NAME

query, ipquery, mkhash, mkdb, mkhosts, cs, csquery, dns, dnstcp, dnsquery, dnsdebug, inform – network database

SYNOPSIS

`ndb/query` [`-am`] [`-f` *dbfile*] *attr value* [*rattr*]
`ndb/ipquery` *attr value rattr...*
`ndb/mkhash` *file attr*
`ndb/mkdb`
`ndb/mkhosts` [*domain* [*dbfile*]]
`ndb/cs` [`-n`] [`-f` *dbfile*] [`-x` *netmtpt*]
`ndb/csquery` [`-s`] [*server* [*addr...*]]
`ndb/dns` [`-norRs`] [`-a` *maxage*] [`-f` *dbfile*] [`-N` *target*] [`-x` *netmtpt*] [`-z` *program*]
`ndb/dnstcp` [`-rR`] [`-f` *dbfile*] [`-x` *netmtpt*] [*conn–dir*]
`ndb/dnsquery`
`ndb/dnsdebug` [`-rx`] [`-f` *dbfile*] [[`@`*server*] *domain–name* [*type*]]
`ndb/inform` [`-x` *netmtpt*]

DESCRIPTION

The network database holds administrative information used by network programs such as *dhcpd*(8), *ipconfig*(8), *con*(1), etc.

Ndb/query searches the database *dbfile* (`/lib/ndb/local` by default) for an attribute of type *attr* and value *value*. If *rattr* is not specified, all entries matched by the search are printed. If *rattr* is specified, the value of the first pair with attribute *rattr* of all the matched entries normally is printed. Under `-m` and *rattr*, the values of all pairs with a *rattr* attribute within the first matching entry are printed. Under `-a` and *rattr*, all values of pairs with a *rattr* attribute within all entries are printed.

Ndb/ipquery uses *ndbipinfo* (see *ndb*(2)) to search for the values of the attributes *rattr* corresponding to the system with entries of attribute type *attr* and value *value*.

Ndb/inform sends an RFC2136 DNS *inform* packet to a nameserver to associate the host's IPv4 address with its DNS name. This is required if the domain's nameserver is a Microsoft Windows Active Directory controller.

Database maintenance

Ndb/mkhash creates a hash file for all entries with attribute *attr* in database file *file*. The hash files are used by *ndb/query* and by the ndb library routines.

Ndb/mkdb is used in concert with *awk*(1) scripts to convert uucp systems files and IP host files into database files. It is very specific to the situation at Murray Hill.

When the database files change underfoot, *ndb/cs* and *ndb/dns* track them properly. Nonetheless, to keep the database searches efficient it is necessary to run *ndb/mkhash* whenever the files are modified. It may be profitable to control this by a frequent *cron*(8) job.

Ndb/mkhosts generates a BSD style `hosts`, `hosts.txt`, and `hosts.equiv` files from an ndb data base file specified on the command line (default `/lib/ndb/local`). For local reasons the files are called `hosts.1127`, `astro.txt`, and `hosts.equiv`.

Connection service

Ndb/cs is a server used by *dial*(2) to translate network names. It is started at boot time. It finds out what networks are configured by looking for `/net/*/clone` when it starts. It can also be told about networks by writing to `/net/cs` a message of the form:

```
add net1 net2 ...
```

Ndb/cs also sets the system name in `/dev/sysname` if it can figure it out. The options are:

`-f`	supplies the name of the data base file to use, default `/lib/ndb/local`.
`-n`	causes cs to do nothing but set the system name.
`-x`	specifies the mount point of the network.

Ndb/csquery queries *ndb/cs* to see how it resolves addresses. *Ndb/csquery* prompts for addresses and prints what *ndb/cs* returns. *Server* defaults to `/net/cs`. If any *addrs* are specified, *ndb/csquery* prints their translations and immediately exits. The exit status will be nil only if all addresses were successfully translated. The `−s` flag sets exit status without printing any results.

Domain name service

Ndb/dns serves *ndb/cs* and remote systems by translating Internet domain names. *Ndb/dns* is started at boot time. By default *dns* serves only requests written to `/net/dns`. Programs must *seek* to offset 0 before reading or writing `/net/dns` or `/net/cs`. The options are:

`−a` sets the maximum time in seconds that an unreferenced domain name will remain cached. The default is one hour (3600).

`−f` supplies the name of the data base file to use, default `/lib/ndb/local`.

`−n` whenever a DNS zone that we serve changes, send UDP NOTIFY messages to any dns slaves for that zone (see the `dnsslave` attribute below).

`−N` sets the goal for the number of domain names cached to *target* rather than the default of 8,000.

`−o` used with `−s`, `−o` causes *dns* to assume that it straddles inside and outside networks and that the outside network is mounted on `/net.alt`. Queries for inside addresses will be sent via `/net/udp` (or `/net/tcp` in response to truncated replies) and those for outside addresses via `/net.alt/udp` (or `/net.alt/tcp`). This makes *dns* suitable for serving non-Plan-9 systems in an organization with firewalls, DNS proxies, etc., particularly if they don't work very well. See 'Straddling Server' below for details.

