NAME

ntp.conf − Network Time Protocol (NTP) daemon configuration file format

SYNOPSIS

/etc/ntpsec/ntp.conf

DESCRIPTION

The ntp.conf configuration file is read at initial startup by the ntpd(8) daemon in order to specify the synchronization sources, modes, and other related information. Usually, it is installed in the /etc directory, but could be installed elsewhere (see the daemon's −c command line option).

The file format is similar to other UNIX configuration files. Comments begin with a '#' character and extend to the end of the line; blank lines are ignored. Configuration commands consist of an initial keyword followed by a list of arguments, some of which may be optional, separated by whitespace. Commands may not be continued over multiple lines. Arguments may be host names, host addresses written in numeric, dotted−quad form, integers, floating point numbers (when specifying times in seconds) and text strings.

Configuration files may have inclusion lines. The syntax is includefile followed by whitespace followed by a file or directory name. The configuration is evaluated as though the text of the file − or all files of the directory with the extension ".conf" − were textually spliced in at the point of the include. Relative paths will work, even when the −c option changes the config directory root.

The rest of this page describes the configuration and control options. The "Notes on Configuring NTP and Setting up an NTP Subnet" page (available as part of the HTML documentation provided under /usr/share/doc/ntp) contains an extended discussion of these options. In addition to the discussion of general *Configuration Options*, there are sections describing the following supported functionality and the options used to control it:

- Authentication Support
- NTS Support
- Monitoring Support
- Access Control Support
- Automatic NTP Configuration Options
- Reference Clock Support
- Miscellaneous Options

Following these is a section describing *Miscellaneous Options*. While there is a rich set of options available, the only required option is one or more pool, server, peer, or broadcast commands.

CONFIGURATION SUPPORT

Following is a description of the configuration commands in NTPv4. There are two classes of commands, association commands that configure a persistent association with a remote server or peer or reference clock, and auxiliary commands that specify environment variables that control various related operations.

Association Commands

Only those options applicable to each command are listed below. Use of options not listed may not be caught as an error, but may result in some weird and even destructive behavior.

In contexts where a host name is expected, a −4 or −−ipv4 qualifier preceding the host name forces DNS resolution to the IPv4 namespace, while a −6 or −−ipv6 qualifier forces DNS resolution to the IPv6 namespace.

In these commands, an *address* can be any of (a) an IPV4 address in a.b.c.d format, (b) an IPV6 address in [a:b:c:d:e:f:g] format, (c) a link−local IPV6 address with an interface specified in [a:b:c:d:e:f:g]%device format, or (d) a DNS hostname.

pool *address* [burst] [iburst] [version *version*] [prefer] [minpoll *minpoll*] [maxpoll *maxpoll*] [preempt]

server *address* [key *key*] [burst] [iburst] [version *version*] [prefer] [minpoll *minpoll*] [maxpoll *maxpoll*]

peer *address* [key *key*] [version *version*] [prefer] [minpoll *minpoll*] [maxpoll *maxpoll*]

unpeer [*address* | *associd* | clock *clocktype* [unit *unitnum*]]

These four commands specify the time server name or address to be used and the mode in which to operate. The *address* can be either a DNS name or an IP address in dotted−quad notation. If it is a refclock, it can be clock followed by a type−unit pair as in the refclock directive; a missing unit clause is interpreted as unit 0.

pool

For server addresses, this command mobilizes a persistent client mode association with a number of remote servers. In this mode the local clock can synchronized to the remote server, but the remote server can never be synchronized to the local clock.

server

For server addresses, this command mobilizes a persistent client mode association with the specified remote server or local radio clock. In this mode the local clock can synchronized to the remote server, but the remote server can never be synchronized to the local clock.

peer

NTP peer mode has been removed for security reasons. peer is now just an alias for the server keyword. See above.

unpeer

This command removes a previously configured association. An address or association ID can be used to identify the association. Either an IP address or DNS name can be used. This command is most useful when supplied via ntpq runtime configuration commands config and config−from−file.

Association Options

bias

Add the command argument, a floating−point value in seconds, to the time offset () computed for this server; this may be useful if you are a client on a network connection such as an ADSL line where there is a predictable asymmetry between upstream and downstream flight times. One way you might see this is if you use a fixed set of others and one has a stable offset that is an outlier from the others; in that case, you might want to use bias to compensate out the offset.

burst

When the server is reachable, send a burst of eight packets instead of the usual one. The packet spacing is normally 2 s; however, the spacing between the first and second packets can be changed with the calldelay command to allow additional time for a modem or ISDN call to complete; this is designed to improve timekeeping quality with the server command.

iburst

When the server is unreachable, send a burst of six packets instead of the usual one. The packet spacing is normally 2 s; however, the spacing between the first and second packets can be changed with the calldelay command to allow additional time for a modem or ISDN call to complete; this is designed to speed the initial synchronization acquisition with the server command, and when ntpd(8) is started with the −q option.

key *key*

All packets sent to and received from the server or peer are to include authentication fields encrypted using the specified *key* identifier with values from 1 to 65535, inclusive. The default is to include no encryption field.

minpoll *minpoll*, maxpoll *maxpoll*

These options specify the minimum and maximum poll intervals for NTP messages, as a power of 2 in seconds. The maximum poll interval defaults to 10 (1,024 s), but can be increased by the *maxpoll* option to an upper limit of 17 (36.4 h). The minimum poll interval defaults to 6 (64 s), but can be decreased by the *minpoll* option to a lower limit of 0 (1 s).

mode *option*

Pass the option to a reference clock driver. This option is valid only with refclock addresses.

noselect

Marks the server as unused, except for display purposes. The server is discarded by the selection algorithm.

prefer

Marks the server as preferred. All other things being equal, this host will be chosen for synchronization among a set of correctly operating hosts. See the "Mitigation Rules and the prefer Keyword" page for further information.

true

Mark the association to assume truechimer status; that is, always survive the selection and clustering algorithms. This option can be used with any association but is most useful for reference clocks with large jitter on the serial port and precision pulse−per−second (PPS) signals. Caution: this option defeats the algorithms designed to cast out falsetickers and can allow these sources to set the system clock. This option is valid only with the server command.

version *version*

Specifies the version number to be used for outgoing NTP packets. Versions 1−4 are the choices, with version 4 the default.

