NetworkManager

NetworkManager (http://projects.gnome.org/NetworkManager/) is a program for providing detection and configuration for systems to automatically connect to network. NetworkManager's functionality can be useful for both wireless and wired networks. For wireless networks, NetworkManager prefers known wireless networks and has the ability to switch to the most reliable network. NetworkManager-aware applications can switch from online and offline mode. NetworkManager also prefers wired connections over wireless ones, has support for modem connections and certain types of VPN. NetworkManager was originally developed by Red Hat and now is hosted by the **GNOME** project.

Related articles

Network configuration

Wireless network configuration

Category:Network configuration

Warning: By default, Wi-Fi passwords are stored in clear text. See section **#Encrypted Wi-Fi passwords**

Contents

Installation

NetworkManager can be **installed** with the package **networkmanager (https://www.archlinux.org/packages/?name=networkmanager)**. The package does not include the tray applet *nm-applet* which is part of the **network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)**. It has functionality for basic DHCP support. For full featured DHCP and if you require IPv6 support, **dhclient (https://www.archlinux.org/packages/?name=dhclient)** integrates it.

Note: You must ensure that no other service that wants to configure the network is running; in fact, multiple networking services will conflict. You can find a list of the currently running services with `systemctl --type=service` and then **stop** them. See **#Configuration** to enable the NetworkManager service.

VPN support

NetworkManager VPN support is based on a plug-in system. If you need VPN support via NetworkManager, you have to install one of the following packages:

- **NetworkManager-openconnect** — Connect to Cisco AnyConnect, Juniper VPNs.

https://git.gnome.org/browse/network-manager-openconnect || networkmanager-openconnect (https://www.archlinux.org/packages/?name=networkmanager-openconnect)

- **NetworkManager-openvpn** — Connect to OpenVPN VPNs.

  https://git.gnome.org/browse/network-manager-openvpn || networkmanager-openvpn (https://www.archlinux.org/packages/?name=networkmanager-openvpn)

- **NetworkManager-pptp** — Connect to PPTP VPNs, Microsoft compatible.

  https://git.gnome.org/browse/network-manager-pptp || networkmanager-pptp (https://www.archlinux.org/packages/?name=networkmanager-pptp)

- **NetworkManager-vpnc** — Connect to IPsec VPNs, Cisco compatible.

  https://git.gnome.org/browse/network-manager-vpnc || networkmanager-vpnc (https://www.archlinux.org/packages/?name=networkmanager-vpnc)

- **NetworkManager-strongswan** — Connect to IKEv2 IPsec VPNs with support for EAP, PSK and certificate authentication.

  https://wiki.strongswan.org/projects/strongswan/wiki/NetworkManager || networkmanager-strongswan (https://www.archlinux.org/packages/?name=networkmanager-strongswan)

- **NetworkManager-fortisslvpn** — Connect to Fortinet SSLVPN VPNs.

  https://git.gnome.org/browse/network-manager-fortisslvpn || **networkmanager-fortisslvpn-git (https://aur.archlinux.org/packages/networkmanager-fortisslvpn-git/)**[AUR]

- **NetworkManager-iodine** — Tunnel IP traffic via DNS using Iodine.

  https://honk.sigxcpu.org/piki/projects/network-manager-iodine/ || **networkmanager-iodine-git (https://aur.archlinux.org/packages/networkmanager-iodine-git/)**[AUR]

- **NetworkManager-libreswan** — Connect to IPsec IKEv1 VPNs, Cisco compatible.

  https://git.gnome.org/browse/network-manager-libreswan || **networkmanager-libreswan (https://aur.archlinux.org/packages/networkmanager-libreswan/)**[AUR]

- **NetworkManager-l2tp** — L2TP compatible VPN plugin .

  https://github.com/nm-l2tp/network-manager-l2tp || **networkmanager-l2tp (https://aur.archlinux.org/packages/networkmanager-l2tp/)**[AUR]

- **NetworkManager-ssh** — Connect using OpenSSH's Tunnel capability.

https://github.com/danfruehauf/NetworkManager-ssh || networkmanager-ssh-git (https://aur.archlinux.org/packages/networkmanager-ssh-git/)AUR

- **NetworkManager-sstp** — SSTP compatible VPN plugin.

  http://sstp-client.sourceforge.net/#Network_Manager_Plugin || networkmanager-sstp (https://aur.archlinux.org/packages/networkmanager-sstp/)AUR

Warning: VPN support is unstable (https://bugzilla.gnome.org/buglist.cgi?quicksearch=networkmanager%20vpn), check the daemon processes options set via the GUI correctly and double-check with each package release.[1] (https://bugzilla.gnome.org/show_bug.cgi?id=755350)

PPPoE / DSL support

Install rp-pppoe (https://www.archlinux.org/packages/?name=rp-pppoe) for PPPoE / DSL connection support. To actually add pppoe connection you must use `nm-connection-editor` from the command line and add new DSL/PPPoE connection.

Front-ends

To configure and have easy access to NetworkManager, most users will want to install an applet. This GUI front-end usually resides in the system tray (or notification area) and allows network selection and configuration of NetworkManager. Various desktop environments have their own applet. Otherwise you can use **#nm-applet**.

GNOME

GNOME has a built-in tool, accessible from the Network settings.

