MyWebUniversity
python
Red Hat Enterprise Linux Manual Pages
FPDF Library PDF generator
FPDF

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'chrony.conf.5' command

$ man chrony.conf.5

CHRONY.CONF(5)   Configuration Files   CHRONY.CONF(5)

NAME

chrony.conf - chronyd configuration file

SYNOPSIS

chrony.conf

DESCRIPTION

This file configures the chronyd daemon. The compiled-in location is /etc/chrony.conf. Other locations can be specified on the chronyd command line with the -f option.

Each directive in the configuration file is placed on a separate line. The following sections describe each of the directives in turn. The directives are not case-sensitive. Generally, the directives can occur in any order in the file and if a directive is specified multiple times, only the last one will be effective. Exceptions are noted in the descriptions.

The configuration directives can also be specified directly on the chronyd command line. In this case each argument is parsed as a new line and the configuration file is ignored.

While the number of supported directives is large, only a few of them are typically needed. See the EXAMPLES section for configuration in typical operating scenarios.

The configuration file might contain comment lines. A comment line is any line that starts with zero or more spaces followed by any one of the following characters: !, ;, #, %. Any line with this format will be ignored.

DIRECTIVES

Time sources

server hostname [option]...

The server directive specifies an NTP server which can be used as a time source. The client-server relationship is strictly hierarchical: a client might synchronise its system time to that of the server, but the server?s system time will never be influenced by that of a client.

The server can be specified by its hostname or IP address. If the hostname cannot be resolved on start, chronyd will try it again in increasing intervals, and also when the online command is issued in chronyc.

The DNS record can change over time. The used address will be replaced with a newly resolved address when the server becomes unreachable (i.e. no valid response to last 8 requests), unsynchronised, a falseticker (i.e. does not agree with a majority of other sources), or the root distance is too large (the limit can be configured by the maxdistance directive). The automatic replacement happens at most once per 30 minutes. It can also be triggered manually for all sources by the refresh command in chronyc.

This directive can be used multiple times to specify multiple servers.

The directive supports the following options:

minpoll poll

This option specifies the minimum interval between requests sent to the server as a power of 2 in seconds. For example, minpoll 5 would mean that the polling interval should not drop below 32 seconds. The default is 6 (64 seconds), the minimum is -7 (1/128th of a second), and the maximum is 24 (6 months).

Note that intervals shorter than 6 (64 seconds) should generally not be used with public servers on the Internet, because it might be considered abuse. A sub-second interval will be enabled only when the server is reachable and the round-trip delay is shorter than 10 milliseconds, i.e. the server should be in a local network.

maxpoll poll

This option specifies the maximum interval between requests sent to the server as a power of 2 in seconds. For example, maxpoll 9 indicates that the polling interval should stay at or below 9 (512 seconds). The default is 10 (1024 seconds), the minimum is -7 (1/128th of a second), and the maximum is 24 (6 months).

iburst

With this option, chronyd will start with a burst of 4-8 requests in order to make the first update of the clock sooner. It will also repeat the burst every time the source is switched from the offline state to online with the online command in chronyc.

burst

With this option, chronyd will send a burst of up to 4 requests when it cannot get a good measurement from the server. The number of requests in the burst is limited by the current polling interval to keep the average interval at or above the minimum interval, i.e. the current interval needs to be at least two times longer than the minimum interval in order to allow a burst with two requests.

key ID

The NTP protocol supports a message authentication code (MAC) to prevent computers having their system time upset by rogue packets being sent to them. The MAC is generated as a function of a key specified in the key file, which is specified by the keyfile directive.

The key option specifies which key (with an ID in the range 1

through 2^32-1) should chronyd use to authenticate requests
sent to the server and verify its responses. The server must
have the same key for this number configured, otherwise no
relationship between the computers will be possible.

If the server is running ntpd and the output size of the hash
function used by the key is longer than 160 bits (e.g. SHA256),
the version option needs to be set to 4 for compatibility.

nts

This option enables authentication using the Network Time
Security (NTS) mechanism. Unlike with the key option, the
server and client do not need to share a key in a key file. NTS
has a Key Establishment (NTS-KE) protocol using the Transport
Layer Security (TLS) protocol to get the keys and cookies
required by NTS for authentication of NTP packets.

certset ID

This option specifies which set of trusted certificates should
be used to verify the server?s certificate when the nts option
is enabled. Sets of certificates can be specified with the
ntstrustedcerts directive. The default set is 0, which by
default contains certificates of the system?s default trusted
certificate authorities.

maxdelay delay

chronyd uses the network round-trip delay to the server to
determine how accurate a particular measurement is likely to
be. Long round-trip delays indicate that the request, or the
response, or both were delayed. If only one of the messages was
delayed the measurement error is likely to be substantial.

For small variations in the round-trip delay, chronyd uses a
weighting scheme when processing the measurements. However,
beyond a certain level of delay the measurements are likely to
be so corrupted as to be useless. (This is particularly so on
wireless networks and other slow links, where a long delay
probably indicates a highly asymmetric delay caused by the

response waiting behind a lot of packets related to a download of some sort).

If the user knows that round trip delays above a certain level should cause the measurement to be ignored, this level can be defined with the maxdelay option. For example, maxdelay 0.3 would indicate that measurements with a round-trip delay greater than 0.3 seconds should be ignored. The default value is 3 seconds and the maximum value is 1000 seconds.

maxdelayratio ratio

This option is similar to the maxdelay option above. chronyd keeps a record of the minimum round-trip delay amongst the previous measurements that it has buffered. If a measurement has a round-trip delay that is greater than the specified ratio times the minimum delay, it will be rejected. By default, this test is disabled.

maxdelaydevratio ratio

If a measurement has a ratio of the increase in the round-trip delay from the minimum delay amongst the previous measurements to the standard deviation of the previous measurements that is greater than the specified ratio, it will be rejected. The default is 10.0.

maxdelayquant p

This option disables the maxdelaydevratio test and specifies the maximum acceptable delay as a quantile of the round-trip delay instead of a function of the minimum delay amongst the buffered measurements. If a measurement has a round-trip delay that is greater than a long-term estimate of the p-quantile, it will be rejected.

The specified p value should be between 0.05 and 0.95. For example, maxdelayquant 0.2 would indicate that only measurements with the lowest 20 percent of round-trip delays should be accepted. Note that it can take many measurements for the estimated quantile to reach the expected value. This option

is intended for synchronisation in mostly static local networks
with very short polling intervals and possibly combined with
the filter option. By default, this test is disabled in favour
of the maxdelaydevratio test.

mindelay delay

This option specifies a fixed minimum round-trip delay to be
used instead of the minimum amongst the previous measurements.
This can be useful in networks with static configuration to
improve the stability of corrections for asymmetric jitter,
weighting of the measurements, and the maxdelayratio and
maxdelaydevratio tests. The value should be set accurately in
order to have a positive effect on the synchronisation.

asymmetry ratio

This option specifies the asymmetry of the network jitter on
the path to the source, which is used to correct the measured
offset according to the delay. The asymmetry can be between
-0.5 and +0.5. A negative value means the delay of packets sent
to the source is more variable than the delay of packets sent
from the source back. By default, chronyd estimates the
asymmetry automatically.

offset offset

This option specifies a correction (in seconds) which will be
applied to offsets measured with this source. It?s particularly
useful to compensate for a known asymmetry in network delay or
timestamping errors. For example, if packets sent to the source
were on average delayed by 100 microseconds more than packets
sent from the source back, the correction would be -0.00005
(-50 microseconds). The default is 0.0.

minsamples samples

Set the minimum number of samples kept for this source. This
overrides the minsamples directive.

maxsamples samples

Set the maximum number of samples kept for this source. This overrides the maxsamples directive.

filter polls

This option enables a median filter to reduce noise in NTP measurements. The filter will process samples collected in the specified number of polls into a single sample. It is intended to be used with very short polling intervals in local networks where it is acceptable to generate a lot of NTP traffic.

offline

If the server will not be reachable when chronyd is started, the offline option can be specified. chronyd will not try to poll the server until it is enabled to do so (by using the online command in chronyc).

auto_offline

With this option, the server will be assumed to have gone offline when sending a request fails, e.g. due to a missing route to the network. This option avoids the need to run the offline command from chronyc when disconnecting the network link. (It will still be necessary to use the online command when the link has been established, to enable measurements to start.)

prefer

Prefer this source over sources without the prefer option.

noselect

Never select this source. This is particularly useful for monitoring.

trust

Assume time from this source is always true. It can be rejected as a falseticker in the source selection only if another source with this option does not agree with it.

require

Require that at least one of the sources specified with this option is selectable (i.e. recently reachable and not a falseticker) before updating the clock. Together with the trust option this might be useful to allow a trusted authenticated source to be safely combined with unauthenticated sources in order to improve the accuracy of the clock. They can be selected and used for synchronisation only if they agree with the trusted and required source.

xleave

This option enables the interleaved mode of NTP. It enables the server to respond with more accurate transmit timestamps (e.g. kernel or hardware timestamps), which cannot be contained in the transmitted packet itself and need to refer to a previous packet instead. This can significantly improve the accuracy and stability of the measurements.

The interleaved mode is compatible with servers that support only the basic mode. Note that even servers that support the interleaved mode might respond in the basic mode as the interleaved mode requires the servers to keep some state for each client and the state might be dropped when there are too many clients (e.g. clientloglimit is too small), or it might be overwritten by other clients that have the same IP address (e.g. computers behind NAT or someone sending requests with a spoofed source address).

The xleave option can be combined with the presend option in order to shorten the interval in which the server has to keep the state to be able to respond in the interleaved mode.

polltarget target

Target number of measurements to use for the regression algorithm which chronyd will try to maintain by adjusting the polling interval between minpoll and maxpoll. A higher target makes chronyd prefer shorter polling intervals. The default is 8 and a useful range is from 6 to 60.

port port

This option allows the UDP port on which the server understands NTP requests to be specified. For normal servers this option should not be required (the default is 123, the standard NTP port).

ntsport port

This option specifies the TCP port on which the server is listening for NTS-KE connections when the nts option is enabled. The default is 4460.

presend poll

If the timing measurements being made by chronyd are the only network data passing between two computers, you might find that some measurements are badly skewed due to either the client or the server having to do an ARP lookup on the other party prior to transmitting a packet. This is more of a problem with long sampling intervals, which might be similar in duration to the lifetime of entries in the ARP caches of the machines.

In order to avoid this problem, the presend option can be used. It takes a single integer argument, which is the smallest polling interval for which an extra pair of NTP packets will be exchanged between the client and the server prior to the actual measurement. For example, with the following option included in a server directive:

```
presend 9
```

when the polling interval is 512 seconds or more, an extra NTP client packet will be sent to the server a short time (2 seconds) before making the actual measurement.

If the presend option is used together with the xleave option, chronyd will send two extra packets instead of one.

minstratum stratum

When the synchronisation source is selected from available sources, sources with lower stratum are normally slightly preferred. This option can be used to increase stratum of the source to the specified minimum, so chronyd will avoid selecting that source. This is useful with low-stratum sources that are known to be unreliable or inaccurate and which should be used only when other sources are unreachable.

version version

This option sets the NTP version of packets sent to the server. This can be useful when the server runs an old NTP implementation that does not respond to requests using a newer version. The default version depends on other options. If the extfield or xleave option is used, the default version is 4. If those options are not used and the key option specifies a key using a hash function with output size longer than 160 bits (e.g. SHA256), the default version is 3 for compatibility with older chronyd servers. In other cases, the default version is 4.

copy

This option specifies that the server and client are closely related, their configuration does not allow a synchronisation loop to form between them, and the client is allowed to assume the reference ID and stratum of the server. This is useful when multiple instances of chronyd are running on one computer (e.g. for security or performance reasons), one primarily operating as a client to synchronise the system clock and other instances started with the -x option to operate as NTP servers for other computers with their NTP clocks synchronised to the first instance.

extfield type

This option enables an NTPv4 extension field specified by its type as a hexadecimal number. It will be included in requests sent to the server and processed in received responses if the server supports it. Note that some server implementations do not respond to requests containing an unknown extension field (chronyd as a server responded to such requests since version 2.0).

The following extension field can be enabled by this option:

F323

This is an experimental extension field for some improvements that were proposed for the next version of the NTP protocol (NTPv5). The field contains root delay and dispersion in higher resolution and a monotonic receive timestamp, which enables a frequency transfer between the server and client. It can significantly improve stability of the synchronization. Generally, it should be expected to work only between servers and clients running the same version of chronyd.

pool name [option]...

The syntax of this directive is similar to that for the server directive, except that it is used to specify a pool of NTP servers rather than a single NTP server. The pool name is expected to resolve to multiple addresses which might change over time. This directive can be used multiple times to specify multiple pools.

All options valid in the server directive can be used in this directive too. There is one option specific to the pool directive:

maxsources sources

This option sets the desired number of sources to be used from the pool. chronyd will repeatedly try to resolve the name until it gets this number of sources responding to requests. The default value is 4 and the maximum value is 16.