Association Auxiliary Commands

mdnstries *number*

If we are participating in mDNS, after we have synched for the first time we attempt to register with the mDNS system. If that registration attempt fails, we try again at one minute intervals for up to *number* times. After all, ntpd may be starting before mDNS. The default value for mdnstries is 5.

Authentication Commands

The following declarations control MAC authentication:

controlkey *key*

Specifies the key identifier to use with the ntpq(1) utility, which uses the standard protocol defined in RFC 5905. The *key* argument is the key identifier for a trusted key, where the value can be in the range 1 to 65,535, inclusive.

keys *keyfile*

Specifies the complete path and location of the key file containing the keys and key identifiers used by ntpd(8), and ntpq(1) when operating with symmetric−key cryptography. This is the same operation as the −k command line option.

trustedkey *key...*

Specifies the key identifiers which are trusted for the purposes of authenticating peers with symmetric key cryptography, as well as keys used by the ntpq(1) program. Multiple keys on the same line should be separated by spaces. Key ranges can be specified as (first ... last). The spaces around the ... are necessary. Multiple trustedkey lines are supported and trusted keys can also be specified on the command line.

The MAC authentication procedures require that both the local and remote servers share the same key and key identifier for this purpose, although different keys can be used with different servers. The *key* arguments are 32−bit unsigned integers with values from 1 to 65,535.

NTS Commands

The following command controls NTS authentication. It overrides normal TLS protocol negotiation, which is not usually necessary.

nts [enable|disable] [mintls *version*] [maxtls *version*] [tlsciphers *name*] [tlsciphersuites *name*]

The options are as follows:

cert *file*
: Present the certificate in *file* as our certificate.

key *file*
: Read the private key to our certificate from *file*.

ca *location*
: Use the file, or directory, specified by *location* to validate NTS−KE server certificates instead of the system default root certificates. If a directory is specified, it must have files named with their hash, as created by openssl rehash.

cookie *location*
: Use the file (or directory) specified by *location* to store the keys used to make and decode cookies. The default is */var/lib/ntpsec/nts−keys*.

enable
: Enable NTS−KE server. When enabled, cert and key are required.

disable
: Disable NTS−KE server.

mintls *string*
: Set the lowest allowable TLS version to negotiate. Will be useful in the wake of a TLS compromise. Reasonable values are *TLS1.2* and *TLS1.3* if your system supports it. TLS 1.3 was first supported in OpenSSL version 1.1.1.

maxtls *string*
: Set the highest allowable TLS version to negotiate. By setting mintls and maxtls equal, you can force the TLS version for testing. Format is as for mintls.

tlsciphers *string*
: An OpenSSL cipher list to configure the allowed ciphers for TLS versions up to and including TLS 1.2. A single NULL cipher disables encryption and use of certificates.

tlsciphersuites *string*
: An OpenSSL ciphersuite list to configure the allowed ciphersuites for TLS 1.3. A single NULL cipher disables encryption and use of certificates.

aead *string*
: Specify the crypto algorithm to be used on the wire. The choices come from RFC 5297. The only options supported are AES_SIV_CMAC_256, AES_SIV_CMAC_384, and AES_SIV_CMAC_512. This slot is dual use. It is the server default if the remote client doesn't request a valid choice and it is also the preference passed to the remote client if the server command doesn't specify a preference. The default is AES_SIV_CMAC_256.

The following options of the server command configure NTS (as a client).

nts
: Use Network Time Security (NTS) for authentication. Normally, this is all you have to do to activate the client side of NTS.

  The hostname following the server command is used as the address of the NTS key exchange server (NTS−KE) rather than the address of a NTP server. The NTS−KE exchange defaults to using the same IP address for the NTP server.

  Note that the server hostname must match the name on the NTS−KE server's certificate.

ask *address*
: (not implemented) Use Network Time Security for authentication. Ask for a specific NTP server, which may differ from the NTS server. Conforms to RFC 3896 section 3.2.2 prescription for the Host

part of a URI: that is, the *address* may be a hostname, an FQDN, an IPv4 numeric address, or an IPv6 numeric address (in square brackets). The address may have the suffix :port to specify a UDP port.

require *address*

(not implemented) Use Network Time Security for authentication and encryption. Require a specific NTP server, which may differ from the NTS server. Address syntax is as for ask.

noval

Do not validate the server certificate.

expire

(not implemented) How long to use a secured NTP association before rekeying with the NTS−KE server.

cert *file*

(not implemented) Present the certificate in *file* as our client certificate, overriding the site default.

ca *location*

Use the file, or directory, specified by *location* to validate the NTS−KE server certificate, overriding the site default. Do not use any other CA. If a directory is specified, it must have files named with their hash, as created by openssl rehash.

aead *string*

Specify the preferred crypto algorithm to be used on the wire. The only options supported are AES_SIV_CMAC_256, AES_SIV_CMAC_384, and AES_SIV_CMAC_512. The server may ignore the request. See the aead option above.