KDE Plasma

Install the `plasma-nm (https://www.archlinux.org/packages/?name=plasma-nm)` package.

nm-applet

`network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)` is a GTK+ 3 front-end which works under Xorg environments with a systray.

To store connection secrets install and configure **GNOME/Keyring**.

Be aware that after enabling the tick-box option `Make available to other users` for a connection, NetworkManager stores the password in plain-text, though the respective file is accessible only to root (or other users via `nm-applet`). See **#Encrypted Wi-Fi passwords**.

In order to run `nm-applet` without a systray, you can use **trayer (https://www.archlinux.org/packages/?name=trayer)** or **stalonetray (https://www.archlinux.org/packages/?name=stalonetray)**. For example, you can add a script like this one in your path:

```
nmgui
```

```
#!/bin/sh
nm-applet    2>&1 > /dev/null &
stalonetray  2>&1 > /dev/null
killall nm-applet
```

When you close the *stalonetray* window, it closes `nm-applet` too, so no extra memory is used once you are done with network settings.

The applet can show notifications for events such as connecting to or disconnecting from a WiFi network. For these notifications to display, ensure that you have a notification server installed - see **Desktop notifications**. If you use the applet without a notification server, you might see some messages in stdout/stderr, and the app might hang. See **[2] (https://bugzilla.gnome.org/show_bug.cgi?id=788313)**.

In order to run `nm-applet` with such notifications disabled, start the applet with the following command:

```
$ nm-applet --no-agent
```

Tip: `nm-applet` might be started automatically with a **autostart desktop file**, to add the --no-agent option modify the Exec line there, i.e.

```
Exec=nm-applet --no-agent
```

Appindicator

Appindicator support is available in *nm-applet* however it is not compiled into the official package, see **FS#51740 (https://bugs.archlinux.org/task/51740)**. To use nm-applet in an Appindicator environment, replace **network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)** with **network-manager-applet-indicator (https://aur.archlinux.org/packages/network-manager-applet-indicator/)** AUR and then start the applet with the following command:

```
$ nm-applet --indicator
```

Command line

The following applications can be useful for configuring and managing networks without X.

nmcli

A command line frontend, *nmcli*, is included with **networkmanager (https://www.archlinux.org/packages/?name=networkmanager)**.

For usage information, see **nmcli(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/nmcli.1)**. Examples:

- To connect to a wifi network:

```
nmcli dev wifi connect <SSID> password <password>
```

- To connect to a hidden network:

```
nmcli dev wifi connect <SSID> password <password> hidden yes
```

- To connect to a wifi on the `wlan1` wifi interface:

```
nmcli dev wifi connect <SSID> password <password> iface wlan1 [profile name]
```

- To disconnect an interface:

```
nmcli dev disconnect iface eth0
```

- To reconnect an interface marked as disconnected:

```
nmcli con up uuid <uuid>
```

- To get a list of UUIDs:

```
nmcli con show
```

- To see a list of network devices and their state:

```
nmcli dev
```

- To turn off wifi:

```
nmcli r wifi off
```

nmtui

A curses based graphical frontend, *nmtui*, is included with **networkmanager (https://www.archlinux.org/packages/?name=networkmanager)**.

For usage information, see **nmtui(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/nmtui.1)**.

nmcli-dmenu

Alternatively there is **networkmanager-dmenu-git (https://aur.archlinux.org/packages/networkmanager-dmenu-git/)**[AUR] which is a small script to manage NetworkManager connections with *dmenu* instead of `nm-applet`. It provides all essential features such as connect to existing NetworkManager wifi or wired connections, connect to new wifi connections, requests passphrase if required, connect to existing VPN connections, enable/disable networking, launch *nm-connection-editor* GUI.

Configuration

NetworkManager will require some additional steps to be able run properly. Make sure you have configured `/etc/hosts` as described in **Network configuration#Set the hostname** section.

Enable NetworkManager

NetworkManager is **controlled** with the `NetworkManager.service` **systemd** unit. Once the NetworkManager daemon is started, it will automatically connect to any available "system connections" that have already been configured. Any "user connections" or unconfigured connections will need *nmcli* or an applet to configure and connect.

NetworkManager has a global configuration file at `/etc/NetworkManager/NetworkManager.conf`. Usually no configuration needs to be done to the global defaults.

Enable NetworkManager Wait Online

If you have services which fail if they are started before the network is up, you may use `NetworkManager-wait-online.service` in addition to `NetworkManager.service`. This is, however, rarely necessary because most networked daemons start up okay, even if the network has not been configured yet.

In some cases, the service will still fail to start successfully on boot due to the timeout setting in `/usr/lib/systemd/system/NetworkManager-wait-online.service` being too short. Change the default timeout from 30 to a higher value.

Set up PolicyKit permissions

See **General troubleshooting#Session permissions** for setting up a working session.

With a working session, you have several options for granting the necessary privileges to NetworkManager:

- *Option 1.* Run a **Polkit** authentication agent when you log in, such as `/usr/lib/polkit-gnome/polkit-gnome-authentication-agent-1` (part of **polkit-gnome (https://www.archlinux.org/packages/?name=polkit-gnome)**). You will be prompted for your password whenever you add or remove a network connection.
- *Option 2.* **Add** yourself to the `wheel` group. You will not have to enter your password, but your user account may be granted other permissions as well, such as the ability to use **sudo** without entering the root password.
- *Option 3.* **Add** yourself to the `network` group and create the following file:

```
/etc/polkit-1/rules.d/50-org.freedesktop.NetworkManager.rules
```

```
polkit.addRule(function(action, subject) {
  if (action.id.indexOf("org.freedesktop.NetworkManager.") == 0 && subject.isInGroup("network")) {
    return polkit.Result.YES;
  }
});
```

All users in the `network` group will be able to add and remove networks without a password. This will not work under **systemd** if you do not have an active session with *systemd-logind*.

