Red Hat Enterprise Linux 8.0 Beta

Configuring and managing networking

A guide to configuring and managing networking in Red Hat Enterprise Linux 8

Last Updated: 2018-12-04

Red Hat Enterprise Linux 8.0 Beta Configuring and managing networking

A guide to configuring and managing networking in Red Hat Enterprise Linux 8

Legal Notice

Abstract

This document describes how to manage networking on Red Hat Enterprise Linux 8. The current version of the document contains only selected preview user stories.

Table of Contents

THIS IS A BETA VERSION!

Thank you for your interest in Red Hat Enterprise Linux 8.0 Beta. Be aware that:

- Beta code should not be used with production data or on production systems.
- Beta does not include a guarantee of support.
- Feedback and bug reports are welcome. Discussions with your account representative, partner contact, and Technical Account Manager (TAM) are also welcome.
- Upgrades to or from a Beta are not supported or recommended.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
    1. Go to the Bugzilla website.
    2. As the Component, use **Documentation**.
    3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
    4. Click **Submit Bug**.

CHAPTER 1. OVERVIEW OF NETWORKING TOPICS

NOTE

The following sections mention some commands to be performed. The commands that need to be entered by the **root** user have # in the prompt, while the commands that can be performed by a regular user, have **$** in their prompt.

1.1. COMPARING IP TO NON-IP NETWORKS

A Network is a system of interconnected devices that can communicate sharing information and resources, such as files, printers, applications, and Internet connection. Each of these devices has a unique Internet Protocol (IP) address to send and receive messages between two or more devices using a set of rules called protocol.

Categories of Network Communication

IP Networks

Networks that communicate through Internet Protocol addresses. An IP network is implemented in the Internet and most internal networks. Ethernet, Cable Modems, DSL Modems, dial up modems, wireless networks, and VPN connections are typical examples.

non-IP Networks

Networks that are used to communicate through a lower layer rather than the transport layer. Note that these networks are rarely used. InfiniBand is a non-IP network.

1.2. COMPARING STATIC TO DYNAMIC IP ADDRESSING

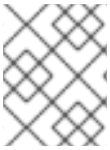
Static IP addressing

When a device is assigned a static IP address, the address does not change over time unless changed manually. It is recommended to use static **IP** addressing if you want:

- To ensure network address consistency for servers such as **DNS**, and authentication servers.
- To use out-of-band management devices that work independently of other network infrastructure.
  All the configuration tools listed in Section 3.1, “Selecting Network Configuration methods” allow assigning static **IP** addresses manually.

Dynamic IP addressing

When a device is assigned a dynamic IP address, the address changes over time. For this reason, it is recommended for devices that connect to the network occasionally because IP address might be changed after rebooting the machine.
Dynamic IP addresses are more flexible, easier to set up and administer. The **Dynamic Host Control Protocol** (**DHCP**) is a traditional method of dynamically assigning network configurations to hosts.



NOTE

There is no strict rule defining when to use static or dynamic IP address. It depends on user’s needs, preferences and the network environment.

1.3. CONFIGURING THE DHCP CLIENT BEHAVIOR

A Dynamic Host Configuration Protocol (DHCP) client requests the dynamic IP address and corresponding configuration information from a DHCP server each time a client connects to the network.

Configuring the DHCP Timeout

When a **DHCP** connection is started, a dhcp client requests an IP address from a **DHCP** server. The time that a dhcp client waits for this request to be completed is 45 seconds by default. This procedure describes how you can configure the **ipv4.dhcp-timeout** property using the **nmcli** tool or the **IPV4_DHCP_TIMEOUT** option in the **/etc/sysconfig/network-scripts/ifcfg-*ifname*** file. For example, using **nmcli**:

```
~]# nmcli connection modify eth1 ipv4.dhcp-timeout 10
```

If an address cannot be obtained during this interval, the IPv4 configuration fails. The whole connection may fail, too, and this depends on the **ipv4.may-fail** property:

- If **ipv4.may-fail** is set to **yes** (default), the state of the connection depends on IPv6 configuration:
  a. If the IPv6 configuration is enabled and successful, the connection is activated, but the IPv4 configuration can never be retried again.
  b. If the IPv6 configuration is disabled or does not get configured, the connection fails.
- If **ipv4.may-fail** is set to **no** the connection is deactivated. In this case:
  a. If the **autoconnect** property of the connection is enabled, **NetworkManager** retries to activate the connection as many times as set in the **autoconnect-retries** property. The default is 4.
  b. If the connection still cannot acquire the dhcp address, auto-activation fails.
  Note that after 5 minutes, the auto-connection process starts again and the dhcp client retries to acquire an address from the dhcp server.

Lease Renewal and Expiration

After a DHCP lease is acquired successfully, **NetworkManager** configures the interface with parameters received from the DHCP server for the given time, and tries to renew the lease periodically. When the lease expires and cannot be renewed, **NetworkManager** continues trying to contact the server up to 8 minutes. If the other IP configuration, either IPv4 or IPv6 is successful, DHCP requests continue as long as the connection is active.

1.3.1. Making DHCPv4 Persistent

To make DHCPv4 persistent both at startup and during the lease renewal processes, set the **ipv4.dhcp-timeout** property either to the maximum for a 32-bit integer (MAXINT32), which is **2147483647**, or to the **infinity** value:

```
~]$ nmcli connection modify eth1 ipv4.dhcp-timeout infinity
```

As a result, **NetworkManager** never stops trying to get or renew a lease from a DHCP server until it is successful.

To ensure a DHCP persistent behavior only during the lease renewal process, you can manually add a static IP to the **IPADDR** property in the **/etc/sysconfig/network-scripts/ifcfg-*ethX*** configuration file or by using **nmcli**:

```
~]$ nmcli connection modify eth0 ipv4.address 192.168.122.88/24
```

When an IP address lease expires, the static IP preserves the IP state as configured or partially configured - you can have an IP address, but you are not connected to the Internet.

Additional Resources

- Section 1.2, “Comparing Static to Dynamic IP Addressing”
- Section 5.1, “Getting started with nmcli”

1.4. SETTING THE WIRELESS REGULATORY DOMAIN

In Red Hat Enterprise Linux, the **crda** package contains the Central Regulatory Domain Agent that provides the kernel with the wireless regulatory rules for a given jurisdiction. It is used by certain **udev** scripts and should not be run manually unless debugging **udev** scripts. The kernel runs **crda** by sending a **udev** event upon a new regulatory domain change. Regulatory domain changes are triggered by the Linux wireless subsystem (IEEE-802.11). This subsystem uses the **regulatory.bin** file to keep its regulatory database information.

The **setregdomain** utility sets the regulatory domain for your system. **Setregdomain** takes no arguments and is usually called through system script such as **udev** rather than manually by the administrator. If a country code look-up fails, the system administrator can define the **COUNTRY** environment variable in the **/etc/sysconfig/regdomain** file.

Additional resources

See the following man pages for more information about the regulatory domain:

- **setregdomain(1)** man page — Sets regulatory domain based on country code.
- **crda(8)** man page — Sends to the kernel a wireless regulatory domain for a given ISO or IEC 3166 alpha2.
- **regulatory.bin(5)** man page — Shows the Linux wireless regulatory database.
- **iw(8)** man page — Shows or manipulates wireless devices and their configuration.

1.5. CONFIGURING NETCONSOLE

The netconsole kernel module enables logging of kernel messages over the network to another computer. It allows kernel debugging in the cases where disk logging fails or using the serial console is not possible.

Prerequisites

It is necessary to have a rsyslog server on the network with the rsyslogd daemon listening on the 514/udp port.

Procedure

1. Configure the **rsyslogd** daemon to listen on the 514/udp port and receive messages from the network by uncommenting the following lines in the **MODULES** section of the

/etc/rsyslog.conf file:

```
$ModLoad imudp
$UDPServerRun 514
```

2. Restart the **rsyslogd** service for the changes to take effect:

```
]# systemctl restart rsyslog
```

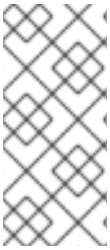
3. Verify that **rsyslogd** is listening on the 514/udp port:

```
]# netstat -l | grep syslog
udp        0      0 0.0.0.0:syslog          0.0.0.0:*
udp6       0      0 [::]:syslog             [::]:*
```

The **0.0.0.0:syslog** and **[::]:syslog** values in the **netstat -l** output mean that **rsyslogd** is listening on default **netconsole** port defined in the **/etc/services** file:

```
]$ cat /etc/services | grep syslog
syslog          514/udp
syslog-conn     601/tcp                 # Reliable Syslog Service
syslog-conn     601/udp                 # Reliable Syslog Service
syslog-tls      6514/tcp                # Syslog over TLS
syslog-tls      6514/udp                # Syslog over TLS
syslog-tls      6514/dccp               # Syslog over TLS
```

Netconsole is configured using the **/etc/sysconfig/netconsole** file, which is a part of the **initscripts** package. This package is installed by default and it also provides the **netconsole** service.



NOTE

By default, **rsyslogd** and **netconsole.service** use port 514. To use a different port, change the following line in **/etc/rsyslog.conf** to the required port number:

```
$UDPServerRun <PORT>
```

Additional Resources

For more information about **netconsole** configuration and troubleshooting tips, see Netconsole Kernel Documentation.

1.6. CONFIGURING A SENDING MACHINE

This procedures describes how to configure a sending machine.

Prerequisites

Section 1.5, “Configuring netconsole”

Procedure

1. Set the value of the **SYSLOGADDR** variable in the **/etc/sysconfig/netconsole** file to match the IP address of the **syslogd** server. For example:

```
SYSLOGADDR=192.168.0.1
```

2. Restart the **netconsole** service for the changes to take effect:

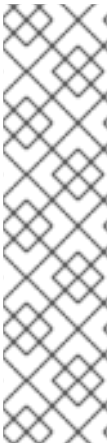
```
]# systemctl restart netconsole.service
```

3. Enable **netconsole.service** to run after rebooting the system:

```
]# systemctl enable netconsole.service
```

4. View the **netconsole** messages from the client in the **/var/log/messages** file (default) or in the file specified in **rsyslog.conf**.

```
]# cat /var/log/messages
```



NOTE

By default, **rsyslogd** and **netconsole.service** use port 514. To use a different port, change the following line in **/etc/rsyslog.conf** to the required port number:

```
$UDPServerRun <PORT>
```

On the sending machine, uncomment and edit the following line in the **/etc/sysconfig/netconsole** file:

```
SYSLOGPORT=514
```

Additional Resources

For more information about **netconsole** configuration and troubleshooting tips, see Netconsole Kernel Documentation.

1.7. USING NETWORK KERNEL TUNABLES WITH SYSCTL

Using certain kernel tunables through the **sysctl** utility, you can adjust network configuration on a running system and directly affect the networking performance.

To change network settings, use the **sysctl** commands. For permanent changes that persist across system restarts, add lines to the **/etc/sysctl.conf** file.

To display a list of all available **sysctl** parameters, enter as **root**:

```
~]# sysctl -a
```

1.8. MANAGING DATA USING THE NCAT UTILITY

The **ncat** networking utility replaces **netcat** in Red Hat Enterprise Linux 7. **ncat** is a reliable back-end tool that provides network connectivity to other applications and users. It reads and writes data across the network from the command line, and uses Transmission Control Protocol (TCP), User Datagram

Protocol (UDP), Stream Control Transmission Protocol (SCTP) or Unix sockets for communication. **ncat** can deal with both **IPv4** and **IPv6**, open connections, send packets, perform port scanning, and supports higher-level features such as **SSL**, and connection broker.

The **nc** command can also be entered as **ncat**, using the identical options. For more information about the **ncat** options, see *the New networking utility (ncat) section in the Migration Planning Guide* and the ***ncat**(1)* man page.

Installing ncat

To install the **ncat** package, enter as **root**:

```
~]# yum install nmap-ncat
```

Brief Selection of ncat Use Cases

Example 1.1. Enabling Communication between a Client and a Server

1. Set a client machine to listen for connections on TCP port *8080*:

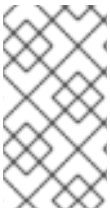
   ```
   ~]$ ncat -l 8080
   ```

2. On a server machine, specify the IP address of the client and use the same port number:

   ```
   ~]$ ncat 10.0.11.60 8080
   ```

   You can send messages on either side of the connection and they appear on both local and remote machines.