The same aead algorithms are also used to encrypt cookies. The default is AES_SIV_CMAC_256. There is no config file option to change it, but you can change it by editing the saved cookie key file, probably */var/lib/ntpsec/nts−keys*. Adjust the *L:* slot to be 48 or 64 and adjust the *I:* slots to have the right number of bytes. Then restart the server. (All old cookies held by clients will be rejected so their next 8 NTP requests will be ignored. They should recover by retrying NTS−KE to get fresh cookies.)

MONITORING SUPPORT

ntpd(8) includes a comprehensive monitoring facility suitable for continuous, long term recording of server and client timekeeping performance. See the statistics command below for a listing and example of each type of statistics currently supported. Statistic files are managed using file generation sets and scripts in the ./scripts directory of this distribution. Using these facilities and UNIX cron(8) jobs, the data can be automatically summarized and archived for retrospective analysis.

Monitoring Commands

statistics *name...*

Enables writing of statistics records. Currently, eight kinds of *name* statistics are supported.

clockstats

Enables recording of clock driver statistics information. Each update received from a clock driver appends a line of the following form to the file generation set named *clockstats*:

```
49213 525.624 SPECTRACOM(1) 93 226 00:08:29.606
```

Item	Units	Description
49213	MJD	modified Julian day number
525.624	s	time of day (s) past midnight UTC
SPECTRACOM(1)		receiver identifier (Spectracom unit 1)
93 226 00:08:29.606		timecode (format varies by refclock)

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next normally shows clock type and unit (but if you are running in strict Classic compatibility mode it will show the magic clock address in dotted−quad notation). The final field is the last timecode received from the clock in decoded ASCII format, where meaningful. For some clock drivers, a good deal of additional information can be gathered and displayed as well. See information specific to each clock for further details.

loopstats

Enables recording of loop filter statistics information. Each update of the local clock outputs a line of the following form to the file generation set named *loopstats*:

```
50935 75440.031 0.000006019 13.778190 0.000351733 0.0133806
```

Item	Units	Description
50935	MJD	date
75440.031	s	time past midnight
0.000006019	s	clock offset
13.778	PPM	drift (frequency offset)
0.000351733	s	RMS jitter
0.013380	PPM	RMS frequency jitter (aka wander)
6	log2 s	clock discipline loop time constant

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next five fields show time offset (seconds), frequency offset (parts per million − PPM), RMS jitter (seconds), Allan deviation (PPM) and clock discipline time constant.

protostats

Record significant peer and system events. Each significant event appends one line to the protostats file set:

49213 525.624 128.4.1.1 963a 8a *message*

Item	Units	Description
49213	MJD	date
525.624	s	time past midnight
128.4.1.1	IP	source address (0.0.0.0 for system)
963a	code	status word
8a	code	event message code
message	text	event message

The event message code and *message* field are described on the "Event Messages and Status Words" page.

peerstats

Enables recording of peer statistics information. This includes statistics records of all peers of an NTP server and of special signals, where present and configured. Each valid update appends a line of the following form to the current element of a file generation set named *peerstats*:

```
48773 10847.650 SPECTRACOM(4) 9714 −0.001605376 0.000000000
   0.001424877 0.000958674
```

Item	Units	Description
48773	MJD	date
10847.650	s	time past midnight
SPECTRACOM(4)		clock name (unit) or source address
9714	hex	status word
−0.001605376	s	clock offset
0.000000000	s	roundtrip delay
0.001424877	s	dispersion
0.000958674	s	RMS jitter

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The third field shows the reference clock type and unit number (but if you are running in the peer address in dotted−quad notation instead) The fourth field is a status word, encoded in hex in the format described in Appendix A of the NTP specification RFC 1305. The final four fields show the offset, delay, dispersion and RMS jitter, all in seconds.

rawstats

Enables recording of raw−timestamp statistics information. This includes statistics records of all peers of an NTP server and of special signals, where present and configured. Each NTP message received from a peer or clock driver appends a line of the following form to the file generation set named *rawstats*:

```
56285 54575.160 128.4.1.1 192.168.1.5 3565350574.400229473
   3565350574.442385200 3565350574.442436000
   3565350575.154505763 0 4 4 1 8 −21 0.000000 0.000320
   PPS 0
```

Item	Units	Description
56285	MJD	date
54575.160	s	time past midnight
128.4.1.1	IP	source address
192.168.1.5	IP	destination address
3565350574.400229473	NTP s	origin timestamp
3565350574.442385200	NTP s	receive timestamp
3565350574.442436000	NTP s	transmit timestamp
3565350575.154505763	NTP s	destination timestamp
0	0: OK, 1: insert pending, 2: delete pending, 3: not synced	leap warning indicator
4	4 was current in 2012	NTP version
4	3: client, 4: server, 6: ntpq	mode
1	1−15, 16: not synced	stratum
8	log2 seconds	poll
−21	log2 seconds	precision
0.000000	seconds	total roundtrip delay from the remote server to the primary reference clock
0.000320	seconds	total dispersion from the remote server to the primary reference clock
.PPS.	IP or text	refid, association ID
0	integer	lost packets since last response

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the remote peer or clock identification followed by the local address in dotted−quad notation. The final four fields show the originate, receive, transmit and final NTP timestamps in order. The timestamp values are as received and before processing by the various data smoothing and mitigation algorithms.

sysstats

Enables recording of ntpd statistics counters on a periodic basis. Each hour a line of the following form is appended to the file generation set named *sysstats*:

```
50928 2132.543 36000 81965 0 9546 56 71793 512 540 10 147 1
```