Network services with NetworkManager dispatcher

There are quite a few network services that you will not want running until NetworkManager brings up an interface. Good examples are **NTPd** and network filesystem mounts of various types (e.g. **netfs**). NetworkManager has the ability to start these services when you connect to a network and stop them when you disconnect. To activate the feature you need to **start** the `NetworkManager-dispatcher.service`.

Once the feature is active, scripts can be added to the `/etc/NetworkManager/dispatcher.d` directory. These scripts must be **owned by root**, otherwise the dispatcher will not execute them. For added security, set group ownership to root as well:

```
# chown root:root scriptname
```

Also, the script must have **write permission for owner only**, otherwise the dispatcher will not execute them:

```
# chmod 755 scriptname
```

The scripts will be run in alphabetical order at connection time, and in reverse alphabetical order at disconnect time. They receive two arguments: the name of the interface (e.g. `eth0`) and the status (*up* or *down* for interfaces and *vpn-up* or *vpn-down* for vpn connections). To ensure what order they come up in, it is common to use numerical characters prior to the name of the script (e.g. `10_portmap` or `30_netfs` (which ensures that the *portmapper* is up before NFS mounts are attempted).

Warning: If you connect to foreign or public networks, be aware of what services you are starting and what servers you expect to be available for them to connect to. You could make a security hole by starting the wrong services while connected to a public network

Avoiding the dispatcher timeout

If the above is working, then this section is not relevant. However, there is a general problem related to running dispatcher scripts which take longer to be executed. Initially an internal timeout of three seconds only was used. If the called script did not complete in time, it was killed. Later the timeout was extended to about 20 seconds (see the **Bugtracker (https://bugzilla.redhat.com/show_bug.cgi?id=982734)** for more information). If the timeout still creates the problem, a work around may be to modify the dispatcher service file `/usr/lib/systemd/system/NetworkManager-dispatcher.service` to remain active after exit:

```
/etc/systemd/system/NetworkManager-dispatcher.service
```

```
.include /usr/lib/systemd/system/NetworkManager-dispatcher.service
[Service]
RemainAfterExit=yes
```

Now start and enable the modified `NetworkManager-dispatcher` service.

Warning: Adding the `RemainAfterExit` line to it will prevent the dispatcher from closing. Unfortunately, the dispatcher **has** to close before it can run your scripts again. With it the dispatcher will not time out but it also will not close, which means that the scripts will only run once per boot. Therefore, do not add the line unless the timeout is definitely causing a problem.

Start OpenNTPD

Install the **networkmanager-dispatcher-openntpd (https://www.archlinux.org/packages/?name=networkmanager-dispatcher-openntpd)** package.

Mount remote folder with sshfs

As the script is run in a very restrictive environment, you have to export `SSH_AUTH_SOCK` in order to connect to your SSH agent. There are different ways to accomplish this, see **this message (https://bbs.archlinux.org/viewtopic.php?pid=1042030#p1042030)** for more information. The example below works with **GNOME Keyring**, and will ask you for the

password if not unlocked already. In case NetworkManager connects automatically on login, it is likely *gnome-keyring* has not yet started and the export will fail (hence the sleep). The `UUID` to match can be found with the command `nmcli con status` or `nmcli con list`.

```
#!/bin/sh
USER='username'
REMOTE='user@host:/remote/path'
LOCAL='/local/path'

interface=$1 status=$2
if [ "$CONNECTION_UUID" = "uuid" ]; then
  case $status in
    up)
      export SSH_AUTH_SOCK=$(find /tmp -maxdepth 1 -type s -user "$USER" -name 'ssh')
      su "$USER" -c "sshfs $REMOTE $LOCAL"
      ;;
    down)
      fusermount -u "$LOCAL"
      ;;
  esac
fi
```

Use dispatcher to automatically toggle Wi-Fi depending on LAN cable being plugged in

The idea is to only turn Wi-Fi on when the LAN cable is unplugged (for example when detaching from a laptop dock), and for Wi-Fi to be automatically disabled, once a LAN cable is plugged in again.

Create the following dispatcher script (**Source (http://superuser.com/questions/233448/disable-wlan-if-wired-cable-network-is-available)**), replacing `LAN_interface` with yours.

```
/etc/NetworkManager/dispatcher.d/wlan_auto_toggle.sh
```

```
#!/bin/sh

if [ "$1" = "LAN_interface" ]; then
    case "$2" in
        up)
            nmcli radio wifi off
            ;;
        down)
            nmcli radio wifi on
            ;;
    esac
fi
```

Note: You can get a list of interfaces using **nmcli**. The ethernet (LAN) interfaces start with `en` , e.g. `enp0s5`

Use dispatcher to connect to a VPN after a network connection is established

In this example we want to connect automatically to a previously defined VPN connection after connecting to a specific Wi-Fi network. First thing to do is to create the dispatcher script that defines what to do after we are connected to the network.

