NAME

systemctl − Control the systemd system and service manager

SYNOPSIS

systemctl [OPTIONS...] COMMAND [UNIT...]

DESCRIPTION

systemctl may be used to introspect and control the state of the "systemd" system and service manager. Please refer to **systemd**(1) for an introduction into the basic concepts and functionality this tool manages.

COMMANDS

The following commands are understood:

Unit Commands (Introspection and Modification)

list−units [*PATTERN*...]

List units that **systemd** currently has in memory. This includes units that are either referenced directly or through a dependency, units that are pinned by applications programmatically, or units that were active in the past and have failed. By default only units which are active, have pending jobs, or have failed are shown; this can be changed with option **−−all**. If one or more *PATTERN*s are specified, only units matching one of them are shown. The units that are shown are additionally filtered by **−−type=** and **−−state=** if those options are specified.

Note that this command does not show unit templates, but only instances of unit templates. Units templates that aren't instantiated are not runnable, and will thus never show up in the output of this command. Specifically this means that foo@.service will never be shown in this list — unless instantiated, e.g. as foo@bar.service. Use **list−unit−files** (see below) for listing installed unit template files.

Produces output similar to

```
 UNIT                     LOAD   ACTIVE SUB     DESCRIPTION
 sys−module−fuse.device   loaded active plugged /sys/module/fuse
 −.mount                  loaded active mounted Root Mount
 boot−efi.mount           loaded active mounted /boot/efi
 systemd−journald.service loaded active running Journal Service
 systemd−logind.service   loaded active running Login Service
user@1000.service         loaded failed failed  User Manager for UID 1000
 ...
 systemd−tmpfiles−clean.timer loaded active waiting Daily Cleanup of Temporary Directories

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high−level unit activation state, i.e. generalization of SUB.
SUB    = The low−level unit activation state, values depend on unit type.

123 loaded units listed. Pass −−all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list−unit−files'.
```

The header and the last unit of a given type are underlined if the terminal supports that. A colored dot is shown next to services which were masked, not found, or otherwise failed.

The LOAD column shows the load state, one of **loaded**, **not−found**, **bad−setting**, **error**, **masked**. The ACTIVE columns shows the general unit state, one of **active**, **reloading**, **inactive**, **failed**, **activating**, **deactivating**. The SUB column shows the unit−type−specific detailed state of the unit, possible values vary by unit type. The list of possible LOAD, ACTIVE, and SUB states is not constant and new systemd releases may both add and remove values.

systemctl −−state=help

command may be used to display the current set of possible values.

This is the default command.

list−automounts [*PATTERN*...]
List automount units currently in memory, ordered by mount path. If one or more *PATTERN*s are specified, only automount units matching one of them are shown. Produces output similar to

```
WHAT       WHERE                  MOUNTED IDLE TIMEOUT UNIT
/dev/sdb1  /mnt/test              no      120s        mnt−test.automount
binfmt_misc /proc/sys/fs/binfmt_misc yes    0           proc−sys−fs−binfmt_misc.automount
```

2 automounts listed.

Also see **−−show−types**, **−−all**, and **−−state=**.

Added in version 252.

list−paths [*PATTERN*...]
List path units currently in memory, ordered by path. If one or more *PATTERN*s are specified, only path units matching one of them are shown. Produces output similar to

```
PATH                          CONDITION        UNIT                          ACTIVATES
/run/systemd/ask−password     DirectoryNotEmpty systemd−ask−password−plymouth.path systemd−ask−password−ply
/run/systemd/ask−password     DirectoryNotEmpty systemd−ask−password−wall.path     systemd−ask−password−wall.s
/var/cache/cups/org.cups.cupsd PathExists       cups.path                     cups.service
```

3 paths listed.

Also see **−−show−types**, **−−all**, and **−−state=**.

Added in version 254.

list−sockets [*PATTERN*...]
List socket units currently in memory, ordered by listening address. If one or more *PATTERN*s are specified, only socket units matching one of them are shown. Produces output similar to

```
LISTEN          UNIT                      ACTIVATES
/dev/initctl    systemd−initctl.socket    systemd−initctl.service
...
[::]:22         sshd.socket               sshd.service
kobject−uevent 1 systemd−udevd−kernel.socket systemd−udevd.service
```

5 sockets listed.

Note: because the addresses might contains spaces, this output is not suitable for programmatic consumption.

Also see **−−show−types**, **−−all**, and **−−state=**.

Added in version 202.

list−timers [*PATTERN*...]
List timer units currently in memory, ordered by the time they elapse next. If one or more *PATTERN*s are specified, only units matching one of them are shown. Produces output similar to

```
NEXT                         LEFT        LAST                         PASSED    UNIT                         ACTIVATES
–                            –           Thu 2017−02−23 13:40:29 EST  3 days ago ureadahead−stop.timer       ureadahead−stop.ser
Sun 2017−02−26 18:55:42 EST  1min 14s left Thu 2017−02−23 13:54:44 EST  3 days ago systemd−tmpfiles−clean.time
Sun 2017−02−26 20:37:16 EST  1h 42min left Sun 2017−02−26 11:56:36 EST  6h ago     apt−daily.timer           apt−da
Sun 2017−02−26 20:57:49 EST  2h 3min left  Sun 2017−02−26 11:56:36 EST  6h ago     snapd.refresh.timer       snapd
```

NEXT shows the next time the timer will run.

LEFT shows how long till the next time the timer runs.

LAST shows the last time the timer ran.

PASSED shows how long has passed since the timer last ran.

UNIT shows the name of the timer

ACTIVATES shows the name the service the timer activates when it runs.

Also see **−−all** and **−−state=**.

Added in version 209.

is−active *PATTERN*...
Check whether any of the specified units are active (i.e. running). Returns an exit code **0** if at least one is active, or non−zero otherwise. Unless **−−quiet** is specified, this will also print the current unit state to standard output.

is−failed [*PATTERN*...]
Check whether any of the specified units is in the "failed" state. If no unit is specified, check whether there are any failed units, which corresponds to the "degraded" state returned by **is−system−running**. Returns an exit code **0** if at least one has failed, non−zero otherwise. Unless **−−quiet** is specified, this will also print the current unit or system state to standard output.

Added in version 197.

status [*PATTERN*...|*PID*...]]
Show runtime status information about the whole system or about one or more units followed by most recent log data from the journal. If no positional arguments are specified, and no unit filter is given with **−−type=**, **−−state=**, or **−−failed**, shows the status of the whole system. If combined with **−−all**, follows that with the status of all units. If positional arguments are specified, each positional argument is treated as either a unit name to show, or a glob pattern to show units whose names match that pattern, or a PID to show the unit containing that PID. When **−−type=**, **−−state=**, or **−−failed** are used, units are additionally filtered by the TYPE and ACTIVE state.

This function is intended to generate human−readable output. If you are looking for computer−parsable output, use **show** instead. By default, this function only shows 10 lines of output and ellipsizes lines to fit in the terminal window. This can be changed with **−−lines** and **−−full**, see above. In addition, **journalctl −−unit=***NAME* or **journalctl −−user−unit=***NAME* use a similar filter for messages and might be more convenient.

Note that this operation only displays *runtime* status, i.e. information about the current invocation of the unit (if it is running) or the most recent invocation (if it is not running anymore, and has not been released from memory). Information about earlier invocations, invocations from previous system boots, or prior invocations that have already been released from memory may be retrieved via **journalctl −−unit=**.

systemd implicitly loads units as necessary, so just running the **status** will attempt to load a file. The command is thus not useful for determining if something was already loaded or not. The units may possibly also be quickly unloaded after the operation is completed if there's no reason to keep it in memory thereafter.

Example 1. Example output from systemctl status

```
$ systemctl status bluetooth
 bluetooth.service − Bluetooth service
   Loaded: loaded (/usr/lib/systemd/system/bluetooth.service; enabled; preset: enabled)
   Active: active (running) since Wed 2017−01−04 13:54:04 EST; 1 weeks 0 days ago
     Docs: man:bluetoothd(8)
 Main PID: 930 (bluetoothd)
   Status: "Running"
    Tasks: 1
   Memory: 648.0K
      CPU: 435ms
   CGroup: /system.slice/bluetooth.service
           930 /usr/lib/bluetooth/bluetoothd

Jan 12 10:46:45 example.com bluetoothd[8900]: Not enough free handles to register service
Jan 12 10:46:45 example.com bluetoothd[8900]: Current Time Service could not be registered
Jan 12 10:46:45 example.com bluetoothd[8900]: gatt−time−server: Input/output error (5)
```

The dot ("") uses color on supported terminals to summarize the unit state at a glance. Along with its color, its shape varies according to its state: "inactive" or "maintenance" is a white circle ("○"), "active" is a green dot (""), "deactivating" is a white dot, "failed" or "error" is a red cross ("×"), and "reloading" is a green clockwise circle arrow ("").