Item	Units	Description
50928	MJD	date
2132.543	s	time past midnight
3600	s	time since reset
81965	#	packets received
0	#	packets for this host
9546	#	current versions
56	#	old version
512	#	access denied
540	#	bad length or format
10	#	bad authentication
4	#	declined
147	#	rate exceeded
1	#	kiss−o'−death packets sent

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The remaining ten fields show the statistics counter values accumulated since the last generated line.

usestats

Enables recording of ntpd resource usage statistics. Each hour a line of the following form is appended to the file generation set named *usestats*:

```
57570 83399.541 3600 0.902 1.451 164 0 0 0 2328 64226 1 0 4308
```

Item	Units	Description
57570	MJD	date
83399.541	s	time past midnight
3600	s	time since reset
0.902	s	ru_utime: CPU seconds − user mode
1.451	s	ru_stime: CPU seconds − system
164	#	ru_minflt: page faults − reclaim/soft (no I/O)
0	#	ru_majflt: page faults − I/O
0	#	ru_nswap: process swapped out
0	#	ru_inblock: file blocks in
2328	#	ru_oublock: file blocks out
64226	#	ru_nvcsw: context switches, wait
1	#	ru_nivcsw: context switches, preempts
0	#	ru_nsignals: signals
4308	#	ru_maxrss: resident set size, kilobytes

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The ru_ tags are the names from the rusage struct. See man getrusage for details. (The

NetBSD and FreeBSD man pages have more details.) The maxrss column is the high water mark since the process was started. The remaining fields show the values used since the last report.

statsdir *directory_path*

Indicates the full path of a directory where statistics files should be created (see below). This keyword allows the (otherwise constant) *filegen* filename prefix to be modified for file generation sets, which is useful for handling statistics logs.

filegen *name* [file *filename*] [type *typename*] [link | nolink] [enable | disable]

Configures setting of the generation file set name. Generation file sets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. Generation file sets provide access to a set of files used to store the actual data. At any time at most one element of the set is being written to. The type given specifies when and how data will be directed to a new element of the set. This way, information stored in elements of a file set that are currently unused are available for administrative operations without the risk of disturbing the operation of ntpd. (Most important: they can be removed to free space for new data produced.)

Note that this command can be sent from the ntpq(1) program running at a remote location.

name

This is the type of the statistics records, as shown in the *statistics* command.

file *filename*

This is the file name for the statistics records. Filenames of set members are built from three concatenated elements *prefix*, *filename* and *suffix*:

Attribute	Description
prefix	This is a constant filename path. It is not subject to modifications via the *filegen* option. It is defined by the server, usually specified as a compile–time constant. It may, however, be configurable for individual file generation sets via other commands. For example, the prefix used with *loopstats* and *peerstats* generation can be configured using the *statsdir* option explained above.
filename	This string is directly concatenated to the prefix mentioned above (no intervening '/'). This can be modified using the file argument to the *filegen* statement. No .. elements are allowed in this component to prevent filenames referring to parts outside the filesystem hierarchy denoted by *prefix*.
suffix	This part is reflects individual elements of a file set. It is generated according to the type of a file set.

type *typename*

A file generation set is characterized by its type. The following types are supported: // The following are tables only because indent lists cannot be // nested more than 2 deep.

link | nolink

It is convenient to be able to access the current element of a file generation set by a fixed name. This feature is enabled by specifying link and disabled using nolink. If link is specified, a hard link from the current file set element to a file without suffix is created. When there is already a file with this name and the number of links of this file is one, it is renamed appending a dot, the letter *C*, and the pid of the ntpd server process. When the number of links is greater than one, the file is unlinked. This allows the current file to be accessed by a constant name.

enable | disable

Enables or disables the recording function. Information is only written to a file generation by specifying enable; output is prevented by specifying disable.

ACCESS CONTROL SUPPORT

The ntpd(8) daemon implements a general purpose address/mask based restriction list. The list contains address/match entries sorted first by increasing address values and then by increasing mask values. A match occurs when the bitwise AND of the mask and the packet source address is equal to the bitwise AND of the mask and address in the list. The list is searched in order with the last match found defining the restriction flags associated with the entry. Additional information and examples can be found in the "Notes on Configuring NTP and Setting up a NTP Subnet" page (available as part of the HTML documentation).

The restriction facility was implemented in conformance with the access policies for the original NSFnet backbone time servers. Later the facility was expanded to deflect cryptographic and clogging attacks. While this facility may be useful for keeping unwanted or broken or malicious clients from congesting innocent servers, it should not be considered an alternative to the NTP authentication facilities. Source address based restrictions are easily circumvented by a determined cracker.

Clients can be denied service because they are explicitly included in the restrict list created by the restrict command or implicitly as the result of cryptographic or rate limit violations. Cryptographic violations include certificate or identity verification failures; rate limit violations generally result from defective NTP implementations that send packets at abusive rates. Some violations cause denied service only for the offending packet, others cause denied service for a timed period and others cause the denied service for an indefinite period. When a client or network is denied access for an indefinite period, the only way at present to remove the restrictions is by restarting the server.

The Kiss−of−Death Packet

Ordinarily, packets denied service are simply dropped with no further action except incrementing statistics counters. Sometimes a more proactive response is needed, such as a server message that explicitly requests the client to stop sending and leave a message for the system operator. A special packet format has been created for this purpose called the "kiss−of−death" (KoD) packet. KoD packets have the leap bits set unsynchronized and stratum set to zero and the reference identifier field set to a four−byte ASCII code. If the noserve or notrust flag of the matching restrict list entry is set, the code is "DENY"; if the limited flag is set and the rate limit is exceeded, the code is "RATE". Finally, if a cryptographic violation occurs, the code is "CRYP".