3. Press **Ctrl+D** to close the TCP connection.



NOTE

To check a UDP port, use the same **nc** commands with the **-u** option. For example:

```
~]$ ncat -u -l 8080
```

Example 1.2. Sending Files

Instead of printing information on the screen, as mentioned in the previous example, you can send all information to a file. For example, to send a file over TCP port *8080* from a client to a server:

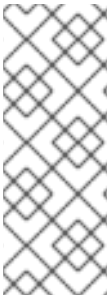
1. On a client machine, to listen a specific port transferring a file to the server machine:

   ```
   ~]$  ncat -l 8080 > outputfile
   ```

2. On a server machine, specify the IP address of the client, the port and the file which is to be transferred:

   ```
   ~]$  ncat -l 10.0.11.60 8080 < inputfile
   ```

After the file is transferred, the connection closes automatically.



NOTE

You can transfer a file in the other direction as well:

```
~]$  ncat -l 8080 < inputfile
```

```
~]$  ncat -l 10.0.11.60 8080 > outputfile
```

Example 1.3. Creating an HTTP proxy server

To create an HTTP proxy server on localhost port *8080*:

```
~]$  ncat -l --proxy-type http localhost 8080
```

Example 1.4. Port Scanning

To view which ports are open, use the **-z** option and specify a range of ports to scan:

```
~]$  ncat -z 10.0.11.60 80-90
    Connection to 192.168.0.1 80 port [tcp/http] succeeded!
```

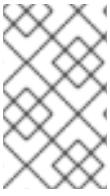
Example 1.5. Setting up Secure Client-Server Communication Using SSL

Set up **SSL** on a server:

```
~]$ ncat -e /bin/bash -k -l 8080 --ssl
```

On a client machine:

```
~]$ ncat --ssl 10.0.11.60 8080
```



NOTE

To ensure true confidentiality of the **SSL** connection, the server requires the **--ssl-cert** and **--ssl-key** options, and the client requires the **--ssl-verify** and **--ssl-trustfile** options.

Additional Resources

For more examples, see the ***ncat****(1)* man page.

CHAPTER 2. GETTING STARTED WITH MANAGING NETWORKING WITH NETWORKMANAGER

2.1. OVERVIEW OF NETWORKMANAGER

Red Hat Enterprise Linux 8 uses the default networking service, **NetworkManager**, which is a dynamic network control and configuration daemon to keep network devices and connections up and active when they are available. The traditional `ifcfg` type configuration files are still supported.

Each network device corresponds to a **NetworkManager** device. The configuration of a network device is completely stored in a single **NetworkManager** connection. You can perform a network configuration applying a **NetworkManager** connection to a **NetworkManager** device.

2.1.1. Benefits of Using NetworkManager

The main benefits of using NetworkManager are:

- Making Network management easier: **NetworkManager** ensures that network connectivity works. When it detects that there is no network configuration in a system but there are network devices, **NetworkManager** creates temporary connections to provide connectivity.
- Providing easy setup of connection to the user: **NetworkManager** offers management through different tools — **GUI, nmtui, nmcli**.
- Supporting configuration flexibility. For example, configuring a WiFi interface, **NetworkManager** scans and shows the available wifi networks. You can select an interface, and **NetworkManager** displays the required credentials providing automatic connection after the reboot process. **NetworkManager** can configure network aliases, IP addresses, static routes, DNS information, and VPN connections, as well as many connection-specific parameters. You can modify the configuration options to reflect your needs.
- Offering an API through D-Bus which allows applications to query and control network configuration and state. In this way, applications can check or configure networking through D-BUS. For example, the `Cockpit` web-based interface, which monitors and configures servers through a web browser, uses the **NetworkManager** D-BUS interface to configure networking.
- Maintaining the state of devices after the reboot process and taking over interfaces which are set into managed mode during restart.
- Handling devices which are not explicitly set unmanaged but controlled manually by the user or another network service.

Additional resources

- Section 2.5, “NetworkManager tools”
- Section 4.1, “Getting started with nmtui”
- Section 5.1, “Getting started with nmcli”]
- For more information on installing and using **Cockpit**, see Getting Started with Cockpit Guide.

2.2. INSTALLING NETWORKMANAGER

NetworkManager is installed by default on Red Hat Enterprise Linux 8 . If it is not, enter as **root**:

```
~]# yum install NetworkManager
```

Additional resources

- Section 2.1, “Overview of NetworkManager”
- Section 2.1.1, “Benefits of Using NetworkManager”

2.3. CHECKING THE STATUS OF NETWORKMANAGER

To check whether **NetworkManager** is running:

```
~]$ systemctl status NetworkManager
NetworkManager.service - Network Manager
   Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled)
   Active: active (running) since Fri, 08 Mar 2013 12:50:04 +0100; 3 days
ago
```

Note that the **systemctl status** command displays **Active: inactive (dead)** when **NetworkManager** is not running.

2.4. STARTING NETWORKMANAGER

To start **NetworkManager**:

```
~]# systemctl start NetworkManager
```

To enable **NetworkManager** automatically at boot time:

```
~]# systemctl enable NetworkManager
```

2.5. NETWORKMANAGER TOOLS

Table 2.1. A Summary of NetworkManager Tools and Applications

Application or Tool	Description
nmcli	A command-line tool which enables users and scripts to interact with **NetworkManager**. Note that **nmcli** can be used on systems without a GUI such as servers to control all aspects of **NetworkManager**. It provides a deeper functionality as GUI tools.
nmtui	A simple curses-based text user interface (TUI) for **NetworkManager**

Application or Tool	Description
nm-connection-editor	A graphical user interface tool for certain tasks not yet handled by the **control-center** utility such as configuring bonds and teaming connections. You can add, remove, and modify network connections stored by **NetworkManager**. To start it, enter **nm-connection-editor** in a terminal.
control-center	A graphical user interface tool provided by the GNOME Shell, available for desktop users. It incorporates a Network settings tool. To start it, press the **Super** key to enter the Activities Overview, type **Network** and then press **Enter**. The Network settings tool appears.
network connection icon	A graphical user interface tool provided by the GNOME Shell representing network connection states as reported by **NetworkManager**. The icon has multiple states that serve as visual indicators for the type of connection you are currently using.

Additional resources

- Section 4.1, “Getting started with nmtui”
- Section 5.1, “Getting started with nmcli”

2.6. RUNNING DISPATCHER SCRIPTS

NetworkManager provides a way to run additional custom scripts to start or stop services based on the connection status. By default, the **/etc/NetworkManager/dispatcher.d/** directory exists and **NetworkManager** runs scripts there, in alphabetical order. Each script must be an executable file **owned by root** and must have **write permission** only for the file owner.

Additional resources

- For more information about running NetworkManager dispatcher scripts, see the Red Hat Knowledgebase solution How to write a NetworkManager dispatcher script to apply ethtool commands.

2.7. USING NETWORKMANAGER WITH SYSCONFIG FILES

The **/etc/sysconfig/** directory is a location for configuration files and scripts. Most network configuration information is stored there, with the exception of VPN, mobile broadband and PPPoE configuration, which are stored in the **/etc/NetworkManager/** subdirectories. For example, interface-specific information is stored in the **ifcfg** files in the **/etc/sysconfig/network-scripts/** directory.

For global settings, use the **/etc/sysconfig/network** file. Information for VPNs, mobile broadband and PPPoE connections is stored in **/etc/NetworkManager/system-connections/**.

In Red Hat Enterprise Linux 8, if you edit an **ifcfg** file, **NetworkManager** is not automatically aware of the change and has to be prompted to notice the change. If you use one of the tools to update **NetworkManager** profile settings, **NetworkManager** does not implement those changes until you reconnect using that profile. For example, if configuration files have been changed using an editor, **NetworkManager** must read the configuration files again.

To ensure this, enter as **root** to reload all connection profiles:

```
~]# nmcli connection reload
```

Alternatively, to reload **only one** changed file, **ifcfg-*ifname***:

```
~]# nmcli con load /etc/sysconfig/network-scripts/ifcfg-ifname
```

Note that you can specify multiple file names using the above command.

To restart the connection after changes are made, use:

```
~]# nmcli con up connection-name
```

2.7.1. Legacy network scripts support

Network scripts are deprecated in Red Hat Enterprise Linux 8 and are no longer provided by default. The basic installation provides a new version of the **ifup** and **ifdown** scripts which call **NetworkManager** through the **nmcli** tool. In Red Hat Enterprise Linux 8, to run the **ifup** and the **ifdown** scripts, **NetworkManager** must be running.

NOTE

Custom commands in **/sbin/ifup-local**, **ifdown-pre-local** and **ifdown-local** scripts are not executed.

If any of these scripts are required, the installation of the deprecated network scripts in the system is still possible with the following command:

```
~]# yum install network-scripts
```

The **ifup** and the **ifdown** scripts link to the installed legacy network scripts.

Calling the legacy network scripts shows a warning about their deprecation.

Additional resources

- **NetworkManager(8)** man page — Describes the network management daemon.
- **NetworkManager.conf(5)** man page — Describes the **NetworkManager** configuration file.
- **/usr/share/doc/initscripts/sysconfig.txt** — Describes **ifcfg** configuration files and their directives as understood by the legacy network service.
- **ifcfg(8)** man page — Describes briefly the **ifcfg** command.

CHAPTER 3. OVERVIEW OF NETWORK CONFIGURATION METHODS

The following section provides an overview of network configuration methods that are available in Red Hat Enterprise Linux 8.

3.1. SELECTING NETWORK CONFIGURATION METHODS

- To configure a network interface using **NetworkManager**, use one of the following tools:
  - the text user interface tool, **nmtui**.
  - the command-line tool, **nmcli**.
  - the graphical user interface tools, **GNOME GUI**.
- To configure a network interface **without** using **NetworkManager**:
  - edit the `ifcfg` files manually.
  - use the `ip` commands. This can be used to assign IP addresses to an interface, but changes are not persistent across reboots; when you reboot, you will lose any changes.
- To configure the network settings when the root filesystem is **not** local:
  - use the kernel command-line.

Additional resources

- Section 4.1, “Getting started with nmtui”
- Section 5.1, “Getting started with nmcli”

CHAPTER 4. CONFIGURING IP NETWORKING WITH NMTUI

The following section provides how you can configure a network interface using the **NetworkManager's** tool, **nmtui**.

4.1. GETTING STARTED WITH NMTUI

nmtui is a simple curses-based text user interface (TUI) for **NetworkManager**.

This procedure describes how to start the text user interface tool, **nmtui**.

Prerequisites

- The **nmtui** tool is used in a terminal window. It is contained in the **NetworkManager-tui** package, but it is not installed along with **NetworkManager** by default. To install **NetworkManager-tui**:

```
~]# yum install NetworkManager-tui
```

- To verify that **NetworkManager** is running, see Section 2.3, "Checking the Status of NetworkManager"

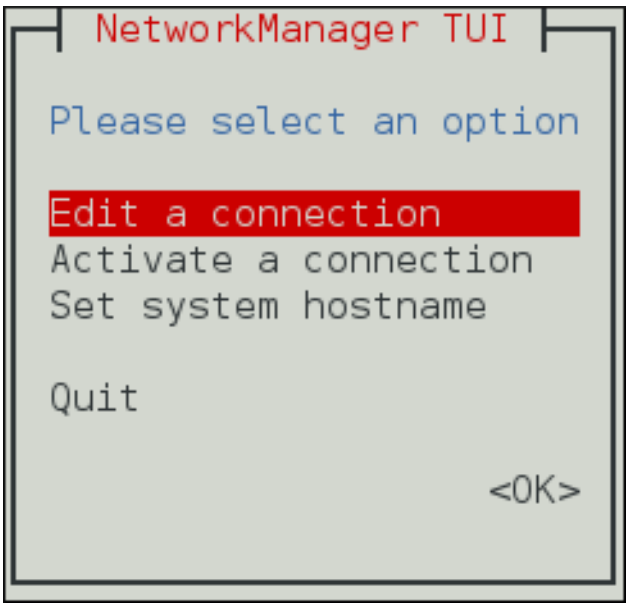
Procedure

1. Start the **nmtui** tool:

```
~]$ nmtui
```

   The text user interface appears.

   Figure 4.1. The NetworkManager Text User Interface starting menu



2. To navigate, use the arrow keys or press **Tab** to step forwards and press **Shift**+**Tab** to step back through the options. Press **Enter** to select an option. The **Space** bar toggles the status of a check box.

4.1.1. Editing a connection with nmtui

Prerequisites

- Section 4.1, “Getting started with nmtui”

To edit a connection using **nmtui**, select the **`Edit a connection`** option in the **`NetworkManager TUI`** menu and press **`Enter`**.

