NAME

journald.conf, journald.conf.d, journald@.conf – Journal service configuration files

SYNOPSIS

/etc/systemd/journald.conf

/etc/systemd/journald.conf.d/*.conf

/run/systemd/journald.conf.d/*.conf

/usr/lib/systemd/journald.conf.d/*.conf

/etc/systemd/journald@*NAMESPACE*.conf

DESCRIPTION

These files configure various parameters of the systemd journal service, **systemd-journald.service**(8). See **systemd.syntax**(7) for a general description of the syntax.

The **systemd−journald** instance managing the default namespace is configured by /etc/systemd/journald.conf and associated drop−ins. Instances managing other namespaces read /etc/systemd/journald@*NAMESPACE*.conf with the namespace identifier filled in. This allows each namespace to carry a distinct configuration. See **systemd-journald.service**(8) for details about journal namespaces.

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/. 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 in which of the subdirectories they reside. 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.

Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. 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 "[Journal]" section:

Storage=

Controls where to store journal data. One of "volatile", "persistent", "auto" and "none". If "volatile", journal log data will be stored only in memory, i.e. below the /run/log/journal hierarchy (which is created if needed). If "persistent", data will be stored preferably on disk, i.e. below the /var/log/journal hierarchy (which is created if needed), with a fallback to /run/log/journal (which is created if needed), during early boot and if the disk is not writable. "auto" is similar to "persistent" but the directory /var/log/journal is not created if needed, so that its existence controls where log data goes. "none" turns off all storage, all log data received will be dropped. Forwarding to other targets, such as the console, the kernel log buffer, or a syslog socket will still work however. Defaults to "auto" in the default journal namespace, and "persistent" in all others.

Compress=

Can take a boolean value. If enabled (the default), data objects that shall be stored in the journal and are larger than the default threshold of 512 bytes are compressed before they are written to the file system. It can also be set to a number of bytes to specify the compression threshold directly. Suffixes like K, M, and G can be used to specify larger units.

Seal=

Takes a boolean value. If enabled (the default), and a sealing key is available (as created by **journalctl**(1)'s **−−setup−keys** command), Forward Secure Sealing (FSS) for all persistent journal files is enabled. FSS is based on **Seekable Sequential Key Generators**[1] by G. A. Marson and B. Poettering (doi:10.1007/978−3−642−40203−6_7) and may be used to protect journal files from unnoticed alteration.

SplitMode=

Controls whether to split up journal files per user, either "uid" or "none". Split journal files are primarily useful for access control: on UNIX/Linux access control is managed per file, and the journal daemon will assign users read access to their journal files. If "uid", all regular users (with UID outside the range of system users, dynamic service users, and the nobody user) will each get their own journal files, and system users will log to the system journal. See **Users, Groups, UIDs and GIDs on systemd systems**[2] for more details about UID ranges. If "none", journal files are not split up by user and all messages are instead stored in the single system journal. In this mode unprivileged users generally do not have access to their own log data. Note that splitting up journal files by user is only available for journals stored persistently. If journals are stored on volatile storage (see *Storage=* above), only a single journal file is used. Defaults to "uid".

RateLimitIntervalSec=, *RateLimitBurst=*

Configures the rate limiting that is applied to all messages generated on the system. If, in the time interval defined by *RateLimitIntervalSec=*, more messages than specified in *RateLimitBurst=* are logged by a service, all further messages within the interval are dropped until the interval is over. A message about the number of dropped messages is generated. This rate limiting is applied per−service, so that two services which log do not interfere with each other's limits. Defaults to 10000 messages in 30s. The time specification for *RateLimitIntervalSec=* may be specified in the following units: "s", "min", "h", "ms", "us". To turn off any kind of rate limiting, set either value to 0.

Note that the effective rate limit is multiplied with a factor derived from the available free disk space for the journal. Currently, this factor is calculated using the base 2 logarithm.

Table 1. Example *RateLimitBurst=* rate modifications by the available disk space

Available Disk Space	Burst Multiplier
<= 1MB	1
<= 16MB	2
<= 256MB	3
<= 4GB	4
<= 64GB	5
<= 1TB	6

If a service provides rate limits for itself through *LogRateLimitIntervalSec=* and/or *LogRateLimitBurst=* in **systemd.exec**(5), those values will override the settings specified here.