A client receiving a KoD performs a set of sanity checks to minimize security exposure, then updates the stratum and reference identifier peer variables, sets the access denied (BOGON4) bit in the peer flash variable and sends a message to the log. As long as the BOGON4 bit is set, the client will send no further packets to the server. The only way at present to recover from this condition is to restart the protocol at both the client and server. This happens automatically at the client when the association times out. It will happen at the server only if the server operator cooperates.

ACCESS CONTROL COMMANDS

discard [average *avg*] [minimum *min*] [monitor *prob*]

Set the parameters of the limited facility which protects the server from client abuse. The average subcommand specifies the minimum average packet spacing, while the minimum subcommand specifies the minimum packet spacing. Packets that violate these minima are discarded and a kiss−o'−death packet returned if enabled. The default minimum average and minimum are 5 and 2, respectively. The monitor subcommand specifies the probability of discard for packets that overflow the rate−control window. The options are:

average *avg*
Specify the minimum average interpacket spacing (minimum average headway time) in log2 s with default 3.

minimum *min*
Specify the minimum interpacket spacing (guard time) in seconds with default 2.

monitor
Specify the probability of being recorded for packets that overflow the MRU list size limit set by mru maxmem or mru maxdepth; this is a performance optimization for servers with aggregate arrivals of 1000 packets per second or more.

restrict *address*[/*cidr*] [mask *mask*] [flag ...]
The *address* argument expressed in dotted−quad (for IPv4) or :−delimited (for IPv6) form is the address of a host or network. Alternatively, the *address* argument can be a valid host DNS name. The *mask* argument expressed in IPv4 or IPv6 numeric address form defaults to all mask bits on, meaning that the *address* is treated as the address of an individual host. Instead of an explicit *mask*, the *address/cidr* may be specified in CIDR notation. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and is always the first entry in the list. Note that text string *default*, with no mask option, may be used to indicate the default entry. In the current implementation, *flag* always restricts access, i.e., an entry with no flags indicates that free access to the server is to be given. The flags are not orthogonal, in that more restrictive flags will often make less restrictive ones redundant. The flags can generally be classed into two categories, those which restrict time service and those which restrict informational queries and attempts to do a run−time reconfiguration of the server. One or more of the following flags may be specified:

flake
Discard received NTP packets with probability 0.1; that is, on average drop one packet in ten. This flag is for testing and amusement. The name comes from Bob Braden's *flakeway*, which once did a similar thing for early Internet testing.

ignore
Deny packets of all kinds, including ntpq(1) queries.

kod
If this flag is set when an access violation occurs, a kiss−o'−death (KoD) packet is sent. KoD packets are rate limited to no more than one per second. If another KoD packet occurs within one second after the last one, the packet is dropped.

limited
Deny service if the packet spacing violates the lower limits specified in the discard command. A history of clients is kept using the monitoring capability of ntpd(8). Thus, monitoring is always active as long as there is a restriction entry with the limited flag.

mssntp
Enable Microsoft Windows MS−SNTP authentication using Active Directory services. **Note: Potential users should be aware that these services involve a TCP connection to another process that could potentially block, denying services to other users. Therefore, this flag should be used only for a dedicated server with no clients other than MS−SNTP.**

nomodify
Deny ntpq(1) queries which attempt to modify the state of the server (i.e., run time reconfiguration). Queries which return information are permitted.

noquery
Deny ntpq(1) queries. Time service is not affected.

nopeer
Deny packets which would result in mobilizing a new association; this includes symmetric active

packets when a configured association does not exist. That used to happen when the remote client used the peer command in its config file. We don’t support that mode. It used to include *pool* servers, but they now poke a hole in any restrictions.

noserve

Deny all packets except ntpq(1) and queries.

notrust

Deny service unless the packet is cryptographically authenticated.

ntpport

This is a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched if the source port in the packet is the standard NTP UDP port (123). Both ntpport and non−ntpport may be specified. The ntpport is considered more specific and is sorted later in the list.

nomrulist

Do not accept MRU−list requests. These can be expensive to service and may generate a high volume of response traffic.

version

Deny packets that do not match the current NTP version.

Default restriction list entries with the flags ignore, interface, ntpport, for each of the local host’s interface addresses are inserted into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present, though if it is otherwise unconfigured; no flags are associated with the default entry (i.e., everything besides your own NTP server is unrestricted).

unrestrict *address*[/*cidr*] [mask *mask*] [flag ...]

Like a restrict command, but turns off the specified flags rather than turning them on (expected to be useful mainly with ntpq :config). An unrestrict with no flags specified removes any rule with matching address and mask. Use only on an address/mask or CIDR−format address mentioned in a previous restrict statement.

AUTOMATIC NTP CONFIGURATION OPTIONS

Manycasting

For a detailed description of manycast operation, see the "Server Discovery" page (available as part of the HTML documentation).

