Basic DIG response

dig www.isc.org

```
; <<>> DiG 9.9.4-RedHat-9.9.4-38.el7_3.2 <<>> www.isc.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13564
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.isc.org.                   IN    A

;; ANSWER SECTION:
www.isc.org.        60    IN    A     149.20.64.69

;; AUTHORITY SECTION:
isc.org.            7200  IN    NS    sfba.sns-pb.isc.org.
isc.org.            7200  IN    NS    ns.isc.afilias-nst.info.
isc.org.            7200  IN    NS    ord.sns-pb.isc.org.
isc.org.            7200  IN    NS    ams.sns-pb.isc.org.

;; ADDITIONAL SECTION:
ord.sns-pb.isc.org.      86400 IN    A     199.6.0.30
ord.sns-pb.isc.org.      86400 IN    AAAA  2001:500:71::30
ams.sns-pb.isc.org.      86400 IN    A     199.6.1.30
ams.sns-pb.isc.org.      86400 IN    AAAA  2001:500:60::30
sfba.sns-pb.isc.org.     86400 IN    A     149.20.64.3
sfba.sns-pb.isc.org.     86400 IN    AAAA  2001:4f8:0:2::19

;; Query time: 54 msec
;; SERVER: 10.20.6.12#53(10.20.6.12)
;; WHEN: Mon Jun 19 21:16:53 UTC 2017
;; MSG SIZE  rcvd: 287
```

Verifying DNSSEC

How to query and verify DNSSEC
A domain with DNSSEC enabled will have the *ad* flag set in the response.

```
; <<>> DiG 9.8.3-P1 <<>> +dnssec www.appfolio.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40107
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;www.appfolio.com.       IN    A

;; ANSWER SECTION:
www.appfolio.com.  528   IN    A     162.216.20.141

;; Query time: 24 msec
;; SERVER: 2001:578:3f::30#53(2001:578:3f::30)
;; WHEN: Thu Jul 27 20:53:55 2017
;; MSG SIZE  rcvd: 61
```

+dnssec flag asks the server to validate the zone data.
If the zone data was incorrect, the server would return a SERVFAIL error.
DNSSEC can be enabled by adding the following lines to the named.conf file,

```
dnssec-enable yes;
Dnssec-validation yes;
```

DIG Options:

dig can be used to perform any standard dns query.

```
dig google.com A +short
74.125.127.113
74.125.127.139
74.125.127.138
74.125.127.102
74.125.127.100
74.125.127.101
dig google.com MX +short
10 aspmx.l.google.com.
40 alt3.aspmx.l.google.com.
50 alt4.aspmx.l.google.com.
20 alt1.aspmx.l.google.com.
30 alt2.aspmx.l.google.com.
dig google.com NS +short
ns2.google.com.
ns1.google.com.
ns4.google.com.
ns3.google.com.
dig google.com ANY +short
40 alt3.aspmx.l.google.com.
30 alt2.aspmx.l.google.com.
20 alt1.aspmx.l.google.com.
50 alt4.aspmx.l.google.com.
10 aspmx.l.google.com.
74.125.127.100
74.125.127.102
74.125.127.113
```

```
74.125.127.138
74.125.127.139
74.125.127.101
ns4.google.com.
ns2.google.com.
ns1.google.com.
ns3.google.com.
```

You can also do a DNS zone transfer (if the domain allows that),
dig xx.yourdomain.com AXFR

+short = short answer
+noall = turn off all results
+multiline = gives SOA records in a verbose multiline format with human-readable comments.
-x = Do a reverse lookup (gets the hostname associated with the IP address)
+nssearch = displays SOA information

```
dig +nocmd ogi.edu any +multiline +noall +answer
ogi.edu.           3600 IN     A 137.53.244.59
ogi.edu.           86400 IN SOA DNS0.ohsu.edu. netcomm.ohsu.edu. (
                        53         ; serial
                        3600       ; refresh (1 hour)
                        3600       ; retry (1 hour)
                        604800     ; expire (1 week)
                        600        ; minimum (10 minutes)
                        )
ogi.edu.           86400 IN NS DNS4.ohsu.edu.
ogi.edu.           86400 IN NS DNS1.ohsu.edu.
ogi.edu.           86400 IN NS DNS3.ohsu.edu.
```

```
dig -x 74.125.127.103 +short
ov-in-f103.1e100.net.
dig @ns2.google.com www.google.com

; <<>> DiG 9.9.4-RedHat-9.9.4-38.el7_3.2 <<>> @ns2.google.com www.google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19200
;; flags: qr aa rd; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.google.com.                IN    A

;; ANSWER SECTION:
www.google.com.          300    IN    A     74.125.127.105
```

```
www.google.com.          300    IN     A      74.125.127.147
www.google.com.          300    IN     A      74.125.127.106
www.google.com.          300    IN     A      74.125.127.104
www.google.com.          300    IN     A      74.125.127.103
www.google.com.          300    IN     A      74.125.127.99

;; Query time: 29 msec
;; SERVER: 216.239.34.10#53(216.239.34.10)
;; WHEN: Mon Jun 19 21:37:44 UTC 2017
;; MSG SIZE  rcvd: 128
```