Note: This script will require **wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)** in order to use `iwgetid` .

Create the dispatcher script

```
/etc/NetworkManager/dispatcher.d/vpn-up
```

```
#!/bin/sh
VPN_NAME="name of VPN connection defined in NetworkManager"
```

```
ESSID="Wi-Fi network ESSID (not connection name)"

interface=$1 status=$2
case $status in
  up|vpn-down)
    if iwgetid | grep -qs ":\"$ESSID\""; then
      nmcli con up id "$VPN_NAME"
    fi
    ;;
  down)
    if iwgetid | grep -qs ":\"$ESSID\""; then
      if nmcli con show --active | grep "$VPN_NAME"; then
        nmcli con down id "$VPN_NAME"
      fi
    fi
    ;;
esac
```

If you would like to attempt to automatically connect to VPN for all Wi-Fi networks, you can use the following definition of the ESSID: `ESSID=$(iwgetid -r)`. Remember to set the script's permissions **accordingly**.

Give the script access to VPN password

Trying to connect with the above script may still fail with `NetworkManager-dispatcher.service` complaining about 'no valid VPN secrets', because of **the way VPN secrets are stored (http://developer.gnome.org/NetworkManager/0.9/secrets-flags.html)**. Fortunately, there are different options to give the above script access to your VPN password.

1: One of them requires editing the VPN connection configuration file to make NetworkManager store the secrets by itself rather than inside a keyring **that will be inaccessible for root (https://bugzilla.redhat.com/show_bug.cgi?id=710552)**: open up `/etc/NetworkManager/system-connections/name of your VPN connection` and change the `password-flags` and `secret-flags` from `1` to `0`.

If that alone doesn't work, you may have to create a `passwd-file` in a safe location with the same permissions and ownership as the dispatcher script, containing the following:

```
/path/to/passwd-file
```

```
vpn.secrets.password:YOUR_PASSWORD
```

The script must be changed accordingly, so that it gets the password from the file:

```
/etc/NetworkManager/dispatcher.d/vpn-up
```

```
#!/bin/sh
VPN_NAME="name of VPN connection defined in NetworkManager"
ESSID="Wi-Fi network ESSID (not connection name)"

interface=$1 status=$2
case $status in
  up|vpn-down)
    if iwgetid | grep -qs ":\"$ESSID\""; then
      nmcli con up id "$VPN_NAME" passwd-file /path/to/passwd-file
    fi
    ;;
  down)
    if iwgetid | grep -qs ":\"$ESSID\""; then
      if nmcli con show --active | grep "$VPN_NAME"; then
        nmcli con down id "$VPN_NAME"
      fi
    fi
    ;;
esac
```

2: Alternatively, change the `password-flags` and put the password directly in the configuration file adding the section `vpn-secrets`:

```
[vpn]
....
password-flags=0

[vpn-secrets]
password=your_password
```

Note: It may now be necessary to re-open the NetworkManager connection editor and save the VPN passwords/secrets again.

Use dispatcher to handle mounting of CIFS shares

Some CIFS shares are only available on certain networks or locations (e.g. at home). You can use the dispatcher to only mount CIFS shares that are present at your current location.

The following script will check if we connected to a specific network and mount shares accordingly:

/etc/NetworkManager/dispatcher.d/mount_cifs

```
#!/bin/bash
if [ "$2" = "up" ]; then
  if [ "$CONNECTION_UUID" = "uuid" ]; then
    mount /your/mount/point &
    # add more shares as needed
```

```
  fi
fi
```

Note: You can get a list of uuids using **nmcli**.

The following script will unmount all CIFS before a disconnect from a specific network:

```
/etc/NetworkManager/dispatcher.d/pre-down.d/mount_cifs
```

```
#!/bin/bash
umount -a -l -t cifs
```

Note: Make sure this script is located in the pre-down.d subdirectory as shown above, otherwise it will unmount all shares on any connection state change.

Note: Ever since NetworkManager 0.9.8, the 'pre-down' and 'down' actions are not executed on shutdown or restart, so the above script will only work if you manually disconnect from the network. See **this bug report (https://bugzilla.gnome.org/show_bug.cgi?id=701242)** for more info.

As before, do not forget to set the script permissions **accordingly**.

See also **NFS#NetworkManager dispatcher** for another example script that parses `/etc/fstab` mounts during dispatcher actions.

Proxy settings

NetworkManager does not directly handle proxy settings, but if you are using GNOME or KDE, you could use **proxydriver (http://marin.jb.free.fr/proxydriver/)** wich handles proxy settings using NetworkManager's informations. proxydriver is found in the package **proxydriver (https://aur.archlinux.org/packages/proxydriver/)**[AUR].

In order for *proxydriver* to be able to change the proxy settings, you would need to execute this command, as part of the GNOME startup process (System -> Preferences -> Startup Applications):

```
xhost +si:localuser:your_username
```

See: **Proxy settings**.

Disable NetworkManager

It might not be obvious, but the service automatically starts through *dbus*. To completely disable it you can **mask** the services `NetworkManager` and `NetworkManager-dispatcher`.

