Red Hat Enterprise Linux Release 9.2 Manual Pages on 'systemd.timer.5' command

$ man systemd.timer.5

SYSTEMD.TIMER(5) systemd.timer SYSTEMD.TIMER(5)

NAME

systemd.timer - Timer unit configuration

SYNOPSIS

timer.timer

DESCRIPTION

A unit configuration file whose name ends in ".timer" encodes information about a timer controlled and supervised by systemd, for timer-based activation.

This man page lists the configuration options specific to this unit type. See systemd.unit(5) for the common options of all unit configuration files. The common configuration items are configured in the generic [Unit] and [Install] sections. The timer specific configuration options are configured in the [Timer] section.

For each timer file, a matching unit file must exist, describing the unit to activate when the timer elapses. By default, a service by the same name as the timer (except for the suffix) is activated. Example: a timer file foo.timer activates a matching service foo.service. The unit to activate may be controlled by Unit= (see below).

Note that in case the unit to activate is already active at the time the timer elapses it is not restarted, but simply left running. There is no concept of spawning new service instances in this case. Due to this, services with RemainAfterExit= set (which stay around continuously even after the service's main process exited) are usually not suitable for activation via repetitive timers, as they will only be activated once, and then stay around forever.

AUTOMATIC DEPENDENCIES

Implicit Dependencies

The following dependencies are implicitly added:

? Timer units automatically gain a Before= dependency on the service they are supposed to activate.

Default Dependencies

The following dependencies are added unless DefaultDependencies=no is set:

? Timer units will automatically have dependencies of type Requires= and After= on sysinit.target, a dependency of type Before= on timers.target, as well as Conflicts= and Before= on shutdown.target to ensure that they are stopped cleanly prior to system shutdown. Only timer units involved with early boot or late system shutdown should disable the DefaultDependencies= option.

? Timer units with at least one OnCalendar= directive acquire a pair of additional After= dependencies on time-set.target and time-sync.target, in order to avoid being started before the system clock has been correctly set. See systemd.special(7) for details on these two targets.

OPTIONS

Timer unit files may include [Unit] and [Install] sections, which are described in systemd.unit(5).

Timer unit files must include a [Timer] section, which carries information about the timer it defines. The options specific to the [Timer] section of timer units are the following:

OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec=, OnUnitInactiveSec=

Defines monotonic timers relative to different starting points:

Table 1. Settings and their starting points

???

```
?Setting          ? Meaning                    ?
??????????????????????????????????????????????????
?OnActiveSec=     ? Defines a timer relative   ?
?                 ? to the moment the timer    ?
?                 ? unit itself is activated.  ?
??????????????????????????????????????????????????
?OnBootSec=       ? Defines a timer relative   ?
?                 ? to when the machine was    ?
?                 ? booted up. In containers,  ?
?                 ? for the system manager     ?
?                 ? instance, this is mapped   ?
?                 ? to OnStartupSec=, making   ?
?                 ? both equivalent.           ?
??????????????????????????????????????????????????
?OnStartupSec=    ? Defines a timer relative   ?
?                 ? to when the service        ?
?                 ? manager was first started. ?
?                 ? For system timer units     ?
?                 ? this is very similar to    ?
?                 ? OnBootSec= as the system   ?
?                 ? service manager is         ?
?                 ? generally started very     ?
?                 ? early at boot. It's        ?
?                 ? primarily useful when      ?
?                 ? configured in units        ?
?                 ? running in the per-user    ?
?                 ? service manager, as the    ?
?                 ? user service manager is    ?
?                 ? generally started on first ?
?                 ? login only, not already    ?
?                 ? during boot.               ?
??????????????????????????????????????????????????
?OnUnitActiveSec= ? Defines a timer relative   ?
```

```
?                  ? to when the unit the timer ?
?                  ? unit is activating was     ?
?                  ? last activated.            ?
??????????????????????????????????????????????????
?OnUnitInactiveSec= ? Defines a timer relative   ?
?                  ? to when the unit the timer ?
?                  ? unit is activating was     ?
?                  ? last deactivated.          ?
??????????????????????????????????????????????????
```

Multiple directives may be combined of the same and of different types, in which case the timer unit will trigger whenever any of the specified timer expressions elapse. For example, by combining OnBootSec= and OnUnitActiveSec=, it is possible to define a timer that elapses in regular intervals and activates a specific service each time. Moreover, both monotonic time expressions and OnCalendar= calendar expressions may be combined in the same timer unit.