The "Loaded:" line in the output will show "loaded" if the unit has been loaded into memory. Other possible values for "Loaded:" include: "error" if there was a problem loading it, "not−found" if no unit file was found for this unit, "bad−setting" if an essential unit file setting could not be parsed and "masked" if the unit file has been masked. Along with showing the path to the unit file, this line will also show the enablement state. Enabled units are included in the dependency network between units, and thus are started at boot or via some other form of activation. See the full table of possible enablement states — including the definition of "masked" — in the documentation for the **is−enabled** command.

The "Active:" line shows active state. The value is usually "active" or "inactive". Active could mean started, bound, plugged in, etc depending on the unit type. The unit could also be in process of changing states, reporting a state of "activating" or "deactivating". A special "failed" state is entered when the service failed in some way, such as a crash, exiting with an error code or timing out. If the failed state is entered the cause will be logged for later reference.

show [*PATTERN*...|*JOB*...]

Show properties of one or more units, jobs, or the manager itself. If no argument is specified, properties of the manager will be shown. If a unit name is specified, properties of the unit are shown, and if a job ID is specified, properties of the job are shown. By default, empty properties are suppressed. Use **−−all** to show those too. To select specific properties to show, use **−−property=**. This command is intended to be used whenever computer−parsable output is required. Use **status** if you are looking for formatted human−readable output.

Many properties shown by **systemctl show** map directly to configuration settings of the system and service manager and its unit files. Note that the properties shown by the command are generally more low−level, normalized versions of the original configuration settings and expose runtime state in addition to configuration. For example, properties shown for service units include the service's current

main process identifier as "MainPID" (which is runtime state), and time settings are always exposed as properties ending in the "...USec" suffix even if a matching configuration options end in "...Sec", because microseconds is the normalized time unit used internally by the system and service manager.

For details about many of these properties, see the documentation of the D−Bus interface backing these properties, see **org.freedesktop.systemd1**(5).

cat *PATTERN*...
Show backing files of one or more units. Prints the "fragment" and "drop−ins" (source files) of units. Each file is preceded by a comment which includes the file name. Note that this shows the contents of the backing files on disk, which may not match the system manager's understanding of these units if any unit files were updated on disk and the **daemon−reload** command wasn't issued since.

Added in version 209.

help *PATTERN*...|*PID*...
Show manual pages for one or more units, if available. If a PID is given, the manual pages for the unit the process belongs to are shown.

Added in version 185.

list−dependencies [*UNIT*...]
Shows units required and wanted by the specified units. This recursively lists units following the *Requires*=, *Requisite*=, *Wants*=, *ConsistsOf*=, *BindsTo*=, and *Upholds*= dependencies. If no units are specified, default.target is implied.

The units that are shown are additionally filtered by **−−type=** and **−−state=** if those options are specified. Note that we won't be able to use a tree structure in this case, so **−−plain** is implied.

By default, only target units are recursively expanded. When **−−all** is passed, all other units are recursively expanded as well.

Options **−−reverse**, **−−after**, **−−before** may be used to change what types of dependencies are shown.

Note that this command only lists units currently loaded into memory by the service manager. In particular, this command is not suitable to get a comprehensive list at all reverse dependencies on a specific unit, as it won't list the dependencies declared by units currently not loaded.

Added in version 198.

start *PATTERN*...
Start (activate) one or more units specified on the command line.

Note that unit glob patterns expand to names of units currently in memory. Units which are not active and are not in a failed state usually are not in memory, and will not be matched by any pattern. In addition, in case of instantiated units, systemd is often unaware of the instance name until the instance has been started. Therefore, using glob patterns with **start** has limited usefulness. Also, secondary alias names of units are not considered.

Option **−−all** may be used to also operate on inactive units which are referenced by other loaded units. Note that this is not the same as operating on "all" possible units, because as the previous paragraph describes, such a list is ill−defined. Nevertheless, **systemctl start −−all** *GLOB* may be useful if all the units that should match the pattern are pulled in by some target which is known to be loaded.

stop *PATTERN*...
Stop (deactivate) one or more units specified on the command line.

This command will fail if the unit does not exist or if stopping of the unit is prohibited (see *RefuseManualStop*= in **systemd.unit**(5)). It will *not* fail if any of the commands configured to stop the unit (*ExecStop*=, etc.) fail, because the manager will still forcibly terminate the unit.

If a unit that gets stopped can still be triggered by other units, a warning containing the names of the triggering units is shown. **−−no−warn** can be used to suppress the warning.

reload *PATTERN*...
Asks all units listed on the command line to reload their configuration. Note that this will reload the service−specific configuration, not the unit configuration file of systemd. If you want systemd to reload the configuration file of a unit, use the **daemon−reload** command. In other words: for the example case of Apache, this will reload Apache's httpd.conf in the web server, not the apache.service systemd unit file.

This command should not be confused with the **daemon−reload** command.

restart *PATTERN*...
Stop and then start one or more units specified on the command line. If the units are not running yet, they will be started.

Note that restarting a unit with this command does not necessarily flush out all of the unit's resources before it is started again. For example, the per−service file descriptor storage facility (see *FileDescriptorStoreMax*= in **systemd.service**(5)) will remain intact as long as the unit has a job pending, and is only cleared when the unit is fully stopped and no jobs are pending anymore. If it is intended that the file descriptor store is flushed out, too, during a restart operation an explicit **systemctl stop** command followed by **systemctl start** should be issued.

try−restart *PATTERN*...
Stop and then start one or more units specified on the command line if the units are running. This does nothing if units are not running.

reload−or−restart *PATTERN*...
Reload one or more units if they support it. If not, stop and then start them instead. If the units are not running yet, they will be started.

try−reload−or−restart *PATTERN*...
Reload one or more units if they support it. If not, stop and then start them instead. This does nothing if the units are not running.

Added in version 229.

isolate *UNIT*
Start the unit specified on the command line and its dependencies and stop all others, unless they have **IgnoreOnIsolate=yes** (see **systemd.unit**(5)). If a unit name with no extension is given, an extension of ".target" will be assumed.

This command is dangerous, since it will immediately stop processes that are not enabled in the new target, possibly including the graphical environment or terminal you are currently using.

Note that this operation is allowed only on units where **AllowIsolate=** is enabled. See **systemd.unit**(5) for details.

kill *PATTERN*...
Send a UNIX process signal to one or more processes of the unit. Use **−−kill−whom=** to select which process to send the signal to. Use **−−signal=** to select the signal to send. Combine with **−−kill−value=** to enqueue a POSIX Realtime Signal with an associated value.

clean *PATTERN*...
Remove the configuration, state, cache, logs or runtime data of the specified units. Use **−−what=** to

select which kind of resource to remove. For service units this may be used to remove the directories configured with *ConfigurationDirectory*=, *StateDirectory*=, *CacheDirectory*=, *LogsDirectory*= and *RuntimeDirectory*=, see **systemd.exec**(5) for details. It may also be used to clear the file descriptor store as enabled via *FileDescriptorStoreMax*=, see **systemd.service**(5) for details. For timer units this may be used to clear out the persistent timestamp data if *Persistent*= is used and **−−what=state** is selected, see **systemd.timer**(5). This command only applies to units that use either of these settings. If **−−what=** is not specified, the cache and runtime data as well as the file descriptor store are removed (as these three types of resources are generally redundant and reproducible on the next invocation of the unit). Note that the specified units must be stopped to invoke this operation.

Added in version 243.

freeze *PATTERN*...
Freeze one or more units specified on the command line using cgroup freezer

Freezing the unit will cause all processes contained within the cgroup corresponding to the unit to be suspended. Being suspended means that unit's processes won't be scheduled to run on CPU until thawed. Note that this command is supported only on systems that use unified cgroup hierarchy. Unit is automatically thawed just before we execute a job against the unit, e.g. before the unit is stopped.

Added in version 246.

thaw *PATTERN*...
Thaw (unfreeze) one or more units specified on the command line.

This is the inverse operation to the **freeze** command and resumes the execution of processes in the unit's cgroup.

Added in version 246.

set−property *UNIT PROPERTY*=*VALUE*...
Set the specified unit properties at runtime where this is supported. This allows changing configuration parameter properties such as resource control settings at runtime. Not all properties may be changed at runtime, but many resource control settings (primarily those in **systemd.resource-control**(5)) may. The changes are applied immediately, and stored on disk for future boots, unless **−−runtime** is passed, in which case the settings only apply until the next reboot. The syntax of the property assignment follows closely the syntax of assignments in unit files.

