NAME

nsd – Name Server Daemon (NSD) version 4.12.0.

SYNOPSIS

nsd [–**4**] [–**6**] [–**a** *ip–address[@port]*] [–**c** *configfile*] [–**d**] [–**h**] [–**i** *identity*] [–**I** *nsid*] [–**l** *logfile*] [–**N** *server–count*] [–**n** *noncurrent–tcp–count*] [–**P** *pidfile*] [–**p** *port*] [–**s** *seconds*] [–**t** *chrootdir*] [–**u** *username*] [–**V** *level*] [–**v**]

DESCRIPTION

NSD is a complete implementation of an authoritative DNS nameserver. Upon startup, **NSD** will read the configuration file and put itself into the background and answers queries on port 53 or a different port specified with **–p** *port* option. By default, **NSD** will bind to all local interfaces available. Use the **–a** *ip–address[@port]* option to specify a single particular interface address to be bound. If this option is given more than once, **NSD** will bind its UDP and TCP sockets to all the specified ip–addresses separately. If IPv6 is enabled when **NSD** is compiled an IPv6 address can also be specified.

OPTIONS

All the options can be specified in the configfile (**–c** argument), except for the **–v** and **–h** options. If options are specified on the commandline, the options on the commandline take precedence over the options in the configfile.

Normally **NSD** should be started with the ‘nsd–control(8) start‘ command invoked from a */etc/rc.d/nsd.sh* script or similar at the operating system startup.

–4 Only listen to IPv4 connections.

–6 Only listen to IPv6 connections.

−a *ip−address[@port]*
: Listen to the specified *ip−address*. The *ip−address* must be specified in numeric format (using the standard IPv4 or IPv6 notation). Optionally, a port number can be given. This flag can be specified multiple times to listen to multiple IP addresses. If this flag is not specified, **NSD** listens to the wildcard interface.

−c *configfile*
: Read specified *configfile* instead of the default *//etc/nsd/nsd.conf*. For format description see nsd.conf(5).

−d Do not fork, stay in the foreground.

−h Print help information and exit.

−i *identity*
: Return the specified *identity* when asked for *CH TXT ID.SERVER* (This option is used to determine which server is answering the queries when they are anycast). The default is the name returned by gethostname(3).

−I *nsid*
: Add the specified *nsid* to the EDNS section of the answer when queried with an NSID EDNS enabled packet. As a sequence of hex characters or with ascii_ prefix and then an ascii string.

−l *logfile*
: Log messages to the specified *logfile*. The default is to log to stderr and syslog. If a **zonesdir:** is specified in the config file this path can be relative to that directory.

−N *count*
: Start *count* **NSD** servers. The default is 1. Starting more than a single server is only

zone.secondary
: number of secondary zones served. These are zones with 'request–xfr' entries. Also output as 'zone.slave' for backwards compatibility.

FILES

//etc/nsd/nsd.conf
: nsd configuration file.

//etc/nsd
: directory with private keys (nsd_server.key and nsd_control.key) and self–signed certificates (nsd_server.pem and nsd_control.pem).

SEE ALSO

nsd.conf(5), *nsd*(8), *nsd–checkconf*(8)

useful on machines with multiple CPUs and/or network adapters.

–n *number*
: The maximum *number* of concurrent TCP connection that can be handled by each server. The default is 100.

–P *pidfile*
: Use the specified *pidfile* instead of the platform specific default, which is mostly */run/nsd/nsd.pid*. If a **zonesdir:** is specified in the config file, this path can be relative to that directory.

–p *port*
: Answer the queries on the specified *port*. Normally this is port 53.

–s *seconds*
: Produce statistics dump every *seconds* seconds. This is equal to sending *SIGUSR1* to the daemon periodically.

–t *chroot*
: Specifies a directory to *chroot* to upon startup. This option requires you to ensure that appropriate syslogd(8) socket (e.g. *chrootdir* /dev/log) is available, otherwise **NSD** won't produce any log output.

–u *username*
: Drop user and group privileges to those of *username* after binding the socket. The *username* must be one of: username, id, or id.gid. For example: nsd, 80, or 80.80.

–V *level*
: This value specifies the verbosity level for (non–debug) logging. Default is 0.

–v
: Print the version number of **NSD** to standard error and exit.

NSD reacts to the following signals:

SIGTERM
: Stop answering queries, shutdown, and exit normally.

SIGHUP
: Reopen logfile (assists rotation) and optionally update TSIG keys and zones.

SIGUSR1
: Dump BIND8–style statistics into the log. Ignored otherwise.

FILES

/run/nsd/nsd.pid
: the process id of the name server.

//etc/nsd/nsd.conf
: default **NSD** configuration file

DIAGNOSTICS

NSD will log all the problems via the standard syslog(8) *daemon* facility, unless the **–d** option is specified.

SEE ALSO

nsd.conf(5), *nsd–checkconf*(8), *nsd–control*(8)

AUTHORS

NSD was written by NLnet Labs and RIPE NCC joint team. Please see CREDITS file in the distribution for further details.

num.tcp6
: number of connections over TCP ip6.

num.tls
: number of connections over TLS ip4. TLS queries are not part of num.tcp.

num.tls6
: number of connections over TLS ip6. TLS queries are not part of num.tcp6.

num.answer_wo_aa
: number of answers with NOERROR rcode and without AA flag, this includes the referrals.

num.rxerr
: number of queries for which the receive failed.

num.txerr
: number of answers for which the transmit failed.

num.raxfr
: number of AXFR requests from clients (that got served with reply).

num.rixfr
: number of IXFR requests from clients (that got served with reply).

num.truncated
: number of answers with TC flag set.

num.dropped
: number of queries that were dropped because they failed sanity check.

zone.primary
: number of primary zones served. These are zones with no 'request–xfr:' entries. Also output as 'zone.master' for backwards compatibility.

size.db.disk
: size of nsd.db on disk, in bytes.

size.db.mem
: size of the DNS database in memory, in bytes.

size.xfrd.mem
: size of memory for zone transfers and notifies in xfrd process, excludes TSIG data, in bytes.

size.config.disk
: size of zonelist file on disk, excludes the nsd.conf size, in bytes.

size.config.mem
: size of config data in memory, kept twice in server and xfrd process, in bytes.

num.type.X
: number of queries with this query type.

num.opcode.X
: number of queries with this opcode.

num.class.X
: number of queries with this query class.

num.rcode.X
: number of answers that carried this return code.

num.edns
: number of queries with EDNS OPT.

num.ednserr
: number of queries which failed EDNS parse.

num.udp
: number of queries over UDP ip4.

num.udp6
: number of queries over UDP ip6.

num.tcp
: number of connections over TCP ip4.

NAME

nsd−checkconf − NSD configuration file checker.

SYNOPSIS

nsd−checkconf [−**v**] [−**f**] [−**h**] [−**o** *option*] [−**z** *zonename*] [−**p** *pattern*] [−**s** *keyname*] [−**t** *tlsauth-name*] *configfile*

DESCRIPTION

nsd−checkconf reads a configuration file. It prints parse errors to standard error, and performs additional checks on the contents. The configfile format is described in nsd.conf(5).

The utility of this program is to check a config file for errors before using it in nsd(8). This program can also be used for shell scripts to access the nsd config file, using the −o and −z options.

OPTIONS

−v
: After reading print the options to standard output in configfile format. Without this option, only success or parse errors are reported.

−f
: Print full pathname when used with files, like with −o pidfile. This includes the chroot in the way it is applied to the pidfile.

−h
: Print usage help information and exit.

−o *option*
: Return only this option from the config file. This option can be used in conjunction with the **−z** and the **−p** option, or without them to query the server: section. The special value *zones* prints out a list of configured zones. The special value *patterns* prints out a list of configured patterns.

  This option can be used to parse the config file from the shell. If the **−z** option is given, but the **−o** option is not given, nothing is printed.

−s *keyname*
: Prints the key secret (base64 blob) configured for this key in the config file. Used to help shell scripts parse the config file.

−t *tls-auth*
: Prints the authentication domain name configured for this tls-auth clause in the config file. Used to help shell scripts parse the config file.

−p *pattern*
: Return the option specified with **−o** for the given pattern name.

−z *zonename*
: Return the option specified with **−o** for zone ’zonename’.

If this option is not given, the server section of the config file is used.

The −o, −s and −z option print configfile options to standard output.