dig -f /Path_to_file

Any DNS server connected to the Internet is likely to have a copy of the InterNIC's `named.root` file that lists the root name servers for the entire Internet. You can always download that file in the boring way from the InterNIC's ftp server. Or, in a true build-it-yourself fashion, you can build it with `dig`.

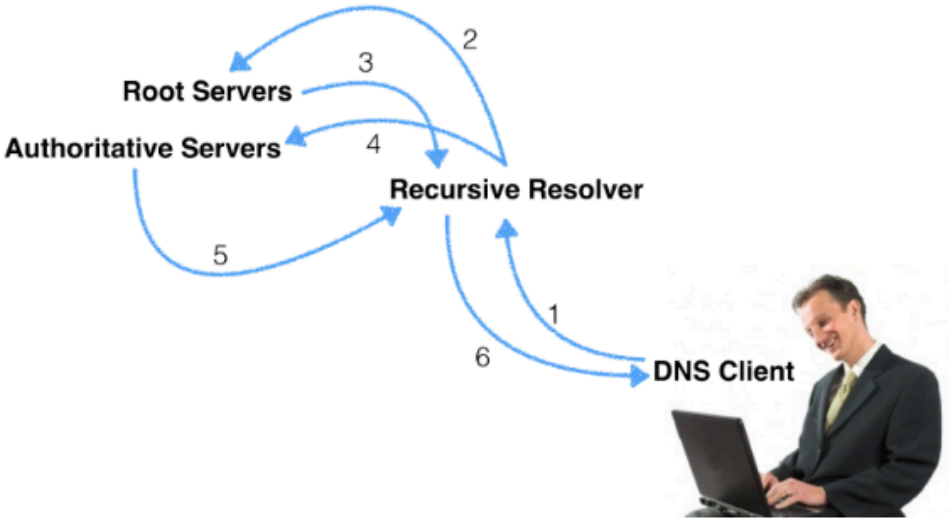
```
# compare with ftp://ftp.internic.net/domain/named.root
dig +nocmd . NS +noall +answer +additional
```

DIG Trace

```
dig something.com +trace
```

1. You as the DNS client (or stub resolver) query your recursive resolver for www.example.com.
2. Your recursive resolver queries the root name server for www.example.com.
3. The root name server refers your recursive resolver to the .com Top-Level Domain (TLD) authoritative server.
4. Your recursive resolver queries the .com TLD authoritative server for www.example.com.
5. The .com TLD authoritative server refers your recursive server to the authoritative servers for example.com.
6. Your recursive resolver queries the authoritative servers for www.example.com, and receives 1.2.3.4 as the answer.
7. Your recursive resolver caches the answer for the duration of the time-to-live (TTL) specified on the record, and returns it to you.