4.1.2. Applying changes to a modified connection with nmtui

To apply changes after a modified connection which is already active requires a reactivation of the connection. In this case, follow the procedure below:

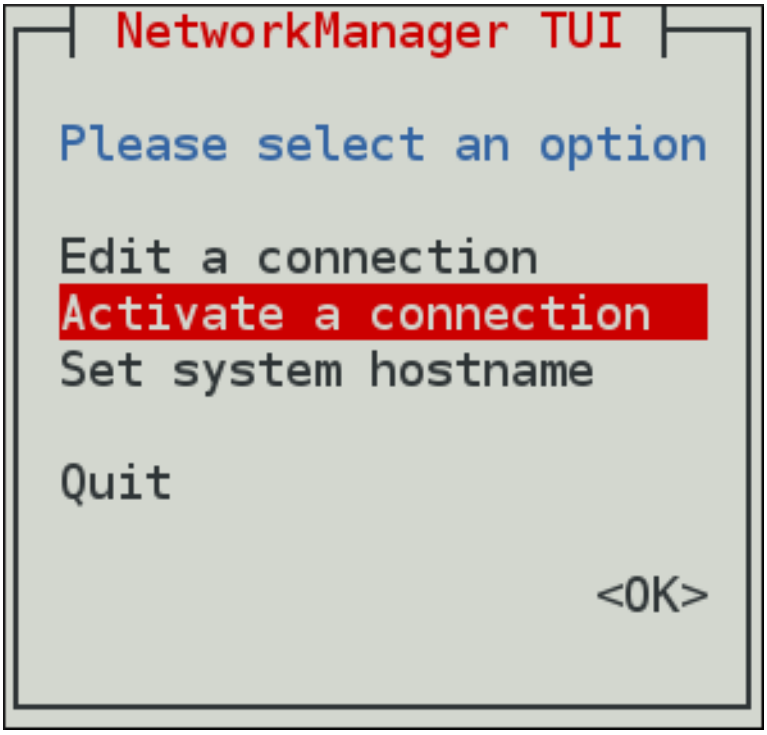
Prerequisites

- Section 4.1, “Getting started with nmtui”

Procedure

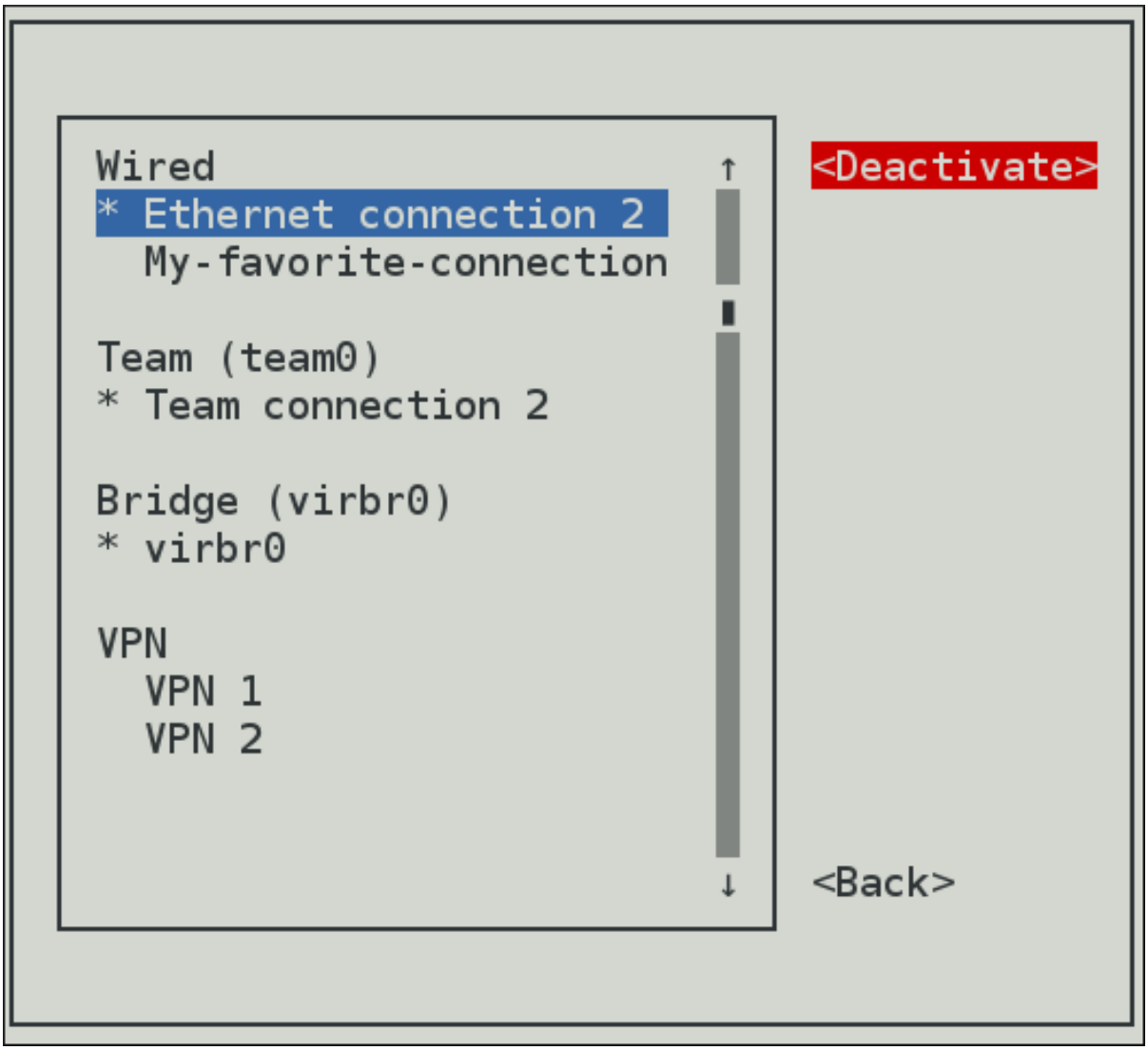
1. Select the **`Activate a connection`** menu entry.

   Figure 4.2. Activating a Connection with nmtui

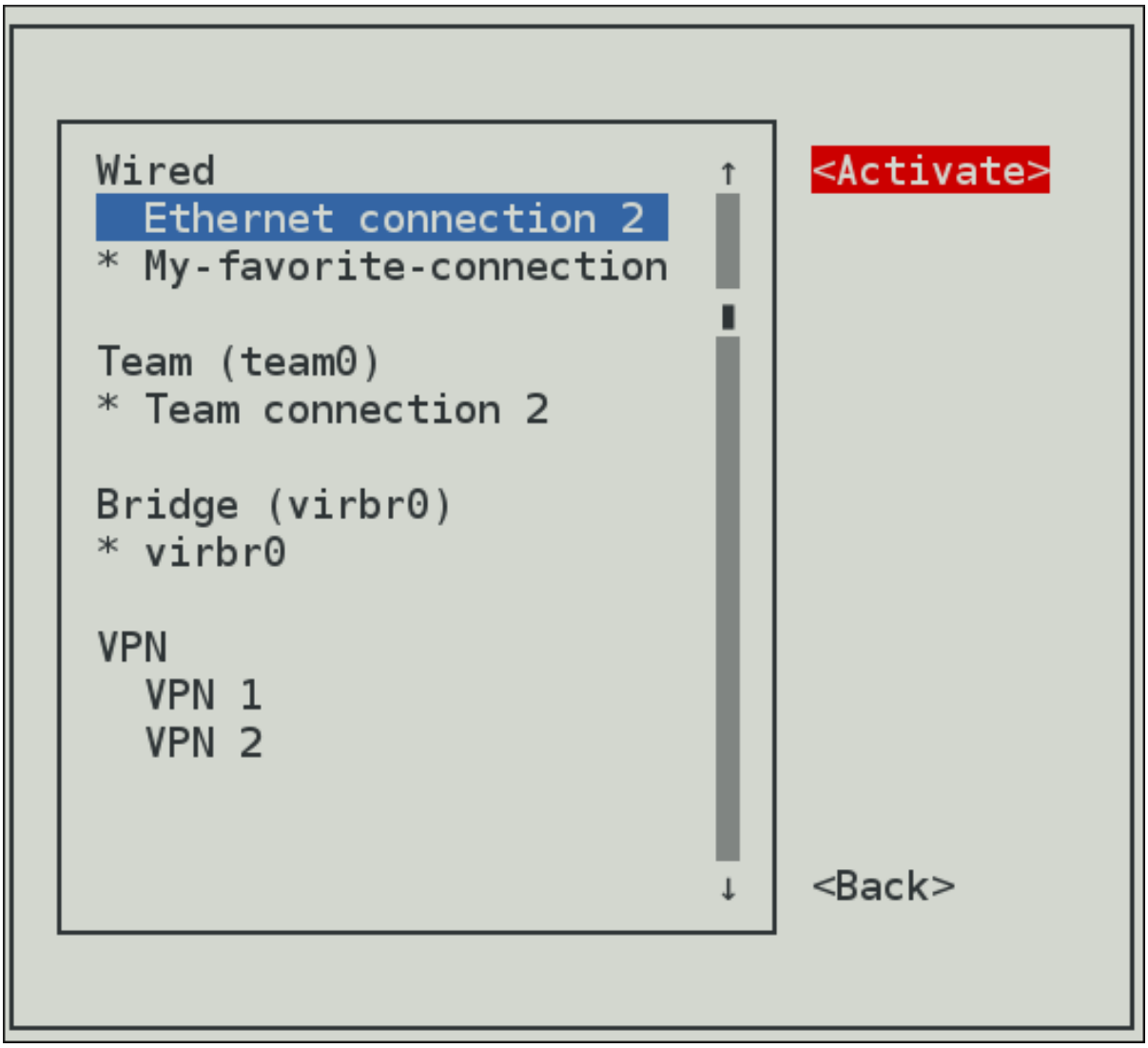
2. Select the modified connection. On the right, click the **`Deactivate`** button.

Figure 4.3. Deactivating a Modified Connection with nmtui



3. Choose the connection again and click the **`Activate`** button.

Figure 4.4. Reactivating a Modified Connection with nmtui



The following commands are also available:

```
~]$ nmtui edit connection-name
```

If no connection name is supplied, the selection menu appears. If the connection name is supplied and correctly identified, the relevant **Edit connection** screen appears.

```
~]$ nmtui connect connection-name
```

If no connection name is supplied, the selection menu appears. If the connection name is supplied and correctly identified, the relevant connection is activated. Any invalid command prints a usage message.

Note that **nmtui** does not support all types of connections. In particular, you cannot edit VPNs, wireless network connections using WPA Enterprise, or Ethernet connections using **802.1X**.

Additional resources

- For more information about the **NetworkManager's** tools, see Section 2.5, "NetworkManager tools"

CHAPTER 5. CONFIGURING NETWORKING WITH NMCLI

The following section provides how you can configure a network interface using **nmcli**.

5.1. GETTING STARTED WITH NMCLI

nmcli (NetworkManager Command Line Interface) is the command-line utility to configure networking through **NetworkManager**. **nmcli** is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status.

The **nmcli** utility can be used by both users and scripts:

- For servers, headless machines, and terminals, **nmcli** can be used to control **NetworkManager** directly, without GUI.
- For scripts, **nmcli** supports options to change the output to a format better suited for script processing.

Each network device corresponds to a **NetworkManager** device. The configuration of a network device is completely stored in a single **NetworkManager** connection. You can perform a network configuration applying a **NetworkManager** connection to a **NetworkManager** device.

To get started with **nmcli** the most common **nmcli** commands are **`nmcli device`** and **`nmcli connection`**:

- The **`nmcli device`** command lists the available network devices in the system.

To view the **NetworkManager** devices:

```
~]$ nmcli device
```

A device can be:

1. **`managed`** - under the **NetworkManager** control. A **`managed`** device may be **`connected`**, meaning that it is activated and configured, or **`disconnected`**, meaning that it is not configured but ready to be activated again.
2. **`unmanaged`** - **NetworkManager** does not control it.

For more details on setting a **`managed`** or **`unmanaged`** device, see Section 5.4, “Setting a device managed or unmanaged with nmcli”.

The **`nmcli device`** command can take many arguments. Most notable are: **`status`**, **`show`**, **`set`**, **`connect`**, **`disconnect`**, **`modify`**, **`delete`**, **`wifi`**. Enter the **`nmcli device help`** command to see the full list.

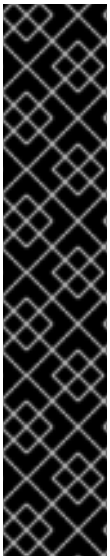
- The **`nmcli connection`** command lists the available connection profiles in **NetworkManager**.