FILES

//etc/nsd/nsd.conf
: default **NSD** configuration file

SEE ALSO

nsd(8), *nsd.conf*(5), *nsd−control*(8)

AUTHORS

NSD was written by NLnet Labs and RIPE NCC joint team. Please see CREDITS file in the distribution for further details.

EXIT CODE

The nsd−control program exits with status code 1 on error, 0 on success.

SET UP

The setup requires a self−signed certificate and private keys for both the server and client. The script *nsd−control−setup* generates these in the default run directory, or with −d in another directory. If you change the access control permissions on the key files you can decide who can use nsd−control, by default owner and group but not all users. The script preserves private keys present in the directory. After running the script as root, turn on **control−enable** in *nsd.conf*.

STATISTIC COUNTERS

The *stats* command shows a number of statistic counters.

num.queries
: number of queries received (the tls, tcp and udp queries added up).

serverX.queries
: number of queries handled by the server process. The number of server processes is set with the config statement **server−count**.

time.boot
: uptime in seconds since the server was started. With fractional seconds.

time.elapsed
: time since the last stats report, in seconds. With fractional seconds. Can be zero if polled quickly and the previous stats command resets the counters, so that the next gets a fully zero, and zero elapsed time, report.

Cookie succeeds with any of the *active* or *staging* cookie secrets. The state of the current cookie secrets can be printed with the **print_cookie_secrets** command.

When there are no cookie secrets configured yet, the <secret> is added as *active*. If there is already an *active* cookie secret, the <secret> is added as *staging* or replacing an existing *staging* secret.

To "roll" a cookie secret used in an anycast set. The new secret has to be added as staging secret to **all** nodes in the anycast set. When **all** nodes can verify DNS Cookies with the new secret, the new secret can be activated with the **activate_cookie_secret** command. After **all** nodes have the new secret *active* for at least one hour, the previous secret can be dropped with the **drop_cookie_secret** command.

Persistence is accomplished by writing to a file which if configured with the **cookie–secret–file** option in the server section of the config file. The default value for that is: //var/db/nsd/cookiesecrets.txt .

drop_cookie_secret
Drop the *staging* cookie secret.

activate_cookie_secret
Make the current *staging* cookie secret *active*, and the current *active* cookie secret *staging*.

print_cookie_secrets
Show the current configured cookie secrets with their status.

NAME

nsd–checkzone – NSD zone file syntax checker.

SYNOPSIS

nsd–checkzone [**–h**] *zonename zonefile*

DESCRIPTION

nsd–checkzone reads a DNS zone file and checks it for errors. It prints errors to stderr. On failure it exits with nonzero exit status.

This is used to check files before feeding them to the nsd(8) daemon.

OPTIONS

–h Print usage help information and exit.

zonename
The name of the zone to check, eg. "example.com".

zonefile
The file to read, eg. "zones/example.com.zone.signed". Use "-" to read from stdin.

–p Print the zone contents to stdout if the zone is ok. This prints the contents as it has been parsed, not literally a copy of the input, but as printed by the formatting routines in NSD, much like the nsd-control command write does.

–i *<oldzonefile>*
Create an IXFR from the differences between the old zone file and the new zone file. The argument to the –i option is the old zone file, the other zonefile argument passed is the new zonefile.

The difference is computed between the two zonefiles by keeping one version of the zone in memory, and another version in a temporary file. The temporary file is located at the zonefile directory. This is also where the result is written, to a file with the zonefile name, ending with '.ixfr'. This is also where NSD reads it when IXFRs are configured for the zone.

The other existing ixfr files are renamed to become older IXFR contents for the zone, if any such files exist. If the output file already exists with the correct contents, no new file is created. The contents of the header of the output file are checked for that, if it already exists.

−n *<ixfr number>*
The number of IXFR versions to store, at most. Default 5. This is the number of files that is created with ixfr contents for the zone. Older stored IXFR versions are deleted when the number is exceeded.

−s *<ixfr size>*
The number of bytes of storage to use for IXFRs. Default is 1048576. If an IXFR is bigger it is not created, and if the sum of IXFR storage exceeds it, older IXFRs versions are deleted.

SEE ALSO

nsd(8), *nsd-checkconf*(8)

AUTHORS

NSD was written by NLnet Labs and RIPE NCC joint team. Please see CREDITS file in the distribution for further details.

print_tsig [<key_name>]
print the secret and algorithm for the TSIG key with that name. Or list all the tsig keys with their name, secret and algorithm.

update_tsig <name> <secret>
Change existing TSIG key with name to the new secret. The secret is a base64 encoded string. The changes are only in-memory and are gone next restart, for lasting changes edit the nsd.conf file or a file included from it.

add_tsig <name> <secret> [algo]
Add a new TSIG key with the given name, secret and algorithm. Without algorithm a default (hmac-sha256) algorithm is used. The secret is a base64 encoded string. The changes are only in-memory and are gone next restart, for lasting changes edit the nsd.conf file or a file included from it.

assoc_tsig <zone> <key_name>
Associate the zone with the given tsig. The access control lists for notify, allow-notify, provide-xfr and request-xfr are adjusted to use the given key.

del_tsig <key_name>
Delete the TSIG key with the given name. Prints error if the key is still in use by some zone. The changes are only in-memory and are gone next restart, for lasting changes edit the nsd.conf file or a file included from it.

add_cookie_secret <secret>
Add or replace a cookie secret persistently. <secret> needs to be an 128 bit hex string.

Cookie secrets can be either *active* or *staging*. *Active* cookie secrets are used to create DNS Cookies, but verification of a DNS argument, all zones are transferred.

force_transfer [<zone>]
Force update secondary zones that are hosted on this server. Even if the primary hosts the same serial number of the zone, a full AXFR is performed to fetch it. If you want to use IXFR and check that the serial number increases, use the ’transfer’ command. With argument that zone is transferred, without argument, all zones are transferred.

zonestatus [<zone>]
Print state of the zone, the serial numbers and since when they have been acquired. Also prints the notify action (to which server), and zone transfer (and from which primary) if there is activity right now. The state of the zone is printed as: ’primary’ (primary zones), ’ok’ (secondary zone is up–to–date), ’expired’ (secondary zone has expired), ’refreshing’ (secondary zone has transfers active). The serial numbers printed are the ’served–serial’ (currently active), the ’commit–serial’ (is in reload), the ’notified–serial’ (got notify, busy fetching the data). The serial numbers are only printed if such a serial number is available. With argument that zone is printed, without argument, all zones are printed.

serverpid
Prints the PID of the server process. This is used for statistics (and only works when NSD is compiled with statistics enabled). This pid is not for sending unix signals, use the pid from nsd.pid for that, that pid is also stable.

verbosity <number>
Change logging verbosity.

NAME
nsd.conf – NSD configuration file

SYNOPSIS
nsd.conf

DESCRIPTION
This file is used to configure nsd(8). It specifies options for the nsd server, zone files, primaries and secondaries.

The file format has attributes and values. Some attributes have attributes inside them. The notation is:

attribute: value

Comments start with # and last to the end of line. Empty lines are ignored, as is whitespace at the beginning of a line. Quotes must be used for values with spaces in them, eg. "file name.zone".

EXAMPLE
An example of a short nsd.conf file is below.

```
# Example nsd.conf file for example.com.
# This is a comment.

server:
        server-count: 1 # use this number of cpu cores
        username:
        zonelistfile: //var/db/nsd/zone.list
        logfile: //var/log/nsd.log
        pidfile: /run/nsd/nsd.pid
        xfrdfile: //var/db/nsd/xfrd.state

zone:
        name: example.com
        zonefile: //etc/nsd/example.com.zone

zone:
        # this server is the primary and 192.0.2.1 is the secondary.
        name: primaryzone.com
        zonefile: //etc/nsd/primaryzone.com.zone
        notify: 192.0.2.1 NOKEY
        provide-xfr: 192.0.2.1 NOKEY

zone:
        # this server is the secondary and 192.0.2.2 is the primary.
        name: secondaryzone.com
        zonefile: //etc/nsd/secondaryzone.com.zone
        allow-notify: 192.0.2.2 NOKEY
        request-xfr: 192.0.2.2 NOKEY
```

Then, use kill −HUP to reload changes from primary zone files. And use kill −TERM to stop the server.