Checking connectivity

NetworkManager can try to reach a page on Internet when connecting to a network. **networkmanager (https://www.archlinux.org/packages/?name=networkmanager)** is configured by default in `/usr/lib/NetworkManager/conf.d/20-connectivity.conf` to

check connectivity to archlinux.org. To use a different webserver or disable connectivity checking create `/etc/NetworkManager/conf.d/20-connectivity.conf`, see "connectivity section" in **NetworkManager.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/NetworkManager.conf.5)**.

For those behind a captive portal, the desktop manager can automatically open a window asking for credentials.

Testing

NetworkManager applets are designed to load upon login so no further configuration should be necessary for most users. If you have already disabled your previous network settings and disconnected from your network, you can now test if NetworkManager will work. The first step is to **start** `NetworkManager.service`.

Some applets will provide you with a `.desktop` file so that the NetworkManager applet can be loaded through the application menu. If it does not, you are going to either have to discover the command to use or logout and login again to start the applet. Once the applet is started, it will likely begin polling network connections with for auto-configuration with a DHCP server.

To start the GNOME applet in non-xdg-compliant window managers like **awesome**:

```
nm-applet --sm-disable &
```

For static IP addresses, you will have to configure NetworkManager to understand them. The process usually involves right-clicking the applet and selecting something like 'Edit Connections'.

Troubleshooting

No prompt for password of secured Wi-Fi networks

When trying to connect to a secured Wi-Fi network, no prompt for a password is shown and no connection is established. This happens when no keyring package is installed. An easy solution is to install `gnome-keyring (https://www.archlinux.org/packages/?name=gnome-keyring)`. If you want the passwords to be stored in encrypted form, follow **GNOME Keyring** to set up the *gnome-keyring-daemon*.

No traffic via PPTP tunnel

PPTP connection logins successfully; you see a ppp0 interface with the correct VPN IP address, but you cannot even ping the remote IP address. It is due to lack of MPPE (Microsoft Point-to-Point Encryption) support in stock Arch pppd. It is recommended to first try with the stock Arch `ppp (https://www.archlinux.org/packages/?name=ppp)` as it may work as intended.

To solve the problem it should be sufficient to install the **ppp-mppe (https://aur.archlinux.org/packages/ppp-mppe/)** AUR[**broken link**: archived in **aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/ppp-mppe)**] package.

See also **WPA2 Enterprise#MS-CHAPv2**.

Network management disabled

When NetworkManager shuts down but the pid (state) file is not removed, you will see a `Network management disabled` message. If this happens, remove the file manually:

```
# rm /var/lib/NetworkManager/NetworkManager.state
```

Problems with internal DHCP client

If you have problems with getting an IP address using the internal DHCP client, consider **dhclient (https://www.archlinux.org/packages/?name=dhclient)** as DHCP client.

After installation, update the NetworkManager config file:

```
/etc/NetworkManager/conf.d/dhcp-client.conf
```

```
[main]
dhcp=dhclient
```

This workaround might solve problems in big wireless networks like eduroam.

Customizing resolv.conf

See the main page: **resolv.conf**. If you use **dhclient (https://www.archlinux.org/packages/?name=dhclient)**, you may try the **networkmanager-dispatch-resolv (https://aur.archlinux.org/packages/networkmanager-dispatch-resolv/)** AUR[**broken link**: archived in **aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/networkmanager-dispatch-resolv)**] package.

DHCP problems with dhclient

If you have problems with getting an IP address via DHCP, try to add the following to your `/etc/dhclient.conf`:

```
interface "eth0" {
  send dhcp-client-identifier 01:aa:bb:cc:dd:ee:ff;
}
```

Where `aa:bb:cc:dd:ee:ff` is the MAC address of this NIC. The MAC address can be found using the `ip link show interface` command from the **iproute2 (https://www.archlinux.org/packages/?name=iproute2)** package.

Hostname problems

NetworkManager utilizes **dhclient (https://www.archlinux.org/packages/?name=dhclient)** in default and falls back to its internal DHCP funtionality, if the former is not installed. To make *dhclient* forward the hostname requires to set a non-default option, *dhcpcd* forwards the hostname by default.

First, check which DHCP client is used (*dhclient* in this example):

```
# journalctl -b | egrep "dhc"
```

```
...
Nov 17 21:03:20 zenbook dhclient[2949]: Nov 17 21:03:20 zenbook dhclient[2949]: Bound to *:546
Nov 17 21:03:20 zenbook dhclient[2949]: Listening on Socket/wlan0
Nov 17 21:03:20 zenbook dhclient[2949]: Sending on   Socket/wlan0
Nov 17 21:03:20 zenbook dhclient[2949]: XMT: Info-Request on wlan0, interval 1020ms.
Nov 17 21:03:20 zenbook dhclient[2949]: RCV: Reply message on wlan0 from fe80::126f:3fff:fe0c:2dc.
```

Configure dhclient to push the hostname to the DHCP server

Copy the example configuration file:

```
# cp /usr/share/dhclient/dhclient.conf.example /etc/dhclient.conf
```

Take a look at the file - there will only really be one line we want to keep and *dhclient* will use it's defaults (as it has been using if you did not have this file) for the other options. This is the important line:

```
/etc/dhclient.conf
```

```
send host-name = pick-first-value(gethostname(), "ISC-dhclient");
```

Force an IP address renewal by your favorite means, and you should now see your hostname on your DHCP server.