The above process basically looks like this:




```
dig www.google.com +trace
***********
; <<>> DiG 9.9.4-RedHat-9.9.4-38.el7_3.2 <<>> www.google.com +trace
;; global options: +cmd
.                   425    IN     NS     k.root-servers.net.
.                   425    IN     NS     d.root-servers.net.
.                   425    IN     NS     f.root-servers.net.
.                   425    IN     NS     i.root-servers.net.
.                   425    IN     NS     c.root-servers.net.
.                   425    IN     NS     l.root-servers.net.
```

```
.                   425   IN    NS    j.root-servers.net.
.                   425   IN    NS    b.root-servers.net.
.                   425   IN    NS    h.root-servers.net.
.                   425   IN    NS    m.root-servers.net.
.                   425   IN    NS    g.root-servers.net.
.                   425   IN    NS    e.root-servers.net.
.                   425   IN    NS    a.root-servers.net.
.                   425   IN    RRSIG NS    8    0    518400    20170626170000
20170613160000                              14796                              .
gtEwIaAnir0PwKAo0JLp3U2ANokZKRr+ZoOYbDK6PK+vDWw3dDdlzZ1W
cyCpne/loaxbCbQKTXobsLvhuCmLQisCmjl2XNEs0aDhkhVWqEffi2up
SPzRM5CdrNv9ZKqMrxAjzeeKdHCRvHrhFph8pvUHIFhNIjGk01xbhyrJ
EJo16LTqny3v/dcnyif5HgiB64iO7icJztCA8mAgbdFj++7tcXte6ieB
3eW+0fxjwgI9lpZwhWNz9k+HbPI6XEz/UesRPZjS3ECg4fx/ka4zFge+
a6MUMJCQf69M0Oz+lOGO6/jm7r/sDfxsoVwSAHentjPrI9hA2XhbmnAH TPArvQ==
********
;; Received 1097 bytes from 10.20.6.12#53(10.20.6.12) in 2 ms

com.                172800IN    NS    a.gtld-servers.net.
com.                172800IN    NS    b.gtld-servers.net.
com.                172800IN    NS    c.gtld-servers.net.
com.                172800IN    NS    d.gtld-servers.net.
com.                172800IN    NS    e.gtld-servers.net.
com.                172800IN    NS    f.gtld-servers.net.
com.                172800IN    NS    g.gtld-servers.net.
com.                172800IN    NS    h.gtld-servers.net.
com.                172800IN    NS    i.gtld-servers.net.
com.                172800IN    NS    j.gtld-servers.net.
com.                172800IN    NS    k.gtld-servers.net.
com.                172800IN    NS    l.gtld-servers.net.
com.                172800IN    NS    m.gtld-servers.net.
com.                86400 IN    DS    30909                8                2
E2D3C916F6DEEAC73294E8268FB5885044A833FC5459588F4A9184CF C41A5766
com.                86400 IN    RRSIG DS    8    1    86400    20170702170000
20170619160000                              14796                              .
M4sA/cDWGMtToCXD3cuNB2LRvbujjnlqdXmegY0pOx0O49gwNikDuKd8
BpGcV1lMAlRmZcbBl8Fm6N8sibpROZL1MkO72BlZhPpg7l4lWlIYHHut
BSqHuidis18PUaW/TzR5qymhVciGUs4T/pgzZqRbmmDw3QQnL6t6XK0U
DBmAHUw+cjjMkZE3tZ8vKmq9WTQnyJWMoqx5RowHLlRQ4wRKEMeelzEu
8rDqKGYUWAN4yBcTpRQpuuZjlZuBEMns1LM6MLsPxe6XiL6AfDqsDGcI
9uUKfSYbtqP3sD3MY5ahEEfsrKQOY09rZGKGJs8QD4EziFAVN5LdYUk9 /zE54A==
******
;; Received 866 bytes from 198.97.190.53#53(h.root-servers.net) in 488 ms

google.com.         172800IN    NS    ns2.google.com.
google.com.         172800IN    NS    ns1.google.com.
google.com.         172800IN    NS    ns3.google.com.
google.com.         172800IN    NS    ns4.google.com.
CK0POJMG874LJREF7EFN8430QVIT8BSM.com.   86400    IN    NSEC3    1    1    0    -
CK0Q1GIN43N1ARRC9OSM6QPQR81H5M9A NS SOA RRSIG DNSKEY NSEC3PARAM
```

```
CK0POJMG874LJREF7EFN8430QVIT8BSM.com.  86400  IN  RRSIG  NSEC3  8  2  86400
20170626045030             20170619034030             27302             com.
CG45EFSkPOKZGrutc7F0yjAlQ1OxTo3nkDAw3J9uyg5MJBVJoTly/kck
ubQZBZR7IL9RsMHTJVfDo5CpVVLmiRbwBhT7bdkS2Cll9CHvX+xo+l6e
qHtiFKwhjrlQguyW8gwNGGkZsfyI2MBZsQ7i1nULJ3x2vD0I4oeXzGGQ xz0=
S848JI1TS2RCEPV5SPG2RJA2T711BO8H.com.   86400   IN   NSEC3   1   1   0   -
S84C439C9HACCNUVH6CBPPTUS93VLTUG NS DS RRSIG
S848JI1TS2RCEPV5SPG2RJA2T711BO8H.com.  86400  IN  RRSIG  NSEC3  8  2  86400
20170623045130             20170616034130             27302             com.
E1ycWmoJxXOhHRLDddmMSwaIdE7BPg/fCnCRBTTBFhZoTR/S3jAm0bDc
GCzT2Inr7DwEW/UqSGx49VMY4z74KVUzorFxZlZumawiOVrXV/1T8hqS
d2tF/56Hu3bDnQJrhTmSKR67GUnAApyqHyjy2CxfLXjJN5lkVeBkuTYa s80=
**********
;; Received 664 bytes from 192.42.93.30#53(g.gtld-servers.net) in 100 ms

www.google.com.          300    IN     A      74.125.127.99
www.google.com.          300    IN     A      74.125.127.104
www.google.com.          300    IN     A      74.125.127.147
www.google.com.          300    IN     A      74.125.127.103
www.google.com.          300    IN     A      74.125.127.105
www.google.com.          300    IN     A      74.125.127.106
**********
;; Received 128 bytes from 216.239.36.10#53(ns3.google.com) in 23 ms
*****
```