Example: **systemctl set−property foobar.service CPUWeight=200**

If the specified unit appears to be inactive, the changes will be only stored on disk as described previously hence they will be effective when the unit will be started.

Note that this command allows changing multiple properties at the same time, which is preferable over setting them individually.

Example: **systemctl set−property foobar.service CPUWeight=200 MemoryMax=2G IPAccounting=yes**

Like with unit file configuration settings, assigning an empty setting usually resets a property to its defaults.

Example: **systemctl set−property avahi−daemon.service IPAddressDeny=**

Added in version 206.

bind *UNIT PATH* [*PATH*]

Bind−mounts a file or directory from the host into the specified unit's mount namespace. The first path argument is the source file or directory on the host, the second path argument is the destination file or directory in the unit's mount namespace. When the latter is omitted, the destination path in the unit's mount namespace is the same as the source path on the host. When combined with the **−−read−only** switch, a ready−only bind mount is created. When combined with the **−−mkdir** switch, the destination path is first created before the mount is applied.

Note that this option is currently only supported for units that run within a mount namespace (e.g.: with **RootImage=**, **PrivateMounts=**, etc.). This command supports bind−mounting directories, regular files, device nodes, **AF_UNIX** socket nodes, as well as FIFOs. The bind mount is ephemeral, and it is undone as soon as the current unit process exists. Note that the namespace mentioned here, where the bind mount will be added to, is the one where the main service process runs. Other processes (those exececuted by **ExecReload=**, **ExecStartPre=**, etc.) run in distinct namespaces.

If supported by the kernel, any prior mount on the selected target will be replaced by the new mount. If not supported, any prior mount will be over−mounted, but remain pinned and inaccessible.

Added in version 248.

mount−image *UNIT IMAGE* [*PATH* [*PARTITION_NAME*:*MOUNT_OPTIONS*]]
Mounts an image from the host into the specified unit's mount namespace. The first path argument is the source image on the host, the second path argument is the destination directory in the unit's mount namespace (i.e. inside **RootImage=/RootDirectory=**). The following argument, if any, is interpreted as a colon−separated tuple of partition name and comma−separated list of mount options for that partition. The format is the same as the service **MountImages=** setting. When combined with the **−−read−only** switch, a ready−only mount is created. When combined with the **−−mkdir** switch, the destination path is first created before the mount is applied.

Note that this option is currently only supported for units that run within a mount namespace (i.e. with **RootImage=**, **PrivateMounts=**, etc.). Note that the namespace mentioned here where the image mount will be added to, is the one where the main service process runs. Note that the namespace mentioned here, where the bind mount will be added to, is the one where the main service process runs. Other processes (those exececuted by **ExecReload=**, **ExecStartPre=**, etc.) run in distinct namespaces.

If supported by the kernel, any prior mount on the selected target will be replaced by the new mount. If not supported, any prior mount will be over−mounted, but remain pinned and inaccessible.

Example:

systemctl mount−image foo.service /tmp/img.raw /var/lib/image root:ro,nosuid

systemctl mount−image −−mkdir bar.service /tmp/img.raw /var/lib/baz/img

Added in version 248.

service−log−level *SERVICE* [*LEVEL*]
If the *LEVEL* argument is not given, print the current log level as reported by service *SERVICE*.

If the optional argument *LEVEL* is provided, then change the current log level of the service to *LEVEL*. The log level should be a typical syslog log level, i.e. a value in the range 0...7 or one of the strings **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**; see **syslog**(3) for details.

The service must have the appropriate *BusName=destination* property and also implement the generic

org.freedesktop.LogControl1(5) interface. (systemctl will use the generic D−Bus protocol to access the org.freedesktop.LogControl1.LogLevel interface for the D−Bus name *destination*.)

Added in version 247.

service−log−target *SERVICE* [*TARGET*]
If the *TARGET* argument is not given, print the current log target as reported by service *SERVICE*.

If the optional argument *TARGET* is provided, then change the current log target of the service to *TARGET*. The log target should be one of the strings **console** (for log output to the service's standard error stream), **kmsg** (for log output to the kernel log buffer), **journal** (for log output to **systemd-journald.service**(8) using the native journal protocol), **syslog** (for log output to the classic syslog socket /dev/log), **null** (for no log output whatsoever) or **auto** (for an automatically determined choice, typically equivalent to **console** if the service is invoked interactively, and **journal** or **syslog** otherwise).

For most services, only a small subset of log targets make sense. In particular, most "normal" services should only implement **console**, **journal**, and **null**. Anything else is only appropriate for low−level services that are active in very early boot before proper logging is established.

The service must have the appropriate *BusName=destination* property and also implement the generic **org.freedesktop.LogControl1**(5) interface. (systemctl will use the generic D−Bus protocol to access the org.freedesktop.LogControl1.LogLevel interface for the D−Bus name *destination*.)

Added in version 247.

reset−failed [*PATTERN***...]**
Reset the "failed" state of the specified units, or if no unit name is passed, reset the state of all units. When a unit fails in some way (i.e. process exiting with non−zero error code, terminating abnormally or timing out), it will automatically enter the "failed" state and its exit code and status is recorded for introspection by the administrator until the service is stopped/re−started or reset with this command.

In addition to resetting the "failed" state of a unit it also resets various other per−unit properties: the start rate limit counter of all unit types is reset to zero, as is the restart counter of service units. Thus, if a unit's start limit (as configured with *StartLimitIntervalSec=*/*StartLimitBurst=*) is hit and the unit refuses to be started again, use this command to make it startable again.

whoami [*PID***...]**
Returns the units the processes referenced by the given PIDs belong to (one per line). If no PID is specified returns the unit the **systemctl** command is invoked in.

Added in version 254.

Unit File Commands

list−unit−files [*PATTERN...*]
List unit files installed on the system, in combination with their enablement state (as reported by **is−enabled**). If one or more *PATTERN*s are specified, only unit files whose name matches one of them are shown (patterns matching unit file system paths are not supported).

Unlike **list−units** this command will list template units in addition to explicitly instantiated units.

Added in version 233.

enable *UNIT***...**, **enable** *PATH***...**
Enable one or more units or unit instances. This will create a set of symlinks, as encoded in the [Install] sections of the indicated unit files. After the symlinks have been created, the system manager configuration is reloaded (in a way equivalent to **daemon−reload**), in order to ensure the changes are taken into account immediately. Note that this does *not* have the effect of also starting any of the units

being enabled. If this is desired, combine this command with the **−−now** switch, or invoke **start** with appropriate arguments later. Note that in case of unit instance enablement (i.e. enablement of units of the form foo@bar.service), symlinks named the same as instances are created in the unit configuration directory, however they point to the single template unit file they are instantiated from.

This command expects either valid unit names (in which case various unit file directories are automatically searched for unit files with appropriate names), or absolute paths to unit files (in which case these files are read directly). If a specified unit file is located outside of the usual unit file directories, an additional symlink is created, linking it into the unit configuration path, thus ensuring it is found when requested by commands such as **start**. The file system where the linked unit files are located must be accessible when systemd is started (e.g. anything underneath /home/ or /var/ is not allowed, unless those directories are located on the root file system).

This command will print the file system operations executed. This output may be suppressed by passing **−−quiet**.

Note that this operation creates only the symlinks suggested in the [Install] section of the unit files. While this command is the recommended way to manipulate the unit configuration directory, the administrator is free to make additional changes manually by placing or removing symlinks below this directory. This is particularly useful to create configurations that deviate from the suggested default installation. In this case, the administrator must make sure to invoke **daemon−reload** manually as necessary, in order to ensure the changes are taken into account.

When using this operation on units without install information, a warning about it is shown. **−−no−warn** can be used to suppress the warning.

Enabling units should not be confused with starting (activating) units, as done by the **start** command. Enabling and starting units is orthogonal: units may be enabled without being started and started without being enabled. Enabling simply hooks the unit into various suggested places (for example, so that the unit is automatically started on boot or when a particular kind of hardware is plugged in). Starting actually spawns the daemon process (in case of service units), or binds the socket (in case of socket units), and so on.

Depending on whether **−−system**, **−−user**, **−−runtime**, or **−−global** is specified, this enables the unit for the system, for the calling user only, for only this boot of the system, or for all future logins of all users. Note that in the last case, no systemd daemon configuration is reloaded.