IPv6 push host name:

```
# cp /usr/share/dhclient/dhclient.conf.example /etc/dhclient6.conf
```

```
/etc/dhclient6.conf
send fqdn.fqdn = pick-first-value(gethostname(), "ISC-dhclient");
```

Configure NetworkManager to use a specific DHCP client

If you want to explicitly set the DHCP client used by NetworkManager, it can be set in the global configuration:

```
/etc/NetworkManager/conf.d/dhcp-client.conf
dhcp=internal
```

The alternative `dhcp=dhclient` is used per default, if this option is not set.

Then **restart** `NetworkManager.service`.

Note: Support for dhcpcd (https://www.archlinux.org/packages/?name=dhcpcd) has been disabled (https://projects.archlinux.org/svntogit/packages.git/commit/trunk?h=packages/networkmanager&id=a1df79cbcebaec0c043789eb31965e57d17b6cdb) in networkmanager (https://www.archlinux.org/packages/?name=networkmanager)-1.0.0-2 (2015-02-14).

Missing default route

On at least one KDE4 system, no default route was created when establishing wireless connections with NetworkManager. Changing the route settings of the wireless connection to remove the default selection "Use only for resources on this connection" solved the issue.

3G modem not detected

See USB 3G Modem#Network Manager.

Switching off WLAN on laptops

Sometimes NetworkManager will not work when you disable your Wi-Fi adapter with a switch on your laptop and try to enable it again afterwards. This is often a problem with *rfkill*. To check if the driver notifies *rfkill* about the wireless adapter's status, use:

```
$ watch -n1 rfkill list all
```

If one identifier stays blocked after you switch on the adapter you could try to manually unblock it with (where X is the number of the identifier provided by the above output):

```
# rfkill event unblock X
```

Static IP address settings revert to DHCP

Due to an unresolved bug, when changing default connections to a static IP address, `nm-applet` may not properly store the configuration change, and will revert to automatic DHCP.

To work around this issue you have to edit the default connection (e.g. "Auto eth0") in `nm-applet`, change the connection name (e.g. "my eth0"), uncheck the "Available to all users" checkbox, change your static IP address settings as desired, and click **Apply**. This will save a new connection with the given name.

Next, you will want to make the default connection not connect automatically. To do so, run `nm-connection-editor` (**not** as root). In the connection editor, edit the default connection (e.g. "Auto eth0") and uncheck "Connect automatically". Click **Apply** and close the connection editor.

Cannot edit connections as normal user

See #Set up PolicyKit permissions.

Forget hidden wireless network

Since hidden networks are not displayed in the selection list of the Wireless view, they cannot be forgotten (removed) with the GUI. You can delete one with the following command:

```
# rm /etc/NetworkManager/system-connections/SSID
```

This works for any other connection.

VPN not working in GNOME

When setting up OpenConnect or vpnc connections in NetworkManager while using GNOME, you will sometimes never see the dialog box pop up and the following error appears in `/var/log/errors.log`:

```
localhost NetworkManager[399]: <error> [1361719690.10506] [nm-vpn-connection.c:1405] get_secrets_cb(): Failed to request VPN secrets #3: (6) No agen
ts were available for this request.
```

This is caused by the GNOME NM Applet expecting dialog scripts to be at `/usr/lib/gnome-shell`, when NetworkManager's packages put them in `/usr/lib/networkmanager`. As a "temporary" fix (this bug has been around for a while now), make the following symlink(s):

- For OpenConnect:
```
ln -s /usr/lib/networkmanager/nm-openconnect-auth-dialog /usr/lib/gnome-shell/
```
- For VPNC (i.e. Cisco VPN):
```
ln -s /usr/lib/networkmanager/nm-vpnc-auth-dialog /usr/lib/gnome-shell/
```

This may need to be done for any other NM VPN plugins as well, but these are the two most common.

Unable to connect to visible European wireless networks

WLAN chips are shipped with a default **regulatory domain**. If your access point does not operate within these limitations, you will not be able to connect to the network. Fixing this is easy:

1. **Install `crda` (https://www.archlinux.org/packages/?name=crda)**
2. Uncomment the correct Country Code in `/etc/conf.d/wireless-regdom`
3. Reboot the system, because the setting is only read on boot

Automatic connect to VPN on boot is not working

The problem occurs when the system (i.e. NetworkManager running as the root user) tries to establish a VPN connection, but the password is not accessible because it is stored in the Gnome keyring of a particular user.

A solution is to keep the password to your VPN in plaintext, as described in step (2.) of **#Use dispatcher to connect to a VPN after a network connection is established**.

You do not need to use the dispatcher described in step (1.) to auto-connect anymore, if you use the new "auto-connect VPN" option from the `nm-applet` GUI.

Systemd Bottleneck

Over time the log files (`/var/log/journal`) can become very large. This can have a big impact on boot performance when using NetworkManager, see: **Systemd#Boot time increasing over time**.

Regular network disconnects, latency and lost packets (WiFi)

NetworkManager does a scan every 2 minutes.

Some WiFi drivers have issues when scanning for base stations whilst connected/associated. Symptoms include VPN disconnects/reconnects and lost packets, web pages failing to load and then refresh fine.

Running `journalctl -f` will indicate that this is taking place, messages like the following will be contained in the logs at regular intervals.

```
NetworkManager[410]: <info>  (wlp3s0): roamed from BSSID 00:14:48:11:20:CF (my-wifi-name) to (none) ((none))
```

There is a patched version of NetworkManager which should prevent this type of scanning: **networkmanager-noscan (https://aur.archlinux.org/packages/networkmanager-noscan/)**[AUR].

