NAME

systemctl − Control the systemd system and service manager

SYNOPSIS

systemctl [OPTIONS...] COMMAND [NAME...]

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.

OPTIONS

The following options are understood:

−t, **−−type=**

The argument should be a comma−separated list of unit types such as **service** and **socket**.

If one of the arguments is a unit type, when listing units, limit display to certain unit types. Otherwise, units of all types will be 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 should be a comma−separated list of unit LOAD, SUB, or ACTIVE states. When listing units, show only those in the specified states. Use **−−state=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.

−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. Those properties are documented 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.

−a, **−−all**

When listing units, show all loaded units, regardless of their state, including inactive units. When showing unit/job/manager properties, show all properties regardless whether they are set or not.

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

−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 (":").

−−reverse

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

−−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**.

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

−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**.

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

−−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" or "flush". 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.

"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 honoured.

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

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

−i, **−−ignore−inhibitors**
When system shutdown or a sleep state is requested, ignore inhibitor locks. Applications can establish inhibitor locks to avoid that certain important operations (such as CD burning or suchlike) are interrupted by system shutdown or a sleep state. 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 (regardless of whether privileged or not) and a list of active locks is printed. However, if **−−ignore−inhibitors** is specified, the locks are ignored and not printed, and the operation attempted

anyway, possibly requiring additional privileges.

−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−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.

−−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.

−−no−wall
Do not send wall message before halt, power−off, 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−who=
When used with **kill**, choose which processes to send a 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**.

−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**.

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

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, 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.

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

−−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.

−−root=
When used with **enable**/**disable**/**is−enabled** (and related commands), use an alternate root path when looking for unit files.

−−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.

−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. 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** command, indicate to the system's firmware to boot into setup mode. Note that this is currently only supported on some EFI systems and only if the system was booted in EFI mode.

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

−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 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*.

−M, **−−machine=**
Execute operation on a local container. Specify a container name to connect to.

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

−−no−legend
Do not print the legend, i.e. column headers and the footer with hints.

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

−−version
Print a short version string and exit.

COMMANDS

The following commands are understood:

Unit Commands

list−units [*PATTERN*...]
List known units (subject to limitations specified with **−t**). If one or more *PATTERN*s are specified, only units matching one of them are shown.

This is the default command.

list−sockets [*PATTERN*...]
List socket units 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.

See also the options **−−show−types**, **−−all**, and **−−state=**.

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

See also the options **−−all** and **−−state=**.

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

Note that glob patterns operate on the set of primary names of currently loaded units. Units which are not active and are not in a failed state usually are not loaded, 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.

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

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*...
Restart one or more units specified on the command line. If the units are not running yet, they will be started.

try−restart *PATTERN*...
Restart 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, restart 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, restart them instead. This does nothing if the units are not running.

isolate *NAME*
Start the unit specified on the command line and its dependencies and stop all others. If a unit name with no extension is given, an extension of ".target" will be assumed.

This is similar to changing the runlevel in a traditional init system. The **isolate** command will immediately stop processes that are not enabled in the new unit, possibly including the graphical environment or terminal you are currently using.

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

kill *PATTERN*...
Send a signal to one or more processes of the unit. Use **−−kill−who=** to select which process to kill. Use **−−signal=** to select the signal to send.

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 are in a "failed" state. 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 state to standard output.

status [*PATTERN*...|*PID*...]]
Show terse runtime status information about one or more units, followed by most recent log data from the journal. If no units are specified, show system status. If combined with **−−all**, also show the status of all units (subject to limitations specified with **−t**). If a PID is passed, show information about the unit the process belongs to.

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.

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 is shown, and if a job ID is specified, properties of the job is 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.

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.

set−property *NAME ASSIGNMENT*...
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 instantly, 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 CPUShares=777**

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. Like unit file configuration settings, assigning the empty list to list parameters will reset the list.

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.

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 restarted or reset with this command.

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

Unit File Commands

list−unit−files [*PATTERN*...]
List installed unit files and their enablement state (as reported by **is−enabled**). If one or more *PATTERN*s are specified, only units whose filename (just the last component of the path) matches one of them are shown.

enable *NAME*...
Enable one or more unit files or unit file instances, as specified on the command line. This will create a number of symlinks as encoded in the "[Install]" sections of the unit files. After the symlinks have been created, the systemd configuration is reloaded (in a way that is equivalent to **daemon−reload**) 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, either **−−now** should be used together with this command, or an additional **start** command must be invoked for the unit. Also note that, in case of instance enablement, symlinks named the same as instances are created in the install location, however they all point to the same template unit file.

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

Note that this operation creates only the suggested symlinks for the units. 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 in the 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 to ensure the changes are taken into account.

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, or only this boot. Note that in the last case, no systemd daemon configuration is reloaded.

Using **enable** on masked units results in an error.

disable *NAME*...
Disables one or more units. This removes all symlinks to the specified unit files from the unit configuration directory, and hence undoes the changes made by **enable**. Note however that this removes all symlinks to the unit files (i.e. including manual additions), not just those actually created by **enable**. This call implicitly reloads the systemd daemon configuration after completing the disabling of the units. Note that this command does not implicitly stop the units that are being disabled. If this is desired, either **−−now** should be used together with this command, or an additional **stop** command should be executed afterwards.

