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 will execute scripts in the /etc/NetworkManager/dispatcher.d directory or subdirectories in alphabetical order in response to network events. Each script should be a regular executable file owned by root. Furthermore, it must not be writable by group or other, and not setuid.

Each script receives two arguments, the first being the interface name of the device an operation just happened on, and second the action. For device actions, the interface is the name of the kernel interface suitable for IP configuration. Thus it is either VPN_IP_IFACE, DEVICE_IP_IFACE, or DEVICE_IFACE, as applicable. For the *hostname* and *connectivity−change* actions it is always "none".

The actions are:

pre−up
: The interface is connected to the network but is not yet fully activated. Scripts acting on this event must be placed or symlinked into the /etc/NetworkManager/dispatcher.d/pre−up.d directory, and NetworkManager will wait for script execution to complete before indicating to applications that the interface is fully activated.

up
: The interface has been activated.

pre−down
: The interface will be deactivated but has not yet been disconnected from the network. Scripts acting on this event must be placed or symlinked into the /etc/NetworkManager/dispatcher.d/pre−down.d directory, and NetworkManager will wait for script execution to complete before disconnecting the interface from its network. Note that this event is not emitted for forced disconnections, like when carrier is lost or a wireless signal fades. It is only emitted when there is an opportunity to cleanly handle a network disconnection event.

down
: The interface has been deactivated.

vpn−pre−up
: The VPN is connected to the network but is not yet fully activated. Scripts acting on this event must be placed or symlinked into the /etc/NetworkManager/dispatcher.d/pre−up.d directory, and NetworkManager will wait for script execution to complete before indicating to applications that the VPN is fully activated.

vpn−up
: A VPN connection has been activated.

vpn−pre−down
: The VPN will be deactivated but has not yet been disconnected from the network. Scripts acting on this event must be placed or symlinked into the /etc/NetworkManager/dispatcher.d/pre−down.d directory, and NetworkManager will wait for script execution to complete before disconnecting the VPN from its network. Note that this event is not emitted for forced disconnections, like when the VPN terminates unexpectedly or general connectivity is lost. It is only emitted when there is an opportunity to cleanly handle a VPN disconnection event.

vpn−down
A VPN connection has been deactivated.

hostname
The system hostname has been updated. Use gethostname(2) to retrieve it. The interface name (first argument) is empty and no environment variable is set for this action.

dhcp4−change
The DHCPv4 lease has changed (renewed, rebound, etc).

dhcp6−change
The DHCPv6 lease has changed (renewed, rebound, etc).

connectivity−change
The network connectivity state has changed (no connectivity, went online, etc).

The environment contains more information about the interface and the connection. The following variables are available for the use in the dispatcher scripts:

NM_DISPATCHER_ACTION
The dispatcher action like "up" or "dhcp4−change", identical to the first command line argument. Since NetworkManager 1.12.0.

CONNECTION_UUID
The UUID of the connection profile.

CONNECTION_ID
The name (ID) of the connection profile.

CONNECTION_DBUS_PATH
The NetworkManager D−Bus path of the connection.

CONNECTION_FILENAME
The backing file name of the connection profile (if any).

CONNECTION_EXTERNAL
If "1", this indicates that the connection describes a network configuration created outside of NetworkManager.

DEVICE_IFACE
The interface name of the control interface of the device. Depending on the device type, this differs from *DEVICE_IP_IFACE*. For example for ADSL devices, this could be 'atm0' or for WWAN devices it might be 'ttyUSB0'.

DEVICE_IP_IFACE
The IP interface name of the device. This is the network interface on which IP addresses and routes will be configured.

IP4_ADDRESS_N
The IPv4 address in the format "address/prefix gateway", where N is a number from 0 to (# IPv4 addresses − 1). gateway item in this variable is deprecated, use IP4_GATEWAY instead.

IP4_NUM_ADDRESSES
The variable contains the number of IPv4 addresses the script may expect.

IP4_GATEWAY
The gateway IPv4 address in traditional numbers−and−dots notation.

IP4_ROUTE_N
The IPv4 route in the format "address/prefix next−hop metric", where N is a number from 0 to (# IPv4 routes − 1).

IP4_NUM_ROUTES
The variable contains the number of IPv4 routes the script may expect.

IP4_NAMESERVERS

The variable contains a space−separated list of the DNS servers.

IP4_DOMAINS
The variable contains a space−separated list of the search domains.

DHCP4_<dhcp−option−name>
If the connection used DHCP for address configuration, the received DHCP configuration is passed in the environment using standard DHCP option names, prefixed with "DHCP4_", like "DHCP4_HOST_NAME=foobar".

IP6_<name> and DHCP6_<name>
The same variables as for IPv4 are available for IPv6, but the prefixes are IP6_ and DHCP6_ instead.

CONNECTIVITY_STATE
The network connectivity state, which can take the values defined by the NMConnectivityState type, from the org.freedesktop.NetworkManager D−Bus API: unknown, none, portal, limited or full. Note: this variable will only be set for connectivity−change actions.

In case of VPN, VPN_IP_IFACE is set, and IP4_*, IP6_* variables with VPN prefix are exported too, like VPN_IP4_ADDRESS_0, VPN_IP4_NUM_ADDRESSES.

Dispatcher scripts are run one at a time, but asynchronously from the main NetworkManager process, and will be killed if they run for too long. If your script might take arbitrarily long to complete, you should spawn a child process and have the parent return immediately. Scripts that are symbolic links pointing inside the /etc/NetworkManager/dispatcher.d/no−wait.d/ directory are run immediately, without waiting for the termination of previous scripts, and in parallel. Also beware that once a script is queued, it will always be run, even if a later event renders it obsolete. (Eg, if an interface goes up, and then back down again quickly, it is possible that one or more "up" scripts will be run after the interface has gone down.)

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 **NetworkManager.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. 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

The following environment variables are supported to help debugging. When used in conjunction with the **−−no−daemon** option (thus echoing PPP and DHCP helper output to stdout) these can quickly help pinpoint the source of connection issues. Also see the **−−log−level** and **−−log−domains** to enable debug logging inside NetworkManager itself.

NM_PPP_DEBUG: When set to anything, causes NetworkManager to turn on PPP debugging in pppd, which logs all PPP and PPTP frames and client/server exchanges.

BUGS

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

SEE ALSO

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

NOTES

1. NetworkManager bug tracker
   https://bugzilla.gnome.org/enter_bug.cgi?product=NetworkManager
2. NetworkManager home page
   https://wiki.gnome.org/Projects/NetworkManager