NAME

systemd-system.conf, system.conf.d, systemd-user.conf, user.conf.d − System and session service manager configuration files

SYNOPSIS

/etc/systemd/system.conf, /etc/systemd/system.conf.d/*.conf, /run/systemd/system.conf.d/*.conf, /lib/systemd/system.conf.d/*.conf

/etc/systemd/user.conf, /etc/systemd/user.conf.d/*.conf, /run/systemd/user.conf.d/*.conf, /usr/lib/systemd/user.conf.d/*.conf

DESCRIPTION

When run as a system instance, systemd interprets the configuration file system.conf and the files in system.conf.d directories; when run as a user instance, systemd interprets the configuration file user.conf and the files in user.conf.d directories. These configuration files contain a few settings controlling basic manager operations. See **systemd.syntax**(5) for a general description of the syntax.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is defined during compilation, so a configuration file is only needed when it is necessary to deviate from those defaults. By default, the configuration file in /etc/systemd/ contains commented out entries showing the defaults as a guide to the administrator. This file can be edited to create local overrides.

When packages need to customize the configuration, they can install configuration snippets in /usr/lib/systemd/*.conf.d/ or /usr/local/lib/systemd/*.conf.d/. Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. The main configuration file is read before any of the configuration directories, and has the lowest precedence; entries in a file in any configuration directory override entries in the single configuration file. Files in the *.conf.d/ configuration subdirectories are sorted by their filename in lexicographic order, regardless of which of the subdirectories they reside in. When multiple files specify the same option, for options which accept just a single value, the entry in the file with the lexicographically latest name takes precedence. For options which accept a list of values, entries are collected as they occur in files sorted lexicographically. It is recommended to prefix all filenames in those subdirectories with a two−digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the "[Manager]" section:

LogLevel=, *LogTarget=*, *LogColor=*, *LogLocation=*, *DumpCore=yes*, *CrashChangeVT=no*, *CrashShell=no*, *CrashReboot=no*, *ShowStatus=yes*, *DefaultStandardOutput=journal*, *DefaultStandardError=inherit*

Configures various parameters of basic manager operation. These options may be overridden by the respective process and kernel command line arguments. See **systemd**(1) for details.

CtrlAltDelBurstAction=

Defines what action will be performed if user presses Ctrl−Alt−Delete more than 7 times in 2s. Can be set to "reboot−force", "poweroff−force", "reboot−immediate", "poweroff−immediate" or disabled with "none". Defaults to "reboot−force".

CPUAffinity=

Configures the CPU affinity for the service manager as well as the default CPU affinity for all forked off processes. Takes a list of CPU indices or ranges separated by either whitespace or commas. CPU ranges are specified by the lower and upper CPU indices separated by a dash. Individual services may override the CPU affinity for their processes with the *CPUAffinity=* setting in unit files, see **systemd.exec**(5).

RuntimeWatchdogSec=, *ShutdownWatchdogSec=*

Configure the hardware watchdog at runtime and at reboot. Takes a timeout value in seconds (or in

other time units if suffixed with "ms", "min", "h", "d", "w"). If *RuntimeWatchdogSec*= is set to a non−zero value, the watchdog hardware (/dev/watchdog or the path specified with *WatchdogDevice*= or the kernel option *systemd.watchdog−device*=) will be programmed to automatically reboot the system if it is not contacted within the specified timeout interval. The system manager will ensure to contact it at least once in half the specified timeout interval. This feature requires a hardware watchdog device to be present, as it is commonly the case in embedded and server systems. Not all hardware watchdogs allow configuration of all possible reboot timeout values, in which case the closest available timeout is picked. *ShutdownWatchdogSec*= may be used to configure the hardware watchdog when the system is asked to reboot. It works as a safety net to ensure that the reboot takes place even if a clean reboot attempt times out. Note that the *ShutdownWatchdogSec*= timeout applies only to the second phase of the reboot, i.e. after all regular services are already terminated, and after the system and service manager process (PID 1) got replaced by the systemd−shutdown binary, see system **bootup**(7) for details. During the first phase of the shutdown operation the system and service manager remains running and hence *RuntimeWatchdogSec*= is still honoured. In order to define a timeout on this first phase of system shutdown, configure *JobTimeoutSec*= and *JobTimeoutAction*= in the "[Unit]" section of the shutdown.target unit. By default *RuntimeWatchdogSec*= defaults to 0 (off), and *ShutdownWatchdogSec*= to 10min. These settings have no effect if a hardware watchdog is not available.