Alternatively, if roaming is not important, the periodic scanning behavior can be disabled by locking the BSSID of the access point in the WiFi connection profile.

Unable to turn on wi-fi with Lenovo laptop (IdeaPad, Legion, etc.)

There is an issue with the `ideapad_laptop` module on some Lenovo models due to the wi-fi driver incorrectly reporting a soft block. The card can still be manipulated with `netctl`, but managers like NetworkManager break. You can verify that this is the problem by checking the output of `rfkill list` after toggling your hardware switch and seeing that the soft block persists.

Unloading the `ideapad_laptop` module should fix this. (**warning**: this may disable the laptop keyboard and touchpad also!).

Tips and tricks

Encrypted Wi-Fi passwords

By default, NetworkManager stores passwords in clear text in the connection files at `/etc/NetworkManager/system-connections/`. To print the stored passwords, use the following command:

```
# grep -H '^psk=' /etc/NetworkManager/system-connections/*
```

The passwords are accessible to the root user in the filesystem and to users with access to settings via the GUI (e.g. `nm-applet`).

It is preferable to save the passwords in encrypted form in a keyring instead of clear text. The downside of using a keyring is that the connections have to be set up for each user.

Using Gnome-Keyring

The keyring daemon has to be started and the keyring needs to be unlocked for the following to work.

Furthermore, NetworkManager needs to be configured not to store the password for all users. Using GNOME `nm-applet`, run `nm-connection-editor` from a terminal, select a network connection, click `Edit`, select the `Wifi-Security` tab and click on the right icon of password and check `Store the password only for this user`.

Using KDE Wallet

Using KDE's **kdeplasma-applets-plasma-nm (https://www.archlinux.org/packages/?name=kdeplasma-applets-plasma-nm)**[**broken link**: archived in **aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/kdeplasma-applets-plasma-nm)**], click the applet, click on the top right `Settings` icon, double click on a network connection, in the `General settings` tab, untick `all users may connect to this network`. If the option is ticked, the passwords will still be stored in clear text, even if a keyring daemon is running.

If the option was selected previously and you un-tick it, you may have to use the `reset` option first to make the password disappear from the file. Alternatively, delete the connection first and set it up again.

Sharing internet connection over Wi-Fi

You can share your internet connection (e.g. 3G or wired) with a few clicks using nm. You will need a supported Wi-Fi card (Cards based on Atheros AR9xx or at least AR5xx are probably best choice). Please note that a **firewall** may interfere with internet sharing.

Ad-hoc

- **Install** the **dnsmasq (https://www.archlinux.org/packages/?name=dnsmasq)** package to be able to actually share the connection.
- Custom `dnsmasq.conf` may interfere with NetworkManager (not sure about this, but I think so).

- Click on applet and choose "Create new wireless network".
- Follow wizard (if using WEP, be sure to use 5 or 13 character long password, different lengths will fail).
- Settings will remain stored for the next time you need it.

Real AP

Support of infrastructure mode (which is needed by Android phones as they intentionally do not support ad-hoc) is added by NetworkManager as of late 2012.

See **Fedora's wiki (https://fedoraproject.org/wiki/Features/RealHotspot)**.

Sharing internet connection over Ethernet

Scenario: your device has internet connection over wi-fi and you want to share the internet connection to other devices over ethernet.

Requirements:

- **Install** the `dnsmasq (https://www.archlinux.org/packages/?name=dnsmasq)` package to be able to actually share the connection.
- Your internet connected device and the other devices are connected over a suitable ethernet cable (this usually means a cross over cable or a switch in between).
- Internet sharing is not blocked by a **firewall**.

Steps:

- Run `nm-connection-editor` from terminal.
- Add a new ethernet connection.
- Give it some sensible name. For example "Shared Internet"
- Go to "IPv4 Settings".
- For "Method:" select "Shared to other computers".
- Save

Now you should have a new option "Shared Internet" under the Wired connections in NetworkManager.

Checking if networking is up inside a cron job or script

Some *cron* jobs require networking to be up to succeed. You may wish to avoid running these jobs when the network is down. To accomplish this, add an **if** test for networking that queries NetworkManager's *nm-tool* and checks the state of networking. The test shown here succeeds if any interface is up, and fails if they are all down. This is convenient for laptops that might be hardwired, might be on wireless, or might be off the network.

```
if [ $(nm-tool|grep State|cut -f2 -d' ') == "connected" ]; then
    #Whatever you want to do if the network is online
else
    #Whatever you want to do if the network is offline - note, this and the else above are optional
fi
```

This useful for a `cron.hourly` script that runs *fpupdate* for the F-Prot virus scanner signature update, as an example. Another way it might be useful, with a little modification, is to differentiate between networks using various parts of the output from *nm-tool*; for example, since the active wireless network is denoted with an asterisk, you could grep for the network name and then grep for a literal asterisk.

Connect to network with secret on boot

By default, NetworkManager will not connect to networks requiring a secret automatically on boot. This is because it locks such connections to the user who makes it by default, only connecting after they have logged in. To change this, do the following:

1. Right click on the `nm-applet` icon in your panel and select Edit Connections and open the Wireless tab
2. Select the connection you want to work with and click the Edit button
3. Check the boxes “Connect Automatically” and “Available to all users”

Log out and log back in to complete.