FILE FORMAT

There must be whitespace between keywords. Attribute keywords end with a colon ':'. An attribute is followed by its containing attributes, or a value.

At the top level, only **server:**, **verify:**, **key:**, **pattern:**, **zone:**, **tls-auth:**, and **remote-control:** are allowed. These are followed by their attributes or a new top-level keyword. The **zone:** attribute is followed by zone options. The **server:** attribute is followed by global options for the **NSD** server. The **verify:** attribute is used to control zone verification. A **key:** attribute is used to define keys for authentication. The **pattern:** attribute is followed by the zone options for zones that use the pattern. A **tls-auth:** attribute is used to define authentication attributes for TLS connections used for XFR-over-TLS.

Files can be included using the **include:** directive. It can appear anywhere, and takes a single filename as an argument. Processing continues as if the text from the included file were copied into the config file at that point. If a chroot is used, an absolute filename is

write [<zone>]
Write zonefiles to disk, or the given zonefile to disk. Zones that have changed (via AXFR or IXFR) are written, or if the zonefile has not been created yet then it is created. Directory components of the zonefile path are created if necessary. With argument that zone is written if it was modified, without argument, all modified zones are written.

notify [<zone>]
Send NOTIFY messages to secondary servers. Sends to the IP addresses configured in the 'notify:' lists for the primary zones hosted on this server. Usually NSD sends NOTIFY messages right away when a primary zone serial is updated. If a zone is given, notifies are sent for that zone. These secondary servers are supposed to initiate a zone transfer request later (to this server or another primary), this can be allowed via the 'provide−xfr:' acl list configuration. With argument that zone is processed, without argument, all zones are processed.

transfer [<zone>]
Attempt to update secondary zones that are hosted on this server by contacting the primaries. The primaries are configured via 'request−xfr:' lists. If a zone is given, that zone is updated. Usually NSD receives a NOTIFY from the primaries (configured via 'allow−notify:' acl list) that a new zone serial has to be transferred. For zones with no content, NSD may have backed off from asking often because the primaries did not respond, but this command will reset the backoff to its initial timeout, for frequent retries. With argument that zone is transferred, without

addzone <zone name> <pattern name>
Add a new zone to the running server. The zone is added to the zonelist file on disk, so it stays after a restart. The pattern name determines the options for the new zone. For secondary zones a zone transfer is immediately attempted. For zones with a zonefile, the zone file is attempted to be read in.

delzone <zone name>
Remove the zone from the running server. The zone is removed from the zonelist file on disk, from the nsd.db file and from the memory. If it had a zonefile, this remains (but may be outdated). Zones configured inside nsd.conf itself cannot be removed this way because the daemon does not write to the nsd.conf file, you need to add such zones to the zonelist file to be able to delete them with the delzone command.

changezone <zone name> <pattern name>
Change a zone to use the pattern for options. The zone is deleted and added in one operation, changing it to use the new pattern for the zone options. Zones configured in nsd.conf cannot be changed like this, instead edit the nsd.conf (or the included file in nsd.conf) and reconfig.

addzones
Add zones read from stdin of nsd−control. Input is read per line, with name space patternname on a line. For bulk additions.

delzones
Remove zones read from stdin of nsd−control. Input is one name per line. For bulk removals.

needed (with the chroot prepended), so that the include can be parsed before and after application of the chroot (and the knowledge of what that chroot is). You can use '*' to include a wildcard match of files, eg. "foo/nsd.d/*.conf". Also '?', '{}', '[]', and '~' work, see **glob**(7). If no files match the pattern, this is not an error.

Server Options

The global options (if not overridden from the NSD command-line) are taken from the **server:** clause. There may only be one **server:** clause.

ip−address: <ip4 or ip6>[@port] [servers] [bindtodevice] [setfib]
NSD will bind to the listed ip−address. Can be given multiple times to bind multiple ip−addresses. Optionally, a port number can be given. If none are given NSD listens to the wildcard interface. Same as command-line option **−a.**

To limit which NSD server(s) listen on the given interface, specify one or more servers separated by whitespace after <ip>[@port]. Ranges can be used as a shorthand to specify multiple consecutive servers. By default every server will listen.

If an interface name is used instead of ip4 or ip6, the list of IP addresses associated with that interface is picked up and used at server start.

For servers with multiple IP addresses that can be used to send traffic to the internet, list them one by one, or the source address of replies could be wrong. This is because if the udp socket associates a source address of

0.0.0.0 then the kernel picks an ip-address with which to send to the internet, and it picks the wrong one. Typically needed for anycast instances. Use ip-transparent to be able to list addresses that turn on later (typical for certain load-balancing).

interface: <ip4 or ip6>[@port] [servers] [bindtodevice] [setfib]
Same as ip–address (for ease of compatibility with unbound.conf).

ip–transparent: <yes or no>
Allows NSD to bind to non local addresses. This is useful to have NSD listen to IP addresses that are not (yet) added to the network interface, so that it can answer immediately when the address is added. Default is no.

ip–freebind: <yes or no>
Set the IP_FREEBIND option to bind to nonlocal addresses and interfaces that are down. Similar to ip–transparent. Default is no.

reuseport: <yes or no>
Use the SO_REUSEPORT socket option, and create file descriptors for every server in the server–count. This improves performance of the network stack. Only really useful if you also configure a server–count higher than 1 (such as, equal to the number of cpus). The default is no. It works on Linux, but does not work on FreeBSD, and likely does not work on other systems.

send–buffer–size: <number>
Set the send buffer size for query-servicing sockets. Set to 0 to use the default settings.

stop Stop the server. The server daemon exits.

reload [<zone>]
Reload zonefiles and reopen logfile. Without argument reads changed zonefiles. With argument reads the zonefile for the given zone and loads it.

reconfig
Reload nsd.conf and apply changes to TSIG keys and configuration patterns, and apply the changes to add and remove zones that are mentioned in the config. Other changes are not applied, such as listening ip address and port and chroot, also per-zone statistics are not applied. The pattern updates means that the configuration options for zones (request–xfr, zonefile, notify, ...) are updated. Also new patterns are available for use with the addzone command.

repattern
Same as the reconfig option.

log_reopen
Reopen the logfile, for log rotate that wants to move the logfile away and create a new logfile. The log can also be reopened with kill –HUP (which also reloads all zonefiles).

status Display server status. Exit code 3 if not running (the connection to the port is refused), 1 on error, 0 if running.

stats Output a sequence of name=value lines with statistics information, requires NSD to be compiled with this option enabled.

stats_noreset
Same as stats, but does not zero the counters.

NAME

nsd–control, nsd–control–setup – NSD remote server control utility.

SYNOPSIS

nsd–control [**–c** *cfgfile*] [**–s** *server*] *command*

DESCRIPTION

nsd–control performs remote administration on the *nsd*(8) DNS server. It reads the configuration file, contacts the nsd server over SSL, sends the command and displays the result. Commands that require a reload are queued and the result indicates the command was accepted.

The available options are:

–h Show the version and commandline option help.

–c *cfgfile*
The config file to read with settings. If not given the default config file //etc/nsd/nsd.conf is used.

–s *server[@port]*
IPv4 or IPv6 address of the server to contact. If not given, the address is read from the config file.