Verifying DNS mappings

An improperly configured DNS setup can be really annoying. You want to make sure that your mappings work both ways:

1. Each hostname should resolve to an address, and that address ought to resolve back to the proper hostname.
2. If an address on your subnet(s) has been assigned a reverse pointer to a hostname, that hostname ought to point back to the original address.

There are exceptions to those two rules, of course. A CNAME will resolve to another hostname first, and only then to an address. Sometimes multiple hostnames will point to the same address, but that address will have only one reverse pointer.

Still, it's good to know that your basic mappings work as expected.

You can script such a test if you build a file containing your known hostnames. The example script below is pretty simple; it will break if fed a CNAME, and it'll report a failure somewhere if multiple hostnames point to the same address. Let's assume the file containing your hostnames is named `named-hosts`.

```
#!/bin/bash
#
# test DNS forward- and reverse-mapping
#

# edit this variable to reflect local class C subnet(s)
NETS="192.168.1 192.168.2"

# Test name to address to name validity
echo
echo -e "\tname -> address -> name"
echo '---------------------------------'
while read H; do
  ADDR=$(dig $H +short)
  if test -n "$ADDR"; then
    HOST=$(dig -x $ADDR +short)
    if test "$H" = "$HOST"; then
      echo -e "ok\t$H -> $ADDR -> $HOST"
    elif test -n "$HOST"; then
      echo -e "fail\t$H -> $ADDR -> $HOST"
    else
      echo -e "fail\t$H -> $ADDR -> [unassigned]"
    fi
  else
    echo -e "fail\t$H -> [unassigned]"
  fi
done < named-hosts

# Test address to name to address validity
echo
echo -e "\taddress -> name -> address"
echo '-------------------------------------'
for NET in $NETS; do
  for n in $(seq 1 254); do
    A=${NET}.${n}
    HOST=$(dig -x $A +short)
    if test -n "$HOST"; then
      ADDR=$(dig $HOST +short)
      if test "$A" = "$ADDR"; then
        echo -e "ok\t$A -> $HOST -> $ADDR"
      elif test -n "$ADDR"; then
        echo -e "fail\t$A -> $HOST -> $ADDR"
      else
        echo -e "fail\t$A -> $HOST -> [unassigned]"
      fi
    fi
  done
done
```

Troubleshooting with DIG

One of the most powerful tools to have at the command line for working with DNS records is the dig command.
dig (Domain Information Groper) is the swiss army knife of DNS administrators. Let's have a look at how we can use it to troubleshoot the most common problems.
Standard output in dig
By default, the output generated by dig is pretty verbose. Here's an example where we look up the MX records for the our domain hostdns.com.

```
$ dig hostdns.com MX

; <<>> DiG 9.10.6 <<>> hostdns.com MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34904
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;hostdns.com. IN MX

;; ANSWER SECTION:
hostdns.com. 3599 IN MX 1 aspmx.l.google.com.
hostdns.com. 3599 IN MX 10 aspmx2.googlemail.com.
hostdns.com. 3599 IN MX 10 aspmx3.googlemail.com.
hostdns.com. 3599 IN MX 5 alt1.aspmx.l.google.com.
hostdns.com. 3599 IN MX 5 alt2.aspmx.l.google.com.

;; Query time: 47 msec
;; SERVER: 192.168.126.5#53(192.168.126.5)
;; WHEN: Thu Oct 17 08:55:03 CEST 2019
;; MSG SIZE rcvd: 170
```

Wow, there's a lot in there!
Let's unpack this and see what's relevant.
Querying a domain and a specific resource type
Our command line tool was `dig hostdns.com MX`. It follows the format of `dig <domain> <resource-type>`. A resource type is any of the possible DNS records, like `A`, `AAAA`, `CNAME`, …
Our output section reflects the query we just asked. It tells us a lot.

