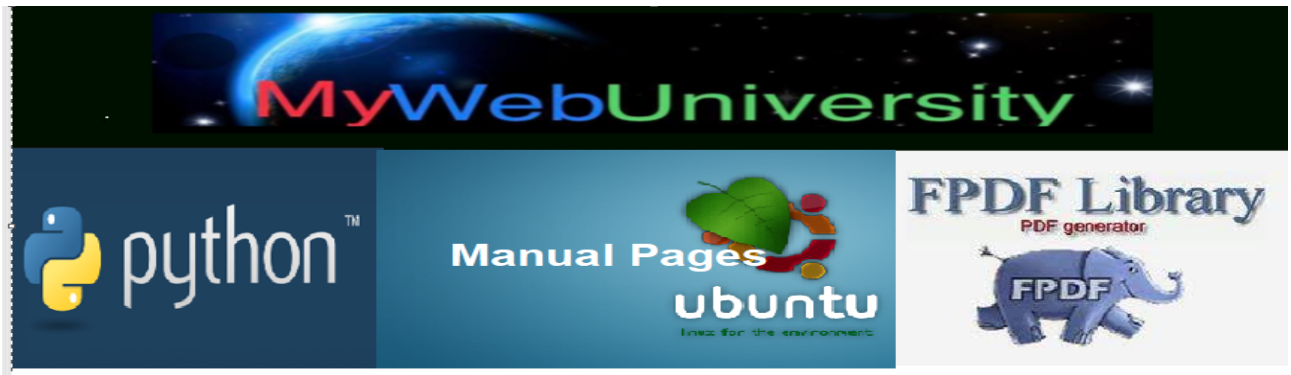

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'systemd-run.1'

$ man systemd-run.1

SYSTEMD-RUN(1) systemd-run SYSTEMD-RUN(1)

NAME

systemd-run - Run programs in transient scope units, service units, or path-, socket-, or timer-triggered service units

SYNOPSIS

systemd-run [OPTIONS...] COMMAND [ARGS...]

systemd-run [OPTIONS...] [PATH OPTIONS...] {COMMAND [ARGS...]}

systemd-run [OPTIONS...] [SOCKET OPTIONS...] {COMMAND [ARGS...]}

systemd-run [OPTIONS...] [TIMER OPTIONS...] {COMMAND [ARGS...]}

DESCRIPTION

systemd-run may be used to create and start a transient .service or .scope unit and run the specified COMMAND in it. It may also be used to create and start a transient .path, .socket, or .timer unit, that activates a .service unit when elapsing.

If a command is run as transient service unit, it will be started and managed by the service manager like any other service, and thus shows up in the output of systemctl list-units like any other unit. It will run in a clean and detached execution environment, with the service manager as its parent process. In this mode, systemd-run will start the service asynchronously in the background and return after the command has begun execution (unless --no-block, --wait, --pipe, or --pty are specified, see below).

If a command is run as transient scope unit, it will be executed by systemd-run itself as parent process and will thus inherit the execution environment of the caller. However, the processes of the command are managed by the service manager similarly to normal services, and will show up in the output of systemctl list-units. Execution in this case is

synchronous, and will return only when the command finishes. This mode is enabled via the --scope switch (see below).

If a command is run with path, socket, or timer options such as --on-calendar= (see below), a transient path, socket, or timer unit is created alongside the service unit for the specified command. Only the transient path, socket, or timer unit is started immediately, the transient service unit will be triggered by the path, socket, or timer unit. If the --unit= option is specified, the COMMAND may be omitted. In this case, systemd-run creates only a .path, .socket, or .timer unit that triggers the specified unit.

By default, services created with systemd-run default to the simple type, see the description of Type= in systemd.service(5) for details. Note that when this type is used, the service manager (and thus the systemd-run command) considers service start-up successful as soon as the fork() for the main service process succeeded, i.e. before the execve() is invoked, and thus even if the specified command cannot be started. Consider using the exec service type (i.e. --property=Type=exec) to ensure that systemd-run returns successfully only if the specified command line has been successfully started.