`−r` act as a resolver only: send 'recursive' queries, asking the other servers to complete lookups. If present, /env/DNSSERVER must be a space-separated list of such DNS servers' IP addresses, otherwise optional *ndb*(6) `dns` attributes name DNS servers to forward queries to.

`−R` ignore the 'recursive' bit on incoming requests. Do not complete lookups on behalf of remote systems.

`−s` also answer domain requests sent to UDP port 53.

`−x` specifies the mount point of the network.

`−z` whenever we receive a UDP NOTIFY message, run *program* with the domain name of the area as its argument.

When the `−r` option is specified, the servers used come from the *dns* attribute in the database. For example, to specify a set of dns servers that will resolve requests for systems on the network *mh−net*:

```
ipnet=mh-net ip=135.104.0.0 ipmask=255.255.0.0
     dns=ns1.cs.bell-labs.com
     dns=ns2.cs.bell-labs.com
dom=ns1.cs.bell-labs.com ip=135.104.1.11
dom=ns2.cs.bell-labs.com ip=135.104.1.12
```

The server for a domain is indicated by a database entry containing both a *dom* and a *ns* attribute.

```
dom=
     ns=A.ROOT-SERVERS.NET
     ns=B.ROOT-SERVERS.NET
     ns=C.ROOT-SERVERS.NET
dom=A.ROOT-SERVERS.NET ip=198.41.0.4
dom=B.ROOT-SERVERS.NET ip=128.9.0.107
dom=C.ROOT-SERVERS.NET ip=192.33.4.12
```

The last three lines provide a mapping for the server names to their ip addresses. This is only a hint and will be superseded from whatever is learned from servers owning the domain.

Authoritative Name Servers

You can also serve a subtree of the domain name space from the local database. You indicate subtrees that you would like to serve by adding an `soa=` attribute to the root entry. For example, the Bell Labs CS research domain is:

```
dom=cs.bell-labs.com soa=
     refresh=3600 ttl=3600
     ns=plan9.bell-labs.com
     ns=ns1.cs.bell-labs.com
     ns=ns2.cs.bell-labs.com
```

```
		mb=presotto@plan9.bell-labs.com
		mx=mail.research.bell-labs.com pref=20
		mx=plan9.bell-labs.com pref=10
		dnsslave=nslocum.cs.bell-labs.com
		dnsslave=vex.cs.bell-labs.com
```

Here, the `mb` entry is the mail address of the person responsible for the domain (default `postmaster`). The `mx` entries list mail exchangers for the domain name and `refresh` and `ttl` define the area refresh interval and the minimum TTL for records in this domain. The `dnsslave` entries specify slave DNS servers that should be notified when the domain changes. The notification also requires the `-n` flag.

Reverse Domains

You can also serve reverse lookups (returning the name that goes with an IP address) by adding an `soa=` attribute to the entry defining the root of the reverse space.

For example, to provide reverse lookup for all addresses in starting with `135.104` or `fd00::`, *ndb* must contain a record like:

```
	dom=104.135.in-addr.arpa soa=
		dom=d.f.ip6.arpa soa=    # special case, rfc 4193
		refresh=3600 ttl=3600
		ns=plan9.bell-labs.com
		ns=ns1.cs.bell-labs.com
		ns=ns2.cs.bell-labs.com
```

Notice the form of the reverse address. For IPv4, it's the bytes of the address range you are serving reversed and expressed in decimal, and with `.in-addr.arpa` appended. For IPv6, it's the nibbles (4-bit fields) of the address range you are serving reversed and expressed in hexadecimal, and with `.ip6.arpa` appended. These are the standard forms for a domain name in a PTR record.

If such an `soa` entry exists in the database, reverse addresses will automatically be generated from any IP addresses in the database that are under this root. For example

```
	dom=ns1.cs.bell-labs.com ip=135.104.1.11
```

will automatically create both forward and reverse entries for `ns1.cs.bell-labs.com`. Unlike other DNS servers, there's no way to generate inconsistent forward and reverse entries.

Classless reverse delegation

Following RFC 2317, it is possible to serve reverse DNS data for IPv4 subnets smaller than /24. Declare the non-/24 subnet, the reverse domain and the individual systems.

For example, this is how to serve RFC-2317 `ptr` records for the subnet `65.14.39.128/123`.

```
	ipnet=our-t1 ip=65.14.39.128 ipmask=/123
	dom=128.39.14.65.in-addr.arpa soa=
		refresh=3600 ttl=3600
		ns=ns1.our-domain.com
		ns=ns2.our-domain.com
	ip=65.14.39.129 dom=router.our-domain.com
```

Delegating Name Service Authority

Delegation of a further subtree to another set of name servers is indicated by an `soa=delegated` attribute.

```
	dom=bignose.cs.research.bell-labs.com
		soa=delegated
		ns=anna.cs.research.bell-labs.com
		ns=dj.cs.research.bell-labs.com
```

Nameservers within the delegated domain (as in this example) must have their IP addresses listed elsewhere in *ndb* files.