An example of the pool directive is

```
pool pool.ntp.org iburst maxsources 3
```

peer hostname [option]...

The syntax of this directive is identical to that for the server directive, except that it specifies a symmetric association with an NTP peer instead of a client/server association with an NTP server. A single symmetric association allows the peers to be both servers and clients to each other. This is mainly useful when the NTP implementation of the peer (e.g. ntpd) supports ephemeral symmetric associations and does not need to be configured with an address of this host. chronyd does not support ephemeral associations.

This directive can be used multiple times to specify multiple peers.

The following options of the server directive do not work in the peer directive: iburst, burst, nts, presend, copy.

When using the xleave option, both peers must support and have enabled the interleaved mode, otherwise the synchronisation will work in one direction only. When a key is specified by the key option to enable authentication, both peers must use the same key and the same key number.

Note that the symmetric mode is less secure than the client/server mode. A denial-of-service attack is possible on unauthenticated symmetric associations, i.e. when the peer was specified without the key option. An attacker who does not see network traffic between two hosts, but knows that they are peering with each other, can periodically send them unauthenticated packets with spoofed source addresses in order to disrupt their NTP state and prevent them from synchronising to each other. When the association is authenticated, an attacker who does see the network traffic, but cannot prevent the packets from reaching the other host, can still disrupt the state by replaying old packets. The attacker has effectively the same power as a man-in-the-middle attacker. A partial protection against this attack is implemented in chronyd, which can protect the peers if they are using the same polling interval and they never sent an authenticated packet with a timestamp from future, but it should not be relied on as it is difficult to ensure the conditions are met. If two hosts should be able to synchronise to each other in both directions, it is recommended to use two separate client/server associations (specified by the server directive on both hosts) instead.

initstepslew step-threshold [hostname]...

(This directive is deprecated in favour of the makestep directive.)

The purpose of the initstepslew directive is to allow chronyd to

make a rapid measurement of the system clock error at boot time,
and to correct the system clock by stepping before normal operation
begins. Since this would normally be performed only at an
appropriate point in the system boot sequence, no other software
should be adversely affected by the step.

If the correction required is less than a specified threshold, a
slew is used instead. This makes it safer to restart chronyd whilst
the system is in normal operation.

The initstepslew directive takes a threshold and a list of NTP
servers as arguments. Each of the servers is rapidly polled several
times, and a majority voting mechanism used to find the most likely
range of system clock error that is present. A step or slew is
applied to the system clock to correct this error. chronyd then
enters its normal operating mode.

An example of the use of the directive is:

```
    initstepslew 30 foo.example.net bar.example.net baz.example.net
```

where 3 NTP servers are used to make the measurement. The 30
indicates that if the system?s error is found to be 30 seconds or
less, a slew will be used to correct it; if the error is above 30
seconds, a step will be used.

The initstepslew directive can also be used in an isolated LAN
environment, where the clocks are set manually. The most stable
computer is chosen as the primary server and the other computers
are its clients. If each of the clients is configured with the
local directive, the server can be set up with an initstepslew
directive which references some or all of the clients. Then, if the
server machine has to be rebooted, the clients can be relied on to
act analogously to a flywheel and preserve the time for a short
period while the server completes its reboot.

The initstepslew directive is functionally similar to a combination
of the makestep and server directives with the iburst option. The
main difference is that the initstepslew servers are used only
before normal operation begins and that the foreground chronyd

process waits for initstepslew to finish before exiting. This
prevent programs started in the boot sequence after chronyd from
reading the clock before it has been stepped. With the makestep
directive, the waitsync command of chronyc can be used instead.

refclock driver parameter[:option]... [option]...

The refclock directive specifies a hardware reference clock to be
used as a time source. It has two mandatory parameters, a driver
name and a driver-specific parameter. The two parameters are
followed by zero or more refclock options. Some drivers have
special options, which can be appended to the driver-specific
parameter using the : character.

This directive can be used multiple times to specify multiple
reference clocks.

There are four drivers included in chronyd:

PPS

Driver for the kernel PPS (pulse per second) API. The parameter
is the path to the PPS device (typically /dev/pps?). As PPS
refclocks do not supply full time, another time source (e.g.
NTP server or non-PPS refclock) is needed to complete samples
from the PPS refclock. An alternative is to enable the local
directive to allow synchronisation with some unknown but
constant offset. The driver supports the following option:

clear

By default, the PPS refclock uses assert events (rising
edge) for synchronisation. With this option, it will use
clear events (falling edge) instead.

Examples:

```
refclock PPS /dev/pps0 lock NMEA refid GPS
refclock SHM 0 offset 0.5 delay 0.2 refid NMEA noselect
refclock PPS /dev/pps1:clear refid GPS2
```

SHM

NTP shared memory driver. This driver uses a shared memory
segment to receive samples from another process (e.g. gpsd).

The parameter is the number of the shared memory segment, typically a small number like 0, 1, 2, or 3. The driver supports the following option:

perm=mode

This option specifies the permissions of the shared memory segment created by chronyd. They are specified as a numeric mode. The default value is 0600 (read-write access for owner only).

Examples:

```
refclock SHM 0 poll 3 refid GPS1
refclock SHM 1:perm=0644 refid GPS2
```

SOCK

Unix domain socket driver. It is similar to the SHM driver, but samples are received from a Unix domain socket instead of shared memory and the messages have a different format. The parameter is the path to the socket, which chronyd creates on start. An advantage over the SHM driver is that SOCK does not require polling and it can receive PPS samples with incomplete time. The format of the messages is described in the refclock_sock.c file in the chrony source code.

An application which supports the SOCK protocol is the gpsd daemon. The path where gpsd expects the socket to be created is described in the gpsd(8) man page. For example:

```
refclock SOCK /var/run/chrony.ttyS0.sock
```

PHC

PTP hardware clock (PHC) driver. The parameter is the path to the device of the PTP clock which should be used as a time source. If the clock is kept in TAI instead of UTC (e.g. it is synchronised by a PTP daemon), the current UTC-TAI offset needs to be specified by the offset option. Alternatively, the pps refclock option can be enabled to treat the PHC as a PPS refclock, using only the sub-second offset for synchronisation.

The driver supports the following options:

nocrossts

This option disables use of precise cross timestamping.

extpps

This option enables a PPS mode in which the PTP clock is timestamping pulses of an external PPS signal connected to the clock. The clock does not need to be synchronised, but another time source is needed to complete the PPS samples. Note that some PTP clocks cannot be configured to timestamp only assert or clear events, and it is necessary to use the width option to filter wrong PPS samples.

pin=index

This option specifies the index of the pin which should be enabled for the PPS timestamping. If the PHC does not have configurable pins (i.e. the channel function is fixed), the index needs to be set to -1 to disable the pin configuration. The default value is 0.

channel=index

This option specifies the index of the channel for the PPS mode. The default value is 0.

clear

This option enables timestamping of clear events (falling edge) instead of assert events (rising edge) in the PPS mode. This may not work with some clocks.

Examples:

```
refclock PHC /dev/ptp0 poll 0 dpoll -2 offset -37
refclock PHC /dev/ptp1:nocrossts poll 3 pps
refclock PHC /dev/ptp2:extpps:pin=1 width 0.2 poll 2
```

The refclock directive supports the following options:

poll poll

Timestamps produced by refclock drivers are not used immediately, but they are stored and processed by a median filter in the polling interval specified by this option. This is defined as a power of 2 and can be negative to specify a sub-second interval. The default is 4 (16 seconds). A shorter interval allows chronyd to react faster to changes in the frequency of the system clock, but it might have a negative effect on its accuracy if the samples have a lot of jitter.

dpoll dpoll

Some drivers do not listen for external events and try to produce samples in their own polling interval. This is defined as a power of 2 and can be negative to specify a sub-second interval. The default is 0 (1 second).

refid refid

This option is used to specify the reference ID of the refclock, as up to four ASCII characters. The default reference ID is composed from the first three characters of the driver name and the number of the refclock. Each refclock must have a unique reference ID.

lock refid

This option can be used to lock a PPS refclock to another refclock, which is specified by its reference ID. In this mode received PPS samples are paired directly with raw samples from the specified refclock.

rate rate

This option sets the rate of the pulses in the PPS signal (in Hz). This option controls how the pulses will be completed with real time. To actually receive more than one pulse per second, a negative dpoll has to be specified (-3 for a 5Hz signal). The default is 1.

maxlockage pulses

This option specifies in number of pulses how old can be samples from the refclock specified by the lock option to be paired with the pulses. Increasing this value is useful when the samples are produced at a lower rate than the pulses. The default is 2.

width width

This option specifies the width of the pulses (in seconds). It is used to filter PPS samples when the driver provides samples for both rising and falling edges. Note that it reduces the maximum allowed error of the time source which completes the PPS samples. If the duty cycle is configurable, 50% should be preferred in order to maximise the allowed error.

pps

This options forces chronyd to treat any refclock (e.g. SHM or PHC) as a PPS refclock. This can be useful when the refclock provides time with a variable offset of a whole number of seconds (e.g. it uses TAI instead of UTC). Another time source is needed to complete samples from the refclock.

offset offset

This option can be used to compensate for a constant error. The specified offset (in seconds) is applied to all samples produced by the reference clock. The default is 0.0.

delay delay

This option sets the NTP delay of the source (in seconds). Half of this value is included in the maximum assumed error which is used in the source selection algorithm. Increasing the delay is useful to avoid having no majority in the source selection or to make it prefer other sources. The default is 1e-9 (1 nanosecond).

stratum stratum

This option sets the NTP stratum of the refclock. This can be useful when the refclock provides time with a stratum other than 0. The default is 0.

precision precision

This option sets the precision of the reference clock (in seconds). The default value is the estimated precision of the system clock.

maxdispersion dispersion

Maximum allowed dispersion for filtered samples (in seconds).

Samples with larger estimated dispersion are ignored. By default, this limit is disabled.

filter samples

This option sets the length of the median filter which is used to reduce the noise in the measurements. With each poll about 40 percent of the stored samples are discarded and one final sample is calculated as an average of the remaining samples. If the length is 4 or more, at least 4 samples have to be collected between polls. For lengths below 4, the filter has to be full. The default is 64.

prefer

Prefer this source over sources without the prefer option.

noselect

Never select this source. This is useful for monitoring or with sources which are not very accurate, but are locked with a PPS refclock.

trust

Assume time from this source is always true. It can be rejected as a falseticker in the source selection only if another source with this option does not agree with it.

require

Require that at least one of the sources specified with this option is selectable (i.e. recently reachable and not a falseticker) before updating the clock. Together with the trust option this can be useful to allow a trusted, but not very precise, reference clock to be safely combined with unauthenticated NTP sources in order to improve the accuracy of the clock. They can be selected and used for synchronisation only if they agree with the trusted and required source.

tai

This option indicates that the reference clock keeps time in TAI instead of UTC and that chronyd should correct its offset by the current TAI-UTC offset. The leapsectz directive must be

used with this option and the database must be kept up to date
in order for this correction to work as expected. This option
does not make sense with PPS refclocks.

local

This option specifies that the reference clock is an
unsynchronised clock which is more stable than the system clock
(e.g. TCXO, OCXO, or atomic clock) and it should be used as a
local standard to stabilise the system clock. The refclock will
bypass the source selection. There should be at most one
refclock specified with this option and it should have the
shortest polling interval among all configured sources.

minsamples samples

Set the minimum number of samples kept for this source. This
overrides the minsamples directive.

maxsamples samples

Set the maximum number of samples kept for this source. This
overrides the maxsamples directive.

manual

The manual directive enables support at run-time for the settime
command in chronyc. If no manual directive is included, any attempt
to use the settime command in chronyc will be met with an error
message.

Note that the settime command can be enabled at run-time using the
manual command in chronyc. (The idea of the two commands is that
the manual command controls the manual clock driver?s behaviour,
whereas the settime command allows samples of manually entered time
to be provided.)

acquisitionport port

By default, chronyd as an NTP client opens a new socket for each
request with the source port chosen randomly by the operating
system. The acquisitionport directive can be used to specify the
source port and use only one socket (per IPv4 or IPv6 address
family) for all configured servers. This can be useful for getting

through some firewalls. It should not be used if not necessary as there is a small impact on security of the client. If set to 0, the source port of the permanent socket will be chosen randomly by the operating system.

It can be set to the same port as is used by the NTP server (which can be configured with the port directive) to use only one socket for all NTP packets.

An example of the acquisitionport directive is:

```
acquisitionport 1123
```

This would change the source port used for client requests to UDP port 1123. You could then persuade the firewall administrator to open that port.

bindacqaddress address

The bindacqaddress directive specifies a local IP address to which chronyd will bind its NTP and NTS-KE client sockets. The syntax is similar to the bindaddress and bindcmdaddress directives.