Using **enable** on masked units is not supported and results in an error.

disable *UNIT*...
Disables one or more units. This removes all symlinks to the unit files backing the specified units from the unit configuration directory, and hence undoes any changes made by **enable** or **link**. Note that this removes *all* symlinks to matching unit files, including manually created symlinks, and not just those actually created by **enable** or **link**. Note that while **disable** undoes the effect of **enable**, the two commands are otherwise not symmetric, as **disable** may remove more symlinks than a prior **enable** invocation of the same unit created.

This command expects valid unit names only, it does not accept paths to unit files.

In addition to the units specified as arguments, all units are disabled that are listed in the *Also*= setting contained in the [Install] section of any of the unit files being operated on.

This command implicitly reloads the system manager configuration after completing the operation. Note that this command does not implicitly stop the units that are being disabled. If this is desired, either combine this command with the **−−now** switch, or invoke the **stop** command with appropriate

arguments later.

This command will print information about the file system operations (symlink removals) executed. This output may be suppressed by passing **−−quiet**.

If a unit gets disabled but its triggering units are still active, a warning containing the names of the triggering units is shown. **−−no−warn** can be used to suppress the warning.

When this command is used with **−−user**, the units being operated on might still be enabled in global scope, and thus get started automatically even after a successful disablement in user scope. In this case, a warning about it is shown, which can be suppressed using **−−no−warn**.

This command honors **−−system**, **−−user**, **−−runtime**, **−−global** and **−−no−warn** in a similar way as **enable**.

Added in version 238.

reenable *UNIT*...
Reenable one or more units, as specified on the command line. This is a combination of **disable** and **enable** and is useful to reset the symlinks a unit file is enabled with to the defaults configured in its [Install] section. This command expects a unit name only, it does not accept paths to unit files.

Added in version 238.

preset *UNIT*...
Reset the enable/disable status one or more unit files, as specified on the command line, to the defaults configured in the preset policy files. This has the same effect as **disable** or **enable**, depending how the unit is listed in the preset files.

Use **−−preset−mode=** to control whether units shall be enabled and disabled, or only enabled, or only disabled.

If the unit carries no install information, it will be silently ignored by this command. *UNIT* must be the real unit name, any alias names are ignored silently.

For more information on the preset policy format, see **systemd.preset**(5).

Added in version 238.

preset−all
Resets all installed unit files to the defaults configured in the preset policy file (see above).

Use **−−preset−mode=** to control whether units shall be enabled and disabled, or only enabled, or only disabled.

Added in version 215.

is−enabled *UNIT*...
Checks whether any of the specified unit files are enabled (as with **enable**). Returns an exit code of 0 if at least one is enabled, non−zero otherwise. Prints the current enable status (see table). To suppress this output, use **−−quiet**. To show installation targets, use **−−full**.

Table 1. is−enabled output

Added in version 238.

mask *UNIT*...
Mask one or more units, as specified on the command line. This will link these unit files to /dev/null, making it impossible to start them. This is a stronger version of **disable**, since it prohibits all kinds of activation of the unit, including enablement and manual activation. Use this option with care. This honors the **−−runtime** option to only mask temporarily until the next reboot of the system. The **−−now** option may be used to ensure that the units are also stopped. This command expects valid unit names only, it does not accept unit file paths.

Note that this will create a symlink under the unit's name in /etc/systemd/system/ (in case **−−runtime** is not specified) or /run/systemd/system/ (in case **−−runtime** is specified). If a matching unit file already exists under these directories this operation will hence fail. This means that the operation is primarily useful to mask units shipped by the vendor (as those are shipped in /usr/lib/systemd/system/ and not the aforementioned two directories), but typically doesn't work for units created locally (as those are typically placed precisely in the two aforementioned directories). Similar restrictions apply for **−−user** mode, in which case the directories are below the user's home directory however.

If a unit gets masked but its triggering units are still active, a warning containing the names of the triggering units is shown. **−−no−warn** can be used to suppress the warning.

Added in version 238.

unmask *UNIT*...
Unmask one or more unit files, as specified on the command line. This will undo the effect of **mask**. This command expects valid unit names only, it does not accept unit file paths.

Added in version 238.

link *PATH*...
Link a unit file that is not in the unit file search path into the unit file search path. This command expects an absolute path to a unit file. The effect of this may be undone with **disable**. The effect of this command is that a unit file is made available for commands such as **start**, even though it is not installed directly in the unit search path. The file system where the linked unit files are located must be accessible when systemd is started (e.g. anything underneath /home/ or /var/ is not allowed, unless those directories are located on the root file system).

Added in version 233.

revert *UNIT*...
Revert one or more unit files to their vendor versions. This command removes drop−in configuration files that modify the specified units, as well as any user−configured unit file that overrides a matching vendor supplied unit file. Specifically, for a unit "foo.service" the matching directories "foo.service.d/" with all their contained files are removed, both below the persistent and runtime configuration directories (i.e. below /etc/systemd/system and /run/systemd/system); if the unit file has a vendor−supplied version (i.e. a unit file located below /usr/) any matching persistent or runtime unit file that overrides it is removed, too. Note that if a unit file has no vendor−supplied version (i.e. is only defined below /etc/systemd/system or /run/systemd/system, but not in a unit file stored below /usr/), then it is not removed. Also, if a unit is masked, it is unmasked.

Effectively, this command may be used to undo all changes made with **systemctl edit**, **systemctl set−property** and **systemctl mask** and puts the original unit file with its settings back in effect.

Added in version 230.

add−wants *TARGET UNIT*..., **add−requires** *TARGET UNIT*...
Adds "Wants=" or "Requires=" dependencies, respectively, to the specified *TARGET* for one or more

units.

This command honors **−−system**, **−−user**, **−−runtime** and **−−global** in a way similar to **enable**.

Added in version 217.

edit *UNIT***...**
Edit a drop−in snippet or a whole replacement file if **−−full** is specified, to extend or override the specified unit.

Depending on whether **−−system** (the default), **−−user**, or **−−global** is specified, this command creates a drop−in file for each unit either for the system, for the calling user, or for all futures logins of all users. Then, the editor (see the "Environment" section below) is invoked on temporary files which will be written to the real location if the editor exits successfully.

If **−−drop−in=** is specified, the given drop−in file name will be used instead of the default override.conf.

If **−−full** is specified, this will copy the original units instead of creating drop−in files.

If **−−force** is specified and any units do not already exist, new unit files will be opened for editing.

If **−−runtime** is specified, the changes will be made temporarily in /run/ and they will be lost on the next reboot.

If the temporary file is empty upon exit, the modification of the related unit is canceled.

After the units have been edited, systemd configuration is reloaded (in a way that is equivalent to **daemon−reload**).

Note that this command cannot be used to remotely edit units and that you cannot temporarily edit units which are in /etc/, since they take precedence over /run/.

Added in version 218.

get−default
Return the default target to boot into. This returns the target unit name default.target is aliased (symlinked) to.

Added in version 205.

set−default *TARGET*
Set the default target to boot into. This sets (symlinks) the default.target alias to the given target unit.

Added in version 205.

Machine Commands

list−machines [*PATTERN*...]
List the host and all running local containers with their state. If one or more *PATTERN*s are specified, only containers matching one of them are shown.

Added in version 212.

Job Commands

list−jobs [*PATTERN*...]
List jobs that are in progress. If one or more *PATTERN*s are specified, only jobs for units matching one of them are shown.

When combined with **−−after** or **−−before** the list is augmented with information on which other job each job is waiting for, and which other jobs are waiting for it, see above.

Added in version 233.

cancel [*JOB***...]**
Cancel one or more jobs specified on the command line by their numeric job IDs. If no job ID is specified, cancel all pending jobs.

Added in version 233.

Environment Commands

systemd supports an environment block that is passed to processes the manager spawns. The names of the variables can contain ASCII letters, digits, and the underscore character. Variable names cannot be empty or start with a digit. In variable values, most characters are allowed, but the whole sequence must be valid UTF−8. (Note that control characters like newline (**NL**), tab (**TAB**), or the escape character (**ESC**), *are* valid ASCII and thus valid UTF−8). The total length of the environment block is limited to **_SC_ARG_MAX** value defined by **sysconf**(3).

show−environment
Dump the systemd manager environment block. This is the environment block that is passed to all processes the manager spawns. The environment block will be dumped in straightforward form suitable for sourcing into most shells. If no special characters or whitespace is present in the variable values, no escaping is performed, and the assignments have the form "VARIABLE=value". If whitespace or characters which have special meaning to the shell are present, dollar−single−quote escaping is used, and assignments have the form "VARIABLE=$'value'". This syntax is known to be supported by **bash**(1), **zsh**(1), **ksh**(1), and **busybox**(1)'s **ash**(1), but not **dash**(1) or **fish**(1).

set−environment *VARIABLE=VALUE***...**
Set one or more systemd manager environment variables, as specified on the command line. This command will fail if variable names and values do not conform to the rules listed above.