SystemMaxUse=, *SystemKeepFree=*, *SystemMaxFileSize=*, *SystemMaxFiles=*, *RuntimeMaxUse=*, *RuntimeKeepFree=*, *RuntimeMaxFileSize=*, *RuntimeMaxFiles=*

Enforce size limits on the journal files stored. The options prefixed with "System" apply to the journal files when stored on a persistent file system, more specifically /var/log/journal. The options prefixed with "Runtime" apply to the journal files when stored on a volatile in−memory file system, more specifically /run/log/journal. The former is used only when /var is mounted, writable, and the directory /var/log/journal exists. Otherwise, only the latter applies. Note that this means that during early boot and if the administrator disabled persistent logging, only the latter options apply, while the former apply if persistent logging is enabled and the system is fully booted up. **journalctl** and **systemd−journald** ignore all files with names not ending with ".journal" or ".journal~", so only such files, located in the appropriate directories, are taken into account when calculating current disk usage.

SystemMaxUse= and *RuntimeMaxUse=* control how much disk space the journal may use up at most. *SystemKeepFree=* and *RuntimeKeepFree=* control how much disk space systemd−journald shall leave free for other uses. **systemd−journald** will respect both limits and use the smaller of the two values.

The first pair defaults to 10% and the second to 15% of the size of the respective file system, but each value is capped to 4G. If the file system is nearly full and either *SystemKeepFree=* or *RuntimeKeepFree=* are violated when systemd−journald is started, the limit will be raised to the percentage that is actually free. This means that if there was enough free space before and journal files were created, and subsequently something else causes the file system to fill up, journald will stop using more space, but it will not be removing existing files to reduce the footprint again, either. Also note that only archived files are deleted to reduce the space occupied by journal files. This means that, in effect, there might still be more space used than *SystemMaxUse=* or *RuntimeMaxUse=* limit after a vacuuming operation is complete.

SystemMaxFileSize= and *RuntimeMaxFileSize=* control how large individual journal files may grow at most. This influences the granularity in which disk space is made available through rotation, i.e. deletion of historic data. Defaults to one eighth of the values configured with *SystemMaxUse=* and *RuntimeMaxUse=*, so that usually seven rotated journal files are kept as history.

Specify values in bytes or use K, M, G, T, P, E as units for the specified sizes (equal to 1024, 1024^2, ... bytes). Note that size limits are enforced synchronously when journal files are extended, and no explicit rotation step triggered by time is needed.

SystemMaxFiles= and *RuntimeMaxFiles=* control how many individual journal files to keep at most. Note that only archived files are deleted to reduce the number of files until this limit is reached; active files will stay around. This means that, in effect, there might still be more journal files around in total than this limit after a vacuuming operation is complete. This setting defaults to 100.

MaxFileSec=

The maximum time to store entries in a single journal file before rotating to the next one. Normally, time−based rotation should not be required as size−based rotation with options such as *SystemMaxFileSize=* should be sufficient to ensure that journal files do not grow without bounds. However, to ensure that not too much data is lost at once when old journal files are deleted, it might make sense to change this value from the default of one month. Set to 0 to turn off this feature. This setting takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

MaxRetentionSec=

The maximum time to store journal entries. This controls whether journal files containing entries older than the specified time span are deleted. Normally, time−based deletion of old journal files should not be required as size−based deletion with options such as *SystemMaxUse=* should be sufficient to ensure that journal files do not grow without bounds. However, to enforce data retention policies, it might make sense to change this value from the default of 0 (which turns off this feature). This setting also takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or " m" to override the default time unit of seconds.

SyncIntervalSec=

The timeout before synchronizing journal files to disk. After syncing, journal files are placed in the OFFLINE state. Note that syncing is unconditionally done immediately after a log message of priority CRIT, ALERT or EMERG has been logged. This setting hence applies only to messages of the levels ERR, WARNING, NOTICE, INFO, DEBUG. The default timeout is 5 minutes.

ForwardToSyslog=, *ForwardToKMsg=*, *ForwardToConsole=*, *ForwardToWall=*

Control whether log messages received by the journal daemon shall be forwarded to a traditional syslog daemon, to the kernel log buffer (kmsg), to the system console, or sent as wall messages to all logged−in users. These options take boolean arguments. If forwarding to syslog is enabled but nothing reads messages from the socket, forwarding to syslog has no effect. By default, only forwarding to syslog and wall is enabled. These settings may be overridden at boot time with the kernel command line options "systemd.journald.forward_to_syslog", "systemd.journald.forward_to_kmsg", "systemd.journald.forward_to_console", and "systemd.journald.forward_to_wall". If the option name is specified without "=" and the following argument, true is assumed. Otherwise, the argument is parsed as a boolean.