For each of the IPv4 and IPv6 protocols, only one bindacqaddress directive can be specified.

bindacqdevice interface

The bindacqdevice directive binds the client sockets to a network device specified by the interface name. This can be useful when the local address is dynamic, or to enable an NTP source specified with a link-local IPv6 address. This directive can specify only one interface and it is supported on Linux only.

An example of the directive is:

```
bindacqdevice eth0
```

dscp point

The dscp directive sets the Differentiated Services Code Point (DSCP) in transmitted NTP packets to the specified value. It can improve stability of NTP measurements in local networks where switches or routers are configured to prioritise forwarding of packets with specific DSCP values. The default value is 0 and the maximum value is 63.

An example of the directive (setting the Expedited Forwarding class) is:

```
dscp 46
```

dumpdir directory

To compute the rate of gain or loss of time, chronyd has to store a measurement history for each of the time sources it uses.

All supported systems, with the exception of macOS 10.12 and earlier, have operating system support for setting the rate of gain or loss to compensate for known errors. (On macOS 10.12 and earlier, chronyd must simulate such a capability by periodically slewing the system clock forwards or backwards by a suitable amount to compensate for the error built up since the previous slew.)

For such systems, it is possible to save the measurement history across restarts of chronyd (assuming no changes are made to the system clock behaviour whilst it is not running). The dumpdir directive defines the directory where the measurement histories are saved when chronyd exits, or the dump command in chronyc is issued.

If the directory does not exist, it will be created automatically.

The -r option of chronyd enables loading of the dump files on start. All dump files found in the directory will be removed after start, even if the -r option is not present.

An example of the directive is:

```
dumpdir /run/chrony
```

A source whose IP address is 1.2.3.4 would have its measurement history saved in the file /run/chrony/1.2.3.4.dat. History of reference clocks is saved to files named by their reference ID in form of refid:XXXXXXXX.dat.

maxsamples samples

The maxsamples directive sets the default maximum number of samples that chronyd should keep for each source. This setting can be overridden for individual sources in the server and refclock directives. The default value is 0, which disables the configurable limit. The useful range is 4 to 64.

As a special case, setting maxsamples to 1 disables frequency tracking in order to make the sources immediately selectable with only one sample. This can be useful when chronyd is started with the -q or -Q option.

minsamples samples

The minsamples directive sets the default minimum number of samples that chronyd should keep for each source. This setting can be overridden for individual sources in the server and refclock directives. The default value is 6. The useful range is 4 to 64.

Forcing chronyd to keep more samples than it would normally keep reduces noise in the estimated frequency and offset, but slows down the response to changes in the frequency and offset of the clock. The offsets in the tracking and sourcestats reports (and the tracking.log and statistics.log files) may be smaller than the actual offsets.

ntsdumpdir directory

This directive specifies a directory for the client to save NTS cookies it received from the server in order to avoid making an NTS-KE request when chronyd is started again. The cookies are saved separately for each NTP source in files named by the IP address of the NTS-KE server (e.g. 1.2.3.4.nts). By default, the client does not save the cookies.

If the directory does not exist, it will be created automatically.

An example of the directive is:

```
ntsdumpdir /var/lib/chrony
```

This directory is used also by the NTS server to save keys.

ntsrefresh interval

This directive specifies the maximum interval between NTS-KE handshakes (in seconds) in order to refresh the keys authenticating NTP packets. The default value is 2419200 (4 weeks) and the maximum value is 2^31-1 (68 years).

ntstrustedcerts [set-ID] file|directory

This directive specifies a file or directory containing certificates (in the PEM format) of trusted certificate authorities (CA) which can be used to verify certificates of NTS servers.

The optional set-ID argument is a number in the range 0 through 2^32-1, which selects the set of certificates where certificates from the specified file or directory are added. The default ID is 0, which is a set containing the system?s default trusted CAs (unless the nosystemcert directive is present). All other sets are empty by default. A set of certificates can be selected for verification of an NTS server by the certset option in the server or pool directive.

This directive can be used multiple times to specify one or more sets of trusted certificates, each containing certificates from one or more files and/or directories.

It is not necessary to restart chronyd in order to reload the certificates if they change (e.g. after a renewal).

An example is:

```
ntstrustedcerts /etc/pki/nts/foo.crt
ntstrustedcerts 1 /etc/pki/nts/bar.crt
ntstrustedcerts 1 /etc/pki/nts/baz.crt
ntstrustedcerts 2 /etc/pki/nts/qux.crt
```

nosystemcert

This directive disables the system?s default trusted CAs. Only certificates specified by the ntstrustedcerts directive will be trusted.

nocerttimecheck limit

This directive disables the checks of the activation and expiration times of certificates for the specified number of clock updates. It allows the NTS authentication mechanism to be used on computers which start with wrong time (e.g. due to not having an RTC or backup battery). Disabling the time checks has important security implications and should be used only as a last resort, preferably with a minimal number of trusted certificates. The default value is 0, which means the time checks are always enabled.

An example of the directive is:

```
nocerttimecheck 1
```

This would disable the time checks until the clock is updated for the first time, assuming the first update corrects the clock and later checks can work with correct time.

Source selection

authselectmode mode

NTP sources can be specified with the key or nts option to enable authentication to limit the impact of man-in-the-middle attacks. The attackers can drop or delay NTP packets (up to the maxdelay and maxdistance limits), but they cannot modify the timestamps contained in the packets. The attack can cause only a limited slew or step, and also cause the clock to run faster or slower than real time (up to double of the maxdrift limit).

When authentication is enabled for an NTP source, it is important to disable unauthenticated NTP sources which could be exploited in the attack, e.g. if they are not reachable only over a trusted network. Alternatively, the source selection can be configured with the require and trust options to synchronise to the unauthenticated sources only if they agree with the authenticated sources and might have a positive impact on the accuracy of the clock. Note that in this case the impact of the attack is higher. The attackers cannot cause an arbitrarily large step or slew, but they have more control over the frequency of the clock and can cause chronyd to report false information, e.g. a significantly smaller root delay and dispersion.

This directive determines the default selection options for authenticated and unauthenticated sources in order to simplify the configuration with the configuration file and chronyc commands. It sets a policy for authentication.

Sources specified with the noselect option are ignored (not counted as either authenticated or unauthenticated), and they always have only the selection options specified in the configuration.

There are four modes:

require

    Authentication is strictly required for NTP sources in this
    mode. If any unauthenticated NTP sources are specified, they
    will automatically get the noselect option to prevent them from
    being selected for synchronisation.

prefer

    In this mode, authentication is optional and preferred. If it
    is enabled for at least one NTP source, all unauthenticated NTP
    sources will get the noselect option.

mix

    In this mode, authentication is optional and synchronisation to
    a mix of authenticated and unauthenticated NTP sources is
    allowed. If both authenticated and unauthenticated NTP sources
    are specified, all authenticated NTP sources and reference
    clocks will get the require and trust options to prevent
    synchronisation to unauthenticated NTP sources if they do not
    agree with a majority of the authenticated sources and
    reference clocks. This is the default mode.

ignore

    In this mode, authentication is ignored in the source
    selection. All sources will have only the selection options
    that were specified in the configuration file, or chronyc
    command. This was the behaviour of chronyd in versions before
    4.0.

As an example, the following configuration using the default mix mode:

```
    server foo.example.net nts
    server bar.example.net nts
    server baz.example.net
    refclock SHM 0
```

is equivalent to the following configuration using the ignore mode:

```
    authselectmode ignore
```

```
server foo.example.net nts require trust
server bar.example.net nts require trust
server baz.example.net
refclock SHM 0 require trust
```

combinelimit limit

When chronyd has multiple sources available for synchronisation, it has to select one source as the synchronisation source. The measured offsets and frequencies of the system clock relative to the other sources, however, can be combined with the selected source to improve the accuracy of the system clock.

The combinelimit directive limits which sources are included in the combining algorithm. Their synchronisation distance has to be shorter than the distance of the selected source multiplied by the value of the limit. Also, their measured frequencies have to be close to the frequency of the selected source. If the selected source was specified with the prefer option, it can be combined only with other sources specified with this option.

By default, the limit is 3. Setting the limit to 0 effectively disables the source combining algorithm and only the selected source will be used to control the system clock.

maxdistance distance

The maxdistance directive sets the maximum root distance of a source to be acceptable for synchronisation of the clock. Sources that have a distance larger than the specified distance will be rejected. The distance estimates the maximum error of the source. It includes the root dispersion and half of the root delay (round-trip time) accumulated on the path to the primary source.

By default, the maximum root distance is 3 seconds.

Setting maxdistance to a larger value can be useful to allow synchronisation with a server that only has a very infrequent connection to its sources and can accumulate a large dispersion between updates of its clock.

maxjitter jitter

The maxjitter directive sets the maximum allowed jitter of the sources to not be rejected by the source selection algorithm. This prevents synchronisation with sources that have a small root distance, but their time is too variable.

By default, the maximum jitter is 1 second.

minsources sources

The minsources directive sets the minimum number of sources that need to be considered as selectable in the source selection algorithm before the local clock is updated. The default value is 1.

Setting this option to a larger number can be used to improve the reliability. More sources will have to agree with each other and the clock will not be updated when only one source (which could be serving incorrect time) is reachable.

reselectdist distance

When chronyd selects a synchronisation source from available sources, it will prefer the one with the shortest synchronisation distance. However, to avoid frequent reselecting when there are sources with similar distance, a fixed distance is added to the distance for sources that are currently not selected. This can be set with the reselectdist directive. By default, the distance is 100 microseconds.

stratumweight distance

The stratumweight directive sets how much distance should be added per stratum to the synchronisation distance when chronyd selects the synchronisation source from available sources.

By default, the weight is 0.001 seconds. This means that the stratum of the sources in the selection process matters only when the differences between the distances are in milliseconds.

System clock

clockprecision precision

The clockprecision directive specifies the precision of the system clock (in seconds). It is used by chronyd to estimate the minimum noise in NTP measurements and randomise low-order bits of timestamps in NTP responses. By default, the precision is measured on start as the minimum time to read the clock.

The measured value works well in most cases. However, it generally overestimates the precision and it can be sensitive to the CPU speed, which can change over time to save power. In some cases with a high-precision clocksource (e.g. the Time Stamp Counter of the CPU) and hardware timestamping, setting the precision on the server to a smaller value can improve stability of clients' NTP measurements. The server?s precision is reported on clients by the ntpdata command.

An example setting the precision to 8 nanoseconds is:

```
clockprecision 8e-9
```

corrtimeratio ratio

When chronyd is slewing the system clock to correct an offset, the rate at which it is slewing adds to the frequency error of the clock. On all supported systems, with the exception of macOS 12 and earlier, this rate can be controlled.

The corrtimeratio directive sets the ratio between the duration in which the clock is slewed for an average correction according to the source history and the interval in which the corrections are done (usually the NTP polling interval). Corrections larger than the average take less time and smaller corrections take more time, the amount of the correction and the correction time are inversely proportional.

Increasing corrtimeratio improves the overall frequency error of the system clock, but increases the overall time error as the corrections take longer.

By default, the ratio is set to 3, the time accuracy of the clock is preferred over its frequency accuracy.

The maximum allowed slew rate can be set by the maxslewrate directive. The current remaining correction is shown in the tracking report as the System time value.

driftfile file

One of the main activities of the chronyd program is to work out the rate at which the system clock gains or loses time relative to real time.

Whenever chronyd computes a new value of the gain or loss rate, it is desirable to record it somewhere. This allows chronyd to begin compensating the system clock at that rate whenever it is restarted, even before it has had a chance to obtain an equally good estimate of the rate during the new run. (This process can take many minutes, at least.)

The driftfile directive allows a file to be specified into which chronyd can store the rate information. Two parameters are recorded in the file. The first is the rate at which the system clock gains or loses time, expressed in parts per million, with gains positive. Therefore, a value of 100.0 indicates that when the system clock has advanced by a second, it has gained 100 microseconds in reality (so the true time has only advanced by 999900 microseconds). The second is an estimate of the error bound around the first value in which the true rate actually lies.

An example of the driftfile directive is:

```
driftfile /var/lib/chrony/drift
```

fallbackdrift min-interval max-interval

Fallback drifts are long-term averages of the system clock drift calculated over exponentially increasing intervals. They are used to avoid quickly drifting away from true time when the clock was not updated for a longer period of time and there was a short-term deviation in the drift before the updates stopped.

The directive specifies the minimum and maximum interval since the last clock update to switch between fallback drifts. They are defined as a power of 2 (in seconds). The syntax is as follows:

```
fallbackdrift 16 19
```

In this example, the minimum interval is 16 (18 hours) and the maximum interval is 19 (6 days). The system clock frequency will be set to the first fallback 18 hours after last clock update, to the second after 36 hours, and so on. This might be a good setting to cover frequency changes due to daily and weekly temperature fluctuations. When the frequency is set to a fallback, the state of the clock will change to ?Not synchronised?.

By default (or if the specified maximum or minimum is 0), no fallbacks are used and the clock frequency changes only with new measurements from NTP sources, reference clocks, or manual input.

leapsecmode mode

A leap second is an adjustment that is occasionally applied to UTC to keep it close to the mean solar time. When a leap second is inserted, the last day of June or December has an extra second 23:59:60.