Wildcards, MX and CNAME records

Wild-carded domain names can also be used. For example, to specify a mail forwarder for all Bell Labs research systems:

```
dom=*.research.bell-labs.com
     mx=research.bell-labs.com
```

'Cname' aliases may be established by adding a cname attribute giving the real domain name; the name attached to the dom attribute is the alias. 'Cname' aliases are severely restricted; the aliases may have no other attributes than dom and are daily further restricted in their use by new RFCs.

```
cname=anna.cs.bell-labs.com dom=www.cs.bell-labs.com
```

makes www.... a synonym for the canonical name anna.....

Straddling Server

Many companies have an inside network protected from outside access with firewalls. They usually provide internal 'root' DNS servers (of varying reliability and correctness) that serve internal domains and pass on DNS queries for outside domains to the outside, relaying the results back and caching them for future use. Some companies don't even let DNS queries nor replies through their firewalls at all, in either direction.

In such a situation, running `dns -so` on a machine that imports access to the outside network via `/net.alt` from a machine that straddles the firewalls, or that straddles the firewalls itself, will let internal machines query such a machine and receive answers from outside nameservers for outside addresses and inside nameservers for inside addresses, giving the appearance of a unified domain name space, while bypassing the corporate DNS proxies or firewalls. This is different from running `dns -s` and `dns -sRx /net.alt -f /lib/ndb/external` on the same machine, which keeps the inside and outside namespaces entirely separate.

Under `-o`, several *sys* names are significant: `inside-dom`, `inside-ns`, and `outside-ns`. *Inside-dom* should contain a series of dom pairs naming domains internal to the organization. *Inside-ns* should contain a series of ip pairs naming the internal DNS 'root' servers. *Outside-ns* should contain a series of ip pairs naming the external DNS servers to consult.

Zone Transfers and TCP

Dnstcp is invoked, usually from `/rc/bin/service/tcp53`, to answer DNS queries with long answers via TCP, notably to transfer a zone within the database *dbfile* (default `/lib/ndb/local`) to its invoker on the network at *netmtpt* (default `/net`). Standard input will be read for DNS requests and the DNS answers will appear on standard output. Recursion is disabled by `-R`; acting as a pure resolver is enabled by `-r`. If *conn-dir* is provided, it is assumed to be a directory within *netmtpt*/`tcp` and is used to find the caller's address.

DNS Queries and Debugging

Ndb/dnsquery can be used to query *ndb/dns* to see how it resolves requests. *Ndb/dnsquery* prompts for commands of the form

domain-name request-type

where *request-type* can be ip, ipv6, mx, ns, cname, ptr.... In the case of the inverse query type, ptr, *dnsquery* will reverse the ip address and tack on the `.in-addr.arpa` if necessary.

Ndb/dnsdebug is like *ndb/dnsquery* but bypasses the local server. It communicates via UDP (and sometimes TCP) with the domain name servers in the same way that the local resolver would and displays all packets received. The query can be specified on the command line or can be prompted for. The queries look like those of *ndb/dnsquery* with one addition. *Ndb/dnsdebug* can be directed to query a particular name server by the command @*name-server*. From that point on, all queries go to that name server rather than being resolved by *dnsdebug*. The @ command returns query resolution to *dnsdebug*. Finally, any command preceded by a @*name-server* sets the name server only for that command.

Normally *dnsdebug* uses the `/net` interface and the database file `/lib/ndb/local`. The `-f` option supplies the name of the data base file to use. The `-r` option is the same as for *ndb/dns*. The `-x` option directs *dnsdebug* to use the `/net.alt` interface and `/lib/ndb/external` database file.

EXAMPLES

Look up helix in *ndb*.

```
% ndb/query sys helix
sys=helix dom=helix.research.bell-labs.com bootf=/mips/9powerboot
     ip=135.104.117.31 ether=080069020427
```

Look up `plan9.bell-labs.com` and its IP address in the DNS.

```
% ndb/dnsquery
> plan9.bell-labs.com ip
plan9.bell-labs.com ip   204.178.31.2
> 204.178.31.2 ptr
2.31.178.204.in-addr.arpa ptr plan9.bell-labs.com
2.31.178.204.in-addr.arpa ptr ampl.com
>
```

Print the names of all systems that boot via PXE.

```
% ndb/query -a bootf /386/9pxeload sys
```

FILES

`/env/DNSSERVER`	resolver's DNS servers' IP addresses.
`/lib/ndb/local`	first database file searched
`/lib/ndb/local.*`	hash files for `/lib/ndb/local`
`/srv/cs`	service file for *ndb/cs*
`/net/cs`	where `/srv/cs` gets mounted
`/srv/dns`	service file for *ndb/dns*
`/net/dns`	where `/srv/dns` gets mounted

SOURCE

`/sys/src/cmd/ndb`

SEE ALSO

ndb(2), *ndb*(6)

BUGS

Ndb databases are case-sensitive; ethernet addresses must be in lower-case hexadecimal.