When forwarding to the console, the TTY to log to can be changed with *TTYPath=*, described below.

When forwarding to the kernel log buffer (kmsg), make sure to select a suitably large size for the log buffer, for example by adding "log_buf_len=8M" to the kernel command line. **systemd** will automatically disable kernel's rate−limiting applied to userspace processes (equivalent to setting "printk.devkmsg=on").

MaxLevelStore=, *MaxLevelSyslog=*, *MaxLevelKMsg=*, *MaxLevelConsole=*, *MaxLevelWall=*
: Controls the maximum log level of messages that are stored in the journal, forwarded to syslog, kmsg, the console or wall (if that is enabled, see above). As argument, takes one of "emerg", "alert", "crit", "err", "warning", "notice", "info", "debug", or integer values in the range of 0–7 (corresponding to the same levels). Messages equal or below the log level specified are stored/forwarded, messages above are dropped. Defaults to "debug" for *MaxLevelStore=* and *MaxLevelSyslog=*, to ensure that the all messages are stored in the journal and forwarded to syslog. Defaults to "notice" for *MaxLevelKMsg=*, "info" for *MaxLevelConsole=*, and "emerg" for *MaxLevelWall=*. These settings may be overridden at boot time with the kernel command line options "systemd.journald.max_level_store=", "systemd.journald.max_level_syslog=", "systemd.journald.max_level_kmsg=", "systemd.journald.max_level_console=", "systemd.journald.max_level_wall=".

ReadKMsg=
: Takes a boolean value. If enabled **systemd−journal** processes /dev/kmsg messages generated by the kernel. In the default journal namespace this option is enabled by default, it is disabled in all others.

TTYPath=
: Change the console TTY to use if *ForwardToConsole=yes* is used. Defaults to /dev/console.

LineMax=
: The maximum line length to permit when converting stream logs into record logs. When a systemd unit's standard output/error are connected to the journal via a stream socket, the data read is split into individual log records at newline ("\n", ASCII 10) and NUL characters. If no such delimiter is read for the specified number of bytes a hard log record boundary is artificially inserted, breaking up overly long lines into multiple log records. Selecting overly large values increases the possible memory usage of the Journal daemon for each stream client, as in the worst case the journal daemon needs to buffer the specified number of bytes in memory before it can flush a new log record to disk. Also note that permitting overly large line maximum line lengths affects compatibility with traditional log protocols as log records might not fit anymore into a single **AF_UNIX** or **AF_INET** datagram. Takes a size in bytes. If the value is suffixed with K, M, G or T, the specified size is parsed as Kilobytes, Megabytes, Gigabytes, or Terabytes (with the base 1024), respectively. Defaults to 48K, which is relatively large but still small enough so that log records likely fit into network datagrams along with extra room for metadata. Note that values below 79 are not accepted and will be bumped to 79.

FORWARDING TO TRADITIONAL SYSLOG DAEMONS

Journal events can be transferred to a different logging daemon in two different ways. With the first method, messages are immediately forwarded to a socket (/run/systemd/journal/syslog), where the traditional syslog daemon can read them. This method is controlled by the *ForwardToSyslog=* option. With a second method, a syslog daemon behaves like a normal journal client, and reads messages from the journal files, similarly to **journalctl**(1). With this, messages do not have to be read immediately, which allows a logging daemon which is only started late in boot to access all messages since the start of the system. In addition, full structured meta−data is available to it. This method of course is available only if the messages are stored in a journal file at all. So it will not work if *Storage=none* is set. It should be noted that usually the *second* method is used by syslog daemons, so the *Storage=* option, and not the *ForwardToSyslog=* option, is relevant for them.

SEE ALSO

systemd(1), **systemd-journald.service**(8), **journalctl**(1), **systemd.journal-fields**(7), **systemd-system.conf**(5)

NOTES

1. Seekable Sequential Key Generators
   https://eprint.iacr.org/2013/397

2. Users, Groups, UIDs and GIDs on systemd systems
   https://systemd.io/UIDS-GIDS