To view the **NetworkManager** connections:

```
~]$ nmcli connection
```

Every connection that is active is displayed as green on top of the list. The inactive connections are displayed as white. The DEVICE field identifies the device on which the connection is applied on.

The **nmcli connection** command can take many arguments to manage connection profiles. Most notable are: **show**, **up**, **down**, **add**, **modify**, **delete**. Enter the **nmcli connection help** command to see the full list.



IMPORTANT

If you use the **nmcli** commands, it is recommended to type a partial **nmcli** command, and then press the **Tab** key to auto-complete the command sequence. If multiple completions are possible, then **Tab** lists them all. This helps users to type commands faster and easier. To enable the **nmcli** auto-complete feature be sure to install the **bash-completion** package:

```
~]# yum install bash-completion
```

After the package installation, **nmcli auto-complete** will be available next time you login into a console. To activate it also in the current console, enter:

```
~]$ source /etc/profile.d/bash_completion.sh
```

The basic format of using **nmcli** is:

```
nmcli [OPTIONS] OBJECT { COMMAND | help }
```

- where [OPTIONS] can be optional options, such as:

  -t, terse

  This mode can be used for computer script processing as you can see a terse output displaying only the values.

  Example 5.1. Viewing a terse output

  ```
  ~]$ nmcli -t device
  ens3:ethernet:connected:Profile 1
  lo:loopback:unmanaged:
  ```

  -f, field

  This option specifies what fields can be displayed in output. For example, NAME,UUID,TYPE,AUTOCONNECT,ACTIVE,DEVICE,STATE. You can use one or more fields. If you want to use more, do not use space after comma to separate the fields.

  Example 5.2. Specifying Fields in the output

  ```
  ~]$ nmcli -f DEVICE,TYPE device
  DEVICE  TYPE
  ens3    ethernet
  lo      loopback
  ```

  or even better for scripting:

  ```
  ~]$ nmcli -t -f DEVICE,TYPE device
  ens3:ethernet
  lo:loopback
  ```

-p, pretty

This option causes **nmcli** to produce human-readable output. For example, values are aligned and headers are printed.

Example 5.3. Viewing an output in pretty mode

```
~]$ nmcli -p device
====================
  Status of devices
====================
DEVICE  TYPE      STATE      CONNECTION
------------------------------------------------------------
--
ens3    ethernet  connected  Profile 1
lo      loopback  unmanaged  --
```

-h, help

Prints help information.

- where OBJECT can be one of the following options: **general**, **networking**, **radio**, **connection**, **device**, **agent**, and **monitor**.

NOTE

You can use any prefix of the above options in your commands. For example, **nmcli con help**, **nmcli c help**, **nmcli connection help** generate the same output.

- where COMMAND, the required **nmcli** command.
- where *help* is to list available actions related to a specified object:

```
~]$ nmcli OBJECT help
```

For example,

```
~]$ nmcli c help
```

Additional resources

- Section 2.5, “NetworkManager tools”
- the ***nmcli****(1)* man page.
- Section 5.3, “Brief Selection of **nmcli** commands”
- Section 5.5, “Creating a connection profile with nmcli”

5.2. OVERVIEW OF NMCLI PROPERTY NAMES AND ALIASES

Prerequisites

`Property` names are specific names that **NetworkManager** uses to identify a common option. Following are some of the important **nmcli `property`** names:

`connection.type`

A type of a specific connection. Allowed values are: **adsl, bond, bond-slave, bridge, bridge-slave, bluetooth, cdma, ethernet, gsm, infiniband, olpc-mesh, team, team-slave, vlan, wifi, wimax**. Each connection type has type-specific command options. You can see the **`TYPE_SPECIFIC_OPTIONS`** list in the ***nmcli****(1)* man page. For example, a **`gsm`** connection requires the access point name specified in an **`apn`**. A **`wifi`** device requires the service set identifier specified in a **`ssid`**.

`connection.interface-name`

A device name relevant for the connection. For example, *eth0*.

`connection.id`

A name used for the connection profile. If you do not specify a connection name, one will be generated as follows:

```
connection.type -connection.interface-name
```

The **`connection.id`** is the name of a *connection profile* and should not be confused with the interface name which denotes a device (**`wlan0`**, **`ens3`**, **`em1`**). However, users can name the connections after interfaces, but they are not the same thing. There can be multiple connection profiles available for a device. This is particularly useful for mobile devices or when switching a network cable back and forth between different devices. Rather than edit the configuration, create different profiles and apply them to the interface as needed. The **`id`** option also refers to the connection profile name.

The most important options for **nmcli** commands such as **show**, **up**, **down** are:

`id`

An identification string assigned by the user to a connection profile. Id can be used in **nmcli** connection commands to identify a connection. The NAME field in the command output always denotes the connection id. It refers to the same connection profile name that the con-name does.

`uuid`

A unique identification string assigned by the system to a connection profile. The **`uuid`** can be used in **`nmcli connection`** commands to identify a connection.

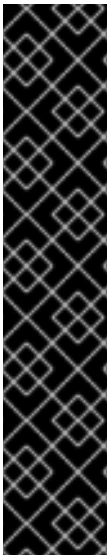
Aliases and Property names

An **`alias`** is an alternative name for a **`property`** name — aliases are translated to properties internally in **nmcli**. **`Aliases`** are more readable but **`property names`** are preferable to use.Both can be used interchangeably.

Alias	Example	Property	Example	Definition

Alias	Example	Property	Example	Definition
type	**type** *bond*	connection.type	**connection.type** *bond*	type of a specific connection. Some of the connection types are: **bond, bridge, ethernet, wifi, infiniband, vlan**
ifname	**ifname** *eth0*	connection.interface-name	**connection.interface-name** *eth0*	name of the device to which a connection belongs to
con-name	**con-name** *"My Connection"*	connection.id	**connection.id** *"My Connection"*	name of a connection

5.3. BRIEF SELECTION OF NMCLI COMMANDS



IMPORTANT

If you use the **nmcli** commands, it is recommended to type a partial **nmcli** command, and then press the **Tab** key to auto-complete the command sequence. If multiple completions are possible, then **Tab** lists them all. This helps users to type commands faster and easier. To enable the **nmcli** auto-complete feature be sure to install the **bash-completion** package:

```
~]# yum install bash-completion
```

After the package installation, **nmcli auto-complete** will be available next time you login into a console. To activate it also in the current console, enter:

```
~]$ source /etc/profile.d/bash_completion.sh
```

The following examples show how to use **nmcli** in specific use cases:

Example 5.4. Viewing all connections

```
~]$ nmcli connection show
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet    ens3
bond0       aaf6eb56-73e5-4746-9037-eed42caa8a65  ethernet    --
```

Example 5.5. Viewing only currently active connections

```
~]$ nmcli connection show --active
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet     ens3
```

Example 5.6. Activating a connection

Use the **up** argument to activate a connection.

```
~]$ nmcli connection show
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet    ens3
bond0       aaf6eb56-73e5-4746-9037-eed42caa8a65  ethernet    --
```

```
~]$ nmcli connection up id bond0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
```

```
~]$ nmcli connection show
NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet    ens3
bond0       aaf6eb56-73e5-4746-9037-eed42caa8a65  ethernet   bond0
```

Example 5.7. Deactivating a specific active connection

Use the **down** argument to deactivate a specific active connection:

```
~]$ nmcli connection down id bond0
```

```
~]$ nmcli connection show
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet    ens3
bond0      aaf6eb56-73e5-4746-9037-eed42caa8a65  ethernet    --
```

Example 5.8. Disconnecting a device preventing it from automatically started again

```
~]$ nmcli device disconnect id bond0
```

NOTE

The **nmcli connection down** command, deactivates a connection from a device without preventing the device from further auto-activation. The **nmcli device disconnect** command, disconnects a device and prevent the device from automatically activating further connections without manual intervention. If the connection has the **connection.autoconnect** flag set to **yes**, the connection automatically starts on the disconnected device again. In this case, use the **nmcli device disconnect** command instead of the **nmcli connection down** command.

Example 5.9. Viewing only devices recognized by NetworkManager and their state

```
~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  Profile 1
lo      loopback  unmanaged  --
```

Example 5.10. Viewing general information for a device

```
~]$ nmcli device show
GENERAL.DEVICE:                         ens3
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         52:54:00:0A:2F:ED
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ens3
[...]
```

Example 5.11. Checking the overall status of NetworkManager

```
~]$ nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

In terse mode:

```
~]$ nmcli -t -f STATE general
connected
```

Example 5.12. Viewing NetworkManager logging status

```
~]$ nmcli general logging
  LEVEL  DOMAINS
  INFO
PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OL
PC,
WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS_PROPS,TEAM,CONCHECK
,DC
B,DISPATCH
```

You can also use the following abbreviations of the **nmcli** commands:

Table 5.1. Abbreviations of some nmcli commands

nmcli command	abbreviation
nmcli general status	nmcli g
nmcli general logging	nmcli g log
nmcli connection show	nmcli con show or nmcli c
nmcli connection show --active	nmcli con show -a or nmcli c -a
nmcli device status	nmcli dev or nmcli d
nmcli device show	nmcli dev show or nmcli d show

Additional resources

- For more information on the comprehensive list of **nmcli** options, see the ***nmcli****(1)* man page.
- For more examples, see the ***nmcli-examples****(5)* man page.
- Section 5.5, “Creating a connection profile with nmcli”

5.4. SETTING A DEVICE MANAGED OR UNMANAGED WITH NMCLI

Prerequisites

- Section 5.1, “Getting started with nmcli”
- Section 5.2, “Overview of nmcli property names and aliases”

To list the currently available network connections:

```
~]$ nmcli con show
NAME              UUID                                  TYPE
DEVICE
Auto Ethernet     9b7f2511-5432-40ae-b091-af2457dfd988  802-3-ethernet  --
ens3              fb157a65-ad32-47ed-858c-102a48e064a2  802-3-ethernet
ens3
MyWiFi            91451385-4eb8-4080-8b82-720aab8328dd  802-11-wireless
wlan0
```

Note that the **NAME** field in the output always denotes the **connection ID** (name). It is not the interface name even though it might look the same. In the second connection shown above, *ens3* in the **NAME** field is the **connection ID** given by the user to the profile applied to the interface *ens3*. In the last connection shown, the user has assigned the connection ID *MyWiFi* to the interface *wlan0*.

Adding an Ethernet connection means creating a configuration profile which is then assigned to a device. Before creating a new profile, review the available devices as follows:

```
~]$ nmcli device status
DEVICE  TYPE      STATE         CONNECTION
```

```
ens3    ethernet  disconnected  --
ens9    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

To set the device unmanaged by the **NetworkManager**:

```
~]$ nmcli device set ifname managed no
```

For example, to set *eth2* unmanaged:

```
~]$ nmcli device status
DEVICE      TYPE      STATE      CONNECTION
bond0       bond      connected  bond0
virbr0      bridge    connected  virbr0
eth1        ethernet  connected  bond-slave-eth1
eth2        ethernet  connected  bond-slave-eth2
eth0        ethernet  unmanaged  --
```

```
~]$ nmcli device set eth2 managed no
```

```
~]$ nmcli device status
DEVICE      TYPE      STATE      CONNECTION
bond0       bond      connected  bond0
virbr0      bridge    connected  virbr0
eth1        ethernet  connected  bond-slave-eth1
eth2        ethernet  unmanaged  --
eth0        ethernet  unmanaged  --
```

NOTE

When you set the device unmanaged, **NetworkManager** does not control it. If the device you want to configure is listed as unmanaged, no **nmcli** command has any effect on this device. However, the device is still connected.

Additional resources

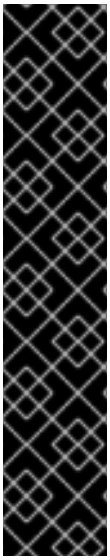
- For more information, see the ***nmcli**(1)* man page.

5.5. CREATING A CONNECTION PROFILE WITH NMCLI

You can create a connection profile to be associated with a device.