For computer clocks that is a problem. The Unix time is defined as number of seconds since 00:00:00 UTC on 1 January 1970 without leap seconds. The system clock cannot have time 23:59:60, every minute has 60 seconds and every day has 86400 seconds by definition. The inserted leap second is skipped and the clock is suddenly ahead of UTC by one second. The leapsecmode directive selects how that error is corrected. There are four options:

system

When inserting a leap second, the kernel steps the system clock backwards by one second when the clock gets to 00:00:00 UTC. When deleting a leap second, it steps forward by one second when the clock gets to 23:59:59 UTC. This is the default mode when the system driver supports leap seconds (i.e. all supported systems with the exception of macOS 12 and earlier).

step

This is similar to the system mode, except the clock is stepped by chronyd instead of the kernel. It can be useful to avoid bugs in the kernel code that would be executed in the system mode. This is the default mode when the system driver does not support leap seconds.

slew

The clock is corrected by slewing started at 00:00:00 UTC when a leap second is inserted or 23:59:59 UTC when a leap second is deleted. This might be preferred over the system and step modes when applications running on the system are sensitive to jumps in the system time and it is acceptable that the clock will be off for a longer time. On Linux with the default maxslewrate value the correction takes 12 seconds.

ignore

No correction is applied to the clock for the leap second. The clock will be corrected later in normal operation when new measurements are made and the estimated offset includes the one second error. This option is particularly useful when multiple chronyd instances are running on the system, one controlling the system clock and others started with the -x option, which should rely on the first instance to correct the system clock and ignore it for the correction of their own NTP clock running on top of the system clock.

When serving time to NTP clients that cannot be configured to correct their clocks for a leap second by slewing, or to clients that would correct at slightly different rates when it is necessary to keep them close together, the slew mode can be combined with the smoothtime directive to enable a server leap smear.

When smearing a leap second, the leap status is suppressed on the server and the served time is corrected slowly by slewing instead of stepping. The clients do not need any special configuration as they do not know there is any leap second and they follow the server time which eventually brings them back to UTC. Care must be taken to ensure they use only NTP servers which smear the leap second in exactly the same way for synchronisation.

This feature must be used carefully, because the server is intentionally not serving its best estimate of the true time.

A recommended configuration to enable a server leap smear is:

```
leapsecmode slew
maxslewrate 1000
smoothtime 400 0.001024 leaponly
```

The first directive is necessary to disable the clock step which would reset the smoothing process. The second directive limits the slewing rate of the local clock to 1000 ppm, which improves the stability of the smoothing process when the local correction starts and ends. The third directive enables the server time smoothing process. It will start when the clock gets to 00:00:00 UTC and it will take 62500 seconds (about 17.36 hours) to finish. The frequency offset will be changing by 0.001024 ppm per second and will reach a maximum of 32 ppm in 31250 seconds. The leaponly option makes the duration of the leap smear constant and allows the clients to safely synchronise with multiple identically configured leap smearing servers.

The duration of the leap smear can be calculated from the specified wander as

```
duration = sqrt(4 / wander)
```

leapsectz timezone

This directive specifies a timezone in the system timezone database which chronyd can use to determine when will the next leap second occur and what is the current offset between TAI and UTC. It will periodically check if 23:59:59 and 23:59:60 are valid times in the timezone. This normally works with the right/UTC timezone.

When a leap second is announced, the timezone needs to be updated at least 12 hours before the leap second. It is not necessary to restart chronyd.

This directive is useful with reference clocks and other time sources which do not announce leap seconds, or announce them too late for an NTP server to forward them to its own clients. Clients of leap smearing servers must not use this directive.

It is also useful when the system clock is required to have correct TAI-UTC offset. Note that the offset is set only when leap seconds are handled by the kernel, i.e. leapsecmode is set to system.

The specified timezone is not used as an exclusive source of information about leap seconds. If a majority of time sources announce on the last day of June or December that a leap second should be inserted or deleted, it will be accepted even if it is not included in the timezone.

An example of the directive is:

```
leapsectz right/UTC
```

The following shell command verifies that the timezone contains leap seconds and can be used with this directive:

```
$ TZ=right/UTC date -d 'Dec 31 2008 23:59:60'
Wed Dec 31 23:59:60 UTC 2008
```

makestep threshold limit

Normally chronyd will cause the system to gradually correct any time offset, by slowing down or speeding up the clock as required. In certain situations, e.g. when chronyd is initially started, the system clock might be so far adrift that this slewing process would take a very long time to correct the system clock.

This directive forces chronyd to step the system clock if the adjustment is larger than a threshold value, but only if there were no more clock updates since chronyd was started than the specified limit. A negative value disables the limit.

On most systems it is desirable to step the system clock only on boot, before starting programs that rely on time advancing monotonically forwards.

An example of the use of this directive is:

```
makestep 0.1 3
```

This would step the system clock if the adjustment is larger than 0.1 seconds, but only in the first three clock updates.

maxchange offset start ignore

This directive sets the maximum offset to be accepted on a clock update. The offset is measured relative to the current estimate of the true time, which is different from the system time if a previous slew did not finish.

The check is enabled after the specified number of clock updates to allow a large initial offset to be corrected on start. Offsets larger than the specified maximum will be ignored for the specified number of times. Another large offset will cause chronyd to give up and exit. A negative value can be used to disable the limit to ignore all large offsets. A syslog message will be generated when an offset is ignored or it causes the exit.

An example of the use of this directive is:

```
maxchange 1000 1 2
```

After the first clock update, chronyd will check the offset on every clock update, it will ignore two adjustments larger than 1000 seconds and exit on another one.

maxclockerror error-in-ppm

The maxclockerror directive sets the maximum assumed frequency error that the system clock can gain on its own between clock updates. It describes the stability of the clock.

By default, the maximum error is 1 ppm.

Typical values for error-in-ppm might be 10 for a low quality clock and 0.1 for a high quality clock using a temperature compensated crystal oscillator.

maxdrift drift-in-ppm

This directive specifies the maximum assumed drift (frequency error) of the system clock. It limits the frequency adjustment that chronyd is allowed to use to correct the measured drift. It is an additional limit to the maximum adjustment that can be set by the system driver (100000 ppm on Linux, 500 ppm on FreeBSD, NetBSD, and macOS 10.13+, 32500 ppm on illumos).

By default, the maximum assumed drift is 500000 ppm, i.e. the adjustment is limited by the system driver rather than this directive.

maxupdateskew skew-in-ppm

One of chronyd's tasks is to work out how fast or slow the computer?s clock runs relative to its reference sources. In addition, it computes an estimate of the error bounds around the estimated value.

If the range of error is too large, it probably indicates that the measurements have not settled down yet, and that the estimated gain or loss rate is not very reliable.

The maxupdateskew directive sets the threshold for determining whether an estimate might be so unreliable that it should not be used. By default, the threshold is 1000 ppm.

Typical values for skew-in-ppm might be 100 for NTP sources polled over a wireless network, and 10 or smaller for sources on a local wired network.

It should be noted that this is not the only means of protection against using unreliable estimates. At all times, chronyd keeps track of both the estimated gain or loss rate, and the error bound on the estimate. When a new estimate is generated following another measurement from one of the sources, a weighted combination algorithm is used to update the master estimate. So if chronyd has an existing highly-reliable master estimate and a new estimate is generated which has large error bounds, the existing master estimate will dominate in the new master estimate.

maxslewrate rate-in-ppm

The maxslewrate directive sets the maximum rate at which chronyd is allowed to slew the time. It limits the slew rate controlled by the correction time ratio (which can be set by the corrtimeratio directive) and is effective only on systems where chronyd is able to control the rate (i.e. all supported systems with the exception of macOS 12 or earlier).

For each system there is a maximum frequency offset of the clock that can be set by the driver. On Linux it is 100000 ppm, on FreeBSD, NetBSD and macOS 10.13+ it is 5000 ppm, and on illumos it is 32500 ppm. Also, due to a kernel limitation, setting maxslewrate on FreeBSD, NetBSD, macOS 10.13+ to a value between 500 ppm and

5000 ppm will effectively set it to 500 ppm.

By default, the maximum slew rate is set to 83333.333 ppm (one twelfth).

tempcomp file interval T0 k0 k1 k2, tempcomp file interval points-file

Normally, changes in the rate of drift of the system clock are caused mainly by changes in the temperature of the crystal oscillator on the motherboard.

If there are temperature measurements available from a sensor close to the oscillator, the tempcomp directive can be used to compensate for the changes in the temperature and improve the stability and accuracy of the clock.

The result depends on many factors, including the resolution of the sensor, the amount of noise in the measurements, the polling interval of the time source, the compensation update interval, how well the compensation is specified, and how close the sensor is to the oscillator. When it is working well, the frequency reported in the tracking.log file is more stable and the maximum reached offset is smaller.

There are two forms of the directive. The first one has six parameters: a path to the file containing the current temperature from the sensor (in text format), the compensation update interval (in seconds), and temperature coefficients T0, k0, k1, k2.

The frequency compensation is calculated (in ppm) as

```
comp = k0 + (T - T0) * k1 + (T - T0)^2 * k2
```

The result has to be between -10 ppm and 10 ppm, otherwise the measurement is considered invalid and will be ignored. The k0 coefficient can be adjusted to keep the compensation in that range.

An example of the use is:

```
tempcomp /sys/class/hwmon/hwmon0/temp2_input 30 26000 0.0 0.000183 0.0
```

The measured temperature will be read from the file in the Linux sysfs filesystem every 30 seconds. When the temperature is 26000 (26 degrees Celsius), the frequency correction will be zero. When it is 27000 (27 degrees Celsius), the clock will be set to run faster by 0.183 ppm, etc.

The second form has three parameters: the path to the sensor file, the update interval, and a path to a file containing a list of (temperature, compensation) points, from which the compensation is linearly interpolated or extrapolated.

An example is:

```
tempcomp /sys/class/hwmon/hwmon0/temp2_input 30 /etc/chrony.tempcomp
```

where the /etc/chrony.tempcomp file could have

```
20000 1.0
21000 0.64
22000 0.36
23000 0.16
24000 0.04
25000 0.0
26000 0.04
27000 0.16
28000 0.36
29000 0.64
30000 1.0
```

Valid measurements with corresponding compensations are logged to the tempcomp.log file if enabled by the log tempcomp directive.

NTP server

allow [all] [subnet]

The allow directive is used to designate a particular subnet from which NTP clients are allowed to access the computer as an NTP server. It also controls access of NTS-KE clients when NTS is enabled on the server.

The default is that no clients are allowed access, i.e. chronyd operates purely as an NTP client. If the allow directive is used, chronyd will be both a client of its servers, and a server to other clients.

This directive can be used multiple times.

Examples of the use of the directive are as follows:

```
    allow 1.2.3.4
    allow 3.4.5.0/24
    allow 3.4.5
    allow 2001:db8::/32
    allow 0/0
    allow ::/0
    allow
```

The first directive allows access from an IPv4 address. The second directive allows access from all computers in an IPv4 subnet specified in the CIDR notation. The third directive specifies the same subnet using a simpler notation where the prefix length is determined by the number of dots. The fourth directive specifies an IPv6 subnet. The fifth and sixth directives allow access from all IPv4 and IPv6 addresses respectively. The seventh directive allows access from all addresses (both IPv4 or IPv6).

A second form of the directive, allow all, has a greater effect, depending on the ordering of directives in the configuration file.

To illustrate the effect, consider the two examples:

```
    allow 1.2.3.4
    deny 1.2.3.0/24
    allow 1.2.0.0/16
```

and

```
    allow 1.2.3.4
    deny 1.2.3.0/24
    allow all 1.2.0.0/16
```

In the first example, the effect is the same regardless of what order the three directives are given in. So the 1.2.0.0/16 subnet is allowed access, except for the 1.2.3.0/24 subnet, which is denied access, however the host 1.2.3.4 is allowed access.

In the second example, the allow all 1.2.0.0/16 directive overrides the effect of any previous directive relating to a subnet within the specified subnet. Within a configuration file this capability is probably rather moot; however, it is of greater use for reconfiguration at run-time via chronyc with the allow all command.

The rules are internally represented as a tree of tables with one level per four bits of the IPv4 or IPv6 address. The order of the allow and deny directives matters if they modify the same records of one table, i.e. if one subnet is included in the other subnet and their prefix lengths are at the same level. For example, 1.2.3.0/28 and 1.2.3.0/29 are in different tables, but 1.2.3.0/25 and 1.2.3.0/28 are in the same table. The configuration can be verified for individual addresses with the accheck command in chronyc.

A hostname can be specified in the directives instead of the IP address, but the name must be resolvable when chronyd is started, i.e. the network is already up and DNS is working. If the hostname resolves to multiple addresses, only the first address (in the order returned by the system resolver) will be allowed or denied.