Added in version 233.

unset−environment *VARIABLE***...**
Unset one or more systemd manager environment variables. If only a variable name is specified, it will be removed regardless of its value. If a variable and a value are specified, the variable is only removed if it has the specified value.

Added in version 233.

import−environment *VARIABLE...*
Import all, one or more environment variables set on the client into the systemd manager environment block. If a list of environment variable names is passed, client−side values are then imported into the manager's environment block. If any names are not valid environment variable names or have invalid values according to the rules described above, an error is raised. If no arguments are passed, the entire environment block inherited by the **systemctl** process is imported. In this mode, any inherited invalid environment variables are quietly ignored.

Importing of the full inherited environment block (calling this command without any arguments) is deprecated. A shell will set dozens of variables which only make sense locally and are only meant for processes which are descendants of the shell. Such variables in the global environment block are confusing to other processes.

Added in version 209.

Manager State Commands

daemon−reload
Reload the systemd manager configuration. This will rerun all generators (see **systemd.generator**(7)),

reload all unit files, and recreate the entire dependency tree. While the daemon is being reloaded, all sockets systemd listens on behalf of user configuration will stay accessible.

This command should not be confused with the **reload** command.

daemon−reexec
Reexecute the systemd manager. This will serialize the manager state, reexecute the process and deserialize the state again. This command is of little use except for debugging and package upgrades. Sometimes, it might be helpful as a heavy−weight **daemon−reload**. While the daemon is being reexecuted, all sockets systemd listening on behalf of user configuration will stay accessible.

log−level [*LEVEL*]
If no argument is given, print the current log level of the manager. If an optional argument *LEVEL* is provided, then the command changes the current log level of the manager to *LEVEL* (accepts the same values as **−−log−level=** described in **systemd**(1)).

Added in version 244.

log−target [*TARGET*]
If no argument is given, print the current log target of the manager. If an optional argument *TARGET* is provided, then the command changes the current log target of the manager to *TARGET* (accepts the same values as **−−log−target=**, described in **systemd**(1)).

Added in version 244.

service−watchdogs [yes|no]
If no argument is given, print the current state of service runtime watchdogs of the manager. If an optional boolean argument is provided, then globally enables or disables the service runtime watchdogs (**WatchdogSec=**) and emergency actions (e.g. **OnFailure=** or **StartLimitAction=**); see **systemd.service**(5). The hardware watchdog is not affected by this setting.

Added in version 244.

System Commands

is−system−running
Checks whether the system is operational. This returns success (exit code 0) when the system is fully up and running, specifically not in startup, shutdown or maintenance mode, and with no failed services. Failure is returned otherwise (exit code non−zero). In addition, the current state is printed in a short string to standard output, see the table below. Use **−−quiet** to suppress this output.

Use **−−wait** to wait until the boot process is completed before printing the current state and returning the appropriate error status. If **−−wait** is in use, states *initializing* or *starting* will not be reported, instead the command will block until a later state (such as *running* or *degraded*) is reached.

Table 2. is−system−running output

Name	Description	Exit Code
initializing	Early bootup, before basic.target is reached or the *maintenance* state entered.	> 0
starting	Late bootup, before the job queue becomes idle for the first time, or one of the rescue targets are reached.	> 0
running	The system is fully operational.	0
degraded	The system is operational but one or more units failed.	> 0
maintenance	The rescue or emergency target is active.	> 0
stopping	The manager is shutting down.	> 0
offline	The manager is not running. Specifically, this is the operational state if an incompatible program is running as system manager (PID 1).	> 0
unknown	The operational state could not be determined, due to lack of resources or another error cause.	> 0

Added in version 215.

default
Enter default mode. This is equivalent to **systemctl isolate default.target**. This operation is blocking by default, use **−−no−block** to request asynchronous behavior.

rescue
Enter rescue mode. This is equivalent to **systemctl isolate rescue.target**. This operation is blocking by default, use **−−no−block** to request asynchronous behavior.

emergency
Enter emergency mode. This is equivalent to **systemctl isolate emergency.target**. This operation is blocking by default, use **−−no−block** to request asynchronous behavior.

halt
Shut down and halt the system. This is mostly equivalent to **systemctl start halt.target −−job−mode=replace−irreversibly −−no−block**, but also prints a wall message to all users. This command is asynchronous; it will return after the halt operation is enqueued, without waiting for it to complete. Note that this operation will simply halt the OS kernel after shutting down, leaving the hardware powered on. Use **systemctl poweroff** for powering off the system (see below).

If combined with **−−force**, shutdown of all running services is skipped, however all processes are killed and all file systems are unmounted or mounted read−only, immediately followed by the system halt. If **−−force** is specified twice, the operation is immediately executed without terminating any processes or unmounting any file systems. This may result in data loss. Note that when **−−force** is specified twice the halt operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

If combined with **−−when=**, shutdown will be scheduled after the given timestamp. And **−−when=cancel** will cancel the shutdown.

poweroff
Shut down and power−off the system. This is mostly equivalent to **systemctl start poweroff.target −−job−mode=replace−irreversibly −−no−block**, but also prints a wall message to all users. This command is asynchronous; it will return after the power−off operation is enqueued, without waiting for it to complete.

This command honors **−−force** and **−−when=** in a similar way as **halt**.

reboot
Shut down and reboot the system.

This command mostly equivalent to **systemctl start reboot.target −−job−mode=replace−irreversibly −−no−block**, but also prints a wall message to all users. This command is asynchronous; it will return after the reboot operation is enqueued, without waiting for it to complete.

If the switch **−−reboot−argument=** is given, it will be passed as the optional argument to the **reboot**(2) system call.

Options **−−boot−loader−entry=**, **−−boot−loader−menu=**, and **−−firmware−setup** can be used to select what to do *after* the reboot. See the descriptions of those options for details.

This command honors **−−force** and **−−when=** in a similar way as **halt**.

If a new kernel has been loaded via **kexec −−load**, a **kexec** will be performed instead of a reboot, unless "SYSTEMCTL_SKIP_AUTO_KEXEC=1" has been set. If a new root file system has been set up on "/run/nextroot/", a **soft−reboot** will be performed instead of a reboot, unless "SYSTEMCTL_SKIP_AUTO_SOFT_REBOOT=1" has been set.

Added in version 246.

kexec
Shut down and reboot the system via **kexec**. This command will load a kexec kernel if one wasn't loaded yet or fail. A kernel may be loaded earlier by a separate step, this is particularly useful if a custom initrd or additional kernel command line options are desired. The **−−force** can be used to continue without a kexec kernel, i.e. to perform a normal reboot. The final reboot step is equivalent to **systemctl start kexec.target −−job−mode=replace−irreversibly −−no−block**.

To load a kernel, an enumeration is performed following the **Boot Loader Specification**[1], and the default boot entry is loaded. For this step to succeed, the system must be using UEFI and the boot loader entries must be configured appropriately. **bootctl list** may be used to list boot entries, see **bootctl**(1).

This command is asynchronous; it will return after the reboot operation is enqueued, without waiting for it to complete.

This command honors **−−force** and **−−when=** similarly to **halt**.

If a new kernel has been loaded via **kexec −−load**, a **kexec** will be performed when **reboot** is invoked, unless "SYSTEMCTL_SKIP_AUTO_KEXEC=1" has been set.

soft−reboot
Shut down and reboot userspace. This is equivalent to **systemctl start soft−reboot.target −−job−mode=replace−irreversibly −−no−block**. This command is asynchronous; it will return after

the reboot operation is enqueued, without waiting for it to complete.

This command honors **−−force** and **−−when=** in a similar way as **halt**.

This operation only reboots userspace, leaving the kernel running. See **systemd-soft-reboot.service**(8) for details.

If a new root file system has been set up on "/run/nextroot/", a **soft−reboot** will be performed when **reboot** is invoked, unless "SYSTEMCTL_SKIP_AUTO_SOFT_REBOOT=1" has been set.

Added in version 254.

exit [*EXIT_CODE*]
Ask the service manager to quit. This is only supported for user service managers (i.e. in conjunction with the **−−user** option) or in containers and is equivalent to **poweroff** otherwise. This command is asynchronous; it will return after the exit operation is enqueued, without waiting for it to complete.

The service manager will exit with the specified exit code, if *EXIT_CODE* is passed.