Prerequisites

- Section 5.1, “Getting started with nmcli”
- Section 5.2, “Overview of nmcli property names and aliases”



IMPORTANT

If you use the **nmcli** commands, it is recommended to type a partial **nmcli** command, and then press the **Tab** key to auto-complete the command sequence. If multiple completions are possible, then **Tab** lists them all. This helps users to type commands faster and easier. To enable the **nmcli** auto-complete feature be sure to install the **bash-completion** package:

```
~]# yum install bash-completion
```

After the package installation, **nmcli auto-complete** will be available next time you login into a console. To activate it also in the current console, enter:

```
~]$ source /etc/profile.d/bash_completion.sh
```

The basic format to **create** a new profile for **NetworkManager** using **nmcli**:

```
nmcli c add {COMMON_OPTIONS} [IP_OPTIONS]/[NETMASK] [GATEWAY]
```

1. where **{COMMON_OPTIONS}** are the aliases or property names, see Aliases and Property names.
2. where **[IP_OPTIONS]** are the IP addresses:
   - For IPv4 addresses: **ip4**
   - For IPv6 addresses: **ip6**
3. where **[NETMASK]** is the network mask width. For example, **255.255.255.0** is the network mask for the prefix *198.51.100.0*/**24**.
4. where **[GATEWAY]** is the gateway information:
   - For IPv4 addresses: **gw4**
   - For IPv6 addresses: **gw6**

```
nmcli connection add type ethernet con-name connection-name ifname
interface-name ip4 address/network mask gw4 address
```

Example 5.13. Creating a connection profile with an IPv4 address

```
~]$ nmcli c add type ethernet ifname eth0 con-name "My Connection" ip4
192.168.2.100/24 gw4 192.168.2.1
Connection 'new-ens33' (f0c23472-1aec-4e84-8f1b-be8a2ecbeade)
successfully added.
```

To activate the created connection:

```
~]$ nmcli c up _"My Connection"
```

To view the created connection:

```
~]$ nmcli c show "My Connection"
```

Note that the **nmcli c show *con-name*** command displays all the properties present in the connection, even those that are empty or have a default value. If the output is longer than a terminal page, **nmcli** generates a pager to allow an easy navigation on the output. In the pager, use arrows to move up and down and the **q** key to quit.

For a more compact display of the connection, use the **-o** option:

```
~]$ nmcli -o c show "My Connection"
```

The **nmcli -o c show *con-name*** command still displays the connection content, but omits empty properties or those that are set to a default value. This usually results in a shorter output that is more readable.

Additional Resources

- See the **nm-settings(5)** man page for more information on properties and their settings.

5.6. USING THE NMCLI INTERACTIVE CONNECTION EDITOR

The **nmcli** tool has an interactive connection editor. It allows you to change connection parameters according to your needs. To use it:

```
~]$ nmcli con edit
```

You should enter a valid **connection type** from the list displayed. Then, you are able to modify its parameters.

```
~]$ nmcli con edit
Valid connection types: generic, 802-3-ethernet (ethernet), pppoe, 802-11-
wireless (wifi), wimax, gsm, cdma, infiniband, adsl, bluetooth, vpn, 802-
11-olpc-mesh (olpc-mesh), vlan, bond, team, bridge, bond-slave, team-
slave, bridge-slave, no-slave, tun, ip-tunnel, macsec, macvlan, vxlan,
dummy
Enter connection type: ethernet

===| nmcli interactive connection editor |===

Adding a new '802-11-wireless' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-11-wireless (wifi),
802-11-wireless-security (wifi-sec), 802-1x, ipv4, ipv6, proxy
nmcli>
```

It is possible now to edit the **ethernet** connection settings. To get the list of available commands, type **help** or **?**:

```
nmcli> ?
```

```
------------------------------------------------------------------------
--------
---[ Main menu ]---
goto     [<setting> | <prop>]        :: go to a setting or property
remove   <setting>[.<prop>] | <prop> :: remove setting or reset property
value
set      [<setting>.<prop> <value>]  :: set property value
describe [<setting>.<prop>]          :: describe property
print    [all | <setting>[.<prop>]]  :: print the connection
verify   [all | fix]                 :: verify the connection
save     [persistent|temporary]      :: save the connection
activate [<ifname>] [/<ap>|<nsp>]    :: activate the connection
back                                 :: go one level up (back)
help/?   [<command>]                 :: print this help
nmcli    <conf-option> <value>       :: nmcli configuration
quit                                 :: exit nmcli
------------------------------------------------------------------------
--------
nmcli>
```

To exit, enter the **quit** command.

Example 5.14. Adding a new `Ethernet` connection using the nmcli Interactive Connection Editor

```
~]$ nmcli con edit
Valid connection types: generic, 802-3-ethernet (ethernet), pppoe, 802-
11-wireless (wifi), wimax, gsm, cdma, infiniband, adsl, bluetooth, vpn,
802-11-olpc-mesh (olpc-mesh), vlan, bond, team, bridge, bond-slave,
team-slave, bridge-slave, no-slave, tun, ip-tunnel, macsec, macvlan,
vxlan, dummy
Enter connection type: ethernet

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet
(ethernet), 802-1x, dcb, ipv4, ipv6, proxy
nmcli> set connection.id new_eth1
nmcli> set connection.interface-name eth1
nmcli> set connection.autoconnect yes
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an
immediate activation of the connection.
Do you still want to save? (yes/no) [yes] yes
Connection 'new_eth1' (34ac8f9a-e9d8-4e0b-9751-d5dc87cc0467)
successfully saved.
nmcli> quit
```

A new network interface configuration file is created in the **/etc/sysconfig/network-scripts** directory:

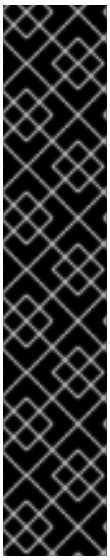
```
~]# ls -lrt /etc/sysconfig/network-scripts/ifcfg*
-rw-r--r--. 1 root root 254 Aug 15  2017 /etc/sysconfig/network-
scripts/ifcfg-lo
-rw-r--r--. 1 root root 304 Apr 26 22:14 /etc/sysconfig/network-
scripts/ifcfg-ens3
-rw-r--r--. 1 root root 266 Aug  6 11:03 /etc/sysconfig/network-
scripts/ifcfg-new_eth1
```

5.7. MODIFYING A CONNECTION PROFILE WITH NMCLI

You can modify the existing configuration of a connection profile.

Prerequisites

- Section 5.1, “Getting started with nmcli”
- Section 5.2, “Overview of nmcli property names and aliases”
- Section 5.5, “Creating a connection profile with nmcli”



IMPORTANT

If you use the **nmcli** commands, it is recommended to type a partial **nmcli** command, and then press the **Tab** key to auto-complete the command sequence. If multiple completions are possible, then **Tab** lists them all. This helps users to type commands faster and easier. To enable the **nmcli** auto-complete feature be sure to install the **bash-completion** package:

```
~]# yum install bash-completion
```

After the package installation, **nmcli auto-complete** will be available next time you login into a console. To activate it also in the current console, enter:

```
~]$ source /etc/profile.d/bash_completion.sh
```

To modify one or more properties of a connection profile, use the following command:

```
nmcli c modify
```

For example, to change the **connection.id** from *"My Connection"* to *"My favorite connection"* and the **connection.interface-name** to *eth1*:

```
~]$ nmcli c modify "My Connection" connection.id "My favorite connection"
connection.interface-name eth1
```

To **apply** changes after a modified connection using **nmcli**, activate again the connection by entering:

```
~]$ [command]~]$nmcli con up "My favorite connection"
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/16)
```

To view the modified connection, enter the **nmcli con show *con-name*** command.

5.8. CONFIGURING A STATIC ETHERNET CONNECTION

5.8.1. Configuring a Static Ethernet Connection with nmcli

Prerequisites

- Section 5.1, “Getting started with nmcli”
- Section 5.5, “Creating a connection profile with nmcli”

Example 5.15. Setting two IPv4 DNS server addresses:

```
~]$ nmcli con mod test-lab ipv4.dns "8.8.8.8 8.8.4.4"
```

Note that this will replace any previously set **DNS** servers.

Alternatively, to set two **IPv6 DNS** server addresses:

```
~]$ nmcli con mod test-lab ipv6.dns "2001:4860:4860::8888
2001:4860:4860::8844"
```

Note that this will replace any previously set **DNS** servers.

To add additional **DNS** servers to any previously set, use the **+** prefix:

```
~]$ nmcli con mod test-lab +ipv4.dns "8.8.8.8 8.8.4.4"
```

```
~]$ nmcli con mod test-lab +ipv6.dns "2001:4860:4860::8888
2001:4860:4860::8844"
```

To activate the new **Ethernet** connection:

```
~]$ nmcli con up test-lab ifname ens9
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/6)
```

To review the status of the devices and connections:

```
~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  my-office
ens9    ethernet  connected  test-lab
lo      loopback  unmanaged  --
```

To view detailed information about the newly configured connection:

```
~]$ nmcli -p con show test-lab
===============================================================================
=====
                      Connection profile details (test-lab)
===============================================================================
=====
connection.id:                          test-lab
connection.uuid:                        05abfd5e-324e-4461-844e-
8501ba704773
connection.interface-name:              ens9
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                   1410428968
connection.read-only:                   no
connection.permissions:
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.secondaries:
connection.gateway-ping-timeout:        0
[output truncated]
```

The use of the **-p, --pretty** option adds a title banner and section breaks to the output.

5.8.2. Configuring a Static Ethernet Connection Using the nmcli Interactive Editor

To configure a static Ethernet connection using the **nmcli** interactive editor:

```
~]$ nmcli con edit type ethernet con-name ens3

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [>setting<.>prop<]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet
(ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> set ipv4.addresses 192.168.122.88/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
nmcli> save temporary
Saving the connection with 'autoconnect=yes'. That might result in an
immediate activation of the connection.
Do you still want to save? [yes] yes
Connection 'ens3' (704a5666-8cbd-4d89-b5f9-fa65a3dbc916) successfully
saved.
nmcli> quit
```

The default action is to save the connection profile as persistent. If required, the profile can be held in memory only, until the next restart, by means of the **save temporary** command.

Additional resources

- See the **nm-settings(5)** man page for more information on properties and their settings.

5.9. CONFIGURING A DYNAMIC ETHERNET CONNECTION

5.9.1. Configuring a Dynamic Ethernet Connection with nmcli

Prerequisites

- Section 5.1, “Getting started with nmcli”
- Section 5.5, “Creating a connection profile with nmcli”

To change the host name sent by a host to a **DHCP** server, modify the **dhcp-hostname** property:

```
~]$ nmcli con modify my-office my-office ipv4.dhcp-hostname host-name
ipv6.dhcp-hostname host-name
```

To change the **IPv4** client ID sent by a host to a **DHCP** server, modify the **dhcp-client-id** property:

```
~]$ nmcli con modify my-office my-office ipv4.dhcp-client-id client-ID-
string
```

There is no **dhcp-client-id** property for **IPv6**, **dhclient** to create an identifier for **IPv6**. See the **dhclient(8)** man page for details.

To ignore the **DNS** servers sent to a host by a **DHCP** server, modify the **ignore-auto-dns** property:

```
~]$ nmcli con modify my-office my-office ipv4.ignore-auto-dns yes
ipv6.ignore-auto-dns yes
```

5.9.2. Configuring a Dynamic Ethernet Connection Using the Interactive Editor

To configure a dynamic Ethernet connection using the interactive editor:

```
~]$ nmcli con edit type ethernet con-name ens3

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet
(ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> describe ipv4.method

=== [method] ===
[NM property description]
IPv4 configuration method.  If 'auto' is specified then the appropriate
automatic method (DHCP, PPP, etc) is used for the interface and most other
properties can be left unset.  If 'link-local' is specified, then a link-
local address in the 169.254/16 range will be assigned to the interface.
```

```
If 'manual' is specified, static IP addressing is used and at least one IP
address must be given in the 'addresses' property.  If 'shared' is
specified (indicating that this connection will provide network access to
other computers) then the interface is assigned an address in the
10.42.x.1/24 range and a DHCP and forwarding DNS server are started, and
the interface is NAT-ed to the current default network connection.
'disabled' means IPv4 will not be used on this connection.  This property
must be set.

nmcli> set ipv4.method auto
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an
immediate activation of the connection.
Do you still want to save? [yes] yes
Connection 'ens3' (090b61f7-540f-4dd6-bf1f-a905831fc287) successfully
saved.
nmcli> quit
```

The default action is to save the connection profile as persistent. If required, the profile can be held in memory only, until the next restart, by means of the **`save temporary`** command.

Additional resources

- See the **`nm-settings(5)`** man page for more information on properties and their settings.

