NAME

NetworkManager − network management daemon

SYNOPSIS

NetworkManager [OPTIONS...]

DESCRIPTION

The NetworkManager daemon attempts to make networking configuration and operation as painless and automatic as possible by managing the primary network connection and other network interfaces, like Ethernet, Wi−Fi, and Mobile Broadband devices. NetworkManager will connect any network device when a connection for that device becomes available, unless that behavior is disabled. Information about networking is exported via a D−Bus interface to any interested application, providing a rich API with which to inspect and control network settings and operation.

DISPATCHER SCRIPTS

NetworkManager−dispatcher service can execute scripts for the user in response to network events. See **NetworkManager-dispatcher**(8) manual.

OPTIONS

The following options are understood:

−−version | **−V**
Print the NetworkManager software version and exit.

−−help | **−h**
Print NetworkManager's available options and exit.

−−no−daemon | **−n**
Do not daemonize.

−−debug | **−d**
Do not daemonize, and direct log output to the controlling terminal in addition to syslog.

−−pid−file | **−p**
Specify location of a PID file. The PID file is used for storing PID of the running process and prevents running multiple instances.

−−state−file
Specify file for storing state of the NetworkManager persistently. If not specified, the default value of /var/lib/NetworkManager/NetworkManager.state is used.

−−config
Specify configuration file to set up various settings for NetworkManager. If not specified, the default value of /etc/NetworkManager/NetworkManager.conf is used with a fallback to the older 'nm−system−settings.conf' if located in the same directory. See**Netw orkManager.conf**(5) for more information on configuration file.

−−configure−and−quit [initrd]
Quit after all devices reach a stable state. The optional initrd parameter enables mode, where no processes are left running after NetworkManager stops, which is useful for running from an initial ramdisk on rearly boot.

−−plugins
List plugins used to manage system−wide connection settings. This list has preference over plugins specified in the configuration file. See main.plugins setting in **NetworkManager.conf**(5) for supported options.

−−log−level
Sets how much information NetworkManager sends to the log destination (usually syslog's "daemon" facility). By default, only informational, warning, and error messages are logged. See the section on logging in **NetworkManager.conf**(5) for more information.

−−log−domains

A comma−separated list specifying which operations are logged to the log destination (usually syslog). By default, most domains are logging−enabled. See the section on logging in **NetworkManager.conf**(5) for more information.

−−print−config
Print the NetworkManager configuration to stdout and exit.

UDEV PROPERTIES

udev(7) device manager is used for the network device discovery. The following property influences how NetworkManager manages the devices:

NM_UNMANAGED
If set to "1" or "true", the device is configured as unmanaged by NetworkManager. Note that the user still can explicitly overrule this configuration via means like **nmcli device set "$DEVICE" managed yes** or "device*.managed=1" in NetworkManager.conf.

SIGNALS

NetworkManager process handles the following signals:

SIGHUP
The signal causes a reload of NetworkManager's configuration. Note that not all configuration parameters can be changed at runtime and therefore some changes may be applied only after the next restart of the daemon. A SIGHUP also involves further reloading actions, like doing a DNS update and restarting the DNS plugin. The latter can be useful for example when using the dnsmasq plugin and changing its configuration in /etc/NetworkManager/dnsmasq.d. However, it also means this will shortly interrupt name resolution. In the future, there may be further actions added. A SIGHUP means to update NetworkManager configuration and reload everything that is supported. Note that this does not reload connections from disk. For that there is a D−Bus API and nmcli's reload action

SIGUSR1
The signal forces a rewrite of DNS configuration. Contrary to SIGHUP, this does not restart the DNS plugin and will not interrupt name resolution. When NetworkManager is not managing DNS, the signal forces a restart of operations that depend on the DNS configuration (like the resolution of the system hostname via reverse DNS, or the resolution of WireGuard peers); therefore, it can be used to tell NetworkManager that the content of resolv.conf was changed externally. In the future, further actions may be added. A SIGUSR1 means to write out data like resolv.conf, or refresh a cache. It is a subset of what is done for SIGHUP without reloading configuration from disk.

SIGUSR2
The signal has no effect at the moment but is reserved for future use.

An alternative to a signal to reload configuration is the Reload D−Bus call. It allows for more fine−grained selection of what to reload, it only returns after the reload is complete, and it is guarded by PolicyKit.

DEBUGGING

NetworkManager only configures your system. So when your networking setup doesn't work as expected, the first step is to look at your system to understand what is actually configured, and whether that is correct. The second step is to find out how to tell NetworkManager to do the right thing.

You can for example try to **ping** hosts (by IP address or DNS name), look at **ip link show**, **ip address show** and **ip route show**, and look at /etc/resolv.conf for name resolution issues. Also look at the connection profiles that you have configured in NetworkManager (**nmcli connection** and **nmcli connection show "$PROFILE"**) and the configured interfaces (**nmcli device**).

If that does not suffice, look at the logfiles of NetworkManager. NetworkManager logs to syslog, so depending on your system configuration you can call **journalctl** to get the logs. By default, NetworkManager logs are not verbose and thus not very helpful for investigating a problem in detail. You can change the logging level at runtime with **nmcli general logging level TRACE domains ALL**. But usually a better way is to collect full logs from the start, by configuring level=TRACE in NetworkManager.conf. See **NetworkManager.conf**(5) manual. Note that trace logs of NetworkManager are verbose and systemd−journald might rate limit some lines. Possibly disable rate limiting first with the

RateLimitIntervalSec and RateLimitBurst options of journald (see **journald.conf**(5) manual).

/VAR/LIB/NETWORKMANAGER/SECRET_KEY AND /ETC/MACHINE−ID

The identity of a machine is important as various settings depend on it. For example, ipv6.addr−gen−mode=stable and ethernet.cloned−mac−address=stable generate identifiers by hashing the machine's identity. See also the connection.stable−id connection property which is a per−profile seed that gets hashed with the machine identity for generating such addresses and identifiers.

If you backup and restore a machine, the identity of the machine probably should be preserved. In that case, preserve the files /var/lib/NetworkManager/secret_key and /etc/machine−id. On the other hand, if you clone a virtual machine, you probably want that the clone has a different identity. There is already existing tooling on Linux for handling /etc/machine−id (see **machine-id**(5)).

The identity of the machine is determined by the /var/lib/NetworkManager/secret_key. If such a file does not exist, NetworkManager will create a file with random content. To generate a new identity just delete the file and after restart a new file will be created. The file should be read−only to root and contain at least 16 bytes that will be used to seed the various places where a stable identifier is used.

Since 1.16.0, NetworkManager supports a version 2 of secret−keys. For such keys /var/lib/NetworkManager/secret_key starts with ASCII "nm−v2:" followed by at least 32 bytes of random data. Also, recent versions of NetworkManager always create such kinds of secret−keys, when the file does not yet exist. With version 2 of the secret−key, /etc/machine−id is also hashed as part of the generation for addresses and identifiers. The advantage is that you can keep /var/lib/NetworkManager/secret_key stable, and only regenerate /etc/machine−id when cloning a VM.

BUGS

Please report any bugs you find in NetworkManager at the **NetworkManager issue tracker**[1].

SEE ALSO

NetworkManager home page[2], **NetworkManager.conf**(5), **NetworkManager-dispatcher**(8), **nmcli**(1), **nmcli-examples**(7), **nm-online**(1), **nm-settings**(5), **nm-applet**(1), **nm-connection-editor**(1), **udev**(7)

NOTES

1. NetworkManager issue tracker
   https://gitlab.freedesktop.org/NetworkManager/NetworkManager/-/issues
2. NetworkManager home page
   https://networkmanager.dev