After systemd-run passes the command to the service manager, the manager performs variable expansion. This means that dollar characters ("$") which should not be expanded need to be escaped as "$$". Expansion can also be disabled using --expand-environment=no.

OPTIONS

The following options are understood:

--scope

Create a transient .scope unit instead of the default transient .service unit (see above).

Added in version 206.

--unit=, -u

Use this unit name instead of an automatically generated one.

Added in version 206.

--property=, -p

Sets a property on the scope or service unit that is created. This option takes an assignment in the same format as systemctl(1)'s set-property command.

Added in version 211.

--description=

Provide a description for the service, scope, path, socket, or timer unit. If not

specified, the command itself will be used as a description. See Description= in systemd.unit(5).

Added in version 206.

--slice=

Make the new .service or .scope unit part of the specified slice, instead of system.slice (when running in --system mode) or the root slice (when running in --user mode).

Added in version 206.

--slice-inherit

Make the new .service or .scope unit part of the slice the systemd-run itself has been invoked in. This option may be combined with --slice=, in which case the slice specified via --slice= is placed within the slice the systemd-run command is invoked in.

Example: consider systemd-run being invoked in the slice foo.slice, and the --slice= argument is bar. The unit will then be placed under foo-bar.slice.

Added in version 246.

--expand-environment=BOOL

Expand environment variables in command arguments. If enabled (the default), environment variables specified as "${VARIABLE}" will be expanded in the same way as in commands specified via ExecStart= in units. With --scope, this expansion is performed by systemd-run itself, and in other cases by the service manager that spawns the command. Note that this is similar to, but not the same as variable expansion in bash(1) and other shells.

See systemd.service(5) for a description of variable expansion. Disabling variable expansion is useful if the specified command includes or may include a "$" sign.

Added in version 254.

-r, --remain-after-exit

After the service process has terminated, keep the service around until it is explicitly stopped. This is useful to collect runtime information about the service after it finished running. Also see RemainAfterExit= in systemd.service(5).

Added in version 207.

--send-sighup

When terminating the scope or service unit, send a SIGHUP immediately after SIGTERM. This is useful to indicate to shells and shell-like processes that the connection has

been severed. Also see SendSIGHUP= in systemd.kill(5).

Added in version 207.

--service-type=

Sets the service type. Also see Type= in systemd.service(5). This option has no effect in conjunction with --scope. Defaults to simple.

Added in version 211.

--uid=, --gid=

Runs the service process under the specified UNIX user and group. Also see User= and Group= in systemd.exec(5).

Added in version 211.

--nice=

Runs the service process with the specified nice level. Also see Nice= in systemd.exec(5).

Added in version 211.

--working-directory=

Runs the service process with the specified working directory. Also see WorkingDirectory= in systemd.exec(5).

Added in version 240.

--same-dir, -d

Similar to --working-directory=, but uses the current working directory of the caller for the service to execute.

Added in version 240.

--root-directory=

Runs the service process with the specified root directory. Also see RootDirectory= in systemd.exec(5).

Note that the path is looked up inside the file system namespace that systemd-run is running in, which might be different that the file system namespace the manager process is running in. Use the RootDirectory= property directly if you want the path to be looked up in the manager process's file system namespace.

Added in version 259.

--same-root-dir, -R

Similar to --root-directory=, but uses the root directory of the systemd-run process as the root directory to execute the service in.

Added in version 259.

-E NAME[=VALUE], --setenv=NAME[=VALUE]

Runs the service process with the specified environment variable set. This parameter may be used more than once to set multiple variables. When "=" and VALUE are omitted, the value of the variable with the same name in the program environment will be used.

Also see Environment= in systemd.exec(5).

Added in version 211.

--pty, -t

When invoking the command, the transient service connects its standard input, output and error to the terminal systemd-run is invoked on, via a pseudo TTY device. This allows running programs that expect interactive user input/output as services, such as interactive command shells.