CHAPTER 6. CONFIGURING NETWORKING WITH GNOME GUI

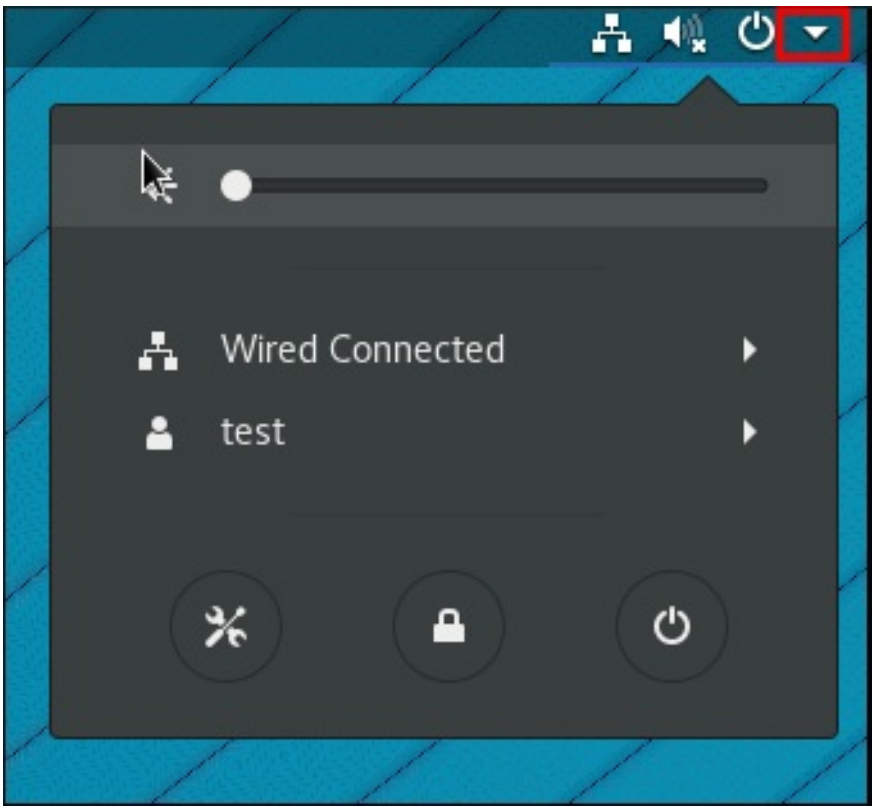
You can configure a network interface using the following **Graphical User Interface** (**GUI**) ways:

- the GNOME Shell **network connection icon** on the top right of the desktop
- the GNOME **control-center** application
- the GNOME **nm-connection-editor** application

6.1. CONNECTING TO A NETWORK USING THE GNOME SHELL NETWORK CONNECTION ICON

To access the **`Network`** settings, click on the GNOME Shell **network connection icon** in the top right-hand corner of the screen to open its menu:

Figure 6.1. The Network Connection Icon Menu



When you click on the GNOME Shell network connection icon, you can see:

- A list of categorized networks you are currently connected to (such as **`Wired`** and **`Wi-Fi`**).
- A list of all **`Available Networks`** that **NetworkManager** has detected. If you are connected to a network, this is indicated on the left of the connection name.
- Options for connecting to any configured Virtual Private Networks (VPNs) and
- An option for selecting the **`Network Settings`** menu entry.

6.2. CREATING A NETWORK CONNECTION USING CONTROL-CENTER

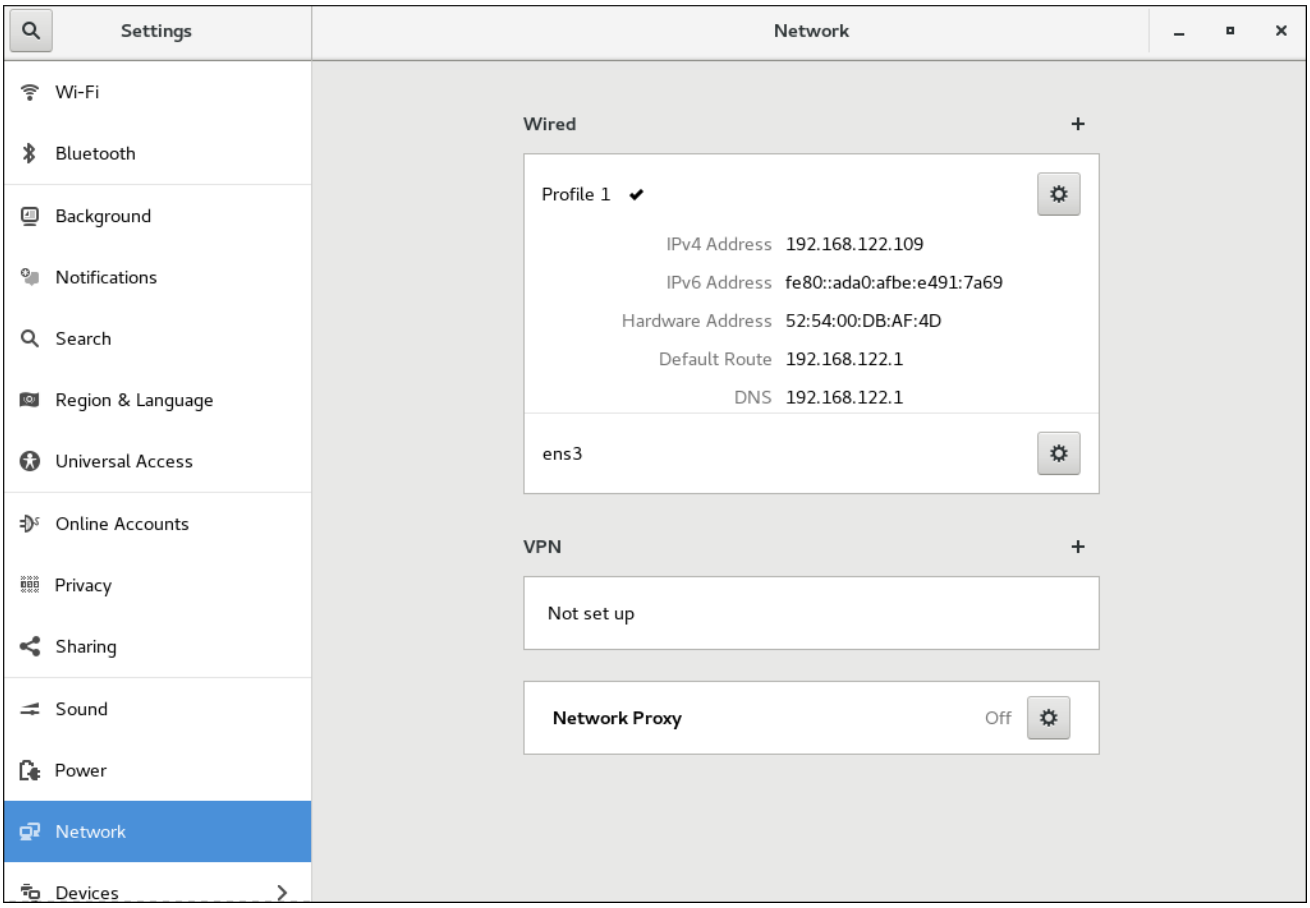
You can create a network connection through the GNOME **control-center** application, which is a graphical user interface that provides a view of available network devices and their current configuration.

This procedures describes how to create a new **`wired`**, **`wireless`**, **`vpn`** connection using **control-center**:

Procedure

1. Press the **`Super`** key to enter the Activities Overview, type **`Settings`**, and press **`Enter`**. Then, select the **`Network`** tab on the left-hand side, and the**`Network`** settings tool appears:

Figure 6.2. Opening the Network Settings Window



1. Click the plus button to add a new connection:
   - **Wired connections**, click the plus button next to **`Wired`** entry and configure the connection.
   - **VPN connections**, click the plus button next to **`VPN`** entry and configure the connection.
   - **Wi-Fi connections**, click the **`Wi-fi`** entry in the **`Settings`** menu and configure the connection.

6.3. CONFIGURING A NETWORK CONNECTION USING CONTROL-CENTER

You can configure a network connection through the GNOME **control-center** application.

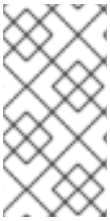
6.3.1. Configuring a Wired (Ethernet) Connection Using control-center

Procedure

1. Press the **Super** key to enter the Activities Overview, type **Settings** and press **Enter**. Then, select the **Network** menu entry on the left-hand side, and the**Network** settings tool appears, see Opening the Network Settings Window

2. Select the **Wired** network interface
   The system creates and configures a single wired *connection profile* called **Wired** by default. More than one profile can be created for an interface and applied as needed. The default profile cannot be deleted but its settings can be changed.

3. Edit the default **Wired** profile by clicking the gear wheel icon to edit an existing connection or click the plus button and then set the configuration options for a new connection.



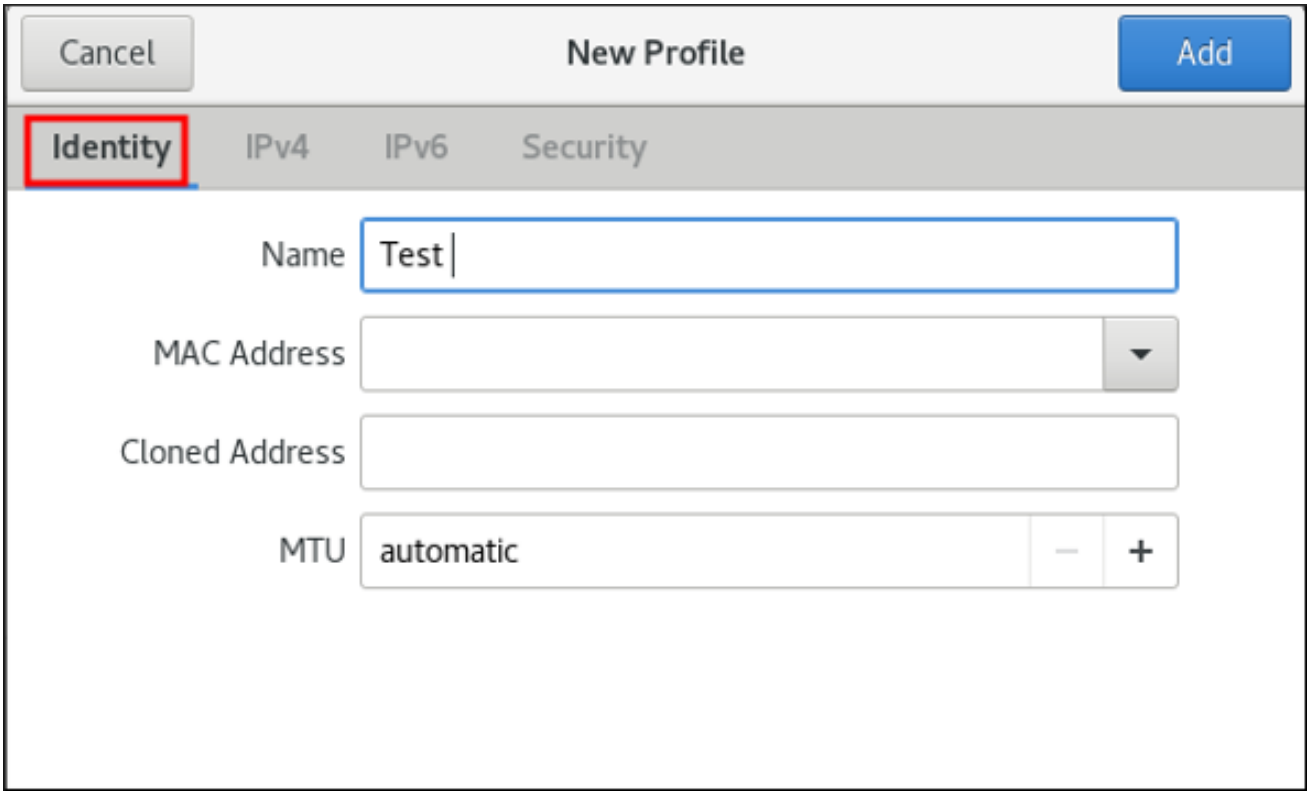
NOTE

When you add a new connection by clicking the plus button, **NetworkManager** creates a new configuration file for that connection and then opens the same dialog that is used for editing an existing connection. The difference between these dialogs is that an existing connection profile has a **Details** menu entry.