Manycast Options

tos [ceiling *ceiling* | floor *floor* | minclock *minclock* | minsane *minsane*]

This command affects the clock selection and clustering algorithms. It can be used to select the quality and quantity of peers used to synchronize the system clock and is most useful in manycast mode. The variables operate as follows:

ceiling *ceiling*

Peers with strata above *ceiling* will be discarded if there are at least *minclock* peers remaining. This value defaults to 15, but can be changed to any number from 1 to 15.

floor *floor*

Peers with strata below *floor* will be discarded if there are at least *minclock* peers remaining. This value defaults to 1, but can be changed to any number from 1 to 15.

minclock *minclock*

The clustering algorithm repeatedly casts out outlier associations until no more than *minclock* associations remain. This value defaults to 3, but can be changed to any number from 1 to the number of configured sources.

minsane *minsane*

This is the minimum number of candidates available to the clock selection algorithm in order to produce one or more truechimers for the clustering algorithm. If fewer than this number are available, the clock is undisciplined and allowed to run free. The default is 1 for legacy purposes. However, according to principles of Byzantine agreement, *minsane* should be at least 4 in order to detect and discard a single falseticker.

REFERENCE CLOCK SUPPORT

For a detailed description of reference−clock configuration, see the "Reference Clock Drivers" page (available as part of the HTML documentation provided in /usr/share/doc/ntp).

REFERENCE CLOCK COMMANDS

refclock *drivername* [unit *u*] [prefer] [subtype *int*] [mode *int*] [minpoll *int*] [maxpoll *int*] [time1 *sec*] [time2 *sec*] [stratum *int*] [refid *string*] [path *filename*] [ppspath *filename*] [baud *number*] [flag1 {0 | 1}] [flag2 {0 | 1}] [flag3 {0 | 1}] [flag4 {0 | 1}]

This command is used to configure reference clocks. The required *drivername* argument is the shortname of a driver type (e.g., shm, nmea, generic; see the Reference Clock Drivers page for a full list. The options are interpreted as follows:

unit

The 0−origin unit number of the device; this modifies the devicename. If not specified, defaults to zero.

prefer

Marks the reference clock as preferred. All other things being equal, this host will be chosen for synchronization among a set of correctly operating hosts and clocks. See the "Mitigation Rules and the prefer Keyword" page (available as part of the HTML documentation provided in /usr/share/doc/ntp) for further information.

subtype *int*

Some drivers (notably the generic and jjy drivers) support multiple device types. This option selects among them in a driver−dependent way.

mode *int*

Specifies a mode number which is interpreted in a device−specific fashion. For instance, it selects a dialing protocol in the ACTS driver and a sentence mix in the nmea driver.

minpoll *int*; maxpoll *int*

These options specify the minimum and maximum polling interval for reference clock messages, as a power of 2 in seconds. For most directly connected reference clocks, both *minpoll* and *maxpoll* default to 6 (64 sec). For modem reference clocks, *minpoll* defaults to 10 (17.1 min) and *maxpoll* defaults to 14 (4.5 hours). The allowable range is 0 (1 sec) to 17 (36.4 hours) inclusive.

time1 *sec*

Specifies a constant to be added to the time offset produced by the driver, a fixed−point decimal number in seconds. Each "g" on the end of the constant adds the number of seconds in a 10−bit GPS era; each "G" adds the number of seconds in a 13−bit GPS era. This is used as a calibration constant to adjust the nominal time offset of a particular clock to agree with an external standard, such as a precision PPS signal. It also provides a way to correct a systematic error or bias due to era wraparound from a GPS device, serial port or operating system latencies, different cable lengths or receiver internal delay. The specified offset is in addition to the propagation delay provided by other means, such as internal DIP switches. Where a calibration for an individual system and driver is available, an approximate correction is noted in the driver documentation pages. Note: To facilitate calibration when more than one radio clock or PPS signal is supported, a special calibration feature is available. It takes the form of an argument to the enable command described in "Miscellaneous Options" page and operates as described in the "Reference Clock Drivers" page.

time2 *secs*

Specifies a fixed−point decimal number in seconds, which is interpreted in a driver−dependent way. See the descriptions of specific drivers in the "Reference Clock Drivers" page.

stratum *int*

Specifies the stratum number assigned to the driver, an integer between 0 and 15. This number overrides the default stratum number ordinarily assigned by the driver itself, usually zero.

refid *string*

Specifies an ASCII string of from one to four characters which defines the reference identifier used by the driver. This string overrides the default identifier ordinarily assigned by the driver itself.

path *filepath*

Overrides the default device location for this refclock.

ppspath *filepath*

Overrides the default PPS device location (if any) for this driver.

baud *number*

Overrides the defaults baud rate for this driver.

flag1 {0 | 1}; flag2 {0 | 1}; flag3 {0 | 1}; flag4 {0 | 1}

These four flags are used for customizing the clock driver. The interpretation of these values, and whether they are used at all, is a function of the particular clock driver. However, by convention flag4 is used to enable recording monitoring data to the *clockstats* file configured with the *filegen* command. Further information on the *filegen* command can be found in "Monitoring Options".

MISCELLANEOUS OPTIONS

calldelay *delay*

This option controls the delay in seconds between the first and second packets sent in burst or iburst mode to allow additional time for a modem or ISDN call to complete.

driftfile *driftfile*

This command specifies the complete path and name of the file used to record the frequency of the local clock oscillator; this is the same operation as the −f command line option. If the file exists, it is read at startup to set the initial frequency and then updated once per hour with the current frequency computed by the daemon. If the file name is specified, but the file itself does not exist, ntpd starts with an initial frequency of zero and creates the file when writing it for the first time. If this command is not given, the daemon will always start with an initial frequency of zero.