Note, if the initstepslew directive is used in the configuration file, each of the computers listed in that directive must allow client access by this computer for it to work.

deny [all] [subnet]

This is similar to the allow directive, except that it denies NTP and NTS-KE client access to a particular subnet or host, rather than allowing it.

The syntax is identical and the directive can be used multiple times too.

There is also a deny all directive with similar behaviour to the allow all directive.

bindaddress address

The bindaddress directive binds the sockets on which chronyd listens for NTP and NTS-KE requests to a local address of the computer. On systems other than Linux, the address of the computer needs to be already configured when chronyd is started.

An example of the use of the directive is:

```
bindaddress 192.168.1.1
```

Currently, for each of the IPv4 and IPv6 protocols, only one bindaddress directive can be specified. Therefore, it is not useful on computers which should serve NTP on multiple network interfaces.

binddevice interface

The binddevice directive binds the NTP and NTS-KE server sockets to a network device specified by the interface name. This directive can specify only one interface and it is supported on Linux only.

An example of the directive is:

```
binddevice eth0
```

broadcast interval address [port]

The broadcast directive is used to declare a broadcast address to which chronyd should send packets in the NTP broadcast mode (i.e. make chronyd act as a broadcast server). Broadcast clients on that subnet will be able to synchronise.

This directive can be used multiple times to specify multiple addresses.

The syntax is as follows:

```
broadcast 32 192.168.1.255
broadcast 64 192.168.2.255 12123
broadcast 64 ff02::101
```

In the first example, the destination port defaults to UDP port 123 (the normal NTP port). In the second example, the destination port is specified as 12123. The first parameter in each case (32 or 64 respectively) is the interval in seconds between broadcast packets being sent. The second parameter in each case is the broadcast address to send the packet to. This should correspond to the broadcast address of one of the network interfaces on the computer where chronyd is running.

You can have more than 1 broadcast directive if you have more than 1 network interface onto which you want to send NTP broadcast packets.

chronyd itself cannot act as a broadcast client; it must always be configured as a point-to-point client by defining specific NTP servers and peers. This broadcast server feature is intended for providing a time source to other NTP implementations.

If ntpd is used as the broadcast client, it will try to measure the round-trip delay between the server and client with normal client mode packets. Thus, the broadcast subnet should also be the subject of an allow directive.

clientloglimit limit

This directive specifies the maximum amount of memory that chronyd is allowed to allocate for logging of client accesses and the state that chronyd as an NTP server needs to support the interleaved mode for its clients. The default limit is 524288 bytes, which enables monitoring of up to 4096 IP addresses at the same time and holding NTP timestamps for up to 4096 clients using the interleaved mode (depending on uniformity of their polling interval). The number of addresses and timestamps is always a power of 2. The maximum effective value is 2147483648 (2 GB), which corresponds to 16777216 addresses and timestamps.

An example of the use of this directive is:

```
clientloglimit 1048576
```

noclientlog

This directive, which takes no arguments, specifies that client accesses are not to be logged. Normally they are logged, allowing statistics to be reported using the clients command in chronyc.

This option also effectively disables server support for the NTP interleaved mode.

local [option]...

The local directive enables a local reference mode, which allows chronyd operating as an NTP server to appear synchronised to real time (from the viewpoint of clients polling it), even when it was never synchronised or the last update of the clock happened a long time ago.

This directive is normally used in an isolated network, where computers are required to be synchronised to one another, but not necessarily to real time. The server can be kept vaguely in line with real time by manual input.

The local directive has the following options:

stratum stratum

This option sets the stratum of the server which will be reported to clients when the local reference is active. The specified value is in the range 1 through 15, and the default value is 10. It should be larger than the maximum expected stratum in the network when external NTP servers are accessible.

Stratum 1 indicates a computer that has a true real-time reference directly connected to it (e.g. GPS, atomic clock, etc.), such computers are expected to be very close to real time. Stratum 2 computers are those which have a stratum 1 server; stratum 3 computers have a stratum 2 server and so on. A value of 10 indicates that the clock is so many hops away from a reference clock that its time is fairly unreliable.

distance distance

This option sets the threshold for the root distance which will activate the local reference. If chronyd was synchronised to some source, the local reference will not be activated until its root distance reaches the specified value (the rate at which the distance is increasing depends on how well the clock was tracking the source). The default value is 1 second.

The current root distance can be calculated from root delay and root dispersion (reported by the tracking command in chronyc) as:

```
distance = delay / 2 + dispersion
```

orphan

This option enables a special ?orphan? mode, where sources with stratum equal to the local stratum are assumed to not serve real time. They are ignored unless no other source is selectable and their reference IDs are smaller than the local reference ID.

This allows multiple servers in the network to use the same local configuration and to be synchronised to one another, without confusing clients that poll more than one server. Each server needs to be configured to poll all other servers with the local directive. This ensures only the server with the smallest reference ID has the local reference active and others are synchronised to it. If that server stops responding, the server with the second smallest reference ID will take over when its local reference mode activates (root distance reaches the threshold configured by the distance option).

The orphan mode is compatible with the ntpd's orphan mode (enabled by the tos orphan command).

An example of the directive is:

```
local stratum 10 orphan distance 0.1
```

ntpsigndsocket directory

This directive specifies the location of the Samba ntp_signd socket when it is running as a Domain Controller (DC). If chronyd is compiled with this feature, responses to MS-SNTP clients will be signed by the smbd daemon.

Note that MS-SNTP requests are not authenticated and any client that is allowed to access the server by the allow directive, or the allow command in chronyc, can get an MS-SNTP response signed with a trust account?s password and try to crack the password in a brute-force attack. Access to the server should be carefully controlled.

An example of the directive is:

```
ntpsigndsocket /var/lib/samba/ntp_signd
```

ntsport port

This directive specifies the TCP port on which chronyd will provide the NTS Key Establishment (NTS-KE) service. The default port is 4460.

The port will be open only when a certificate and key is specified by the ntsservercert and ntsserverkey directives.

ntsservercert file

This directive specifies a file containing a certificate in the PEM format for chronyd to operate as an NTS server. The file should also include any intermediate certificates that the clients will need to validate the server?s certificate. The file needs to be readable by the user under which chronyd is running after dropping root privileges.

This directive can be used multiple times to specify multiple certificates for different names of the server.

The files are loaded only once. chronyd needs to be restarted in order to load a renewed certificate. The ntsdumpdir and dumpdir directives with the -r option of chronyd are recommended for a near-seamless server operation.

ntsserverkey file

This directive specifies a file containing a private key in the PEM format for chronyd to operate as an NTS server. The file needs to be readable by the user under which chronyd is running after dropping root privileges. For security reasons, it should not be readable by other users.

This directive can be used multiple times to specify multiple keys. The number of keys must be the same as the number of certificates and the corresponding files must be specified in the same order.

ntsprocesses processes

This directive specifies how many helper processes will chronyd operating as an NTS server start for handling client NTS-KE requests in order to improve performance with multi-core CPUs and multithreading. If set to 0, no helper process will be started and all NTS-KE requests will be handled by the main chronyd process. The default value is 1.

maxntsconnections connections

This directive specifies the maximum number of concurrent NTS-KE connections per process that the NTS server will accept. The

default value is 100. The maximum practical value is half of the
system FD_SETSIZE constant (usually 1024).

ntsdumpdir directory

This directive specifies a directory where chronyd operating as an NTS server can save the keys which encrypt NTS cookies provided to clients. The keys are saved to a single file named ntskeys. When chronyd is restarted, reloading the keys allows the clients to continue using old cookies and avoids a storm of NTS-KE requests. By default, the server does not save the keys.

An example of the directive is:

```
ntsdumpdir /var/lib/chrony
```

This directory is used also by the NTS client to save NTS cookies.

ntsntpserver hostname

This directive specifies the hostname (as a fully qualified domain name) or address of the NTP server(s) which is provided in the NTS-KE response to the clients. It allows the NTS-KE server to be separated from the NTP server. However, the servers need to share the keys, i.e. external key management needs to be enabled by setting ntsrotate to 0. By default, no hostname or address is provided to the clients, which means they should use the same server for NTS-KE and NTP.

ntsrotate interval

This directive specifies the rotation interval (in seconds) of the server key which encrypts the NTS cookies. New keys are generated automatically from the /dev/urandom device. The server keeps two previous keys to give the clients time to get new cookies encrypted by the latest key. The interval is measured as the server?s operating time, i.e. the actual interval can be longer if chronyd is not running continuously. The default interval is 604800 seconds (1 week). The maximum value is 2^31-1 (68 years).

The automatic rotation of the keys can be disabled by setting ntsrotate to 0. In this case the keys are assumed to be managed externally. chronyd will not save the keys to the ntskeys file and

will reload the keys from the file when the rekey command is issued
in chronyc. The file can be periodically copied from another server
running chronyd (which does not have ntsrotate set to 0) in order
to have one or more servers dedicated to NTS-KE. The NTS-KE servers
need to be configured with the ntsntpserver directive to point the
clients to the right NTP server.

An example of the directive is:

```
ntsrotate 2592000
```

port port

This option allows you to configure the port on which chronyd will
listen for NTP requests. The port will be open only when an address
is allowed by the allow directive or the allow command in chronyc,
an NTP peer is configured, or the broadcast server mode is enabled.
The default value is 123, the standard NTP port. If set to 0,
chronyd will never open the server port and will operate strictly
in a client-only mode. The source port used in NTP client requests
can be set by the acquisitionport directive.

ratelimit [option]...

This directive enables response rate limiting for NTP packets. Its
purpose is to reduce network traffic with misconfigured or broken
NTP clients that are polling the server too frequently. The limits
are applied to individual IP addresses. If multiple clients share
one IP address (e.g. multiple hosts behind NAT), the sum of their
traffic will be limited. If a client that increases its polling
rate when it does not receive a reply is detected, its rate
limiting will be temporarily suspended to avoid increasing the
overall amount of traffic. The maximum number of IP addresses which
can be monitored at the same time depends on the memory limit set
by the clientloglimit directive.

The ratelimit directive supports a number of options (which can be
defined in any order):

interval interval

This option sets the minimum interval between responses. It is defined as a power of 2 in seconds. The default value is 3 (8 seconds). The minimum value is -19 (524288 packets per second) and the maximum value is 12 (one packet per 4096 seconds). Note that with values below -4 the rate limiting is coarse (responses are allowed in bursts, even if the interval between them is shorter than the specified interval).

burst responses

This option sets the maximum number of responses that can be sent in a burst, temporarily exceeding the limit specified by the interval option. This is useful for clients that make rapid measurements on start (e.g. chronyd with the iburst option). The default value is 8. The minimum value is 1 and the maximum value is 255.

leak rate

This option sets the rate at which responses are randomly allowed even if the limits specified by the interval and burst options are exceeded. This is necessary to prevent an attacker who is sending requests with a spoofed source address from completely blocking responses to that address. The leak rate is defined as a power of 1/2 and it is 2 by default, i.e. on average at least every fourth request has a response. The minimum value is 1 and the maximum value is 4.

An example use of the directive is:

```
ratelimit interval 1 burst 16
```

This would reduce the response rate for IP addresses sending packets on average more than once per 2 seconds, or sending packets in bursts of more than 16 packets, by up to 75% (with default leak of 2).

ntsratelimit [option]...

This directive enables rate limiting of NTS-KE requests. It is similar to the ratelimit directive, except the default interval is 6 (1 connection per 64 seconds).

An example of the use of the directive is:

```
ntsratelimit interval 3 burst 1
```

smoothtime max-freq max-wander [leaponly]

The smoothtime directive can be used to enable smoothing of the time that chronyd serves to its clients to make it easier for them to track it and keep their clocks close together even when large offset or frequency corrections are applied to the server?s clock, for example after being offline for a longer time.

BE WARNED: The server is intentionally not serving its best estimate of the true time. If a large offset has been accumulated, it can take a very long time to smooth it out. This directive should be used only when the clients are not configured to also poll another NTP server, because they could reject this server as a falseticker or fail to select a source completely.

The smoothing process is implemented with a quadratic spline function with two or three pieces. It is independent from any slewing applied to the local system clock, but the accumulated offset and frequency will be reset when the clock is corrected by stepping, e.g. by the makestep directive or the makestep command in chronyc. The process can be reset without stepping the clock by the smoothtime reset command.

The first two arguments of the directive are the maximum frequency offset of the smoothed time to the tracked NTP time (in ppm) and the maximum rate at which the frequency offset is allowed to change (in ppm per second). leaponly is an optional third argument which enables a mode where only leap seconds are smoothed out and normal offset and frequency changes are ignored. The leaponly option is useful in a combination with the leapsecmode slew directive to allow the clients to use multiple time smoothing servers safely.

The smoothing process is activated automatically when 1/10000 of the estimated skew of the local clock falls below the maximum rate of frequency change. It can be also activated manually by the smoothtime activate command, which is particularly useful when the clock is synchronised only with manual input and the skew is always larger than the threshold. The smoothing command can be used to monitor the process.