Basic Configuration Options

You can see the following configuration settings in the **Wired** dialog, by selecting the **Identity** menu entry:

Figure 6.3. Basic Configuration options of a Wired Connection



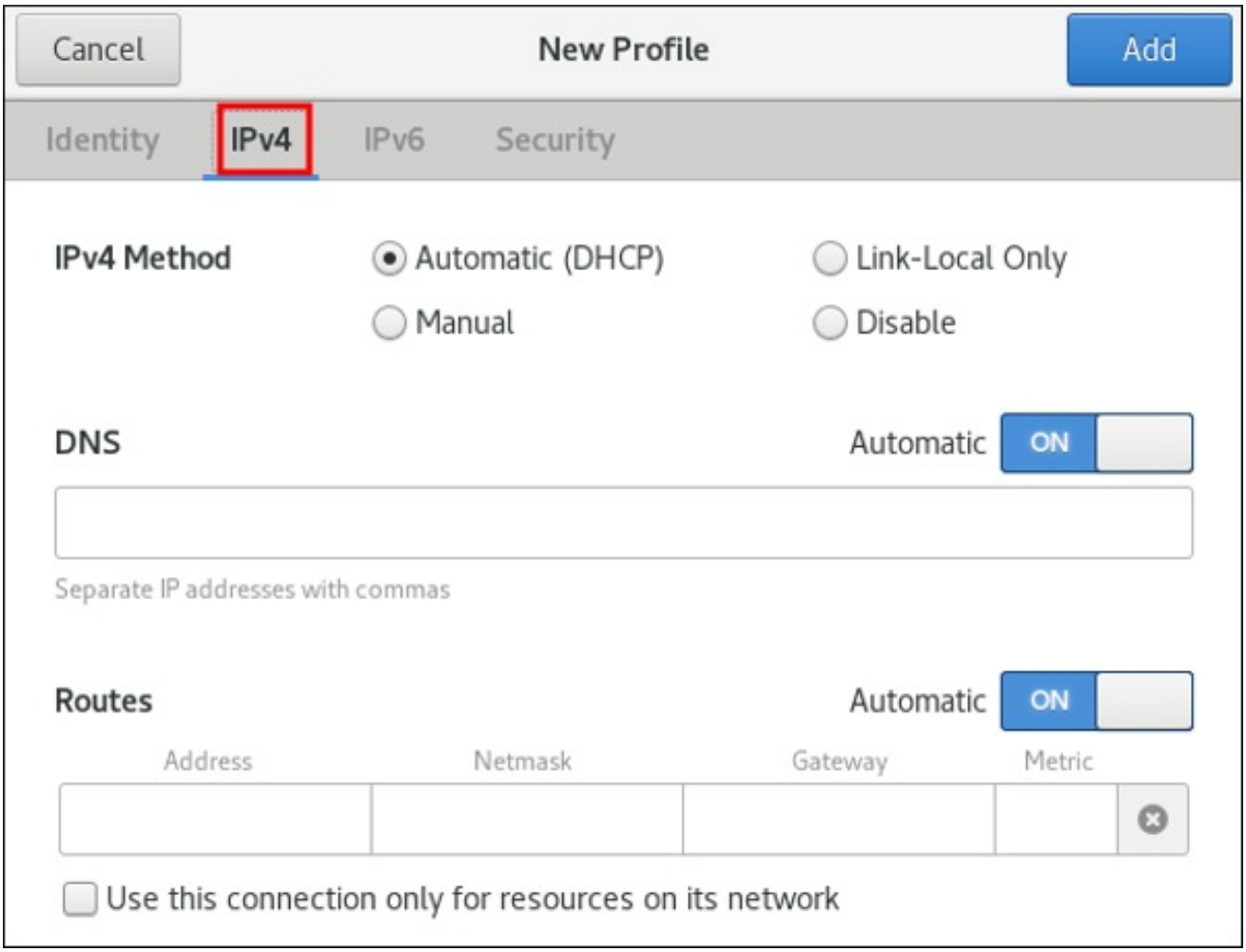
- **Name** — Enter a descriptive name for your network connection. This name will be used to list this connection in the menu of the **Network** window.
- **MAC Address** — Select the MAC address of the interface this profile must be applied to.
- **Cloned Address** — If required, enter a different MAC address to use.

- **MTU** — If required, enter a specific *maximum transmission unit* (**MTU**) to use. The MTU value represents the size in bytes of the largest packet that the link layer will transmit. This value defaults to **1500** and does not generally need to be specified or changed.

Configuring IPv4 Settings for Wired with control-center

You can further configure **IPv4** settings in a wired connection. In the **Wired** dialog, click the **IPv4** menu entry:

Figure 6.4. Configuring IPv4 Settings



The **IPv4** menu entry allows you to configure:

- the **IPv4 Method** used to connect to a network
- **DNS** and
- **Routes**

IPv4 Method

Automatic (DHCP) — Choose this option if the network you are connecting to uses Router Advertisements (RA) or a **DHCP** server to assign dynamic **IP** addresses.

Link-Local Only — Choose this option if the network you are connecting to does not have a **DHCP** server and you do not want to assign **IP** addresses manually. Random addresses will be assigned as per *RFC 3927* with prefix **169.254/16**.

Manual — Choose this option if you want to assign **IP** addresses manually.

Disable — **IPv4** is disabled for this connection.

DNS

In the **DNS** section, when **Automatic** is **ON**. Switch Automatic to **OFF** to enter the IP address of a DNS server you want to use separating the IPs by comma.

Routes

NOTE

In the **Routes** section, when **Automatic** is **ON**, routes from Router Advertisements (RA) or DHCP are used, but you can also add additional static routes. When **OFF**, only static routes are used.

Address — Enter the **IP** address of a remote network, sub-net, or host.

Netmask — The netmask or prefix length of the **IP** address entered above.

Gateway — The **IP** address of the gateway leading to the remote network, sub-net, or host entered above.

Metric — A network cost, a preference value to give to this route. Lower values will be preferred over higher values.

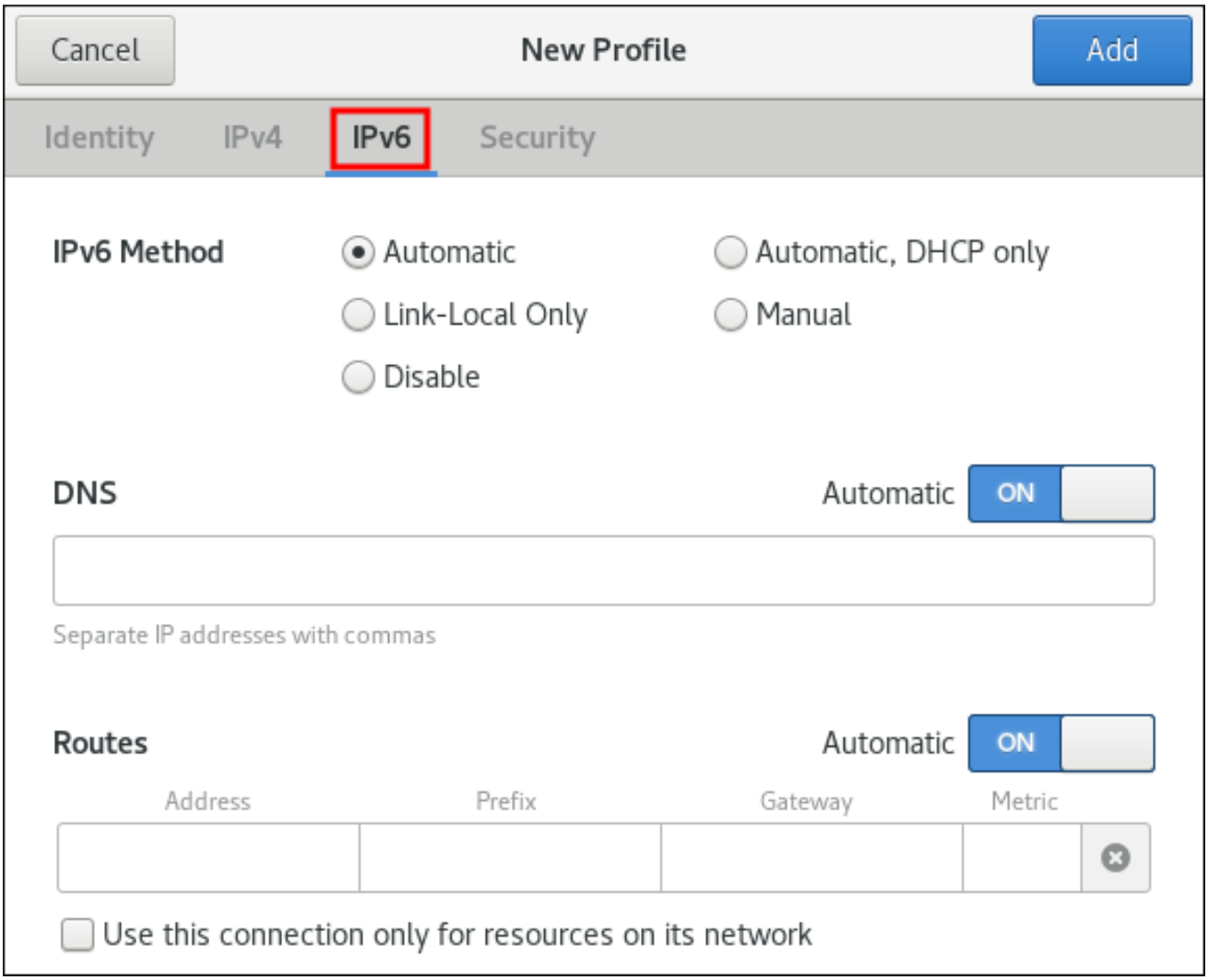
Use this connection only for resources on its network

Select this check box to prevent the connection from becoming the default route. Typical examples are where a connection is a VPN tunnel or a leased line to a head office and you do not want any Internet-bound traffic to pass over the connection. Selecting this option means that only traffic specifically destined for routes learned automatically over the connection or entered here manually will be routed over the connection.

Configuring IPv6 Settings for Wired with control center

Alternatively, to configure **IPv6** settings in a wired connection, click the **IPv6** menu entry:

Figure 6.5. Configuring IPv6 Settings



The **IPv6** menu entry allows you to configure:

- the **IPv6 Method** used to connect to a network
- **DNS** and
- **Routes**

IPv6 Method

Automatic — Choose this option to use **IPv6** Stateless Address AutoConfiguration (SLAAC) to create an automatic, stateless configuration based on the hardware address and Router Advertisements (RA).

Automatic, DHCP only — Choose this option to not use RA, but request information from **DHCPv6** directly to create a stateful configuration.

Link-Local Only — Choose this option if the network you are connecting to does not have a **DHCP** server and you do not want to assign **IP** addresses manually. Random addresses will be assigned as per *RFC 4862* with prefix **FE80::0**.

Manual — Choose this option if you want to assign **IP** addresses manually.

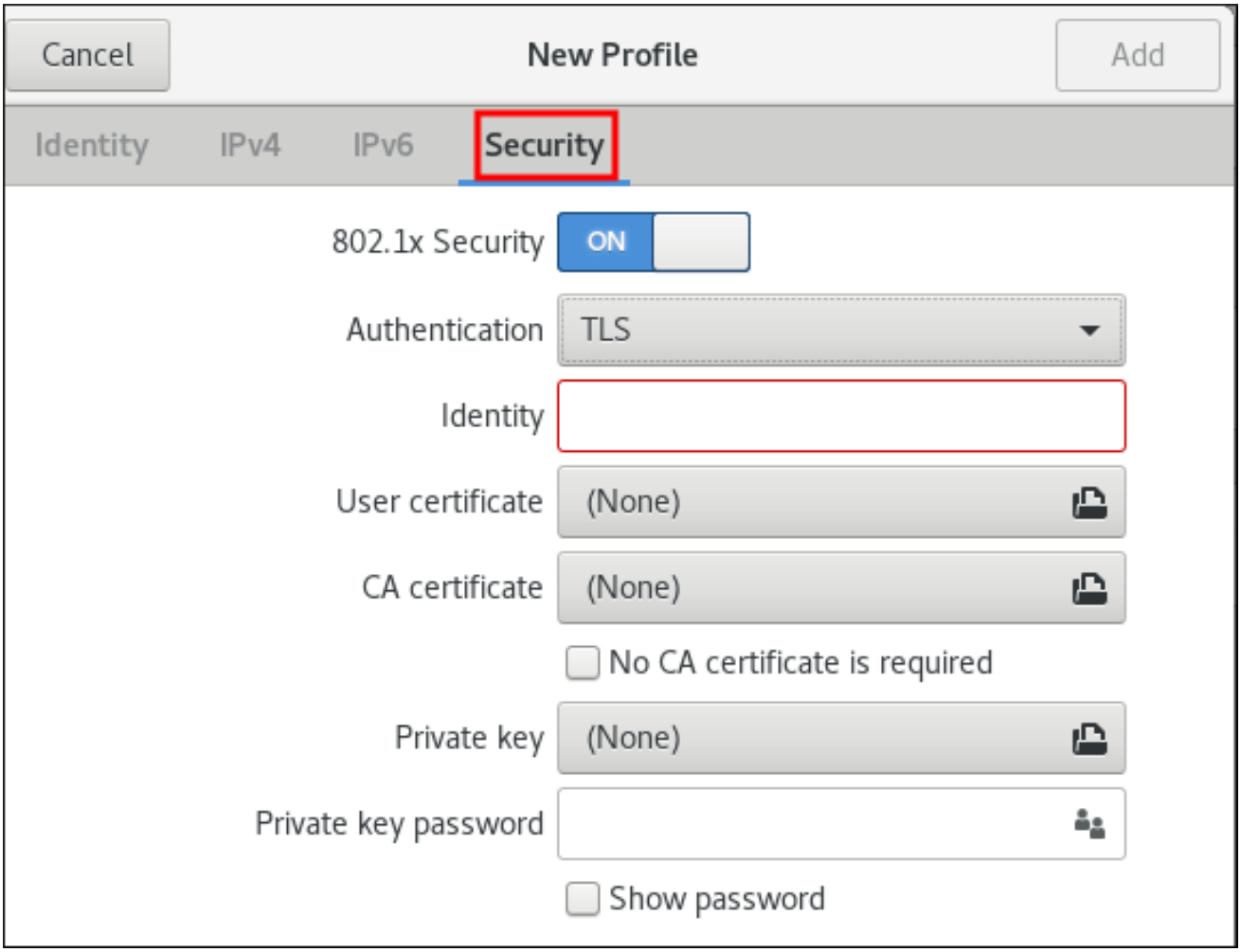
Disabled — **IPv6** is disabled for this connection.