COMMANDS

There are several commands that the server understands.

start Start the server. Simply execs *nsd*(8). The nsd executable is not searched for in the **PATH** set in the environment. Instead the default location relative to the installation prefix specified at compile-time. The executable location can be overridden by setting *NSD_PATH* in the environment. It is started with the config file specified using *–c* or the default config file.

receive–buffer–size: <number>
Set the receive buffer size for query-servicing sockets. Set to 0 to use the default settings.

debug–mode: <yes or no>
Turns on debugging mode for nsd, does not fork a daemon process. Default is no. Same as command-line option **–d.** If set to yes it does not fork and stays in the foreground, which can be helpful for command-line debugging, but is also used by certain server supervisor processes to ascertain that the server is running.

do–ip4: <yes or no>
If yes, NSD listens to IPv4 connections. Default yes.

do–ip6: <yes or no>
If yes, NSD listens to IPv6 connections. Default yes.

database: <filename>
This option is ignored by NSD versions 4.8.0 and newer, because the database feature has been removed.

zonelistfile: <filename>
By default *//var/db/nsd/zone.list* is used. The specified file is used to store the dynamically added list of zones. The list is written to by NSD to add and delete zones. It is a text file with a zone–name and pattern–name on each line. This file is used for the nsd–control addzone and delzone commands.

identity: <string>
Returns the specified identity when asked for CH TXT ID.SERVER. Default is the name as returned by gethostname(3). Same as command-line option **–i**. See hide–identity to set the server to not respond to such queries.

version: <string>
Returns the specified version string when asked for CH TXT version.server, and version.bind queries. Default is the compiled package version. See hide–version to set the server to not respond to such queries.

nsid: <string>
Add the specified nsid to the EDNS section of the answer when queried with an NSID EDNS enabled packet. As a sequence of hex characters or with ascii_ prefix and then an ascii string. Same as command-line option **–I**.

logfile: <filename>
Log messages to the logfile. The default is to log to stderr and syslog (with facility LOG_DAEMON). Same as command-line option **–l**.

log–only–syslog: <yes or no>
Log messages only to syslog. Useful with systemd so that print to stderr does not cause duplicate log strings in journald. Before syslog has been opened, the server uses stderr. Stderr is also used if syslog is not available. Default is no.

server–count: <number>
Start this many NSD servers. Default is 1. Same as command-line option **–N**.

cpu–affinity: <number> <number> ...
Overall CPU affinity for NSD server(s). Default is no affinity.

server–N–cpu–affinity: <number>
Bind NSD server specified by N to a specific core. Default is to have affinity set to every core specified in cpu–affinity. This setting only takes effect if cpu–affinity is enabled.

```
zone:
        name: "example.nl"
        zonefile: "example.nl"
        # allow anybody to request xfr.
        provide–xfr: 0.0.0.0/0 NOKEY
        provide–xfr: ::0/0 NOKEY

# to list a secondary server you would in general give
# provide–xfr: 1.2.3.4 tsig–key.name.
# notify: 1.2.3.4 NOKEY
```

Other

NSD is an authoritative only DNS server. This means that it is meant as a primary or secondary server for zones, providing DNS data to DNS resolvers and caches. BIND9 can function as an authoritative DNS server, the configuration options for that are compared with those for NSD in this section. However, BIND9 can also function as a resolver or cache. The configuration options that BIND9 has for the resolver or caching thus have no equivalents for NSD.

FILES

//etc/nsd/nsd.conf
default **NSD** configuration file

SEE ALSO

nsd(8), *nsd–checkconf*(8), *nsd–checkzone*(8), *nsd–control*(8)

AUTHORS

NSD was written by a combined team from NLnet Labs and RIPE NCC. Please see the CREDITS file in the distribution for further details.

BUGS

nsd.conf is parsed by a primitive parser. Error messages may not be to the point.

```
# Config file for example.org
key:
          name: tsig.example.org.
          algorithm: hmac−md5
          secret: "aaaaaabbbbbbccccccdddddd"

zone:
          name: "example.org"
          zonefile: "secondary/example.org.signed"
          # the primary is allowed to notify and will provide zone data.
          allow−notify: 162.0.4.49 NOKEY
          request−xfr: 162.0.4.49 tsig.example.org.
```

Notice that the primary is listed twice, once to allow it to send notifies to this secondary server and once to tell the secondary server where to look for updates zone data. More allow−notify and request−xfr lines can be added to specify more primaries.

It is possible to specify extra allow−notify lines for addresses that are also allowed to send notifications to this secondary server.

Primary zones

For a primary zone in BIND9, the secondary servers are listed. These secondary servers are sent notifications of updated and are allowed to request transfer of the zone data. In NSD these two properties need to be configured separately.

Here is an example of a primary zone in BIND9 syntax.

```
zone "example.nl" {
          type primary;
          file "example.nl";
};
```

In NSD syntax this becomes:

xfrd−cpu−affinity: <number>
Bind xfrd to a specific core. Default is to have affinity set to every core specified in cpu−affinity. This setting only takes effect if cpu−affinity is enabled.

tcp−count: <number>
The maximum number of concurrent, active TCP connections by each server. Default is 100. Same as command-line option **−n**. That is the number of requests from clients to this server.

tcp−reject−overflow: <yes or no>
If set to yes, TCP connections made beyond the maximum set by tcp-count will be dropped immediately (accepted and closed). Default is no.

tcp−query−count: <number>
The maximum number of queries served on a single TCP connection. Default is 0, meaning there is no maximum.

tcp−timeout: <number>
Overrides the default TCP timeout. This also affects zone transfers over TCP. The default is 120 seconds.

tcp-mss: <number>
Maximum segment size (MSS) of TCP socket on which the server responds to queries. Value lower than common MSS on Ethernet (1220 for example) will address path MTU problem. Note that not all platform supports socket option to set MSS (TCP_MAXSEG). Default is system default MSS determined by interface MTU and negotiation between server and client.

outgoing–tcp–mss: <number>
Maximum segment size (MSS) of TCP socket for outgoing XFR request to other nameservers. Value lower than common MSS on Ethernet (1220 for example) will address path MTU problem. Note that not all platform supports socket option to set MSS (TCP_MAXSEG). Default is system default MSS determined by interface MTU and negotiation between NSD and other servers.

xfrd–tcp–max: <number>
Number of sockets for xfrd to use for outgoing zone transfers. Default 128. Increase it to allow more zone transfer sockets, like to 256. That is for zone transfers requested by this server from other servers. To save memory, this can be lowered, set it lower together with some other settings to have reduced memory footprint for NSD. xfrd–tcp–max: 32 and xfrd–tcp–pipeline: 128 and rrl–size: 1000

This reduces memory footprint, other memory usage is caused mainly by the server–count setting, the number of server processes, and the tcp–count setting, which keeps buffers per server process, and by the size of the zone data.

xfrd–tcp–pipeline: <number>
Number of simultaneous outgoing zone transfers that are possible on the tcp sockets of xfrd. Max is 65536, default is 128. That is for zone transfers requested by this server from other servers.

ipv4–edns–size: <number>
Preferred EDNS buffer size for IPv4. Default 1232.

statements.

In BIND9 you only need to provide allow–notify elements for any extra sources of notifications (i.e. the operators), NSD needs to have allow–notify for both primaries and operators. BIND9 allows additional transfer sources, in NSD you list those as request–xfr.

Here is an example of a secondary zone in BIND9 syntax.

```
# Config file for example.org
options {
        dnssec–enable yes;
};

key tsig.example.org. {
        algorithm hmac–md5;
        secret "aaaaaabbbbbbccccccdddddd";
};

server 162.0.4.49 {
        keys { tsig.example.org. ; };
};

zone "example.org" {
        type secondary;
        file "secondary/example.org.signed";
        primaries { 162.0.4.49; };
};
```

For NSD, DNSSEC is enabled automatically for zones that are signed. The dnssec–enable statement in the options clause is not needed. In NSD keys are associated with an IP address in the access control list statement, therefore the server{} statement is not needed. Below is the same example in an NSD config file.

dnstap-tls-client-key-file: <file name>
The key file for client authentication, or "" disabled.

dnstap-tls-client-cert-file: <file name>
The cert file for client authentication, or "" disabled.

dnstap-send-identity: <yes or no>
If enabled, the server identity is included in the log messages. Default is no.

dnstap-send-version: <yes or no>
If enabled, the server version if included in the log messages. Default is no.

dnstap-identity: <string>
The identity to send with messages, if "" the hostname is used. Default is "".

dnstap-version: <string>
The version to send with messages, if "" the package version is used. Default is "".

dnstap-log-auth-query-messages: <yes or no>
Enable to log auth query messages. Default is no. These are client queries to NSD.

dnstap-log-auth-response-messages: <yes or no>
Enable to log auth response messages. Default is no. These are responses from NSD to clients.

NSD CONFIGURATION FOR BIND9 HACKERS

BIND9 is a name server implementation with its own configuration file format, named.conf(5). BIND9 types zones as 'Primary' or 'Secondary'.