The file format consists of a single line containing a single floating point number, which records the frequency offset measured in parts−per−million (PPM). The file is updated by first writing the current drift value into a temporary file and then renaming this file to replace the old version; this implies that ntpd(8) must have write permission for the directory the drift file is located in, and that file system links, symbolic or otherwise, should be avoided.

enable [auth | calibrate | kernel | monitor | ntp | stats]; disable [auth | calibrate | kernel | monitor | ntp | stats]

Provides a way to enable or disable various server options. Flags not mentioned are unaffected. Note that all of these flags can be controlled remotely using the ntpq(1) utility program.

auth

Enables the server to synchronize with unconfigured peers only if the peer has been correctly authenticated. The default for this flag is enable.

calibrate

Enables the calibrate feature for reference clocks. The default for this flag is disable.

kernel

Enables the kernel time discipline, if available. The default for this flag is enable if support is available, otherwise disable.

monitor

Enables the monitoring facility. See the ntpq(1) program and the monlist command for further information. The default for this flag is enable.

ntp

Enables time and frequency discipline. In effect, this switch opens and closes the feedback loop, which is useful for testing. The default for this flag is enable.

stats

Enables the statistics facility. See the "Monitoring Options" section for further information. The default for this flag is disable.

includefile *includefile*

This command allows additional configuration commands to be included from a separate file. Include files may be nested to a depth of five; upon reaching the end of any include file, command processing resumes in the previous configuration file. Relative pathnames are evaluated not with respect to the current working directory but with respect to the directory name of the last pushed file in the stack. This option is useful for sites that run ntpd(8) on multiple hosts, with (mostly) common options (e.g., a restriction list).

interface [listen | ignore | drop] [all | ipv4 | ipv6 | wildcard | *name* | *address*[*/prefixlen*]]

This command controls which network addresses ntpd opens, and whether the input is dropped without processing. The first parameter determines the action on addresses which match the second parameter. That parameter specifies a class of addresses, or a specific interface name, or an address. In the address case, *prefixlen* determines how many bits must match for this rule to apply. ignore prevents opening matching addresses, drop causes ntpd to open the address and drop all received packets without examination. Multiple interface commands can be used. The last rule which matches a particular address determines the action for it. interface commands are disabled if any of the −I, −−interface,−L, or −−novirtualips command−line options are used. If none of those options are used, and no interface actions are specified in the configuration file, all available network addresses are opened. The nic command is an alias for interface.

leapfile *leapfile*

This command loads the NIST leap seconds file and initializes the leapsecond values for the next leap second time, expiration time and TAI offset. The file can be obtained using ntpleapfetch.

The *leapfile* is scanned when ntpd processes the leapfile directive or when ntpd detects that *leapfile* has changed. ntpd checks once a day to see if the *leapfile* has changed.

leapsmearinterval *interval*

This **experimental** option is only available if ntpd was built with the −−enable−leap−smear option, It specifies the interval over which a leap second correction will be applied. Recommended values for this option are between 7200 (2 hours) and 86400 (24 hours). DO NOT USE THIS OPTION ON PUBLIC−ACCESS SERVERS! See http://bugs.ntp.org/2855 for more information.

logconfig *configkeyword*

This command controls the amount and type of output written to the system *syslog(3)* facility or the alternate log file. By default, all output is turned on. All *configkeyword* keywords can be prefixed with ‘=’, ‘’ and ‘−’, where ‘=’ sets the syslog(3) priority mask, ‘’ adds and ‘−’ removes messages. syslog(3) messages can be controlled in four classes (clock,peer,sys and sync). Within these classes four types of messages can be controlled: informational messages (info), event messages (events), statistics messages (statistics) and status messages (status).

Configuration keywords are formed by concatenating the message class with the event class. The *all* prefix can be used instead of a message class. A message class may also be followed by the *all* keyword to enable/disable all messages of the respective message class. Thus, a minimal log configuration could look like this:

```
logconfig =syncstatus +sysevents
```

This would just list the synchronizations state of ntpd(8) and the major system events. For a simple reference server, the following minimum message configuration could be useful:

```
logconfig =syncall +clockall
```

This configuration will list all clock information and synchronization information. All other events and messages about peers, system events and so on is suppressed.

logfile *logfile*

This command specifies the location of an alternate log file to be used instead of the default system *syslog(3)* facility; this is the same operation as the −l command line option.

If your ntpd runs for a long time, you probably want to use logrotate or newsyslog to switch to a new log file occasionally. SIGHUP will reopen the log file.

mru [maxdepth *count* | maxmem *kilobytes* | mindepth *count* | maxage *seconds* | minage *seconds* | initalloc *count* | initmem *kilobytes* | incalloc *count* | incmem *kilobytes*]

Controls size limits of the monitoring facility Most Recently Used (MRU) list of client addresses, which is also used by the rate control facility.

maxdepth *count*, maxmem *kilobytes*

Equivalent upper limits on the size of the MRU list, in terms of entries or kilobytes. The actual limit will be up to incalloc entries or incmem kilobytes larger. As with all of the mru options offered in units of entries or kilobytes, if both maxdepth and maxmem are used, the last one used controls. The default is 1024 kilobytes.

mindepth *count*

The lower limit on the MRU list size. When the MRU list has fewer than mindepth entries, existing entries are never removed to make room for newer ones, regardless of their age. The default is 600 entries.

maxage *seconds*, minage *seconds*

If an address is not in the list, there are several possible ways to find a slot for it.