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

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

reenable *NAME*...
Reenable one or more unit files, as specified on the command line. This is a combination of **disable** and **enable** and is useful to reset the symlinks a unit is enabled with to the defaults configured in the "[Install]" section of the unit file.

preset *NAME*...
Reset 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.

For more information on the preset policy format, see **systemd.preset**(5). For more information on the concept of presets, please consult the **Preset**[1] document.

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.

is−enabled *NAME*...

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**.

Table 1. is−enabled output

Name	Description	Exit Code
"enabled"	Enabled via .wants/, .requires/ or alias symlinks (permanently in /etc/systemd/system/, or transiently in /run/systemd/system/).	0
"enabled−runtime"		
"linked"	Made available through one or more symlinks to the unit file (permanently in /etc/systemd/system/ or transiently in /run/systemd/system/), even though the unit file might reside outside of the unit file search path.	> 0
"linked−runtime"		
"masked"	Completely disabled, so that any start operation on it fails (permanently in /etc/systemd/system/ or transiently in /run/systemd/systemd/).	> 0
"masked−runtime"		
"static"	The unit file is not enabled, and has no provisions for enabling in the "[Install]" section.	0
"indirect"	The unit file itself is not enabled, but it has a non−empty *Also*= setting in the "[Install]" section, listing other unit files that might be enabled.	0
"disabled"	Unit file is not enabled, but contains an "[Install]" section with installation instructions.	> 0
"bad"	Unit file is invalid or another error occurred. Note that **is−enabled** will not actually return this state, but print an error message instead. However the unit file listing printed by **list−unit−files** might show it.	> 0

mask *NAME*...

Mask one or more unit files, as specified on the command line. This will link these units 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 can be used to ensure that the units are also stopped.

unmask *NAME*...
Unmask one or more unit files, as specified on the command line. This will undo the effect of **mask**.

link *FILENAME*...
Link a unit file that is not in the unit file search paths into the unit file search path. This requires an absolute path to a unit file. The effect of this can be undone with **disable**. The effect of this command is that a unit file is available for **start** and other commands although it is not installed directly in the unit search path.

add−wants *TARGET NAME*..., **add−requires** *TARGET NAME*...
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**.

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

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.

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

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

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.

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.

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.

Environment Commands

show−environment

Dump the systemd manager environment block. The environment block will be dumped in straight−forward form suitable for sourcing into a shell script. This environment block will be passed to all processes the manager spawns.

set−environment *VARIABLE=VALUE*...

Set one or more systemd manager environment variables, as specified on the command line.

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.

import−environment [*VARIABLE...*]

Import all, one or more environment variables set on the client into the systemd manager environment block. If no arguments are passed, the entire environment block is imported. Otherwise, a list of one or more environment variable names should be passed, whose client−side values are then imported into the manager's environment block.

Manager Lifecycle 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.

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.

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

default
Enter default mode. This is mostly equivalent to **isolate default.target**.

rescue
Enter rescue mode. This is mostly equivalent to **isolate rescue.target**, but also prints a wall message to all users.

emergency
Enter emergency mode. This is mostly equivalent to **isolate emergency.target**, but also prints a wall message to all users.

halt
Shut down and halt the system. This is mostly equivalent to **start halt.target --job-mode=replace-irreversibly**, but also prints a wall message to all users. 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.

poweroff
Shut down and power-off the system. This is mostly equivalent to **start poweroff.target --job-mode=replace-irreversibly**, but also prints a wall message to all users. 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 powering off. 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.

reboot [*arg***]**
Shut down and reboot the system. This is mostly equivalent to **start reboot.target**

−−job−mode=replace−irreversibly, but also prints a wall message to all users. 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 reboot. 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.

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

kexec
Shut down and reboot the system via kexec. This is mostly equivalent to **start kexec.target −−job−mode=replace−irreversibly**, but also prints a wall message to all users. 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 reboot.

exit [*EXIT_CODE***]**
Ask the systemd 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.

The systemd manager can exit with a non−zero exit code if the optional argument *EXIT_CODE* is given.

switch−root *ROOT* **[***INIT***]**
Switches to a different root directory and executes a new system manager process below it. This is intended for usage in initial RAM disks ("initrd"), and will transition from the initrd's system manager process (a.k.a. "init" process) to the main system manager process. 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 the latter is omitted or the empty string, a systemd binary will automatically be searched for and used as init. If the system manager path is omitted or equal to the empty string, 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.

suspend
Suspend the system. This will trigger activation of the special suspend.target target.

hibernate
Hibernate the system. This will trigger activation of the special hibernate.target target.

hybrid−sleep
Hibernate and suspend the system. This will trigger activation of the special hybrid−sleep.target target.

Parameter Syntax
Unit commands listed above take either a single unit name (designated as *NAME*), 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 currently loaded units; 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 currently loaded units, 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 loaded are not considered for glob expansion.

For unit file commands, the specified *NAME* 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

EXIT STATUS

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

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).

$SYSTEMD_PAGER
: Pager to use when **−−no−pager** is not given; overrides *$PAGER*. Setting this to an empty string or the value "cat" is equivalent to passing **−−no−pager**.

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

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. Preset
   http://freedesktop.org/wiki/Software/systemd/Preset