An example suitable for clients using ntpd and 1024 second polling interval could be:

```
smoothtime 400 0.001
```

An example suitable for clients using chronyd on Linux could be:

```
smoothtime 50000 0.01
```

Command and monitoring access

bindcmdaddress address

The bindcmdaddress directive specifies a local IP address to which chronyd will bind the UDP socket listening for monitoring command packets (issued by chronyc). On systems other than Linux, the address of the interface needs to be already configured when chronyd is started.

This directive can also change the path of the Unix domain command socket, which is used by chronyc to send configuration commands. The socket must be in a directory that is accessible only by the root or chrony user. The directory will be created on start if it does not exist. The compiled-in default path of the socket is /run/chrony/chronyd.sock. The socket can be disabled by setting the path to /.

By default, chronyd binds the UDP sockets to the addresses 127.0.0.1 and ::1 (i.e. the loopback interface). This blocks all access except from localhost. To listen for command packets on all interfaces, you can add the lines:

```
bindcmdaddress 0.0.0.0
bindcmdaddress ::
```

to the configuration file.

For each of the IPv4, IPv6, and Unix domain protocols, only one bindcmdaddress directive can be specified.

An example that sets the path of the Unix domain command socket is:

```
bindcmdaddress /var/run/chrony/chronyd.sock
```

bindcmddevice interface

The bindcmddevice directive binds the UDP command sockets to a network device specified by the interface name. This directive can specify only one interface and it is supported on Linux only.

An example of the directive is:

```
bindcmddevice eth0
```

cmdallow [all] [subnet]

This is similar to the allow directive, except that it allows monitoring access (rather than NTP client access) to a particular subnet or host. (By ?monitoring access? is meant that chronyc can be run on those hosts and retrieve monitoring data from chronyd on this computer.)

The syntax is identical to the allow directive.

There is also a cmdallow all directive with similar behaviour to the allow all directive (but applying to monitoring access in this case, of course).

Note that chronyd has to be configured with the bindcmdaddress directive to not listen only on the loopback interface to actually allow remote access.

cmddeny [all] [subnet]

This is similar to the cmdallow directive, except that it denies monitoring access to a particular subnet or host, rather than allowing it.

The syntax is identical.

There is also a cmddeny all directive with similar behaviour to the cmdallow all directive.

cmdport port

The cmdport directive allows the port that is used for run-time monitoring (via the chronyc program) to be altered from its default (323). If set to 0, chronyd will not open the port, this is useful to disable chronyc access from the Internet. (It does not disable the Unix domain command socket.)

An example shows the syntax:

```
cmdport 257
```

This would make chronyd use UDP 257 as its command port. (chronyc would need to be run with the -p 257 switch to inter-operate correctly.)

cmdratelimit [option]...

This directive enables response rate limiting for command packets. It is similar to the ratelimit directive, except responses to localhost are never limited and the default interval is -4 (16 packets per second).

An example of the use of the directive is:

```
cmdratelimit interval 2
```

Real-time clock (RTC)

hwclockfile file

The hwclockfile directive sets the location of the adjtime file which is used by the hwclock program on Linux. chronyd parses the file to find out if the RTC keeps local time or UTC. It overrides the rtconutc directive.

The compiled-in default value is '/etc/adjtime'.

An example of the directive is:

```
hwclockfile /etc/adjtime
```

rtcautotrim threshold

The rtcautotrim directive is used to keep the RTC close to the system clock automatically. When the system clock is synchronised and the estimated error between the two clocks is larger than the specified threshold, chronyd will trim the RTC as if the trimrtc command in chronyc was issued. The trimming operation is accurate to only about 1 second, which is the minimum effective threshold. This directive is effective only with the rtcfile directive.

An example of the use of this directive is:

```
rtcautotrim 30
```

This would set the threshold error to 30 seconds.

rtcdevice device

The rtcdevice directive sets the path to the device file for accessing the RTC. The default path is /dev/rtc.

rtcfile file

The rtcfile directive defines the name of the file in which chronyd can save parameters associated with tracking the accuracy of the RTC.

An example of the directive is:

```
rtcfile /var/lib/chrony/rtc
```

chronyd saves information in this file when it exits and when the writertc command is issued in chronyc. The information saved is the RTC?s error at some epoch, that epoch (in seconds since January 1 1970), and the rate at which the RTC gains or loses time.

So far, the support for real-time clocks is limited; their code is even more system-specific than the rest of the software. You can only use the RTC facilities (the rtcfile directive and the -s command-line option to chronyd) if the following three conditions apply:

1. You are running Linux.
2. The kernel is compiled with extended real-time clock support (i.e. the /dev/rtc device is capable of doing useful things).
3. You do not have other applications that need to make use of /dev/rtc at all.

rtconutc

chronyd assumes by default that the RTC keeps local time (including any daylight saving changes). This is convenient on PCs running Linux which are dual-booted with Windows.

If you keep the RTC on local time and your computer is off when daylight saving (summer time) starts or ends, the computer?s system time will be one hour in error when you next boot and start chronyd.

An alternative is for the RTC to keep Universal Coordinated Time (UTC). This does not suffer from the 1 hour problem when daylight saving starts or ends.

If the rtconutc directive appears, it means the RTC is required to keep UTC. The directive takes no arguments. It is equivalent to specifying the -u switch to the Linux hwclock program.

Note that this setting is overridden by the hwclockfile file and is not relevant for the rtcsync directive.

rtcsync

The rtcsync directive enables a mode where the system time is periodically copied to the RTC and chronyd does not try to track its drift. This directive cannot be used with the rtcfile directive.

On Linux, the RTC copy is performed by the kernel every 11 minutes.

On macOS, chronyd will perform the RTC copy every 60 minutes when the system clock is in a synchronised state.

On other systems this directive does nothing.

Logging

log [option]...

The log directive indicates that certain information is to be logged. The log files are written to the directory specified by the logdir directive. A banner is periodically written to the files to indicate the meanings of the columns.

rawmeasurements

This option logs the raw NTP measurements and related information to a file called measurements.log. An entry is made for each packet received from the source. This can be useful when debugging a problem. An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2016-11-09 05:40:50 203.0.113.15    N  2 111 111 1111  10 10 1.0 \
   -4.966e-03  2.296e-01  1.577e-05  1.615e-01  7.446e-03 CB00717B 4B D K
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2015-10-13]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [05:40:50]
3. IP address of server or peer from which measurement came

[203.0.113.15]

4. Leap status (N means normal, + means that the last minute of the current month has 61 seconds, - means that the last minute of the month has 59 seconds, ? means the remote computer is not currently synchronised.) [N]
5. Stratum of remote computer. [2]
6. RFC 5905 tests 1 through 3 (1=pass, 0=fail) [111]
7. RFC 5905 tests 5 through 7 (1=pass, 0=fail) [111]
8. Results of the maxdelay, maxdelayratio, and maxdelaydevratio (or maxdelayquant) tests, and a test for synchronisation loop (1=pass, 0=fail). The first test from these four also checks the server precision, response time, and whether an interleaved response is acceptable for synchronisation. [1111]
9. Local poll [10]
10. Remote poll [10]
11. ?Score? (an internal score within each polling level used to decide when to increase or decrease the polling level. This is adjusted based on number of measurements currently being used for the regression algorithm). [1.0]
12. The estimated local clock error (theta in RFC 5905). Positive indicates that the local clock is slow of the remote source. [-4.966e-03]
13. The peer delay (delta in RFC 5905). [2.296e-01]
14. The peer dispersion (epsilon in RFC 5905). [1.577e-05]
15. The root delay (DELTA in RFC 5905). [1.615e-01]
16. The root dispersion (EPSILON in RFC 5905). [7.446e-03]
17. Reference ID of the server?s source as a hexadecimal number. [CB00717B]
18. NTP mode of the received packet (1=active peer, 2=passive peer, 4=server, B=basic, I=interleaved). [4B]
19. Source of the local transmit timestamp (D=daemon, K=kernel, H=hardware). [D]

20. Source of the local receive timestamp (D=daemon, K=kernel, H=hardware). [K]

measurements

This option is identical to the rawmeasurements option, except it logs only valid measurements from synchronised sources, i.e. measurements which passed the RFC 5905 tests 1 through 7. This can be useful for producing graphs of the source?s performance.

statistics

This option logs information about the regression processing to a file called statistics.log. An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2016-08-10 05:40:50 203.0.113.15     6.261e-03 -3.247e-03 \
     2.220e-03  1.874e-06  1.080e-06 7.8e-02  16   0   8  0.00
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2015-07-22]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [05:40:50]
3. IP address of server or peer from which measurement comes [203.0.113.15]
4. The estimated standard deviation of the measurements from the source (in seconds). [6.261e-03]
5. The estimated offset of the source (in seconds, positive means the local clock is estimated to be fast, in this case). [-3.247e-03]
6. The estimated standard deviation of the offset estimate (in seconds). [2.220e-03]
7. The estimated rate at which the local clock is gaining or losing time relative to the source (in seconds per second, positive means the local clock is gaining). This is relative to the compensation currently being applied to the local clock, not to the local clock without any

compensation. [1.874e-06]

8. The estimated error in the rate value (in seconds per second). [1.080e-06].
9. The ratio of |old_rate - new_rate| / old_rate_error. Large values indicate the statistics are not modelling the source very well. [7.8e-02]
10. The number of measurements currently being used for the regression algorithm. [16]
11. The new starting index (the oldest sample has index 0; this is the method used to prune old samples when it no longer looks like the measurements fit a linear model). [0, i.e. no samples discarded this time]
12. The number of runs. The number of runs of regression residuals with the same sign is computed. If this is too small it indicates that the measurements are no longer represented well by a linear model and that some older samples need to be discarded. The number of runs for the data that is being retained is tabulated. Values of approximately half the number of samples are expected. [8]
13. The estimated or configured asymmetry of network jitter on the path to the source which was used to correct the measured offsets. The asymmetry can be between -0.5 and +0.5. A negative value means the delay of packets sent to the source is more variable than the delay of packets sent from the source back. [0.00, i.e. no correction for asymmetry]

selection

This option logs information about selection of sources for synchronisation to a file called selection.log. Note that the rate of entries written to this file grows quadratically with the number of specified sources (each measurement triggers the selection for all sources). An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2022-05-01 02:01:20 203.0.113.15    * -----  377  1.00  \
     4.228e+01 -1.575e-04  1.239e-04
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2022-05-01]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [02:01:20]
3. IP address or reference ID of the source. [203.0.113.15]
4. State of the source indicated with one of the following symbols. [*]

    Not considered selectable for synchronisation:

    ? N - has the noselect option.

    ? s - is not synchronised.

    ? M - does not have enough measurements.

    ? d - has a root distance larger than the maximum distance (configured by the maxdistance directive).

    ? ~ - has a jitter larger than the maximum jitter (configured by the maxjitter directive).

    ? w - waits for other sources to get out of the M state.

    ? S - has older measurements than other sources.

    ? O - has a stratum equal or larger than the orphan stratum (configured by the local directive).

    ? T - does not fully agree with sources that have the trust option.

    ? x - does not agree with other sources (falseticker).

    Considered selectable for synchronisation, but not currently used:

    ? W - waits for other sources to be selectable (required by the minsources directive, or the require option of another source).

? P - another selectable source is preferred due to the prefer option.

? U - waits for a new measurement (after selecting a different best source).

? D - has, or recently had, a root distance which is too large to be combined with other sources (configured by the combinelimit directive).

Used for synchronisation of the local clock:

? + - combined with the best source.

? * - selected as the best source to update the reference data (e.g. root delay, root dispersion).

5. Reachability register printed as an octal number. The register has 8 bits and is updated on every received or missed packet from the source. A value of 377 indicates that a valid reply was received for all from the last eight transmissions. [377]
6. Current score against the source in the * state. The scoring system avoids frequent reselection when multiple sources have a similar root distance. A value larger than 1 indicates this source was better than the * source in recent selections. If the score reaches 10, the best source will be reselected and the scores will be reset to 1. [1.00]
7. Interval since the last measurement of the source in seconds. [4.228e+01]
8. Lower endpoint of the interval which was expected to contain the true offset of the local clock determined by the root distance of the source. [-1.575e-04]
9. Upper endpoint of the interval which was expected to contain the true offset of the local clock determined by the root distance of the source. [1.239e-04]

tracking

This option logs changes to the estimate of the system?s gain or loss rate, and any slews made, to a file called tracking.log. An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2017-08-22 13:22:36 203.0.113.15     2     -3.541      0.075 -8.621e-06 N \
            2  2.940e-03 -2.084e-04  1.534e-02  3.472e-04  8.304e-03
```

The columns are as follows (the quantities in square brackets are the values from the example line above) :

1. Date [2017-08-22]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [13:22:36]
3. The IP address of the server or peer to which the local system is synchronised. [203.0.113.15]
4. The stratum of the local system. [2]
5. The local system frequency (in ppm, positive means the local system runs fast of UTC). [-3.541]
6. The error bounds on the frequency (in ppm). [0.075]
7. The estimated local offset at the epoch, which is normally corrected by slewing the local clock (in seconds, positive indicates the clock is fast of UTC). [-8.621e-06]
8. Leap status (N means normal, + means that the last minute of this month has 61 seconds, - means that the last minute of the month has 59 seconds, ? means the clock is not currently synchronised.) [N]
9. The number of combined sources. [2]
10. The estimated standard deviation of the combined offset (in seconds). [2.940e-03]
11. The remaining offset correction from the previous update (in seconds, positive means the system clock is slow of UTC). [-2.084e-04]
12. The total of the network path delays to the reference clock to which the local clock is ultimately synchronised (in seconds). [1.534e-02]
13. The total dispersion accumulated through all the servers

back to the reference clock to which the local clock is ultimately synchronised (in seconds). [3.472e-04]

14. The maximum estimated error of the system clock in the interval since the previous update (in seconds). It includes the offset, remaining offset correction, root delay, and dispersion from the previous update with the dispersion which accumulated in the interval. [8.304e-03]

rtc

This option logs information about the system?s real-time clock. An example line (which actually appears as a single line in the file) from the rtc.log file is shown below.