The arguments to the directives are time spans configured in seconds. Example: "OnBootSec=50" means 50s after boot-up. The argument may also include time units. Example: "OnBootSec=5h 30min" means 5 hours and 30 minutes after boot-up. For details about the syntax of time spans, see systemd.time(7).

If a timer configured with OnBootSec= or OnStartupSec= is already in the past when the timer unit is activated, it will immediately elapse and the configured unit is started. This is not the case for timers defined in the other directives.

These are monotonic timers, independent of wall-clock time and timezones. If the computer is temporarily suspended, the monotonic clock generally pauses, too. Note that if WakeSystem= is used, a different monotonic clock is selected that continues to advance while the system is suspended and thus can be used as the trigger to resume the system.

If the empty string is assigned to any of these options, the list of timers is reset (both monotonic timers and OnCalendar= timers, see below), and all prior assignments will have no effect.

Note that timers do not necessarily expire at the precise time configured with these settings, as they are subject to the AccuracySec= setting below.

OnCalendar=

Defines realtime (i.e. wallclock) timers with calendar event expressions. See systemd.time(7) for more information on the syntax of calendar event expressions. Otherwise, the semantics are similar to OnActiveSec= and related settings.

Note that timers do not necessarily expire at the precise time configured with this setting, as it is subject to the AccuracySec= setting below.

May be specified more than once, in which case the timer unit will trigger whenever any of the specified expressions elapse. Moreover calendar timers and monotonic timers (see above) may be combined within the same timer unit.

If the empty string is assigned to any of these options, the list of timers is reset (both OnCalendar= timers and monotonic timers, see above), and all prior assignments will have no effect.

AccuracySec=

Specify the accuracy the timer shall elapse with. Defaults to 1min. The timer is scheduled to elapse within a time window starting with the time specified in OnCalendar=, OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec= or OnUnitInactiveSec= and ending the time configured with AccuracySec= later. Within this time window, the expiry time will be placed at a host-specific, randomized, but stable position that is synchronized between all local timer units. This is done in order to optimize power consumption to suppress unnecessary CPU wake-ups. To get best accuracy, set this option to 1us. Note that the timer is still subject to the timer slack configured via systemd-system.conf(5)'s TimerSlackNSec= setting. See prctl(2) for details. To optimize power consumption, make sure to set this value as high as possible and as low as necessary.

Note that this setting is primarily a power saving option that allows coalescing CPU wake-ups. It should not be confused with RandomizedDelaySec= (see below) which adds a random value to the time the timer shall elapse next and whose purpose is the opposite: to stretch elapsing of timer events over a longer period to reduce workload spikes. For further details and explanations and how both settings play together, see below.

RandomizedDelaySec=

Delay the timer by a randomly selected, evenly distributed amount of time between 0 and the specified time value. Defaults to 0, indicating that no randomized delay shall be applied. Each timer unit will determine this delay randomly before each iteration, and the delay will simply be added on top of the next determined elapsing time, unless modified with FixedRandomDelay=, see below.

This setting is useful to stretch dispatching of similarly configured timer events over a certain time interval, to prevent them from firing all at the same time, possibly resulting in resource congestion.