WatchdogDevice=
Configure the hardware watchdog device that the runtime and shutdown watchdog timers will open and use. Defaults to /dev/watchdog. This setting has no effect if a hardware watchdog is not available.

CapabilityBoundingSet=
Controls which capabilities to include in the capability bounding set for PID 1 and its children. See **capabilities**(7) for details. Takes a whitespace−separated list of capability names as read by **cap_from_name**(3). Capabilities listed will be included in the bounding set, all others are removed. If the list of capabilities is prefixed with ˜, all but the listed capabilities will be included, the effect of the assignment inverted. Note that this option also affects the respective capabilities in the effective, permitted and inheritable capability sets. The capability bounding set may also be individually configured for units using the *CapabilityBoundingSet*= directive for units, but note that capabilities dropped for PID 1 cannot be regained in individual units, they are lost for good.

NoNewPrivileges=
Takes a boolean argument. If true, ensures that PID 1 and all its children can never gain new privileges through **execve**(2) (e.g. via setuid or setgid bits, or filesystem capabilities). Defaults to false. General purpose distributions commonly rely on executables with setuid or setgid bits and will thus not function properly with this option enabled. Individual units cannot disable this option. Also see **No New Privileges Flag**[1].

SystemCallArchitectures=
Takes a space−separated list of architecture identifiers. Selects from which architectures system calls may be invoked on this system. This may be used as an effective way to disable invocation of non−native binaries system−wide, for example to prohibit execution of 32−bit x86 binaries on 64−bit x86−64 systems. This option operates system−wide, and acts similar to the *SystemCallArchitectures*= setting of unit files, see **systemd.exec**(5) for details. This setting defaults to the empty list, in which case no filtering of system calls based on architecture is applied. Known architecture identifiers are "x86", "x86−64", "x32", "arm" and the special identifier "native". The latter implicitly maps to the native architecture of the system (or more specifically, the architecture the system manager was compiled for). Set this setting to "native" to prohibit execution of any non−native binaries. When a binary executes a system call of an architecture that is not listed in this setting, it will be immediately terminated with the SIGSYS signal.

TimerSlackNSec=
Sets the timer slack in nanoseconds for PID 1, which is inherited by all executed processes, unless overridden individually, for example with the *TimerSlackNSec*= setting in service units (for details see **systemd.exec**(5)). The timer slack controls the accuracy of wake−ups triggered by system timers. See

prctl(2) for more information. Note that in contrast to most other time span definitions this parameter takes an integer value in nano−seconds if no unit is specified. The usual time units are understood too.

DefaultTimerAccuracySec=
Sets the default accuracy of timer units. This controls the global default for the *AccuracySec=* setting of timer units, see **systemd.timer**(5) for details. *AccuracySec=* set in individual units override the global default for the specific unit. Defaults to 1min. Note that the accuracy of timer units is also affected by the configured timer slack for PID 1, see *TimerSlackNSec=* above.

DefaultTimeoutStartSec=, *DefaultTimeoutStopSec=*, *DefaultRestartSec=*
Configures the default timeouts for starting and stopping of units, as well as the default time to sleep between automatic restarts of units, as configured per−unit in *TimeoutStartSec=*, *TimeoutStopSec=* and *RestartSec=* (for services, see **systemd.service**(5) for details on the per−unit settings). Disabled by default, when service with *Type=oneshot* is used. For non−service units, *DefaultTimeoutStartSec=* sets the default *TimeoutSec=* value. *DefaultTimeoutStartSec=* and *DefaultTimeoutStopSec=* default to 90s. *DefaultRestartSec=* defaults to 100ms.