Automatically unlock keyring after login

NetworkManager requires access to the login keyring to connect to networks requiring a secret. Under most circumstances, this keyring is unlocked automatically at login, but if it isn't, and NetworkManager isn't connecting on login, you can try the following.

GNOME

Note: The following method is dated and known not to work on at least one machine!

- In `/etc/pam.d/gdm` (or your corresponding daemon in `/etc/pam.d`), add these lines at the end of the "auth" and "session" blocks if they do not exist already:

```
auth            optional        pam_gnome_keyring.so
session         optional        pam_gnome_keyring.so  auto_start
```

- In `/etc/pam.d/passwd`, use this line for the 'password' block:

```
password    optional    pam_gnome_keyring.so
```

Next time you log in, you should be asked if you want the password to be unlocked automatically on login.

SLiM login manager

See **SLiM#Gnome Keyring**.

Troubleshooting

While you may type both values at connection time, `kdeplasma-applets-plasma-nm (https://www.archlinux.org/packages/?name=kdeplasma-applets-plasma-nm)` [broken link: archived in aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/kdeplasma-applets-plasma-nm)] 0.9.3.2-1 and above are capable of retrieving OpenConnect username and password directly from KWallet.

Open "KDE Wallet Manager" and look up your OpenConnect VPN connection under "Network Management|Maps". Click "Show values" and enter your credentials in key "VpnSecrets" in this form (replace *username* and *password* accordingly):

```
form:main:username%SEP%username%SEP%form:main:password%SEP%password
```

Next time you connect, username and password should appear in the "VPN secrets" dialog box.

Ignore specific devices

Sometimes it may be desired that NetworkManager ignores specific devices and does not try to configure addresses and routes for them. You can quickly and easily ignore devices by MAC or interface-name by using the following in `/etc/NetworkManager/NetworkManager.conf` :

```
[keyfile]
unmanaged-devices=mac:00:22:68:1c:59:b1;mac:00:1E:65:30:D1:C4;interface-name:eth0
```

After you have put this in, **restart** NetworkManager, and you should be able to configure interfaces without NetworkManager altering what you have set.

Enable DNS Caching

See **dnsmasq#NetworkManager** to enable the plugin that allows DNS caching using **dnsmasq**.

Configuring MAC Address Randomization

MAC randomization can be used for increased privacy by not disclosing your real MAC address to the network.

NetworkManager supports two types MAC Address Randomization: randomization during scanning, and for network connections. Both modes can be configured by modifying `/etc/NetworkManager/NetworkManager.conf` or by creating a separate configuration file in `/etc/NetworkManager/conf.d` which is recommended since the aforementioned config file may be overwritten by NetworkManager.

Randomization during Wi-Fi scanning is enabled by default, but it may be disabled by adding the following lines to `/etc/NetworkManager/NetworkManager.conf` or a dedicated configuration file under `/etc/NetworkManager/conf.d`. This results in a randomly generated MAC address being used when probing for wireless networks.

```
[device]
wifi.scan-rand-mac-address=no
```

Tip: Disabling MAC address randomization may be needed for stable connection. See **[3] (https://bbs.archlinux.org/viewtopic.php?id=220101)**.

MAC randomization for network connections can be set to different modes for both wireless and ethernet interfaces. See **the Gnome blog post (https://blogs.gnome.org/thaller/2016/08/26/mac-address-spoofing-in-networkmanager-1-4-0/)** for more details on the different modes.

In terms of MAC randomization the most important modes are stable and random. Stable generates a random MAC address when you connect to a new network and associates the two permanently. This means that you will use the same MAC address every time you connect to that network. In contrast, random will generate a new MAC address every time you connect to a network, new or previously known. You can configure the MAC randomization by adding the desired configuration under `/etc/NetworkManager/conf.d` .

```
[device-mac-randomization]
# "yes" is already the default for scanning
wifi.scan-rand-mac-address=yes

[connection-mac-randomization]
# Randomize MAC for every ethernet connection
ethernet.cloned-mac-address=random
# Generate a random MAC for each WiFi and associate the two permanently.
wifi.cloned-mac-address=stable
```

You can read more about it **here (https://blogs.gnome.org/thaller/2016/08/26/mac-address-spoofing-in-networkmanager-1-4-0/)**

Enable IPv6 Privacy Extensions

See **IPv6#NetworkManager**.

Working with wired connections

By default, NetworkManager generates a connection profile for each wired ethernet connection it finds. At the point when generating the connection, it does not know whether there will be more ethernet adapters available. Hence, it calls the first wired connection "Wired connection 1". You can avoid generating this connection, by configuring `no-auto-default` (see **NetworkManager.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/NetworkManager.conf.5)**), or by simply deleting it. Then NetworkManager will remember not to generate a connection for this interface again.

You can also edit the connection (and persist it to disk) or delete it. NetworkManager will not re-generate a new connection. Then you can change the name to whatever you want. You can use something like nm-connection-editor for this task.

See also

- **NetworkManager for Administrators Part 1 (http://blogs.gnome.org/dcbw/2015/02/16/networkmanager-for-administrators-part-1/)**
- **Wikipedia:NetworkManager**

Retrieved from "https://wiki.archlinux.org/index.php?title=NetworkManager&oldid=508623"
