please note that you should not do split horizon DNS on one machine for security reasons, because an attacker who breaks into your DNS machine will have access to your LAN via the eth1. It is a better idea to configure a tinydns on a bastion host in your firewall perimeter network and have a dnscache/tinydns combo on another host in your LAN.

[Benett Todd mailed this description of split horizon DNS to the mailing list. You might find it easier to understand.]

How does "split horizon" DNS work with only one network interface?

For security reasons you should not use split horizon DNS with only one network interface. If you want it nonetheless, you need to bring up a second network interface, which is called "dummy interface" or "alias interface", depending on your operating system terminology. Please consult your OS documentation. Then simply use the same configuration as above.

How do I tell dnscache to accept queries from 10.2.3.64-127 (CIDR)?

Just chdir to `/service/dnscache/root/ip` and create `10.2.3.64`, `10.2.3.65`, etc. You can do this painlessly with a script like this (assuming a Bourne shell)

```
cd /service/dnscache/root/ip
touch 10.2.3.64
for i in `awk 'BEGIN { for( i=65; i<=127; i++ ) print i }'`; do
    ln 10.2.3.64 10.2.3.$i
done
```

Using ln instead of just touching all the files saves inodes. Thanks to Mate Wierdl for posting this suggestion on the mailing list.

How do I convert a BIND zone file to tinydns?

Allow zone transfer in bind and use `axfr-get` from the djbdns distribution to copy the data into tinydns format.

Then use Bennett Todd's scripts, in particular tinydns-data-compactor and tinydns-data-beautify to clean up the data. [local copy]

If you plan to do this regularly, you might want to install axfr-get on the BIND machine locally and use rsync over ssh to copy the converted zones to the tinydns secondary. That improves security and performance greatly.

Thanks to Raul Miller for this answer!

How do I tell BIND slaves to update their zones?

tinydns.org has a perl script at http://tinydns.org/dnsnotify to send DNS NOTIFY packets. [local copy]

My tinydns is running as a secondary for a BIND that sends me NOTIFY messages. How do I catch those and respond?

tinydns logs NOTIFY queries like this:

```
HexSourceIP:HexSourcePort:QID I 0006 domain.xy
```

Set up log/run to single out these lines (multilog can do that) and act accordingly. Or take a look at this perl script someone posted to the mailing list once, which is a wrapper for multilog.

BIND 9 won't accept AXFRs from djbdns!

Greg Hewgill posted a patch to djbdns-1.04 to the mailing list. I heard the BIND people fixed this in recent BIND 9 versions (it was, of course, a BIND bug).

Where is the RPM?

There are several unofficial ones from several sources (usually linked to from tinydns.org). However, running a name server is only for seasoned administrators.

Compiling djbdns yourself is trivial. If you feel you are not up to it, please reconsider if you should be administrating a name server. An incorrectly setup name server can do bad things to many people.

Tinydns does not answer at all when someone lamely delegates to it?

Yes. You can add this line to your `data` file to simulate BIND behaviour:

```
&::a.root-servers.net
```

The tinydns zone file looks ugly!

First, it's not a zone file. data contains all your records, the zones are implicit.

Second, uglyness lies in the eye of the beholder. When first looking at the BIND zone files, most people are repulsed, too.

This chart may help you when reading data:

This	Creates This
.	SOA, NS, A
&	NS, A
@	MX, A
=	PTR, A
+	A
'	TXT
^	PTR
C	CNAME
Z	SOA
%	(client location conditional expression, does not create any records)
#	(comment, does not create any records)
-	(used to temporarily disable A records, does not create any records)

:	User-defined
6	AAAA, PTR (with my patch)
3	AAAA (with my patch)

But see also http://cr.yp.to/djbdns/frombind.html!

See Also

1. Frequently Given Answers written by JdeBP contain several answers about DNS.
2. a package with man pages for djbdns
3. official djbdns home page
4. unofficial djbdns home page
5. my IPv6 patches for djbdns
6. "life with djbdns"
7. http://koeln.ccc.de/~drt/dnscachefortheclueless.html contains a script that automatically downloads and installs djbdns and daemontools, starts a dnscache process and fixes /etc/resolv.conf