Added in version 227.

switch−root [*ROOT* [*INIT*]]
Switches to a different root directory and executes a new system manager process below it. This is intended for use in the initrd, and will transition from the initrd's system manager process (a.k.a. "init" process, PID 1) to the main system manager process which is loaded from the actual host root files system. This call takes two arguments: the directory that is to become the new root directory, and the path to the new system manager binary below it to execute as PID 1. If both are omitted or the former is an empty string it defaults to /sysroot/. If the latter is omitted or is an empty string, a systemd binary will automatically be searched for and used as service manager. If the system manager path is omitted, equal to the empty string or identical to the path to the systemd binary, the state of the initrd's system manager process is passed to the main system manager, which allows later introspection of the state of the services involved in the initrd boot phase.

Added in version 209.

suspend
Suspend the system. This will trigger activation of the special target unit suspend.target. This command is asynchronous, and will return after the suspend operation is successfully enqueued. It will not wait for the suspend/resume cycle to complete.

hibernate
Hibernate the system. This will trigger activation of the special target unit hibernate.target. This command is asynchronous, and will return after the hibernation operation is successfully enqueued. It will not wait for the hibernate/thaw cycle to complete.

hybrid−sleep
Hibernate and suspend the system. This will trigger activation of the special target unit hybrid−sleep.target. This command is asynchronous, and will return after the hybrid sleep operation is successfully enqueued. It will not wait for the sleep/wake−up cycle to complete.

Added in version 196.

suspend−then−hibernate
Suspend the system and hibernate it after the delay specified in systemd−sleep.conf. This will trigger activation of the special target unit suspend−then−hibernate.target. This command is asynchronous, and will return after the hybrid sleep operation is successfully enqueued. It will not wait for the sleep/wake−up or hibernate/thaw cycle to complete.

Added in version 240.

Parameter Syntax

Unit commands listed above take either a single unit name (designated as *UNIT*), or multiple unit specifications (designated as *PATTERN*...). In the first case, the unit name with or without a suffix must be given. If the suffix is not specified (unit name is "abbreviated"), systemctl will append a suitable suffix, ".service" by default, and a type−specific suffix in case of commands which operate only on specific unit types. For example,

systemctl start sshd

and

systemctl start sshd.service

are equivalent, as are

systemctl isolate default

and

systemctl isolate default.target

Note that (absolute) paths to device nodes are automatically converted to device unit names, and other (absolute) paths to mount unit names.

systemctl status /dev/sda
systemctl status /home

are equivalent to:

systemctl status dev−sda.device
systemctl status home.mount

In the second case, shell−style globs will be matched against the primary names of all units currently in memory; literal unit names, with or without a suffix, will be treated as in the first case. This means that literal unit names always refer to exactly one unit, but globs may match zero units and this is not considered an error.

Glob patterns use **fnmatch**(3), so normal shell−style globbing rules are used, and "*", "?", "[]" may be used. See **glob**(7) for more details. The patterns are matched against the primary names of units currently in memory, and patterns which do not match anything are silently skipped. For example:

systemctl stop sshd@*.service

will stop all sshd@.service instances. Note that alias names of units, and units that aren't in memory are not considered for glob expansion.

For unit file commands, the specified *UNIT* should be the name of the unit file (possibly abbreviated, see above), or the absolute path to the unit file:

systemctl enable foo.service

or

systemctl link /path/to/foo.service

OPTIONS

The following options are understood:

−t, **−−type=**
: The argument is a comma−separated list of unit types such as **service** and **socket**. When units are listed with **list−units**, **list−dependencies**, **show**, or **status**, only units of the specified types will be shown. By default, units of all types are shown.

  As a special case, if one of the arguments is **help**, a list of allowed values will be printed and the program will exit.

−−state=
: The argument is a comma−separated list of unit LOAD, SUB, or ACTIVE states. When listing units with **list−units**, **list−dependencies**, **show** or **status**, show only those in the specified states. Use **−−state=failed** or **−−failed** to show only failed units.

  As a special case, if one of the arguments is **help**, a list of allowed values will be printed and the program will exit.

  Added in version 206.

−p, **−−property=**
: When showing unit/job/manager properties with the **show** command, limit display to properties specified in the argument. The argument should be a comma−separated list of property names, such as "MainPID". Unless specified, all known properties are shown. If specified more than once, all properties with the specified names are shown. Shell completion is implemented for property names.

  For the manager itself, **systemctl show** will show all available properties, most of which are derived or closely match the options described in **systemd-system.conf**(5).

  Properties for units vary by unit type, so showing any unit (even a non−existent one) is a way to list properties pertaining to this type. Similarly, showing any job will list properties pertaining to all jobs. Properties for units are documented in **systemd.unit**(5), and the pages for individual unit types **systemd.service**(5), **systemd.socket**(5), etc.

−P
: Equivalent to **−−value −−property=**, i.e. shows the value of the property without the property name or "=". Note that using **−P** once will also affect all properties listed with **−p/−−property=**.

  Added in version 246.

−a, **−−all**
: When listing units with **list−units**, also show inactive units and units which are following other units. When showing unit/job/manager properties, show all properties regardless whether they are set or not.

  To list all units installed in the file system, use the **list−unit−files** command instead.

  When listing units with **list−dependencies**, recursively show dependencies of all dependent units (by default only dependencies of target units are shown).

  When used with **status**, show journal messages in full, even if they include unprintable characters or are very long. By default, fields with unprintable characters are abbreviated as "blob data". (Note that the pager may escape unprintable characters again.)

−r, **−−recursive**
: When listing units, also show units of local containers. Units of local containers will be prefixed with the container name, separated by a single colon character (":").

Added in version 212.

−−reverse
Show reverse dependencies between units with **list−dependencies**, i.e. follow dependencies of type *WantedBy*=, *RequiredBy*=, *UpheldBy*=, *PartOf*=, *BoundBy*=, instead of *Wants*= and similar.

Added in version 203.

−−after
With **list−dependencies**, show the units that are ordered before the specified unit. In other words, recursively list units following the *After*= dependency.

Note that any *After*= dependency is automatically mirrored to create a *Before*= dependency. Temporal dependencies may be specified explicitly, but are also created implicitly for units which are *WantedBy*= targets (see **systemd.target**(5)), and as a result of other directives (for example *RequiresMountsFor*=). Both explicitly and implicitly introduced dependencies are shown with **list−dependencies**.

When passed to the **list−jobs** command, for each printed job show which other jobs are waiting for it. May be combined with **−−before** to show both the jobs waiting for each job as well as all jobs each job is waiting for.

Added in version 203.

−−before
With **list−dependencies**, show the units that are ordered after the specified unit. In other words, recursively list units following the *Before*= dependency.

When passed to the **list−jobs** command, for each printed job show which other jobs it is waiting for. May be combined with **−−after** to show both the jobs waiting for each job as well as all jobs each job is waiting for.

Added in version 212.

−−with−dependencies
When used with **status**, **cat**, **list−units**, and **list−unit−files**, those commands print all specified units and the dependencies of those units.

Options **−−reverse**, **−−after**, **−−before** may be used to change what types of dependencies are shown.

Added in version 245.

−l, **−−full**
Do not ellipsize unit names, process tree entries, journal output, or truncate unit descriptions in the output of **status**, **list−units**, **list−jobs**, and **list−timers**.

Also, show installation targets in the output of **is−enabled**.

−−value
When printing properties with **show**, only print the value, and skip the property name and "=". Also see option **−P** above.

Added in version 230.

−−show−types
When showing sockets, show the type of the socket.

Added in version 202.

−−job−mode=
When queuing a new job, this option controls how to deal with already queued jobs. It takes one of "fail", "replace", "replace−irreversibly", "isolate", "ignore−dependencies", "ignore−requirements", "flush", "triggering", or "restart−dependencies". Defaults to "replace", except when the **isolate** command is used which implies the "isolate" job mode.

If "fail" is specified and a requested operation conflicts with a pending job (more specifically: causes an already pending start job to be reversed into a stop job or vice versa), cause the operation to fail.

If "replace" (the default) is specified, any conflicting pending job will be replaced, as necessary.

If "replace−irreversibly" is specified, operate like "replace", but also mark the new jobs as irreversible. This prevents future conflicting transactions from replacing these jobs (or even being enqueued while the irreversible jobs are still pending). Irreversible jobs can still be cancelled using the **cancel** command. This job mode should be used on any transaction which pulls in shutdown.target.

"isolate" is only valid for start operations and causes all other units to be stopped when the specified unit is started. This mode is always used when the **isolate** command is used.

"flush" will cause all queued jobs to be canceled when the new job is enqueued.

If "ignore−dependencies" is specified, then all unit dependencies are ignored for this new job and the operation is executed immediately. If passed, no required units of the unit passed will be pulled in, and no ordering dependencies will be honored. This is mostly a debugging and rescue tool for the administrator and should not be used by applications.