This option will result in systemd-run synchronously waiting for the transient service to terminate, similar to specifying --wait. If specified along with --wait, systemd-run will not exit when manually disconnecting from the pseudo TTY device.

Note that machinectl(1)'s shell command is usually a better alternative for requesting a new, interactive login session on the local host or a local container.

See below for details on how this switch combines with --pipe.

Added in version 219.

--pty-late, -T

Much like --pty, but the PTY communication only begins once the unit start is complete. This effectively means PTY input during ExecStartPre= of a service is not available while using --pty-late, while it is available using --pty, for most service types.

However, if this option is used, access to the invoking TTY is available for password queries (see --no-ask-password below) during unit start-up.

Added in version 258.

--pipe, -P

If specified, standard input, output, and error of the transient service are inherited from the systemd-run command itself. This allows systemd-run to be used within shell pipelines.

Note that this mode is not suitable for interactive command shells and similar, as the service process will not become a TTY controller when invoked on a terminal. Use --pty instead in that case.

When both --pipe and --pty are used in combination the more appropriate option is automatically determined and used. Specifically, when invoked with standard input, output and error connected to a TTY --pty is used, and otherwise --pipe.

This option will result in systemd-run synchronously waiting for the transient service to terminate, similar to specifying --wait.

When this option is used the original file descriptors systemd-run receives are passed to the service processes as-is. If the service runs with different privileges than systemd-run, this means the service might not be able to reopen the passed file descriptors, due to normal file descriptor access restrictions. If the invoked process is a shell script that uses the echo "hello" >/dev/stderr construct for writing messages to stderr, this might cause problems, as this only works if stderr can be reopened. To mitigate this use the construct echo "hello" >&2 instead, which is mostly equivalent and avoids this pitfall.

Added in version 235.

--shell, -S

A shortcut for "--pty --same-dir --wait --collect --service-type=exec $SHELL", i.e. requests an interactive shell in the current working directory, running in service context, accessible with a single switch.

Added in version 240.

--quiet, -q

Suppresses additional informational output while running. This is particularly useful in combination with --pty when it will suppress the initial message explaining how to terminate the TTY connection.

Added in version 219.

-v, --verbose

Display unit log output while running.

Added in version 258.

--on-active=, --on-boot=, --on-startup=, --on-unit-active=, --on-unit-inactive=

Defines a monotonic timer relative to different starting points for starting the specified command. See OnActiveSec=, OnBootSec=, OnStartupSec=, OnUnitActiveSec= and OnUnitInactiveSec= in systemd.timer(5) for details. These options are shortcuts for --timer-property= with the relevant properties. These options may not be combined with --scope or --pty.

Added in version 218.

--on-calendar=

Defines a calendar timer for starting the specified command. See OnCalendar= in systemd.timer(5). This option is a shortcut for --timer-property=OnCalendar=. This option may not be combined with --scope or --pty.

Added in version 218.

--on-clock-change, --on-timezone-change

Defines a trigger based on system clock jumps or timezone changes for starting the specified command. See OnClockChange= and OnTimezoneChange= in systemd.timer(5). These options are shortcuts for --timer-property=OnClockChange=yes and --timer-property=OnTimezoneChange=yes. These options may not be combined with --scope or --pty.

Added in version 242.

--path-property=, --socket-property=, --timer-property=

Sets a property on the path, socket, or timer unit that is created. This option is similar to --property=, but applies to the transient path, socket, or timer unit rather than the transient service unit created. This option takes an assignment in the same format as systemctl(1)'s set-property command. These options may not be combined with --scope or --pty.

Added in version 218.

--no-block

Do not synchronously wait for the unit start operation to finish. If this option is not specified, the start request for the transient unit will be verified, enqueued and systemd-run 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.

Added in version 220.

--wait

Synchronously wait for the transient service to terminate. If this option is specified, the start request for the transient unit is verified, enqueued, and waited for. Subsequently the invoked unit is monitored, and it is waited until it is deactivated again (most likely because the specified command completed). On exit, terse information about the unit's runtime is shown, including total runtime (as well as CPU, memory, IO, and IP accounting data, if the corresponding cgroup accounting settings are enabled) and

the exit code and status of the main process. This output may be suppressed with --quiet. This option may not be combined with --no-block, --scope or the various path, socket, or timer options.