```
2015-07-22 05:40:50     -0.037360 1       -0.037434\
          -37.948  12   5  120
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2015-07-22]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [05:40:50]
3. The measured offset between the RTC and the system clock in seconds. Positive indicates that the RTC is fast of the system time [-0.037360].
4. Flag indicating whether the regression has produced valid coefficients. (1 for yes, 0 for no). [1]
5. Offset at the current time predicted by the regression process. A large difference between this value and the measured offset tends to indicate that the measurement is an outlier with a serious measurement error. [-0.037434]
6. The rate at which the RTC is losing or gaining time relative to the system clock. In ppm, with positive indicating that the RTC is gaining time. [-37.948]
7. The number of measurements used in the regression. [12]
8. The number of runs of regression residuals of the same sign. Low values indicate that a straight line is no longer

a good model of the measured data and that older measurements should be discarded. [5]

9. The measurement interval used prior to the measurement being made (in seconds). [120]

refclocks

This option logs the raw and filtered reference clock measurements to a file called refclocks.log. An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2009-11-30 14:33:27.000000 PPS2    7 N 1  4.900000e-07 -6.741777e-07  1.000e-06
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2009-11-30]
2. Hour:Minute:Second.Microsecond. Note that the date-time pair is expressed in UTC, not the local time zone. [14:33:27.000000]
3. Reference ID of the reference clock from which the measurement came. [PPS2]
4. Sequence number of driver poll within one polling interval for raw samples, or - for filtered samples. [7]
5. Leap status (N means normal, + means that the last minute of the current month has 61 seconds, - means that the last minute of the month has 59 seconds). [N]
6. Flag indicating whether the sample comes from PPS source. (1 for yes, 0 for no, or - for filtered sample). [1]
7. Local clock error measured by reference clock driver, or - for filtered sample. [4.900000e-07]
8. Local clock error with applied corrections. Positive indicates that the local clock is slow. [-6.741777e-07]
9. Assumed dispersion of the sample. [1.000e-06]

tempcomp

This option logs the temperature measurements and system rate compensations to a file called tempcomp.log. An example line (which actually appears as a single line in the file) from the log file is shown below.

```
2015-04-19 10:39:48  2.8000e+04  3.6600e-01
```

The columns are as follows (the quantities in square brackets are the values from the example line above):

1. Date [2015-04-19]
2. Hour:Minute:Second. Note that the date-time pair is expressed in UTC, not the local time zone. [10:39:48]
3. Temperature read from the sensor. [2.8000e+04]
4. Applied compensation in ppm, positive means the system clock is running faster than it would be without the compensation. [3.6600e-01]

An example of the directive is:

```
log measurements statistics tracking
```

logbanner entries

A banner is periodically written to the log files enabled by the log directive to indicate the meanings of the columns.

The logbanner directive specifies after how many entries in the log file should be the banner written. The default is 32, and 0 can be used to disable it entirely.

logchange threshold

This directive sets the threshold for the adjustment of the system clock that will generate a syslog message. Clock errors detected via NTP packets, reference clocks, or timestamps entered via the settime command of chronyc are logged.

By default, the threshold is 1 second.

An example of the use is:

```
logchange 0.1
```

which would cause a syslog message to be generated if a system clock error of over 0.1 seconds starts to be compensated.

logdir directory

This directive specifies the directory for writing log files enabled by the log directive. If the directory does not exist, it will be created automatically.

An example of the use of this directive is:

```
logdir /var/log/chrony
```

mailonchange email threshold

This directive defines an email address to which mail should be sent if chronyd applies a correction exceeding a particular threshold to the system clock.

An example of the use of this directive is:

```
mailonchange root@localhost 0.5
```

This would send a mail message to root if a change of more than 0.5 seconds were applied to the system clock.

This directive cannot be used when a system call filter is enabled by the -F option as the chronyd process will not be allowed to fork and execute the sendmail binary.

Miscellaneous

confdir directory...

The confdir directive includes configuration files with the .conf suffix from a directory. The files are included in the lexicographical order of the file names.

Multiple directories (up to 10) can be specified with a single confdir directive. In this case, if multiple directories contain a file with the same name, only the first file in the order of the specified directories will be included. This enables a fragmented configuration where existing fragments can be replaced by adding files to a different directory.

This directive can be used multiple times.

An example of the directive is:

```
confdir /etc/chrony.d
```

sourcedir directory...

The sourcedir directive is identical to the confdir directive, except the configuration files have the .sources suffix, they can only specify NTP sources (i.e. the server, pool, and peer directives), they are expected to have all lines terminated by the newline character, and they can be reloaded by the reload sources command in chronyc. It is particularly useful with dynamic sources like NTP servers received from a DHCP server, which can be written to a file specific to the network interface by a networking script.

This directive can be used multiple times.

An example of the directive is:

```
sourcedir /var/run/chrony-dhcp
```

include pattern

The include directive includes a configuration file, or multiple configuration files if a wildcard pattern is specified. Unlike with the confdir directive, the full name of the files needs to be specified and at least one file is required to exist.

This directive can be used multiple times.

An example of the directive is:

```
include /etc/chrony.d/*.conf
```

hwtimestamp interface [option]...

This directive enables hardware timestamping of NTP packets sent to and received from the specified network interface. The network interface controller (NIC) uses its own clock to accurately timestamp the actual transmissions and receptions, avoiding processing and queueing delays in the kernel, network driver, and hardware. This can significantly improve the accuracy of the timestamps and the measured offset, which is used for synchronisation of the system clock. In order to get the best results, both sides receiving and sending NTP packets (i.e. server and client, or two peers) need to use HW timestamping. If the server or peer supports the interleaved mode, it needs to be enabled by the xleave option in the server or the peer directive.

This directive is supported on Linux 3.19 and newer. The NIC must support HW timestamping, which can be verified with the ethtool -T command. The list of capabilities should include hardware-raw-clock, hardware-transmit, and hardware-receive. The receive filter all, or ntp, is necessary for timestamping of received NTP packets. Timestamping of packets received on bridged and bonded interfaces is supported on Linux 4.13 and newer. If HW timestamping does not work for received packets, chronyd will use kernel receive timestamps instead. Transmit-only HW timestamping can still be useful to improve stability of the synchronisation.

chronyd does not synchronise the NIC clock. It assumes the clock is running free. Multiple instances of chronyd can use the same interface with enabled HW timestamping. Applications which need HW timestamping with a synchronised clock (e.g. a PTP daemon) should use a virtual clock running on top of the physical clock created by writing to /sys/class/ptp/ptpX/n_vclocks. This feature is available on Linux 5.14 and newer.

If the kernel supports software timestamping, it will be enabled for all interfaces. The source of timestamps (i.e. hardware, kernel, or daemon) is indicated in the measurements.log file if enabled by the log measurements directive, and the ntpdata report in chronyc.

This directive can be used multiple times to enable HW timestamping on multiple interfaces. If the specified interface is *, chronyd will try to enable HW timestamping on all available interfaces.

The hwtimestamp directive has the following options:

minpoll poll

This option specifies the minimum interval between readings of the NIC clock. It?s defined as a power of two. It should correspond to the minimum polling interval of all NTP sources and the minimum expected polling interval of NTP clients. The default value is 0 (1 second) and the minimum value is -6 (1/64th of a second).

minsamples samples

This option specifies the minimum number of readings kept for tracking of the NIC clock. The default value is 2.

maxsamples samples

This option specifies the maximum number of readings kept for tracking of the NIC clock. The default value is 16.

precision precision

This option specifies the assumed precision of reading of the NIC clock. The default value is 100e-9 (100 nanoseconds).

txcomp compensation

This option specifies the difference in seconds between the actual transmission time at the physical layer and the reported transmit timestamp. This value will be added to transmit timestamps obtained from the NIC. The default value is 0.

rxcomp compensation

This option specifies the difference in seconds between the reported receive timestamp and the actual reception time at the physical layer. This value will be subtracted from receive timestamps obtained from the NIC. The default value is 0.

nocrossts

Some hardware can precisely cross timestamp the NIC clock with the system clock. This option disables the use of the cross timestamping.

rxfilter filter

This option selects the receive timestamping filter. The filter can be one of the following:

all

Enables timestamping of all received packets.

ntp

Enables timestamping of received NTP packets.

ptp

Enables timestamping of received PTP packets.

none

Disables timestamping of received packets.

The most specific filter for timestamping of NTP packets supported by the NIC is selected by default. Some NICs can timestamp PTP packets only. By default, they will be configured with the none filter and expected to provide hardware timestamps for transmitted packets only. Timestamping of PTP packets is useful with NTP-over-PTP enabled by the ptpport directive, or when another application is receiving PTP packets on the interface. Forcing timestamping of all packets with the all filter could be useful if the NIC supported both the all and ntp filters, and it should timestamp both NTP and PTP packets, or NTP packets on a different UDP port.

Examples of the directive are:

```
hwtimestamp eth0
hwtimestamp eth1 txcomp 300e-9 rxcomp 645e-9
hwtimestamp *
```

keyfile file

This directive is used to specify the location of the file containing symmetric keys which are shared between NTP servers and clients, or peers, in order to authenticate NTP packets with a message authentication code (MAC) using a cryptographic hash function or cipher.

The format of the directive is shown in the example below:

```
keyfile /etc/chrony.keys
```

The argument is simply the name of the file containing the ID-key pairs. The format of the file is shown below:

```
10 tulip
11 hyacinth
20 MD5 ASCII:crocus
25 SHA1 HEX:933F62BE1D604E68A81B557F18CFA200483F5B70
30 AES128 HEX:7EA62AE64D190114D46D5A082F948EC1
31 AES256 HEX:37DDCBC67BB902BCB8E995977FAB4D2B5642F5B32EBCEEE421921D97E5CBFE39
 ...
```

Each line consists of an ID, optional type, and key.

The ID can be any positive integer in the range 1 through 2^32-1.

The type is a name of a cryptographic hash function or cipher which is used to generate and verify the MAC. The default type is MD5, which is always supported. If chronyd was built with enabled support for hashing using a crypto library (Nettle, GnuTLS, NSS, or LibTomCrypt), the following functions are available: MD5, SHA1, SHA256, SHA384, SHA512. Depending on which library and version is chronyd using, some of the following hash functions and ciphers may also be available: SHA3-224, SHA3-256, SHA3-384, SHA3-512, TIGER, WHIRLPOOL, AES128, AES256.

The key can be specified as a string of ASCII characters not containing white space with an optional ASCII: prefix, or as a hexadecimal number with the HEX: prefix. The maximum length of the line is 2047 characters. If the type is a cipher, the length of the key must match the cipher (i.e. 128 bits for AES128 and 256 bits for AES256).

It is recommended to use randomly generated keys, specified in the hexadecimal format, which are at least 128 bits long (i.e. they have at least 32 characters after the HEX: prefix). chronyd will log a warning to syslog on start if a source is specified in the configuration file with a key shorter than 80 bits.

The recommended key types are AES ciphers and SHA3 hash functions. MD5 should be avoided unless no other type is supported on the server and client, or peers.

The keygen command of chronyc can be used to generate random keys for the key file. By default, it generates 160-bit MD5 or SHA1 keys.

For security reasons, the file should be readable only by root and the user under which chronyd is normally running (to allow chronyd to re-read the file when the rekey command is issued by chronyc).

lock_all

The lock_all directive will lock the chronyd process into RAM so that it will never be paged out. This can result in lower and more consistent latency. The directive is supported on Linux, FreeBSD, NetBSD, and illumos.

pidfile file

Unless chronyd is started with the -Q option, it writes its process

ID (PID) to a file, and checks this file on startup to see if another chronyd might already be running on the system. By default, the file used is /run/chrony/chronyd.pid. The pidfile directive allows the name to be changed, e.g.:

```
pidfile /run/chronyd.pid
```

ptpport port

The ptpport directive enables chronyd to send and receive NTP messages contained in PTP event messages (NTP-over-PTP) to enable hardware timestamping on NICs which cannot timestamp NTP packets, but can timestamp unicast PTP packets. The port recognized by the NICs is 319 (PTP event port). The default value is 0 (disabled).