"ignore−requirements" is similar to "ignore−dependencies", but only causes the requirement dependencies to be ignored, the ordering dependencies will still be honored.

"triggering" may only be used with **systemctl stop**. In this mode, the specified unit and any active units that trigger it are stopped. See the discussion of *Triggers*= in **systemd.unit**(5) for more information about triggering units.

"restart−dependencies" may only be used with **systemctl start**. In this mode, dependencies of the specified unit will receive restart propagation, as if a restart job had been enqueued for the unit.

Added in version 209.

−T, **−−show−transaction**
When enqueuing a unit job (for example as effect of a **systemctl start** invocation or similar), show brief information about all jobs enqueued, covering both the requested job and any added because of unit dependencies. Note that the output will only include jobs immediately part of the transaction requested. It is possible that service start−up program code run as effect of the enqueued jobs might request further jobs to be pulled in. This means that completion of the listed jobs might ultimately entail more jobs than the listed ones.

Added in version 242.

−−fail
Shorthand for **−−job−mode=**fail.

When used with the **kill** command, if no units were killed, the operation results in an error.

Added in version 227.

−−check−inhibitors=

When system shutdown or sleep state is requested, this option controls checking of inhibitor locks. It takes one of "auto", "yes" or "no". Defaults to "auto", which will behave like "yes" for interactive invocations (i.e. from a TTY) and "no" for non−interactive invocations. "yes" lets the request respect inhibitor locks. "no" lets the request ignore inhibitor locks.

Applications can establish inhibitor locks to prevent certain important operations (such as CD burning) from being interrupted by system shutdown or sleep. Any user may take these locks and privileged users may override these locks. If any locks are taken, shutdown and sleep state requests will normally fail (unless privileged). However, if "no" is specified or "auto" is specified on a non−interactive requests, the operation will be attempted. If locks are present, the operation may require additional privileges.

Option **−−force** provides another way to override inhibitors.

Added in version 248.

−i

Shortcut for **−−check−inhibitors=no**.

Added in version 198.

−−dry−run

Just print what would be done. Currently supported by verbs **halt**, **poweroff**, **reboot**, **kexec**, **suspend**, **hibernate**, **hybrid−sleep**, **suspend−then−hibernate**, **default**, **rescue**, **emergency**, and **exit**.

Added in version 236.

−q, **−−quiet**

Suppress printing of the results of various commands and also the hints about truncated log lines. This does not suppress output of commands for which the printed output is the only result (like **show**). Errors are always printed.

−−no−warn

Don't generate the warnings shown by default in the following cases:

- when **systemctl** is invoked without procfs mounted on /proc/,
- when using **enable** or **disable** on units without install information (i.e. don't have or have an empty [Install] section),
- when using **disable** combined with **−−user** on units that are enabled in global scope,
- when a **stop**−ped, **disable**−d, or **mask**−ed unit still has active triggering units.

Added in version 253.

−−no−block

Do not synchronously wait for the requested operation to finish. If this is not specified, the job will be verified, enqueued and **systemctl** will wait until the unit's start−up is completed. By passing this argument, it is only verified and enqueued. This option may not be combined with **−−wait**.

−−wait

Synchronously wait for started units to terminate again. This option may not be combined with **−−no−block**. Note that this will wait forever if any given unit never terminates (by itself or by getting stopped explicitly); particularly services which use "RemainAfterExit=yes".

When used with **is−system−running**, wait until the boot process is completed before returning.

Added in version 232.

−−user

Talk to the service manager of the calling user, rather than the service manager of the system.

−−system
Talk to the service manager of the system. This is the implied default.

−−failed
List units in failed state. This is equivalent to **−−state=failed**.

Added in version 233.

−−no−wall
Do not send wall message before halt, power−off and reboot.

−−global
When used with **enable** and **disable**, operate on the global user configuration directory, thus enabling or disabling a unit file globally for all future logins of all users.

−−no−reload
When used with **enable** and **disable**, do not implicitly reload daemon configuration after executing the changes.

−−no−ask−password
When used with **start** and related commands, disables asking for passwords. Background services may require input of a password or passphrase string, for example to unlock system hard disks or cryptographic certificates. Unless this option is specified and the command is invoked from a terminal, **systemctl** will query the user on the terminal for the necessary secrets. Use this option to switch this behavior off. In this case, the password must be supplied by some other means (for example graphical password agents) or the service might fail. This also disables querying the user for authentication for privileged operations.

−−kill−whom=
When used with **kill**, choose which processes to send a UNIX process signal to. Must be one of **main**, **control** or **all** to select whether to kill only the main process, the control process or all processes of the unit. The main process of the unit is the one that defines the life−time of it. A control process of a unit is one that is invoked by the manager to induce state changes of it. For example, all processes started due to the *ExecStartPre=*, *ExecStop=* or *ExecReload=* settings of service units are control processes. Note that there is only one control process per unit at a time, as only one state change is executed at a time. For services of type *Type=forking*, the initial process started by the manager for *ExecStart=* is a control process, while the process ultimately forked off by that one is then considered the main process of the unit (if it can be determined). This is different for service units of other types, where the process forked off by the manager for *ExecStart=* is always the main process itself. A service unit consists of zero or one main process, zero or one control process plus any number of additional processes. Not all unit types manage processes of these types however. For example, for mount units, control processes are defined (which are the invocations of /usr/bin/mount and /usr/bin/umount), but no main process is defined. If omitted, defaults to **all**.

Added in version 252.

−−kill−value=*INT*
If used with the **kill** command, enqueues a signal along with the specified integer value parameter to the specified process(es). This operation is only available for POSIX Realtime Signals (i.e. **−−signal=SIGRTMIN+...** or **−−signal=SIGRTMAX−...**), and ensures the signals are generated via the **sigqueue**(3) system call, rather than **kill**(3). The specified value must be a 32−bit signed integer, and may be specified either in decimal, in hexadecimal (if prefixed with "0x"), octal (if prefixed with "0o") or binary (if prefixed with "0b")

If this option is used the signal will only be enqueued on the control or main process of the unit, never on other processes belonging to the unit, i.e. **−−kill−whom=all** will only affect main and control processes but no other processes.

Added in version 254.

−s, **−−signal=**
When used with **kill**, choose which signal to send to selected processes. Must be one of the well−known signal specifiers such as **SIGTERM**, **SIGINT** or **SIGSTOP**. If omitted, defaults to **SIGTERM**.

The special value "help" will list the known values and the program will exit immediately, and the special value "list" will list known values along with the numerical signal numbers and the program will exit immediately.

−−what=
Select what type of per−unit resources to remove when the **clean** command is invoked, see above. Takes one of **configuration**, **state**, **cache**, **logs**, **runtime**, **fdstore** to select the type of resource. This option may be specified more than once, in which case all specified resource types are removed. Also accepts the special value **all** as a shortcut for specifying all six resource types. If this option is not specified defaults to the combination of **cache**, **runtime** and **fdstore**, i.e. the three kinds of resources that are generally considered to be redundant and can be reconstructed on next invocation. Note that the explicit removal of the **fdstore** resource type is only useful if the *FileDescriptorStorePreserve=* option is enabled, since the file descriptor store is otherwise cleaned automatically when the unit is stopped.

Added in version 243.

−f, **−−force**
When used with **enable**, overwrite any existing conflicting symlinks.

When used with **edit**, create all of the specified units which do not already exist.

When used with **halt**, **poweroff**, **reboot** or **kexec**, execute the selected operation without shutting down all units. However, all processes will be killed forcibly and all file systems are unmounted or remounted read−only. This is hence a drastic but relatively safe option to request an immediate reboot. If **−−force** is specified twice for these operations (with the exception of **kexec**), they will be executed immediately, without terminating any processes or unmounting any file systems. Warning: specifying **−−force** twice with any of these operations might result in data loss. Note that when **−−force** is specified twice the selected operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

−−message=
When used with **halt**, **poweroff** or **reboot**, set a short message explaining the reason for the operation. The message will be logged together with the default shutdown message.

Added in version 225.

−−now
When used with **enable**, the units will also be started. When used with **disable** or **mask**, the units will also be stopped. The start or stop operation is only carried out when the respective enable or disable operation has been successful.

Added in version 220.

−−root=
When used with **enable/disable/is−enabled** (and related commands), use the specified root path when looking for unit files. If this option is present, **systemctl** will operate on the file system directly, instead of communicating with the **systemd** daemon to carry out changes.

−−image=*image*
Takes a path to a disk image file or block device node. If specified, all operations are applied to file system in the indicated disk image. This option is similar to **−−root=**, but operates on file systems

stored in disk images or block devices. The disk image should either contain just a file system or a set of file systems within a GPT partition table, following the **Discoverable Partitions Specification**[2]. For further information on supported disk images, see **systemd-nspawn**(1)'s switch of the same name.