Secondary zones

For a secondary zone, the primary servers are listed. The primary servers are queried for zone data, and are listened to for update notifications. In NSD these two properties need to be configured separately, by listing the primary address in allow–notify and request–xfr

ipv6–edns–size: <number>
Preferred EDNS buffer size for IPv6. Default 1232.

pidfile: <filename>
Use the pid file instead of the platform specific default, usually "*/run/nsd/nsd.pid*". Same as command-line option **–P**. With "" there is no pidfile, for some startup management setups, where a pidfile is not useful to have. The default can be set at compile time, sometimes to "". Then the config option and commandline option can be used to specify that a pidfile is used, different from its compile time default value. The file is not chowned to the user from the **username:** option, for permission safety reasons. It remains owned to the user by which the server was started. The file may not be removed after the server is finished and quit, since permissions for the username may not make this possible.

port: <number>
Answer queries on the specified port. Default is 53. Same as command-line option **–p**.

statistics: <number>
If not present no statistics are dumped. Statistics are produced every number seconds. Same as command-line option **–s**.

chroot: <directory>
NSD will chroot on startup to the specified directory. Note that if elsewhere in the configuration you specify an absolute pathname to a file inside the chroot, you have to prepend the **chroot** path. That way, you can switch the chroot option on and off without having to modify anything else in the configuration. Set the value to "" (the empty string) to disable

the chroot. By default "" is used. Same as command-line option **−t**.

username: <username>
After binding the socket, drop user privileges and assume the username. Can be username, id or id.gid. Same as command-line option **−u**.

zonesdir: <directory>
Change the working directory to the specified directory before accessing zone files. Also, NSD will access **zonelistfile**, **logfile**, **pidfile**, **xfrdfile**, **xfrdir**, **server-key-file**, **server-cert-file**, **control-key-file** and **control-cert-file** relative to this directory. Set the value to "" (the empty string) to disable the change of working directory. By default "*//etc/nsd*" is used.

difffile: <filename>
Ignored, for compatibility with NSD3 config files.

xfrdfile: <filename>
The soa timeout and zone transfer daemon in NSD will save its state to this file. State is read back after a restart. The state file can be deleted without too much harm, but timestamps of zones will be gone. If it is configured as "", the state file is not used, all secondary zones are checked for updates upon startup. For more details see the section on zone expiry behavior of NSD. Default is *//var/db/nsd/xfrd.state*.

xfrdir: <directory>
The zone transfers are stored here before they are processed. A directory is created here that is removed when NSD exits. Default is */tmp*.

the client authentication.

client−key−pw: <string>
If the client−key file uses a password to decrypt the key before it can be used, then the password can be specified here as a string. It is possible to include other config files with the include: option, and this can be used to move that sensitive data to another file, if you wish.

DNSTAP Logging Options

DNSTAP support, when compiled in, is enabled in the **dnstap:** section. This starts a collector process that writes the log information to the destination.

dnstap-enable: <yes or no>
If dnstap is enabled. Default no. If yes, it connects to the dnstap server and if any of the dnstap-log-..-messages options is enabled it sends logs for those messages to the server.

dnstap-socket-path: <file name>
Sets the unix socket file name for connecting to the server that is listening on that socket. Default is "//var/run/nsd-dnstap.sock".

dnstap-ip: <"" or addr[@port]>
If disabled with "", the socket path is used. With a value, like address or address@port, like "127.0.0.1@3333" TCP or TLS is used. Default is "".

dnstap-tls: <yes or no>
If enabled, TLS is used to the address specified in **dnstap-ip**. Otherwise, TCP is used. Default is yes.

dnstap-tls-server-name: <string>
The name for authenticating the upstream server. With "" disabled.

that file. In this way the key secret and the rest of the configuration file, which may have different security policies, can be split apart. The content of the secret is the agreed base64 secret content. To make it up, enter a password (its length must be a multiple of 4 characters, A–Za–z0–9), or use dev-random output through a base64 encode filter.

TLS Auth Declarations

The **tls-auth:** clause establishes attributes to use when authenticating the far end of a TLS connection as well as to define credentials to authenticate to a remote server. It is used in access control lists for XFR-over-TLS. It has the following attributes.

name: <string>
The tls-auth name. Used to refer to this TLS authentication information in the access control list.

auth–domain–name: <string>
The authentication domain name as defined in RFC8310. Used to verify the certificate of the remote connecting server. When used by a primary server in **provide-xfr** it verifies the secondary. When used by a secondary server in **request-xfr** it verifies the primary.

client–cert: <file name of clientcert.pem>
If you want to use mutual TLS authentication, this is where the client certificates can be configured that NSD uses to connect to the upstream server to download the zone. The client public key pem cert file can be configured here. Also configure a private key with client–key.

client–key: <file name of clientkey.key>
If you want to use mutual TLS authentication, the private key file can be configured here for

xfrd–reload–timeout: <number>
If this value is –1, xfrd will not trigger a reload after a zone transfer. If positive xfrd will trigger a reload after a zone transfer, then it will wait for the number of seconds before it will trigger a new reload. Setting this value throttles the reloads to once per the number of seconds. The default is 1 second.

verbosity: <level>
This value specifies the verbosity level for (non–debug) logging. Default is 0. 1 gives more information about incoming notifies and zone transfers. 2 lists soft warnings that are encountered. 3 prints more information. Same as command-line option **–V**.

Verbosity 0 will print warnings and errors, and other events that are important to keep NSD running.

Verbosity 1 prints additionally messages of interest. Successful notifies, successful incoming zone transfer (the zone is updated), failed incoming zone transfers or the inability to process zone updates.

Verbosity 2 prints additionally soft errors, like connection resets over TCP. And notify refusal, and axfr request refusals.

hide–version: <yes or no>
Prevent NSD from replying with the version string on CHAOS class queries. Default is no.

hide–identity: <yes or no>
Prevent NSD from replying with the identity string on CHAOS class queries. Default is no.

drop–updates: <yes or no>
If set to yes, drop received packets with the UPDATE opcode. Default is no.

use–systemd: <yes or no>
This option is deprecated and ignored. If compiled with libsystemd, NSD signals readiness to systemd and use of the option is not necessary.

log–time–ascii: <yes or no>
Log time in ascii, if "no" then in seconds epoch. Default is yes. This chooses the format when logging to file. The printout via syslog has a timestamp formatted by syslog.

log–time–iso: <yes or no>
Log time in ISO8601 format, if **log–time–ascii:** yes is also set. Default is no.

round–robin: <yes or no>
Enable round robin rotation of records in the answer. This changes the order of records in the answer and this may balance load across them. The default is no.

minimal–responses: <yes or no>
Enable minimal responses for smaller answers. This makes packets smaller. Extra data is only added for referrals, when it is really necessary. This is different from the ––enable-minimal-responses configure time option, that reduces packets, but exactly to the fragmentation length, the nsd.conf option reduces packets as small as possible. The default is no.

confine–to–zone: <yes or no>
If set to yes, additional information will not be added to the response if the apex zone of the additional information does not match the apex zone of the initial query (E.G. CNAME

Catalog zones contain a list of zones that are served. Use **allow–query: 0.0.0.0/0 BLOCKED** and **allow–query: ::0/0 BLOCKED** in a catalog zone zone or pattern clause to prevent revealing the catalog. Also consider using transfers over TLS to further protect the catalog against eavesdroppers.

catalog–member–pattern: <pattern–name>
If this option is provided for a catalog consumer zone, members of that catalog that have a missing or an invalid group property will be added using pattern <pattern–name>.

catalog–producer–zone: <zone–name>
This option can only be used in a pattern. Adding a zone using "*nsd–control addzone <zone> <pattern>*" with a <pattern> containing this option, will cause a catalog member entry to be created in the catalog producer zone <zone–name>. <zone–name> must exist and must be a valid catalog producer zone.

Key Declarations

The **key:** clause establishes a key for use in access control lists. It has the following attributes.

name: <string>
The key name. Used to refer to this key in the access control list. The key name has to be correct for tsig to work. This is because the key name is output on the wire.

algorithm: <string>
Authentication algorithm for this key. Such as hmac–md5, hmac–sha1, hmac–sha224, hmac–sha256, hmac–sha384 and hmac–sha512. Can also be abbreviated as 'sha1', 'sha256'. Default is sha256. Algorithms are only available when they were compiled in (available in the crypto library).

secret: <base64 blob>
The base64 encoded shared secret. It is possible to put the **secret:** declaration (and base64 blob) into a different file, and then to **include:**

verifier−timeout: <seconds>
Number of seconds before verifier is forcefully terminated. Specify 0 (zero) to not use a specific timeout. Default is verifier−timeout from **verify:**.

catalog: <consumer or producer>
If set to *consumer*, catalog zone processing is enabled for the zone. Only a single zone may be configured as a catalog consumer zone. When more than one catalog consumer zone is configured, none of them will be processed. Member zones of the catalog will use the pattern specified by the group property, or if a group property is missing or invalid, the pattern specified by the **catalog−member−pattern** option is used. Group properties are valid if there is only a single value matching the name of a for member zones valid pattern.