The NTP-over-PTP support is experimental. The protocol and configuration can change in future. It should be used only in local networks and expected to work only between servers and clients running the same version of chronyd.

The PTP port will be open even if chronyd is not configured to operate as a server or client. The directive does not change the default protocol of specified NTP sources. Each NTP source that should use NTP-over-PTP needs to be specified with the port option set to the PTP port. To actually enable hardware timestamping on NICs which can timestamp PTP packets only, the rxfilter option of the hwtimestamp directive needs to be set to ptp.

An example of client configuration is:

```
server foo.example.net minpoll 0 maxpoll 0 xleave port 319
hwtimestamp * rxfilter ptp
ptpport 319
```

sched_priority priority

On Linux, FreeBSD, NetBSD, and illumos, the sched_priority directive will select the SCHED_FIFO real-time scheduler at the specified priority (which must be between 0 and 100). On macOS, this option must have either a value of 0 (the default) to disable the thread time constraint policy or 1 for the policy to be enabled.

On systems other than macOS, this directive uses the pthread_setschedparam() system call to instruct the kernel to use the SCHED_FIFO first-in, first-out real-time scheduling policy for chronyd with the specified priority. This means that whenever chronyd is ready to run it will run, interrupting whatever else is running unless it is a higher priority real-time process. This should not impact performance as chronyd resource requirements are modest, but it should result in lower and more consistent latency since chronyd will not need to wait for the scheduler to get around to running it. You should not use this unless you really need it. The pthread_setschedparam(3) man page has more details.

On macOS, this directive uses the thread_policy_set() kernel call to specify real-time scheduling. As noted above, you should not use this directive unless you really need it.

user user

The user directive sets the name of the system user to which chronyd will switch after start in order to drop root privileges.

On Linux, chronyd needs to be compiled with support for the libcap library. On macOS, FreeBSD, NetBSD and illumos chronyd forks into two processes. The child process retains root privileges, but can only perform a very limited range of privileged system calls on behalf of the parent.

The compiled-in default value is chrony.

EXAMPLES

NTP client with permanent connection to NTP servers

This section shows how to configure chronyd for computers that are connected to the Internet (or to any network containing true NTP servers which ultimately derive their time from a reference clock) permanently or most of the time.

To operate in this mode, you will need to know the names of the NTP servers you want to use. You might be able to find names of suitable servers by one of the following methods:

? Your institution might already operate servers on its network.

Contact your system administrator to find out.

? Your ISP probably has one or more NTP servers available for its customers.

? Somewhere under the NTP homepage there is a list of public stratum 1 and stratum 2 servers. You should find one or more servers that are near to you. Check that their access policy allows you to use their facilities.

? Use public servers from the pool.ntp.org <https://www.pool.ntp.org/> project.

Assuming that your NTP servers are called foo.example.net, bar.example.net and baz.example.net, your chrony.conf file could contain as a minimum:

```
server foo.example.net
server bar.example.net
server baz.example.net
```

However, you will probably want to include some of the other directives. The driftfile, makestep and rtcsync might be particularly useful. Also, the iburst option of the server directive is useful to speed up the initial synchronisation. The smallest useful configuration file would look something like:

```
server foo.example.net iburst
server bar.example.net iburst
server baz.example.net iburst
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
```

When using a pool of NTP servers (one name is used for multiple servers which might change over time), it is better to specify them with the pool directive instead of multiple server directives. The configuration file could in this case look like:

```
pool pool.ntp.org iburst
driftfile /var/lib/chrony/drift
makestep 1.0 3
```

```
rtcsync
```

If the servers (or pool) support the Network Time Security (NTS) authentication mechanism and chronyd is compiled with NTS support, the nts option will enable a secure synchronisation to the servers. The configuration file could look like:

```
server foo.example.net iburst nts
server bar.example.net iburst nts
server baz.example.net iburst nts
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
```

NTP client with infrequent connection to NTP servers

This section shows how to configure chronyd for computers that have occasional connections to NTP servers. In this case, you will need some additional configuration to tell chronyd when the connection goes up and down. This saves the program from continuously trying to poll the servers when they are inaccessible.

Again, assuming that your NTP servers are called foo.example.net, bar.example.net and baz.example.net, your chrony.conf file would now contain:

```
server foo.example.net offline
server bar.example.net offline
server baz.example.net offline
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
```

The offline keyword indicates that the servers start in an offline state, and that they should not be contacted until chronyd receives notification from chronyc that the link to the Internet is present. To tell chronyd when to start and finish sampling the servers, the online and offline commands of chronyc need to be used.

To give an example of their use, assuming that pppd is the program being used to connect to the Internet and that chronyc has been installed at /usr/bin/chronyc, the script /etc/ppp/ip-up would include:

```
/usr/bin/chronyc online
```

and the script /etc/ppp/ip-down would include:

```
/usr/bin/chronyc offline
```

chronyd's polling of the servers would now only occur whilst the machine is actually connected to the Internet.

Isolated networks

This section shows how to configure chronyd for computers that never have network connectivity to any computer which ultimately derives its time from a reference clock.

In this situation, one computer is selected to be the primary timeserver. The other computers are either direct clients of the server, or clients of clients.

The local directive enables a local reference mode, which allows chronyd to appear synchronised even when it is not.

The rate value in the server?s drift file needs to be set to the average rate at which the server gains or loses time. chronyd includes support for this, in the form of the manual directive and the settime command in the chronyc program.

If the server is rebooted, chronyd can re-read the drift rate from the drift file. However, the server has no accurate estimate of the current time. To get around this, the system can be configured so that the server can initially set itself to a ?majority-vote? of selected clients' times; this allows the clients to ?flywheel? the server while it is rebooting.

The smoothtime directive is useful when the clocks of the clients need to stay close together when the local time is adjusted by the settime command. The smoothing process needs to be activated by the smoothtime activate command when the local time is ready to be served. After that point, any adjustments will be smoothed out.

A typical configuration file for the server (called ntp.local) might be (assuming the clients and the server are in the 192.168.165.x subnet):

```
initstepslew 1 client1 client3 client6
```

```
    driftfile /var/lib/chrony/drift
    local stratum 8
    manual
    allow 192.168.165.0/24
    smoothtime 400 0.01
    rtcsync
```

For the clients that have to resynchronise the server when it restarts, the configuration file might be:

```
    server ntp.local iburst
    driftfile /var/lib/chrony/drift
    allow 192.168.165.0/24
    makestep 1.0 3
    rtcsync
```

The rest of the clients would be the same, except that the allow directive is not required.

If there is no suitable computer to be designated as the primary server, or there is a requirement to keep the clients synchronised even when it fails, the orphan option of the local directive enables a special mode where the server is selected from multiple computers automatically. They all need to use the same local configuration and poll one another. The server with the smallest reference ID (which is based on its IP address) will take the role of the primary server and others will be synchronised to it. When it fails, the server with the second smallest reference ID will take over and so on.

A configuration file for the first server might be (assuming there are three servers called ntp1.local, ntp2.local, and ntp3.local):

```
    initstepslew 1 ntp2.local ntp3.local
    server ntp2.local
    server ntp3.local
    driftfile /var/lib/chrony/drift
    local stratum 8 orphan
    manual
    allow 192.168.165.0/24
    rtcsync
```

The other servers would be the same, except the hostnames in the initstepslew and server directives would be modified to specify the other servers. Their clients might be configured to poll all three servers.

RTC tracking

This section considers a computer which has occasional connections to the Internet and is turned off between ?sessions?. In this case, chronyd relies on the computer?s RTC to maintain the time between the periods when it is powered up. It assumes that Linux is run exclusively on the computer. Dual-boot systems might work; it depends what (if anything) the other system does to the RTC. On 2.6 and later kernels, if your motherboard has a HPET, you will need to enable the HPET_EMULATE_RTC option in your kernel configuration. Otherwise, chronyd will not be able to interact with the RTC device and will give up using it.

When the computer is connected to the Internet, chronyd has access to external NTP servers which it makes measurements from. These measurements are saved, and straight-line fits are performed on them to provide an estimate of the computer?s time error and rate of gaining or losing time.

When the computer is taken offline from the Internet, the best estimate of the gain or loss rate is used to free-run the computer until it next goes online.

Whilst the computer is running, chronyd makes measurements of the RTC (via the /dev/rtc interface, which must be compiled into the kernel). An estimate is made of the RTC error at a particular RTC second, and the rate at which the RTC gains or loses time relative to true time.

When the computer is powered down, the measurement histories for all the NTP servers are saved to files, and the RTC tracking information is also saved to a file (if the rtcfile directive has been specified). These pieces of information are also saved if the dump and writertc commands respectively are issued through chronyc.

When the computer is rebooted, chronyd reads the current RTC time and
the RTC information saved at the last shutdown. This information is
used to set the system clock to the best estimate of what its time
would have been now, had it been left running continuously. The
measurement histories for the servers are then reloaded.

The next time the computer goes online, the previous sessions'
measurements can contribute to the line-fitting process, which gives a
much better estimate of the computer?s gain or loss rate.

One problem with saving the measurements and RTC data when the machine
is shut down is what happens if there is a power failure; the most
recent data will not be saved. Although chronyd is robust enough to
cope with this, some performance might be lost. (The main danger arises
if the RTC has been changed during the session, with the trimrtc
command in chronyc. Because of this, trimrtc will make sure that a
meaningful RTC file is saved after the change is completed).

The easiest protection against power failure is to put the dump and
writertc commands in the same place as the offline command is issued to
take chronyd offline; because chronyd free-runs between online
sessions, no parameters will change significantly between going offline
from the Internet and any power failure.

A final point regards computers which are left running for extended
periods and where it is desired to spin down the hard disc when it is
not in use (e.g. when not accessed for 15 minutes). chronyd has been
planned so it supports such operation; this is the reason why the RTC
tracking parameters are not saved to disc after every update, but only
when the user requests such a write, or during the shutdown sequence.

The only other facility that will generate periodic writes to the disc
is the log rtc facility in the configuration file; this option should
not be used if you want your disc to spin down.

To illustrate how a computer might be configured for this case, example
configuration files are shown.

For the chrony.conf file, the following can be used as an example.

```
    server foo.example.net maxdelay 0.4 offline
```

```
server bar.example.net maxdelay 0.4 offline
server baz.example.net maxdelay 0.4 offline
logdir /var/log/chrony
log statistics measurements tracking
driftfile /var/lib/chrony/drift
makestep 1.0 3
maxupdateskew 100.0
dumpdir /var/lib/chrony
rtcfile /var/lib/chrony/rtc
```

pppd is used for connecting to the Internet. This runs two scripts /etc/ppp/ip-up and /etc/ppp/ip-down when the link goes online and offline respectively.

The relevant part of the /etc/ppp/ip-up file is:

```
/usr/bin/chronyc online
```

and the relevant part of the /etc/ppp/ip-down script is:

```
/usr/bin/chronyc -m offline dump writertc
```

chronyd is started during the boot sequence with the -r and -s options. It might need to be started before any software that depends on the system clock not jumping or moving backwards, depending on the directives in chronyd's configuration file.

For the system shutdown, chronyd should receive a SIGTERM several seconds before the final SIGKILL; the SIGTERM causes the measurement histories and RTC information to be saved.

Public NTP server

chronyd can be configured to operate as a public NTP server, e.g. to join the pool.ntp.org <https://www.pool.ntp.org/en/join.html> project. The configuration is similar to the NTP client with permanent connection, except it needs to allow client access from all addresses. It is recommended to find at least four good servers (e.g. from the pool, or on the NTP homepage). If the server has a hardware reference clock (e.g. a GPS receiver), it can be specified by the refclock directive.

The amount of memory used for logging client accesses can be increased in order to enable clients to use the interleaved mode even when the server has a large number of clients, and better support rate limiting if it is enabled by the ratelimit directive. The system timezone database, if it is kept up to date and includes the right/UTC timezone, can be used as a reliable source to determine when a leap second will be applied to UTC. The -r option with the dumpdir directive shortens the time in which chronyd will not be able to serve time to its clients when it needs to be restarted (e.g. after upgrading to a newer version, or a change in the configuration).

The configuration file could look like:

```
server foo.example.net iburst
server bar.example.net iburst
server baz.example.net iburst
server qux.example.net iburst
makestep 1.0 3
rtcsync
allow
clientloglimit 100000000
leapsectz right/UTC
driftfile /var/lib/chrony/drift
dumpdir /run/chrony
```

SEE ALSO

chronyc(1), chronyd(8)

BUGS

For instructions on how to report bugs, please visit https://chrony.tuxfamily.org/.

AUTHORS

chrony was written by Richard Curnow, Miroslav Lichvar, and others.

chrony 4.3 2022-08-29 CHRONY.CONF(5)