

NAME

systemd-notify – Notify service manager about start-up completion and other daemon status changes

SYNOPSIS

systemd-notify [**OPTIONS...**] [**VARIABLE=VALUE...**]

DESCRIPTION

systemd-notify may be called by daemon scripts to notify the init system about status changes. It can be used to send arbitrary information, encoded in an environment-block-like list of strings. Most importantly, it can be used for start-up completion notification.

This is mostly just a wrapper around **sd_notify()** and makes this functionality available to shell scripts. For details see **sd_notify(3)**.

The command line may carry a list of environment variables to send as part of the status update.

Note that systemd will refuse reception of status updates from this command unless *NotifyAccess=* is set for the service unit this command is called from.

Note that **sd_notify()** notifications may be attributed to units correctly only if either the sending process is still around at the time PID 1 processes the message, or if the sending process is explicitly runtime-tracked by the service manager. The latter is the case if the service manager originally forked off the process, i.e. on all processes that match *NotifyAccess=main* or *NotifyAccess=exec*. Conversely, if an auxiliary process of the unit sends an **sd_notify()** message and immediately exits, the service manager might not be able to properly attribute the message to the unit, and thus will ignore it, even if *NotifyAccess=all* is set for it.

systemd-notify will first attempt to invoke **sd_notify()** pretending to have the PID of the invoking process. This will only succeed when invoked with sufficient privileges. On failure, it will then fall back to invoking it under its own PID. This behaviour is useful in order that when the tool is invoked from a shell script the shell process — and not the **systemd-notify** process — appears as sender of the message, which in turn is helpful if the shell process is the main process of a service, due to the limitations of *NotifyAccess=all* described above.

OPTIONS

The following options are understood:

--ready

Inform the init system about service start-up completion. This is equivalent to **systemd-notify READY=1**. For details about the semantics of this option see **sd_notify(3)**.

--pid=

Inform the init system about the main PID of the daemon. Takes a PID as argument. If the argument is omitted, the PID of the process that invoked **systemd-notify** is used. This is equivalent to **systemd-notify MAINPID=\$PID**. For details about the semantics of this option see **sd_notify(3)**.

--uid=USER

Set the user ID to send the notification from. Takes a UNIX user name or numeric UID. When specified the notification message will be sent with the specified UID as sender, in place of the user the command was invoked as. This option requires sufficient privileges in order to be able manipulate the user identity of the process.

--status=

Send a free-form status string for the daemon to the init systemd. This option takes the status string as argument. This is equivalent to **systemd-notify STATUS=...**. For details about the semantics of this option see **sd_notify(3)**.

--booted

Returns 0 if the system was booted up with systemd, non-zero otherwise. If this option is passed, no message is sent. This option is hence unrelated to the other options. For details about the semantics of this option, see **sd_booted(3)**. An alternate way to check for this state is to call **systemctl(1)** with the **is-system-running** command. It will return "offline" if the system was not booted with systemd.

-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.

EXAMPLE

Example 1. Start-up Notification and Status Updates

A simple shell daemon that sends start-up notifications after having set up its communication channel. During runtime it sends further status updates to the init system:

```
#!/bin/bash

mkfifo /tmp/waldo
systemd-notify --ready --status="Waiting for data..."

while : ; do
    read a < /tmp/waldo
    systemd-notify --status="Processing $a"

    # Do something with $a ...

    systemd-notify --status="Waiting for data..."
done
```

SEE ALSO

systemd(1), systemctl(1), systemd.unit(5), sd_notify(3), sd_booted(3)