A zone with the option set to *producer*, can be used to produce a catalog zone. Member zones for catalog producer zones can be added with "*nsd−control addzone <zone> <pattern>*", where <pattern> has a **catalog−producer−zone** option pointing to a catalog producer zone. Members will get a group property with the pattern name as value. Catalog producer zones must be primary zones and may not have a **request−xfr** option. Catalog producer zones will *not* read content from zone files, but will reconstruct the zone on startup from the member zone entries in //var/db/nsd/zone.list, specified with the **zonelistfile** option.

The status of both catalog consumer and producer zones can be verified with *nsd−control zonestatus*. It will show the number of member zones and, if the catalog zone is invalid, the reason for it to be invalid is shown. *nsd−control zonestatus* will also show the entry of a catalog member zone in the catalog (consumer or producer) zone as **catalog-member-id:**.

A catalog zone can either be catalog consumer zone or a catalog producer zone but not both. Likewise, catalog member zones can be either a member of catalog consumer zone or a catalog producer zone but not both.

resolution). Default is no.

refuse−any: <yes or no>
Refuse queries of type ANY. This is useful to stop query floods trying to get large responses. Note that rrl ratelimiting also has type ANY as a ratelimiting type. It sends truncation in response to UDP type ANY queries, and it allows TCP type ANY queries like normal. The default is no. With the option turned off, NSD behaves according to RFC 8482 4.1. It minimizes the response with one RRset. Popular and not large types, like A, AAAA and MX are preferred, and large types, like DNSKEY and RRSIG are picked with a lower preference than other types. This makes the response smaller.

reload−config: <yes or no>
Reload configuration file and update TSIG keys and zones on SIGHUP. Default is no.

zonefiles−check: <yes or no>
Make NSD check the mtime of zone files on start and sighup. If you disable it it starts faster (less disk activity in case of a lot of zones). The default is yes. The nsd−control reload command reloads zone files regardless of this option.

zonefiles−write: <seconds>
Write updated secondary zones to their zonefile every N seconds. If the zone or pattern's "zonefile" option is set to "" (empty string), no zonefile is written. The default is 3600 (1 hour).

rrl−size: <numbuckets>
This option gives the size of the hashtable. Default 1000000. More buckets use more memory, and reduce the chance of hash

collisions.

rrl–ratelimit: <qps>
The max qps allowed (from one query source). Default is on (with a suggested 200 qps). If set to 0 then it is disabled (unlimited rate), also set the whitelist–ratelimit to 0 to disable ratelimit processing. If you set verbosity to 2 the blocked and unblocked subnets are logged. Blocked queries are blocked and some receive TCP fallback replies. Once the rate limit is reached, NSD begins dropping responses. However, one in every "rrl–slip" number of responses is allowed, with the TC bit set. If slip is set to 2, the returned response rate will be halved. If it's set to 3, the returned response rate will be one–third, and so on. If you set rrl–slip to 10, traffic is reduced to 1/10th. Ratelimit options rrl–ratelimit, rrl–size and rrl–whitelist–ratelimit are updated when nsd–control reconfig is done (also the zone–specific ratelimit options are updated).

rrl–slip: <numpackets>
This option controls the number of packets discarded before we send back a SLIP response (a response with "truncated" bit set to one). 0 disables the sending of SLIP packets, 1 means every query will get a SLIP response. Default is 2, cuts traffic in half and legit users have a fair chance to get a +TC response.

rrl–ipv4–prefix–length: <subnet>
IPv4 prefix length. Addresses are grouped by netblock. Default 24.

rrl–ipv6–prefix–length: <subnet>
IPv6 prefix length. Addresses are grouped by netblock. Default 64.

zonestats: <name>
When compiled with ––enable–zone–stats NSD can collect statistics per zone. This name gives the group where statistics are added to. The groups are output from nsd–control stats and stats_noreset. Default is "". You can use "%s" to use the name of the zone to track its statistics. If not compiled in, the option can be given but is ignored.

include–pattern: <pattern–name>
The options from the given pattern are included at this point. The referenced pattern must be defined above this zone.

rrl–whitelist: <rrltype>
This option causes queries of this rrltype to be whitelisted, for this zone. They receive the whitelist–ratelimit. You can give multiple lines, each enables a new rrltype to be whitelisted for the zone. Default has none whitelisted. The rrltype is the query classification that the NSD RRL employs to make different types not interfere with one another. The types are logged in the loglines when a subnet is blocked (in verbosity 2). The RRL classification types are: nxdomain, error, referral, any, rrsig, wildcard, nodata, dnskey, positive, all.

multi–primary–check: <yes or no>
Default no. If enabled, checks all primaries for the last version. It uses the higher version of all the configured primaries. Useful if you have multiple primaries that have different version numbers served.

verify–zone: <yes or no>
Enable or disable verification for this zone. Default is value–zones configured in **verify:**.

verifier: <command>
Command to execute to assess this zone. Default is verifier configured in **verify:**.

verifier-feed-zone: <yes or no>
Feed updated zone to verifier over standard input. Default is verifier–feed–zone configured in **verify:**.

create–ixfr: <yes or no>
If enabled, IXFR data is created when a zonefile is read by the server. This requires store–ixfr to be set to yes, so that the IXFR contents are saved to disk. Default is off. If the server is not running, the nsd–checkzone –i option can be used to create an IXFR file. When an IXFR is created, the server spools a version of the zone to a temporary file, at the location where the ixfr files are stored. This creates IXFR data when the zone is read from file, but not when a zone is read by AXFR transfer from a server, because then the topmost server that originates the data is the one place where IXFR differences are computed and those differences are then transmitted verbatim to all the other servers.

max–refresh–time: <seconds>
Limit refresh time for secondary zones. This is the timer which checks to see if the zone has to be refetched when it expires. Normally the value from the SOA record is used, but this option restricts that value.

min–refresh–time: <seconds>
Limit refresh time for secondary zones.

max–retry–time: <seconds>
Limit retry time for secondary zones. This is the timer which retries after a failed fetch attempt for the zone. Normally the value from the SOA record is used, followed by an exponential backoff, but this option restricts that value.

min–retry–time: <seconds>
Limit retry time for secondary zones.

min–expire–time: <seconds or refresh+retry+1>
Limit expire time for secondary zones. The value can be expressed either by a number of seconds, or the string "refresh+retry+1". With the latter the expire time will be lower bound to the refresh plus the retry value from the SOA record, plus 1. The refresh and retry values will be subject to the bounds configured with max–refresh–time, min–refresh–time, max–retry–time and min–retry–time if given.

rrl–whitelist–ratelimit: <qps>
The max qps for query sorts for a source, which have been whitelisted. Default on (with a suggested 2000 qps). With the rrl–whitelist option you can set specific queries to receive this qps limit instead of the normal limit. With the value 0 the rate is unlimited.

answer–cookie: <yes or no>
Enable to answer to requests containing DNS Cookies as specified in RFC7873. Default is no.

DNS Cookies increase transaction security and provide limited protection against denial-off-service amplification attacks. Server cookies will be created and included in responses. Server cookies are created based on the client cookie in the request, the current time, the client's IP address and a secret. When a client includes a valid server cookie in successive requests, the client will not be subjected to Request Rate Limiting (see **rrl–ratelimit**).

Servers in an anycast deployment need to be able to verify each other's server cookies. For this they need to share the secret used to construct and verify the cookies. These cookie secrets can be specified in the configuration files with the **cookie–secret** and **cookie–staging–secret** options.

If no cookie secrets are provided via configuration file, server cookie secrets can be added, dropped and activated with the *nsd–control*(8) tool. These secrets will be stored persistently in the cookie secret file for which the location can be specified with the **cookie–secret–file** option.

If no cookie secrets are provided via configuration file, and there is no or an empty cookie secret file, a random cookie secret is generated.

cookie–secret: <128 bit hex string>
The cookie secret with which server cookies are created and can be verified. If a **cookie–secret** is specified via configuration file, cookie secrets from the cookie secret file will be ignored.

cookie–staging–secret: <128 bit hex string>
A cookie secret with which server cookies can be verified, but will not be created. This is helpful in rolling cookie secrets in anycast setups.