DefaultStartLimitIntervalSec=, *DefaultStartLimitBurst=*
Configure the default unit start rate limiting, as configured per−service by *StartLimitIntervalSec=* and *StartLimitBurst=*. See **systemd.service**(5) for details on the per−service settings. *DefaultStartLimitIntervalSec=* defaults to 10s. *DefaultStartLimitBurst=* defaults to 5.

DefaultEnvironment=
Sets manager environment variables passed to all executed processes. Takes a space−separated list of variable assignments. See **environ**(7) for details about environment variables.

Example:

DefaultEnvironment="VAR1=word1 word2" VAR2=word3 "VAR3=word 5 6"

Sets three variables "VAR1", "VAR2", "VAR3".

DefaultCPUAccounting=, *DefaultBlockIOAccounting=*, *DefaultMemoryAccounting=*, *DefaultTasksAccounting=*, *DefaultIOAccounting=*, *DefaultIPAccounting=*
Configure the default resource accounting settings, as configured per−unit by *CPUAccounting=*, *BlockIOAccounting=*, *MemoryAccounting=*, *TasksAccounting=*, *IOAccounting=* and *IPAccounting=*. See **systemd.resource-control**(5) for details on the per−unit settings. *DefaultTasksAccounting=* defaults to yes, *DefaultMemoryAccounting=* to yes. *DefaultCPUAccounting=* defaults to yes if enabling CPU accounting doesn't require the CPU controller to be enabled (Linux 4.15+ using the unified hierarchy for resource control), otherwise it defaults to no. The other three settings default to no.

DefaultTasksMax=
Configure the default value for the per−unit *TasksMax=* setting. See **systemd.resource-control**(5) for details. This setting applies to all unit types that support resource control settings, with the exception of slice units.

DefaultLimitCPU=, *DefaultLimitFSIZE=*, *DefaultLimitDATA=*, *DefaultLimitSTACK=*, *DefaultLimitCORE=*, *DefaultLimitRSS=*, *DefaultLimitNOFILE=*, *DefaultLimitAS=*, *DefaultLimitNPROC=*, *DefaultLimitMEMLOCK=*, *DefaultLimitLOCKS=*, *DefaultLimitSIGPENDING=*, *DefaultLimitMSGQUEUE=*, *DefaultLimitNICE=*, *DefaultLimitRTPRIO=*, *DefaultLimitRTTIME=*
These settings control various default resource limits for units. See **setrlimit**(2) for details. The resource limit is possible to specify in two formats, **value** to set soft and hard limits to the same value, or **soft:hard** to set both limits individually (e.g. DefaultLimitAS=4G:16G). Use the string *infinity* to configure no limit on a specific resource. The multiplicative suffixes K (=1024), M (=1024*1024) and so on for G, T, P and E may be used for resource limits measured in bytes (e.g. DefaultLimitAS=16G). For the limits referring to time values, the usual time units ms, s, min, h and so on may be used (see **systemd.time**(7) for details). Note that if no time unit is specified for *DefaultLimitCPU=* the default unit of seconds is implied, while for *DefaultLimitRTTIME=* the default unit of microseconds is

implied. Also, note that the effective granularity of the limits might influence their enforcement. For example, time limits specified for *DefaultLimitCPU*= will be rounded up implicitly to multiples of 1s. These settings may be overridden in individual units using the corresponding LimitXXX= directives. Note that these resource limits are only defaults for units, they are not applied to PID 1 itself.

SEE ALSO

systemd(1), **systemd.directives**(7), **systemd.exec**(5), **systemd.service**(5), **environ**(7), **capabilities**(7)

NOTES

1. No New Privileges Flag
   https://www.kernel.org/doc/html/latest/userspace-api/no_new_privs.html