Added in version 232.

-G, --collect

Unload the transient unit after it completed, even if it failed. Normally, without this option, all units that ran and failed are kept in memory until the user explicitly resets their failure state with systemctl reset-failed or an equivalent command. On the other hand, units that ran successfully are unloaded immediately. If this option is turned on the "garbage collection" of units is more aggressive, and unloads units regardless of whether they exited successfully or failed. This option is a shortcut for --property=CollectMode=inactive-or-failed, see the explanation for CollectMode= in systemd.unit(5) for further information.

Added in version 236.

--job-mode=MODE

When queuing a new job, this option controls how to deal with already queued jobs. The option takes the same mode values as systemctl(1)'s --job-mode= option. The default job mode is "fail".

Running --job-mode=help shows a list of available job modes.

Added in version 258.

--ignore-failure

By default, if the specified command fails the invoked unit will be marked failed (though possibly still unloaded, see --collect=, above), and this is reported in the logs. If this switch is specified this is suppressed and any non-success exit status/code of the command is treated as success.

Added in version 256.

--background=COLOR

Change the terminal background color to the specified ANSI color as long as the session lasts. The color specified should be an ANSI X3.64 SGR background color, i.e. strings such as "40", "41", ..., "47", "48;2;...", "48;5;...". See ANSI Escape Code (Wikipedia)[1] for details.

Added in version 256.

--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.

-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.

-C, --capsule=

Execute operation on a capsule. Specify a capsule name to connect to. See capsule@.service(5) for details about capsules.

Added in version 256.

--no-ask-password

Do not query the user for authentication for privileged operations.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--json=MODE

Shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks), "pretty" (for a pretty version of the same, with indentation and line breaks) or "off" (to turn off JSON output, the default).

--no-pager

Do not pipe output into a pager. This currently only applies to --help. (The pager is not started during normal operation.)

Added in version 258.

All command line arguments after the first non-option argument become part of the command line of the launched process.

EXIT STATUS

On success, 0 is returned. If systemd-run failed to start the service, a non-zero return value will be returned. If systemd-run waits for the service to terminate, the return value will be propagated from the service. 0 will be returned on success, including all the cases where systemd considers a service to have exited cleanly, see the discussion of SuccessExitStatus= in systemd.service(5).

EXAMPLES

Example 1. Logging environment variables provided by systemd to services

```
# systemd-run env
Running as unit: run-19945.service
# journalctl -u run-19945.service
Sep 08 07:37:21 bupkis systemd[1]: Starting /usr/bin/env...
Sep 08 07:37:21 bupkis systemd[1]: Started /usr/bin/env.
Sep 08 07:37:21 bupkis env[19948]: PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
Sep 08 07:37:21 bupkis env[19948]: LANG=en_US.UTF-8
Sep 08 07:37:21 bupkis env[19948]: BOOT_IMAGE=/vmlinuz-3.11.0-0.rc5.git6.2.fc20.x86_64
```

Example 2. Limiting resources available to a command

```
# systemd-run -p IOWeight=10 updatedb
```

This command invokes the updatedb(8) tool, but lowers the block I/O weight for it to 10. See systemd.resource-control(5) for more information on the IOWeight= property.

Example 3. Running commands at a specified time

The following command will touch a file after 30 seconds.

```
# date; systemd-run --on-active=30 --timer-property=AccuracySec=100ms /bin/touch /tmp/foo
Mon Dec  8 20:44:24 KST 2014
Running as unit: run-71.timer
Will run service as unit: run-71.service
# journalctl -b -u run-71.timer
```

-- Journal begins at Fri 2014-12-05 19:09:21 KST, ends at Mon 2014-12-08 20:44:54 KST. --

Dec 08 20:44:38 container systemd[1]: Starting /bin/touch /tmp/foo.