Added in version 252.

−−image−policy=*policy*
Takes an image policy string as argument, as per **systemd.image-policy**(7). The policy is enforced when operating on the disk image specified via **−−image=**, see above. If not specified defaults to the "*" policy, i.e. all recognized file systems in the image are used.

−−runtime
When used with **enable**, **disable**, **edit**, (and related commands), make changes only temporarily, so that they are lost on the next reboot. This will have the effect that changes are not made in subdirectories of /etc/ but in /run/, with identical immediate effects, however, since the latter is lost on reboot, the changes are lost too.

Similarly, when used with **set−property**, make changes only temporarily, so that they are lost on the next reboot.

−−preset−mode=
Takes one of "full" (the default), "enable−only", "disable−only". When used with the **preset** or **preset−all** commands, controls whether units shall be disabled and enabled according to the preset rules, or only enabled, or only disabled.

Added in version 215.

−n, **−−lines=**
When used with **status**, controls the number of journal lines to show, counting from the most recent ones. Takes a positive integer argument, or 0 to disable journal output. Defaults to 10.

−o, **−−output=**
When used with **status**, controls the formatting of the journal entries that are shown. For the available choices, see **journalctl**(1). Defaults to "short".

−−firmware−setup
When used with the **reboot**, **poweroff**, or **halt** command, indicate to the system's firmware to reboot into the firmware setup interface for the next boot. Note that this functionality is not available on all systems.

Added in version 220.

−−boot−loader−menu=*timeout*
When used with the **reboot**, **poweroff**, or **halt** command, indicate to the system's boot loader to show the boot loader menu on the following boot. Takes a time value as parameter — indicating the menu timeout. Pass zero in order to disable the menu timeout. Note that not all boot loaders support this functionality.

Added in version 242.

−−boot−loader−entry=*ID*
When used with the **reboot**, **poweroff**, or **halt** command, indicate to the system's boot loader to boot into a specific boot loader entry on the following boot. Takes a boot loader entry identifier as argument, or "help" in order to list available entries. Note that not all boot loaders support this functionality.

Added in version 242.

−−reboot−argument=
This switch is used with **reboot**. The value is architecture and firmware specific. As an example,

"recovery" might be used to trigger system recovery, and "fota" might be used to trigger a “firmware over the air” update.

Added in version 246.

−−plain
When used with **list−dependencies**, **list−units** or **list−machines**, the output is printed as a list instead of a tree, and the bullet circles are omitted.

Added in version 203.

−−timestamp=
Change the format of printed timestamps. The following values may be used:

pretty (this is the default)
"Day YYYY−MM−DD HH:MM:SS TZ"

Added in version 248.

unix
"@seconds−since−the−epoch"

Added in version 251.

us, **µs**
"Day YYYY−MM−DD HH:MM:SS.UUUUUU TZ"

Added in version 248.

utc
"Day YYYY−MM−DD HH:MM:SS UTC"

Added in version 248.

us+utc, **µs+utc**
"Day YYYY−MM−DD HH:MM:SS.UUUUUU UTC"

Added in version 248.

Added in version 247.

−−mkdir
When used with **bind**, creates the destination file or directory before applying the bind mount. Note that even though the name of this option suggests that it is suitable only for directories, this option also creates the destination file node to mount over if the object to mount is not a directory, but a regular file, device node, socket or FIFO.

Added in version 248.

−−marked
Only allowed with **reload−or−restart**. Enqueues restart jobs for all units that have the "needs−restart" mark, and reload jobs for units that have the "needs−reload" mark. When a unit marked for reload does not support reload, restart will be queued. Those properties can be set using **set−property Markers=...**.

Unless **−−no−block** is used, **systemctl** will wait for the queued jobs to finish.

Added in version 248.

−−read−only

When used with **bind**, creates a read−only bind mount.

Added in version 248.

−−drop−in=*NAME*
When used with **edit**, use *NAME* as the drop−in file name instead of override.conf.

Added in version 253.

−−when=
When used with **halt**, **poweroff**, **reboot** or **kexec**, schedule the action to be performed at the given timestamp, which should adhere to the syntax documented in **systemd.time**(7) section "PARSING TIMESTAMPS". Specially, if "show" is given, the currently scheduled action will be shown, which can be canceled by passing an empty string or "cancel".

Added in version 254.

−H, **−−host=**
Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":", and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl −H** *HOST*. Put IPv6 addresses in brackets.

−M, **−−machine=**
Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "−−user −−machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

−−no−pager
Do not pipe output into a pager.

−−legend=*BOOL*
Enable or disable printing of the legend, i.e. column headers and the footer with hints. The legend is printed by default, unless disabled with **−−quiet** or similar.

−h, **−−help**
Print a short help text and exit.

−−version
Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non−zero failure code otherwise.

systemctl uses the return codes defined by LSB, as defined in **LSB 3.0.0**[3].

Table 3. LSB return codes

Value	Description in LSB	Use in systemd
0	"program is running or service is OK"	unit is active
1	"program is dead and /var/run pid file exists"	unit *not* failed (used by **is−failed**)
2	"program is dead and /var/lock lock file exists"	unused
3	"program is not running"	unit is not active
4	"program or service status is unknown"	no such unit

The mapping of LSB service states to systemd unit states is imperfect, so it is better to not rely on those return values but to look for specific unit states and substates instead.

ENVIRONMENT

$SYSTEMD_EDITOR

Editor to use when editing units; overrides *$EDITOR* and *$VISUAL*. If neither *$SYSTEMD_EDITOR* nor *$EDITOR* nor *$VISUAL* are present or if it is set to an empty string or if their execution failed, systemctl will try to execute well known editors in this order: **editor**(1), **nano**(1), **vim**(1), **vi**(1).

Added in version 218.

$SYSTEMD_LOG_LEVEL

The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog**(3) for more information.

$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This setting is only useful when messages are written directly to the terminal, because **journalctl**(1) and other tools that display logs will color messages based on the log level on their own.

$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This setting is only useful when messages are written directly to the terminal or a file, because **journalctl**(1) and other tools that display logs will attach timestamps based on the entry metadata on their own.

$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

Note that the log location is often attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console−prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see **syslog**(3), **kmsg** (log to the kernel circular log buffer), **journal** (log to the journal), **journal−or−kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

$SYSTEMD_PAGER

Pager to use when **−−no−pager** is not given; overrides *$PAGER*. If neither *$SYSTEMD_PAGER* nor *$PAGER* are set, a set of well−known pager implementations are tried in turn, including **less**(1) and

more(1), until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **−−no−pager**.

Note: if *$SYSTEMD_PAGERSECURE* is not set, *$SYSTEMD_PAGER* (as well as *$PAGER*) will be silently ignored.

$SYSTEMD_LESS
Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K
This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of *$SYSTEMD_LESS* does not include "K", and the pager that is invoked is **less**, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X
This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See **less**(1) for more discussion.

$SYSTEMD_LESSCHARSET
Override the charset passed to **less** (by default "utf−8", if the invoking terminal is determined to be UTF−8 compatible).

$SYSTEMD_PAGERSECURE
Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If *$SYSTEMD_PAGERSECURE* is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see **geteuid**(2) and **sd_pid_get_owner_uid**(3). In secure mode, **LESSSECURE=1** will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When *$SYSTEMD_PAGERSECURE* is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only **less**(1) implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under **sudo**(8) or **pkexec**(1), care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as describe above. Setting *SYSTEMD_PAGERSECURE=0* or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the *$SYSTEMD_PAGER* or *$PAGER* variables are to be honoured, *$SYSTEMD_PAGERSECURE* must be set too. It might be reasonable to completely disable the pager using **−−no−pager** instead.

$SYSTEMD_COLORS
Takes a boolean argument. When true, **systemd** and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on *$TERM* and what the console is connected to.

$SYSTEMD_URLIFY
The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that **systemd** makes based on *$TERM* and other conditions.

SEE ALSO

systemd(1), **journalctl**(1), **loginctl**(1), **machinectl**(1), **systemd.unit**(5), **systemd.resource-control**(5), **systemd.special**(7), **wall**(1), **systemd.preset**(5), **systemd.generator**(7), **glob**(7)

NOTES

1. Boot Loader Specification
   https://uapi-group.org/specifications/specs/boot_loader_specification
2. Discoverable Partitions Specification
   https://uapi-group.org/specifications/specs/discoverable_partitions_specification
3. LSB 3.0.0
   http://refspecs.linuxbase.org/LSB_3.0.0/LSB-PDA/LSB-PDA/iniscrptact.html