A **cookie–staging–secret** can only be configured when there is also a **cookie–secret** configured.

cookie–secret–file: <filename>
File from which the secrets are read used in DNS Cookie calculations. Secrets will only be read from this file if no cookie secrets are given in the configuration file via the **cookie–secret** and **cookie–staging–secret** options. Default is "//var/db/nsd/cookiesecrets.txt"

In NSD version 4.10.1 and earlier, the default location of the cookie secret file was "//etc/nsd/nsd_cookiesecrets.txt". For migration purposes, cookie secrets will be read from that location if no value is given for the **cookie–secret–file** option and when the current default location ("//var/db/nsd/cookiesecrets.txt") does not exist.

must present a certificate with a SAN (Subject Alternative Name) DNS entry or CN (Common Name) entry equal to **auth-domain-name** of the defined **tls-auth**. The certificate validify is also verified with **tls-cert-bundle**. If authentication of the secondary, based on the specified tls-auth authentication information, fails the XFR zone transfer will be refused. If the connection is performed on the **tls-port** then no authentication will be performed and the transfer will not be refused. To enforce only authenticated zone transfers, **tls-auth-xfr-only** should also be enabled. Support for TLS 1.3 is required for XFR-over-TLS.

outgoing–interface: <ip–address>
Access control list. The listed address is used to request AXFR|IXFR (in case of a secondary) or used to send notifies (in case of a primary).

The ip–address is a plain IP address (IPv4 or IPv6). A port number can be added using a suffix of @number, for example 1.2.3.4@5300.

store–ixfr: <yes or no>
If enabled, IXFR contents are stored and provided to the set of clients specified in the provide–xfr statement. Default is no. IXFR content is a smaller set of changes that differ between zone versions, whereas an AXFR contains the full contents of the zone.

ixfr–number: <number>
The number of IXFR versions to store for this zone, at most. Default is 5.

ixfr–size: <number>
The max storage to use for IXFR versions for this zone, in bytes. Default is 1048576. A value of 0 means unlimited. If you want to turn off IXFR storage, set the store–ixfr option to no. NSD does not elide IXFR contents from versions that add and remove the same data. It stores and transmits IXFRs as they were transmitted by the upstream server.

size–limit–xfr: <number>
This option should be accompanied by request–xfr. It specifies XFR temporary file size limit. It can be used to stop very large zone retrieval, that could otherwise use up a lot of memory and disk space. If this option is 0, unlimited. Default value is 0.

notify: <ip–address> <key–name | NOKEY>
Access control list. The listed address (a secondary) is notified of updates to this zone via UDP. A port number can be added using a suffix of @number, for example 1.2.3.4@5300. The specified key is used to sign the notify. Only on secondary configurations will NSD be able to detect zone updates (as it gets notified itself, or refreshes after a time).

notify–retry: <number>
This option should be accompanied by notify. It sets the number of retries when sending notifies.

provide–xfr: <ip–spec> <key–name | NOKEY | BLOCKED> [tls–auth–name]
Access control list. The listed address (a secondary) is allowed to request XFR from this server. Zone data will be provided to the address. The specified key is used during XFR. For unlisted or BLOCKED addresses no data is provided and requests are discarded. BLOCKED supersedes other entries and other entries are scanned for a match in the order of the statements.

The ip–spec is either a plain IP address (IPv4 or IPv6), or can be a subnet of the form 1.2.3.4/24, or masked like 1.2.3.4&255.255.255.0 or a range of the form 1.2.3.4–1.2.3.25. A port number can be added using a suffix of @number, for example 1.2.3.4@5300 or 1.2.3.4/24@5300 for port 5300. Note the ip–spec ranges do not use spaces around the /, &, @ and – symbols.

If a tls-auth-name is given then TLS authentication of the secondary will be performed for zone transfer requests for the zone. The remote end must connect to the **tls-auth-port** and

The content of the cookie secret file must be manipulated with the **add_cookie_secret**, **drop_cookie_secret** and **activate_cookie_secret** commands to the *nsd–control*(8) tool. Please see that manpage how to perform a safe cookie secret rollover.

tls–service–key: <filename>
If enabled, the server provides TLS service on TCP sockets with the TLS service port number. The port number (853) is configured with tls–port. To turn it on, create an interface: option line in config with @port appended to the IP-address. This creates the extra socket on which the DNS over TLS service is provided.

The file is the private key for the TLS session. The public certificate is in the tls-service-pem file. Default is "", turned off. Requires a restart (a reload is not enough) if changed, because the private key is read while root permissions are held and before chroot (if any).

tls–service–pem: <filename>
The public key certificate pem file for the tls service. Default is "", turned off.

tls–service–ocsp: <filename>
The ocsp pem file for the tls service, for OCSP stapling. Default is "", turned off. An external process prepares and updates the OCSP stapling data. Like this,

```
openssl ocsp -no_nonce \
        -respout /path/to/ocsp.pem \
        -CAfile /path/to/ca_and_any_intermediate.pem \
        -issuer /path/to/direct_issuer.pem \
        -cert /path/to/cert.pem \
        -url "$( openssl x509 -noout -ocsp_uri -in /path/to/cer
```

tls−port: <number>
The port number on which to provide TCP TLS service, default is 853, only interfaces configured with that port number as @number get DNS over TLS service.

tls−auth−port: <number>
The port number on which to provide TCP TLS service to authenticated clients only. If you want to use mutual TLS authentication in Transfer over TLS (XoT) connections, this is where the primary server enables a dedicated port for this purpose. Certificates in **tls-cert-bundle** are used for verifying the authenticity of a client or a secondary server.

Client (secondary) must enable **tls-auth**, configure **client-cert** and **client-key** and enable **tls-auth** in zone configuration in order to authenticate to a remote (primary) server.

tls−auth−xfr−only: <yes or no>
Allow zone transfers only on the **tls-auth-port** port and only to authenticated clients. This works globally for all zones. A **provide-xfr** access control list with **tls-auth** is also required to allow and verify a connection. Requests for zone transfers on other ports are refused.

tls−cert−bundle: <filename>
If null or "", the default verify locations are used. Set it to the certificate bundle file, for example "/etc/pki/tls/certs/ca-bundle.crt". These certificates are used for authenticating Transfer over TLS (XoT) connections.

proxy−protocol−port: <number>
The port number for proxy protocol service. If the statement is given multiple times, additional port numbers can be used for proxy

1.2.3.4@5300 or 1.2.3.4/24@5300 for port 5300. Note the ip−spec ranges do not use spaces around the /, &, @ and − symbols.

request−xfr: [AXFR|UDP] <ip−address> <key−name | NOKEY> [tls−auth−name]
Access control list. The listed address (the primary) is queried for AXFR/IXFR on update. A port number can be added using a suffix of @number, for example 1.2.3.4@5300. The specified key is used during AXFR/IXFR. If tls-auth-name is included, the specified tls-auth clause will be used to perform authenticated XFR-over-TLS.

If the AXFR option is given, the server will not be contacted with IXFR queries but only AXFR requests will be made to the server. This allows an NSD secondary to have a primary server that runs NSD. If the AXFR option is left out then both IXFR and AXFR requests are made to the primary server.

If the UDP option is given, the secondary will use UDP to transmit the IXFR requests. You should deploy TSIG when allowing UDP transport, to authenticate notifies and zone transfers. Otherwise, NSD is more vulnerable for Kaminsky−style attacks. If the UDP option is left out then IXFR will be transmitted using TCP.

If a tls-auth-name is given then TLS (by default on port 853) will be used for all zone transfers for the zone. If authentication of the primary, based on the specified tls-auth authentication information, fails the XFR request will not be sent. Support for TLS 1.3 is required for XFR-over-TLS.

allow−axfr−fallback: <yes or no>
This option should be accompanied by request−xfr. It (dis)allows NSD (as secondary) to fallback to AXFR if the primary name server does not support IXFR. Default is yes.

allow–query option is specified, then the specified addresses in the **allow–query** options are allowed to query the server for the zone. Queries from unlisted or specifically BLOCKED addresses are discarded. If NOKEY is given no TSIG signature is required. BLOCKED supersedes other entries, other entries are scanned for a match in the order of the statements. Without **allow–query** options, queries are allowed from any IP address without TSIG key (which is the default).