1. If the list has fewer than mindepth entries, a slot is allocated from the free list; this is the normal case for a server without a lot of clients. If clients come and go, for example, laptops going between home and work, the default setup shows only the long term average.
2. If the age of the oldest slot is greater than maxage, the oldest slot is recycled (default 3600 seconds).
3. If the freelist is not empty, a slot is allocated from the free list.
4. If the freelist is empty but not full (see maxmem), more memory is allocated (see incmem) and, a new slot is used.
5. If the age of the oldest slot is more than minage, the oldest slot is recycled (default 64 seconds).
6. Otherwise, no slot is available.

initalloc *count*, initmem *kilobytes*

Initial memory allocation at the time the monitoring facility is first enabled, in terms of entries or kilobytes. The default is 4 kilobytes.

incalloc *count*, incmem *kilobytes*

Size of additional memory allocations when growing the MRU list, in entries or kilobytes. The default is 4 kilobytes.

nonvolatile *threshold*

Specify the *threshold* in seconds to write the frequency file, with a default of 1e−7 (0.1 PPM). The frequency file is inspected each hour. If the difference between the current frequency and the last value written exceeds the threshold, the file is written, and the threshold becomes the new threshold value. If the threshold is not exceeded, it is reduced by half; this is intended to reduce the frequency of unnecessary file writes for embedded systems with nonvolatile memory.

phone *dial* ...

This command is used in conjunction with the ACTS modem driver (type modem) or the JJY driver (type jjy). For ACTS, the arguments consist of a maximum of 10 telephone numbers used to dial USNO, NIST or European time services. For the jjy driver in modes 100−180, the argument is one telephone number used to dial the telephone JJY service. The Hayes command ATDT is normally prepended to the number, which can contain other modem control codes as well.

reset [allpeers] [auth] [ctl] [io] [mem] [sys] [timer]

Reset one or more groups of counters maintained by ntpd and exposed by ntpq.

setvar *variable* [*default*]

This command adds a system variable. These variables can be used to distribute additional information such as the access policy. If the variable of the form *name=value* is followed by the default keyword, the variable will be listed as part of the default system variables (ntpq(1) rv command). These additional variables serve informational purposes only. They are not related to the protocol other that they can be listed. The known protocol variables will always override any variables defined via the setvar mechanism. There are three special variables that contain the names of all variable of the same group. The sys_var_list holds the names of all system variables. The peer_var_list holds the names of all peer variables and the clock_var_list holds the names of the reference clock variables.

tinker [allan *allan* | dispersion *dispersion* | freq *freq* | huffpuff *huffpuff* | panic *panic* | step *step* | stepback *stepback* | stepfwd *stepfwd* | stepout *stepout*]

This command can be used to alter several system variables in very exceptional circumstances. It should occur in the configuration file before any other configuration options. The default values of these variables have been carefully optimized for a wide range of network speeds and reliability expectations. In general, they interact in intricate ways that are hard to predict, and some combinations can result in some very nasty behavior. Very rarely is it necessary to change the default values; but, some folks cannot resist twisting the knobs anyway, and this command is for them. Emphasis added: twisters are on their own and can expect no help from the support group.

The variables operate as follows:

allan *allan*

The argument becomes the new value for the minimum Allan intercept, which is a parameter of the PLL/FLL clock discipline algorithm. The value in log2 seconds defaults to 11 (2048 s), which is also the lower limit.

dispersion *dispersion*

The argument becomes the new value for the dispersion increase rate, normally .000015 s/s.

freq *freq*

The argument becomes the initial value of the frequency offset in parts−per−million; this overrides the value in the frequency file, if present, and avoids the initial training state if it is not.

huffpuff *huffpuff*

The argument becomes the new value for the experimental huff−n'−puff filter span, which determines the most recent interval the algorithm will search for a minimum delay. The lower limit is 900 s (15 m), but a more reasonable value is 7200 (2 hours). There is no default since the filter is not enabled unless this command is given.

panic *panic*

The argument is the panic threshold, normally 1000 s. If set to zero, the panic sanity check is disabled, and a clock offset of any value will be accepted.

step *step*

The argument is the step threshold, which by default is 0.128 sec. It can be set to any positive number in seconds. If set to zero, step adjustments will never occur. Note: The kernel time discipline is disabled if the step threshold is set to zero or greater than the default.

stepback *stepback*

The argument is the step threshold for the backward direction, which by default is 0.128 sec. It can be set to any positive number in seconds. If both the forward and backward step thresholds are set to zero, step adjustments will never occur. Note: The kernel time discipline is disabled if each direction of step threshold are either set to zero or greater than .5 second.

stepfwd *stepfwd*

As for stepback, but for the forward direction.

stepout *stepout*

The argument is the stepout timeout, which by default is 900 s. It can be set to any positive number in seconds. If set to zero, the stepout pulses will not be suppressed.

rlimit [memlock *megabytes* | stacksize *4kPages* | filenum *filedescriptors*]

memlock *megabytes*

Ignored for backward compatibility.

stacksize *4kPages*

Specifies the maximum size of the process stack on systems with the mlockall() function. Defaults to 50 4k pages.

filenum *filedescriptors*

Specifies the maximum number of file descriptors ntpd may have open at once. Defaults to the system default.

FILES

/etc/ntpsec/ntp.conf

the default name of the configuration file

/etc/ntpsec/ntp.keys

private keys

One of the following exit values will be returned:

0 (EXIT_SUCCESS)

Successful program execution.

1 (EXIT_FAILURE)

The operation failed or the command syntax was not valid.

SEE ALSO

ntpd(8), ntpq(1).

In addition to the manual pages provided, comprehensive documentation is available on the world wide web at https://www.ntpsec.org. A snapshot of this documentation is available in HTML format in /usr/share/doc/ntp.

David L. Mills, *Network Time Protocol (Version 4)*, RFC 5905

BUGS

The syntax checking is not picky; some combinations of ridiculous and even hilarious options and modes may not be detected.