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34904
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1
```

In the very first block, we see the query status: we received a valid reply (`NOERROR`). This could also have been a `NXDOMAIN` to indicate the (sub)domain name we were query'ing did not exist.
Below that are a series of interesting _flags_. The most common ones you'll find are:
– `qr`: _query response_, indicates that what is shown next is the response to our DNS query (_`dig` can also show us the query, not just the response_).
– `rd`: _recursion desired_, we asked for a specific query, and our nameservers are allowed to forward this query to their upstream nameservers if they don't have the response.
– `ra`: _recursion available_, the nameserver that received our query indicated that it _can_ do a recursive lookup (this can be explicitly denied in the config)
The next most important section is the actual reply. In our example, it looked like this.
;; ANSWER SECTION:
hostdns.com. 3599 IN MX 1 aspmx.l.google.com.
hostdns.com. 3599 IN MX 10 aspmx2.googlemail.com.
hostdns.com. 3599 IN MX 10 aspmx3.googlemail.com.
hostdns.com. 3599 IN MX 5 alt1.aspmx.l.google.com.
hostdns.com. 3599 IN MX 5 alt2.aspmx.l.google.com.
This indicates that our query (the `MX` records of the domain `hostdns.com`) could be found. There were 5 results, each with a different MX priority.
The format of the reply is as follows:
<domain> <TTL> IN <type> <answer>
The `TTL` (Time To Live) indicates how long the DNS response should be cached (or considered "valid"). The data in `<answer>` is what we're actually after and contains the mailservers responsible for our domain.
Below that, we see some debug & diagnostic data:
;; Query time: 47 msec
;; SERVER: 192.168.126.5#53(192.168.126.5)
The nameserver took 47msec to reply. Since we didn't explicitly give it a nameserver to query, it used our OS default – in this case the query was answered by a nameserver on IP `192.168.126.5`.
Querying the full hierarchy of nameservers
Whenever you're debugging DNS issues, it's wise to do a full "trace" of the records. DNS works hierarchically, it has a set of root nameservers, below that some TLD-specific ones, then nameservers for your own domain, … The _tree_ for `hostdns.com` looks like this.
root nameservers:
a.root-servers.net.
b.root-servers.net.
...
_ .com nameservers:
a.gtld-servers.net.
b.gtld-servers.net.
...
_ hostdns.com nameservers:
ns-47.awsdns-05.com.
ns-934.awsdns-52.net.

We can see this information too when we query using the `+trace` flag. If we then only ask for the `NS` (nameserver) records, our response is filtered to just what we need.

```
$ dig +trace hostdns.com NS
. 84482 IN NS a.root-servers.net.
. 84482 IN NS b.root-servers.net.
[...]

com. 172800 IN NS a.gtld-servers.net.
com. 172800 IN NS b.gtld-servers.net.
[...]

hostdns.com. 172800 IN NS ns-47.awsdns-05.com.
hostdns.com. 172800 IN NS ns-934.awsdns-52.net.
[...]
```

Why is this useful? Your local nameserver (or your ISP's nameservers) might be caching old records, using `+trace` we force each nameserver in the tree to reply on their own. If you get the correct result with `+trace` and the wrong one without that flag, there's a good chance there's DNS caching (and long TTLs) involved and you need to wait things out. Once the TTL has expired, your DNS might be updated.

Querying a specific nameserver

Another useful trick is to query a specific nameserver. Let's say you have 4 nameservers. How do you know all 4 give the same reply? You can use the `@` flag in `dig` to query a nameserver you pick.

```
$ dig +short +noshort @ns-47.awsdns-05.com. hostdns.com A
hostdns.com. 3600 IN A 206.189.247.1
$ dig +short +noshort @ns-1693.awsdns-19.co.uk. hostdns.com A
hostdns.com. 3600 IN A 206.189.247.1
$ dig +short +noshort @ns-47.awsdns-05.com. hostdns.com A
hostdns.com. 3600 IN A 206.189.247.1
$ dig +short +noshort @ns-934.awsdns-52.net. hostdns.com A
hostdns.com. 3600 IN A 206.189.247.1
```

Notice the use of @ns-47.awsdns-05.com. in our dig command? That tells our dig client to query that specific nameserver instead.

The extra flags of +short +noshort are a useful addition, it keeps our output very short and to-the-point. No debug or verbose output, just the query response we asked for.

With these tips, you should be able to get some basic troubleshooting going. You'll be able to better understand the output of the `dig` command, see the entire _tree_ of nameservers that can reply and query specific nameservers for troubleshooting.