The ip–spec is either a plain IP address (IPv4 or IPv6), or can be a subnet of the form 1.2.3.4/24, or masked like 1.2.3.4&255.255.255.0 or a range of the form 1.2.3.4–1.2.3.25. Note the ip–spec ranges do not use spaces around the /, &, @ and – symbols.

allow–notify: <ip–spec> <key–name | NOKEY | BLOCKED>

Access control list. The listed (primary) address is allowed to send notifies to this (secondary) server via UDP or TCP. Notifies from unlisted or specifically BLOCKED addresses are discarded. If NOKEY is given no TSIG signature is required. BLOCKED supersedes other entries, other entries are scanned for a match in the order of the statements.

The ip–spec is either a plain IP address (IPv4 or IPv6), or can be a subnet of the form 1.2.3.4/24, or masked like 1.2.3.4&255.255.255.0 or a range of the form 1.2.3.4–1.2.3.25. A port number can be added using a suffix of @number, for example

protocol service. The interface definitions that use this port number expect PROXYv2 proxy protocol traffic, for UDP, TCP and for TLS service.

metrics–enable: <yes or no>

Enable the prometheus metrics HTTP endpoint. It exposes the same statistics as the *nsd–control*(8) stats_noreset command, but with metric names following the prometheus specification. (Requires libevent2)

Beware, that when using *nsd–control*(8) stats (instead of stats_noreset), the statistics will be reset for the HTTP metrics endpoint as well.

metrics–interface: <ip4 or ip6 | interface name>

NSD will bind to the listed addresses or interfaces to serve the prometheus metrics. Can be given multiple times to bind multiple ip–addresses. Use 0.0.0.0 and ::0 to bind to the wildcard interface.

If an interface name is used instead of ip4 or ip6, the list of IP addresses associated with that interface is picked up and used at server start.

Default is 127.0.0.1 and ::1.

metrics–port: <number>

The port number for the HTTP service. Default is 9100.

metrics–path: <string>

The HTTP path to expose the metrics at. Default is "/metrics".

Remote Control

The **remote–control:** clause is used to set options for using the *nsd–control*(8) tool to give commands to the running NSD server. It is disabled by default, and listens for localhost by default. It uses TLS over TCP where the server and client authenticate to each other with self–signed certificates. The self–signed certificates can be generated with the *nsd–control–setup* tool. The key files are read by NSD before the chroot and before dropping user permissions, so they can be outside the chroot and readable by the superuser only.

control–enable: <yes or no>
Enable remote control, default is no.

control–interface: <ip4 or ip6 | interface name | absolute path>
NSD will bind to the listed addresses to service control requests (on TCP). Can be given multiple times to bind multiple ip–addresses. Use 0.0.0.0 and ::0 to service the wildcard interface. If none are given NSD listens to the localhost 127.0.0.1 and ::1 interfaces for control, if control is enabled with control–enable.

If an interface name is used instead of ip4 or ip6, the list of IP addresses associated with that interface is picked up and used at server start.

With an absolute path, a unix local named pipe is used for control. The file is created with user and group that is configured and access bits are set to allow members of the group access. Further access can be controlled by setting permissions on the directory containing the control socket file. The key and cert files are not used when control is via the named pipe, because access control is via file and directory permission.

"example.com", "sub.example.net.". This attribute must be present in each zone.

zonefile: <filename>
The file containing the zone information. If this attribute is present it is used to read and write the zone contents. If the attribute is absent it prevents writing out of the zone.

The string is processed so that one string can be used (in a pattern) for a lot of different zones. If the label or character does not exist the percent-character is replaced with a period for output (i.e. for the third character in a two letter domain name).

%s is replaced with the zone name.

%1 is replaced with the first character of the zone name.

%2 is replaced with the second character of the zone name.

%3 is replaced with the third character of the zone name.

%z is replaced with the toplevel domain name of the zone.

%y is replaced with the next label under the toplevel domain.

%x is replaced with the next-next label under the toplevel domain.

allow–query: <ip–spec> <key–name | NOKEY | BLOCKED>
Access control list. When at least one

name: <string>
The name of the pattern. This is a (case sensitive) string. The pattern names that start with "_implicit_" are used internally for zones that have no pattern (they are defined in nsd.conf directly).

include–pattern: <pattern–name>
The options from the given pattern are included at this point in this pattern. The referenced pattern must be defined above this one.

<zone option>: <value>
The zone options such as **zonefile**, **allow–query**, **allow–notify**, **request–xfr**, **allow–axfr–fallback**, **notify**, **notify–retry**, **provide–xfr**, **store–ixfr**, **ixfr–number**, **ixfr–size**, **create–ixfr**, **zonestats**, **outgoing–interface**, **verify–zone**, **verifier**, **verifier–feed–zone**, **verifier–timeout**, **catalog**, and **catalog–member–pattern** can be given. They are applied to the patterns and zones that include this pattern.

Zone Options

For every zone the options need to be specified in one **zone:** clause. The access control list elements can be given multiple times to add multiple servers. These elements need to be added explicitly.

For zones that are configured in the *nsd.conf* config file their settings are hardcoded (in an implicit pattern for themselves only) and they cannot be deleted via delzone, but remove them from the config file and repattern.

name: <string>
The name of the zone. This is the domain name of the apex of the zone. May end with a '.' (in FQDN notation). For example

control–port: <number>
The port number for remote control service. 8952 by default.

server–key–file: <filename>
Path to the server private key, by default *//etc/nsd/nsd_server.key*. This file is generated by the *nsd–control–setup* utility. This file is used by the nsd server, but not by *nsd–control*.

server–cert–file: <filename>
Path to the server self signed certificate, by default *//etc/nsd/nsd_server.pem*. This file is generated by the *nsd–control–setup* utility. This file is used by the nsd server, and also by *nsd–control*.

control–key–file: <filename>
Path to the control client private key, by default *//etc/nsd/nsd_control.key*. This file is generated by the *nsd–control–setup* utility. This file is used by *nsd–control*.

control–cert–file: <filename>
Path to the control client certificate, by default *//etc/nsd/nsd_control.pem*. This certificate has to be signed with the server certificate. This file is generated by the *nsd–control–setup* utility. This file is used by *nsd–control*.

Verifier options

The **verify:** clause is used to enable or disable zone verification, configure listen interfaces and control the global defaults.

enable: <yes or no>
Enable zone verification. Default is no.

port: <number>
The port to answer verifier queries on. Default is 5347.

ip–address:
Interfaces to bind for zone verification (default are the localhost interfaces, usually 127.0.0.1 and ::1). To bind to multiple IP addresses, list them one by one. Optionally, Socket options cannot be specified for verify ip-address

verify–zones: <yes or no>
Verify zones by default.

verifier: <command>
When an update is received for the zone (by IXFR or AXFR) this program will be run to assess the zone with the update. If the program exits with a status code of 0, the zone is considered good and will be served. Any other status code will designate the zone bad and the received update will be discarded. The zone will continue to be served but without the update.

The following environment variables are available to verifiers:

VERIFY_ZONE
The domain name of the zone to be verified.

VERIFY_ZONE_ON_STDIN
When the zone can be read from standard input (stdin), this variable is set to "yes", otherwise it is set to "no".

VERIFY_IP_ADDRESSES
The first address on which the zones to be assessed will be served. If IPv6 is available an IPv6 address will be preferred over IPv4.

VERIFY_PORT
The port number for **VERIFY_IP_ADDRESS**.

VERIFY_IPV6_ADDRESS
The first IPv6 address on which the zones to be assessed will be served.

VERIFY_IPV6_PORT
The port number for **VERIFY_IPV6_ADDRESS**.

VERIFY_IPV4_ADDRESS
The first IPv4 address of which the zones to be assessed will be served.

VERIFY_IPV4_PORT
The port number for **VERIFY_IPV4_ADDRESS**.

verifier–count: <number>
Maximum number of verifiers to run concurrently. Default is 1.

verifier–feed–zone: <yes or no>
Feed the updated zone to the verifier over standard input (stdin).

verifier–timeout: <seconds>
The maximum number of seconds a verifier is allowed to run for assessing one zone. If the verifier takes longer, it will be terminated and the zone update will be discarded. The default is 0 seconds which means the verifier may take as long as it needs.

Pattern Options

The **pattern:** clause is used to denote a set of options to apply to some zones. The same zone options as for a zone are allowed.