Note the relation to AccuracySec= above: the latter allows the service manager to coalesce timer events within a specified time range in order to minimize wakeups, while this setting does the opposite: it stretches timer events over an interval, to make it unlikely that they fire simultaneously. If RandomizedDelaySec= and AccuracySec= are used in conjunction, first the randomized delay is added, and then the result is possibly further shifted to coalesce it with other timer events happening on the system. As mentioned above AccuracySec= defaults to 1 minute and RandomizedDelaySec= to 0, thus encouraging coalescing of timer events. In order to optimally stretch timer events over a certain range of time, set AccuracySec=1us and RandomizedDelaySec= to some higher value.

FixedRandomDelay=

Takes a boolean argument. When enabled, the randomized offset specified by RandomizedDelaySec= is reused for all firings of the same timer. For a given timer unit, the offset depends on the machine ID, user identifier and timer name, which means that it is stable between restarts of the manager. This effectively creates a fixed offset for an individual timer, reducing the jitter in firings of this timer, while still avoiding firing at the same time as other similarly configured timers.

This setting has no effect if RandomizedDelaySec= is set to 0. Defaults to false.

OnClockChange=, OnTimezoneChange=

These options take boolean arguments. When true, the service unit will be triggered when the system clock (CLOCK_REALTIME) jumps relative to the monotonic clock (CLOCK_MONOTONIC), or when the local system timezone is modified. These options can be used alone or in combination with other timer expressions (see above) within the same timer unit. These options default to false.

Unit=

The unit to activate when this timer elapses. The argument is a unit name, whose suffix is not ".timer". If not specified, this value defaults to a service that has the same name as the timer unit, except for the suffix. (See above.) It is recommended that the unit name that is activated and the unit name of the timer unit are named identically, except for the suffix.

Persistent=

Takes a boolean argument. If true, the time when the service unit was last triggered is stored on disk. When the timer is activated, the service unit is triggered immediately if it would have been triggered at least once during the time when the timer was inactive. Such triggering is nonetheless subject to the delay imposed by RandomizedDelaySec=. This is useful to catch up on missed runs of the service when the system was powered down. Note that this setting only has an effect on timers configured with

OnCalendar=. Defaults to false.

Use systemctl clean --what=state ... on the timer unit to remove the timestamp file maintained by this option from disk. In particular, use this command before uninstalling a timer unit. See systemctl(1) for details.

WakeSystem=

Takes a boolean argument. If true, an elapsing timer will cause the system to resume from suspend, should it be suspended and if the system supports this. Note that this option will only make sure the system resumes on the appropriate times, it will not take care of suspending it again after any work that is to be done is finished. Defaults to false.

Note that this functionality requires privileges and is thus generally only available in the system service manager.

Note that behaviour of monotonic clock timers (as configured with OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec=, OnUnitInactiveSec=, see above) is altered depending on this option. If false, a monotonic clock is used that is paused during system suspend (CLOCK_MONOTONIC), if true a different monotonic clock is used that continues advancing during system suspend (CLOCK_BOOTTIME), see clock_getres(2) for details.

RemainAfterElapse=

Takes a boolean argument. If true, a timer will stay loaded, and its state remains queryable even after it elapsed and the associated unit (as configured with Unit=, see above) deactivated again. If false, an elapsed timer unit that cannot elapse anymore is unloaded once its associated unit deactivated again. Turning this off is particularly useful for transient timer units. Note that this setting has an effect when repeatedly starting a timer unit: if RemainAfterElapse= is on, starting the timer a second time has no effect. However, if RemainAfterElapse= is off and the timer unit was already unloaded, it can be started again, and thus the service can be triggered multiple times. Defaults to true.

Check systemd.unit(5), systemd.exec(5), and systemd.kill(5) for more settings.

SEE ALSO

Environment variables with details on the trigger will be set for triggered units. See the "Environment Variables Set on Triggered Units" section in systemd.exec(1) for more details.

systemd(1), systemctl(1), systemd.unit(5), systemd.service(5), systemd.time(7), systemd.directives(7), systemd-system.conf(5), prctl(2)