Dec 08 20:44:38 container systemd[1]: Started /bin/touch /tmp/foo.

journalctl -b -u run-71.service

-- Journal begins at Fri 2014-12-05 19:09:21 KST, ends at Mon 2014-12-08 20:44:54 KST. --

Dec 08 20:44:48 container systemd[1]: Starting /bin/touch /tmp/foo...

Dec 08 20:44:48 container systemd[1]: Started /bin/touch /tmp/foo.

Example 4. Allowing access to the tty

The following command invokes bash(1) as a service passing its standard input, output and error to the calling TTY.

systemd-run -t --send-sighup bash

Example 5. Start screen as a user service

$ systemd-run --scope --user screen

Running scope as unit run-r14b0047ab6df45bfb45e7786cc839e76.scope.

$ screen -ls

There is a screen on:

492..laptop (Detached)

1 Socket in /var/run/screen/S-fatima.

This starts the screen process as a child of the systemd --user process that was started by user@.service, in a scope unit. A systemd.scope(5) unit is used instead of a systemd.service(5) unit, because screen will exit when detaching from the terminal, and a service unit would be terminated. Running screen as a user unit has the advantage that it is not part of the session scope. If KillUserProcesses=yes is configured in logind.conf(5), the default, the session scope will be terminated when the user logs out of that session.

The user@.service is started automatically when the user first logs in, and stays around as long as at least one login session is open. After the user logs out of the last session, user@.service and all services underneath it are terminated. This behavior is the default, when "lingering" is not enabled for that user. Enabling lingering means that user@.service is started automatically during boot, even if the user is not logged in, and that the service is not terminated when the user logs out.

Enabling lingering allows the user to run processes without being logged in, for example to allow screen to persist after the user logs out, even if the session scope is terminated. In the default configuration, users can enable lingering for themselves:

```
$ loginctl enable-linger
```

Example 6. Variable expansion by the manager

```
$ systemd-run -t echo "<${INVOCATION_ID}>" '<${INVOCATION_ID}>'
    <> <5d0149bfa2c34b79bccb13074001eb20>
```

The first argument is expanded by the shell (double quotes), but the second one is not expanded by the shell (single quotes). echo(1) is called with ["/usr/bin/echo", "<>", "<${INVOCATION_ID}>"] as the argument array, and then systemd(1) generates ${INVOCATION_ID} and substitutes it in the command-line. This substitution could not be done on the client side, because the target ID that will be set for the service is not known before the call is made.

Example 7. Variable expansion and output redirection using a shell

Variable expansion by systemd(1) can be disabled with --expand-environment=no. Disabling variable expansion can be useful if the command to execute contains dollar characters and escaping them would be inconvenient. For example, when a shell is used:

```
$ systemd-run --expand-environment=no -t bash \
    -c 'echo $SHELL $$ >/dev/stdout'
/bin/bash 12345
```

The last argument is passed verbatim to the bash(1) shell which is started by the service unit. The shell expands "$SHELL" to the path of the shell, and "$$" to its process number, and then those strings are passed to the echo built-in and printed to standard output (which, in this case, is connected to the calling terminal).

Example 8. Return value

```
$ systemd-run --user --wait true
$ systemd-run --user --wait -p SuccessExitStatus=11 bash -c 'exit 11'
$ systemd-run --user --wait -p SuccessExitStatus=SIGUSR1 --expand-environment=no \
    bash -c 'kill -SIGUSR1 $$'
```

Those three invocations will succeed, i.e. terminate with an exit code of 0.

SEE ALSO

systemd(1), systemctl(1), systemd.unit(5), systemd.service(5), systemd.scope(5), systemd.slice(5), systemd.exec(5), systemd.resource-control(5), systemd.timer(5), systemd-mount(1), machinectl(1), run0(1)

NOTES

1. ANSI Escape Code (Wikipedia)

https://en.wikipedia.org/wiki/ANSI_escape_code#SGR_(Select_Graphic_Rendition)_parameters