Configuring 802.1X Security for Wired with control-center

802.1X security is the name of the IEEE standard for *port-based Network Access Control* (**PNAC**). It is also called *WPA Enterprise*. 802.1X security is a way of controlling access to a *logical network* from a physical one. All clients who want to join the logical network must authenticate with the server (a router, for example) using the correct 802.1X authentication method.

To configure **`802.1X Security`** settings in a wired connection, click the **`Security`** menu entry:

Figure 6.6. Configuring 802.1X Security for a Wired with control-center



To enable settings configuration, set the symbolic power button to **`ON`**, and select from one of following authentication methods:

- **`TLS`** for *Transport Layer Security* and proceed to Configuring TLS Settings
- **`PWD`** for *Password* and proceed to Configuring PWD Settings
- **`FAST`** for *Flexible Authentication through Secure Tunneling* and proceed to Configuring FAST Settings
- Select **`Tunneled TLS`** for *Tunneled Transport Layer Security*, otherwise known as TTLS, or EAP-TTLS and proceed to Configuring Tunneled TLS Settings
- Select **`Protected EAP (PEAP)`** for *Protected Extensible Authentication Protocol* and proceed to Configuring Protected EAP PEAP Settings

Configuring TLS Settings

With Transport Layer Security (TLS), the client and server mutually authenticate using the TLS protocol.

Using TLS security requires the overhead of a public key infrastructure (PKI) to manage certificates. The benefit of using TLS security is that a compromised password does not allow access to the (W)LAN: an intruder must also have access to the authenticating client's private key.

NetworkManager does not determine the version of TLS supported. **NetworkManager** gathers the parameters entered by the user and passes them to the daemon, **wpa_supplicant**, that handles the procedure. It in turn uses OpenSSL to establish the TLS tunnel. OpenSSL itself negotiates the SSL/TLS protocol version. It uses the highest version both ends support.

To configure TLS settings, follow the procedure described in Section 6.3.1, "Configuring a Wired (Ethernet) Connection Using control-center". The following configuration settings are available:

`Identity`

Provide the identity of this server.

`User certificate`

Click to browse for, and select, a personal X.509 certificate file encoded with *Distinguished Encoding Rules* (**DER**) or *Privacy Enhanced Mail* (**PEM**).

`CA certificate`

Click to browse for, and select, an X.509 *certificate authority* certificate file encoded with *Distinguished Encoding Rules* (**DER**) or *Privacy Enhanced Mail* (**PEM**).

`Private key`

Click to browse for, and select, a *private key* file encoded with *Distinguished Encoding Rules* (**DER**), *Privacy Enhanced Mail* (**PEM**), or the *Personal Information Exchange Syntax Standard* (**PKCS #12**).

`Private key password`

Enter the password for the private key in the **`Private key`** field. Select **`Show password`** to make the password visible as you type it.

Configuring PWD Settings

With Password (PWD), you can specify the username and the password.

`Username`

Enter the user name to be used in the authentication process.

`Password`

Enter the password to be used in the authentication process.

Configuring FAST Settings

To configure FAST settings, follow the procedure described in Section 6.3.1, "Configuring a Wired (Ethernet) Connection Using control-center". The following configuration settings are available:

`Anonymous Identity`

Provide the identity of this server.

`Allow automatic PAC provisioning`

Select the check box to enable and then select from Anonymous, Authenticated, and Both.

`PAC file`

Click to browse for, and select, a *protected access credential* (**PAC**) file.

`Inner authentication`

`GTC` — Generic Token Card.

`MSCHAPv2` — Microsoft Challenge Handshake Authentication Protocol version 2.

Username

Enter the user name to be used in the authentication process.

Password

Enter the password to be used in the authentication process.

Configuring Tunneled TLS Settings

To configure Tunneled TLS settings, follow the procedure described in Section 6.3.1, "Configuring a Wired (Ethernet) Connection Using control-center". The following configuration settings are available:

Anonymous identity

This value is used as the unencrypted identity.

CA certificate

Click to browse for, and select, a Certificate Authority's certificate.

Inner authentication

PAP — Password Authentication Protocol.

MSCHAP — Challenge Handshake Authentication Protocol.

MSCHAPv2 — Microsoft Challenge Handshake Authentication Protocol version 2.

MSCHAPv2 (no EAP) — Microsoft Challenge Handshake Authentication Protocol version 2 without Extensive Authentication Protocol.

CHAP — Challenge Handshake Authentication Protocol.

MD5 — Message Digest 5, a cryptographic hash function.

GTC — Generic Token Card.

Username

Enter the user name to be used in the authentication process.

Password

Enter the password to be used in the authentication process.

Configuring Protected EAP (PEAP) Settings

To configure Protected EAP (PEAP) settings, follow the procedure described in Section 6.3.1, "Configuring a Wired (Ethernet) Connection Using control-center". The following configuration settings are available:

Anonymous Identity

This value is used as the unencrypted identity.

CA certificate

Click to browse for, and select, a Certificate Authority's certificate.

PEAP version

The version of Protected EAP to use. Automatic, 0 or 1.

Inner authentication

MSCHAPv2 — Microsoft Challenge Handshake Authentication Protocol version 2.

MD5 — Message Digest 5, a cryptographic hash function.

GTC — Generic Token Card.

Username

Enter the user name to be used in the authentication process.

Password

Enter the password to be used in the authentication process.

CHAPTER 7. GETTING STARTED WITH IPVLAN

This document describes the IPVLAN driver.

7.1. IPVLAN OVERVIEW

IPVLAN is a driver for a virtual network device that can be used in container environment to access the host network. IPVLAN exposes a single MAC address to the external network regardless the number of IPVLAN device created inside the host network. This means that a user can have multiple IPVLAN devices in multiple containers and the corresponding switch reads a single MAC address. IPVLAN driver is useful when the local switch imposes constraints on the total number of MAC addresses that it can manage.

7.2. IPVLAN MODES

The following modes are available for IPVLAN:

- **L2 mode**
  In IPVLAN **L2 mode**, virtual devices receive and respond to Address Resolution Protocol (ARP) requests. The **`netfilter`** framework runs only inside the container that owns the virtual device. No **`netfilter`** chains are executed in the default namespace on the containerized traffic. Using **L2 mode** provides good performance, but less control on the network traffic.

- **L3 mode**
  In **L3 mode**, virtual devices process only **L3** traffic and above. Virtual devices do not respond to ARP request and users must configure the neighbour entries for the IPVLAN IP addresses on the relevant peers manually. The egress traffic of a relevant container is landed on the **`netfilter`** POSTROUTING and OUTPUT chains in the default namespace while the ingress traffic is threaded in the same way as **L2 mode**. Using **L3 mode** provides good control but decreases the network traffic performance.

- **L3S mode**
  In **L3S mode**, virtual devices process the same way as in **L3 mode**, except that both egress and ingress traffics of a relevant container are landed on **`netfilter`** chain in the default namespace. **L3S mode** behaves in a similar way to **L3 mode** but provides greater control of the network.

NOTE

The IPVLAN virtual device does not receive broadcast and multicast traffic in case of **L3** and **L3S modes**.

CHAPTER 8. USING AND CONFIGURING FIREWALLS

A *firewall* is a way to protect machines from any unwanted traffic from outside. It enables users to control incoming network traffic on host machines by defining a set of *firewall rules*. These rules are used to sort the incoming traffic and either block it or allow through.

8.1. GETTING STARTED WITH `FIREWALLD`

`firewalld` is a firewall service daemon that provides a dynamic customizable host-based firewall with a **`D-Bus`** interface. Being dynamic, it enables creating, changing, and deleting the rules without the necessity to restart the firewall daemon each time the rules are changed.

`firewalld` uses the concepts of *zones* and *services*, that simplify the traffic management. Zones are predefined sets of rules. Network interfaces and sources can be assigned to a zone. The traffic allowed depends on the network your computer is connected to and the security level this network is assigned. Firewall services are predefined rules that cover all necessary settings to allow incoming traffic for a specific service and they apply within a zone.

Services use one or more *ports* or *addresses* for network communication. Firewalls filter communication based on ports. To allow network traffic for a service, its ports must be *open*. **`firewalld`** blocks all traffic on ports that are not explicitly set as open. Some zones, such as *trusted*, allow all traffic by default.

8.1.1. Zones

`firewalld` can be used to separate networks into different zones according to the level of trust that the user has decided to place on the interfaces and traffic within that network. A connection can only be part of one zone, but a zone can be used for many network connections.

NetworkManager notifies **`firewalld`** of the zone of an interface. You can assign zones to interfaces with **NetworkManager**, with the **firewall-config** tool, or the **`firewall-cmd`** command-line tool. The latter two only edit the appropriate **NetworkManager** configuration files. If you change the zone of the interface using **`firewall-cmd`** or **firewall-config**, the request is forwarded to **NetworkManager** and is not handled by **`firewalld`**.

The predefined zones are stored in the **`/usr/lib/firewalld/zones/`** directory and can be instantly applied to any available network interface. These files are copied to the **`/etc/firewalld/zones/`** directory only after they are modified. The following table describes the default settings of the predefined zones:

`block`

Any incoming network connections are rejected with an icmp-host-prohibited message for **`IPv4`** and icmp6-adm-prohibited for **`IPv6`**. Only network connections initiated from within the system are possible.

`dmz`

For computers in your demilitarized zone that are publicly-accessible with limited access to your internal network. Only selected incoming connections are accepted.

`drop`

Any incoming network packets are dropped without any notification. Only outgoing network connections are possible.

`external`

For use on external networks with masquerading enabled, especially for routers. You do not trust the other computers on the network to not harm your computer. Only selected incoming connections are accepted.

`home`

For use at home when you mostly trust the other computers on the network. Only selected incoming connections are accepted.

`internal`

For use on internal networks when you mostly trust the other computers on the network. Only selected incoming connections are accepted.

`public`

For use in public areas where you do not trust other computers on the network. Only selected incoming connections are accepted.

`trusted`

All network connections are accepted.

`work`

For use at work where you mostly trust the other computers on the network. Only selected incoming connections are accepted.

One of these zones is set as the *default* zone. When interface connections are added to **NetworkManager**, they are assigned to the default zone. On installation, the default zone in **`firewalld`** is set to be the **`public`** zone. The default zone can be changed.

NOTE

The network zone names have been chosen to be self-explanatory and to allow users to quickly make a reasonable decision. To avoid any security problems, review the default zone configuration and disable any unnecessary services according to your needs and risk assessments.

8.1.2. Predefined services

A service can be a list of local ports, protocols, source ports, and destinations, as well as a list of firewall helper modules automatically loaded if a service is enabled. Using services saves users time because they can achieve several tasks, such as opening ports, defining protocols, enabling packet forwarding and more, in a single step, rather than setting up everything one after another.

Service configuration options and generic file information are described in the **`firewalld.service(5)`** man page. The services are specified by means of individual XML configuration files, which are named in the following format: ***`service-name`*`.xml`**. Protocol names are preferred over service or application names in **`firewalld`**.

8.1.3. Runtime and permanent settings

Any changes committed in *runtime* mode only apply while **`firewalld`** is running. When **`firewalld`** is restarted, the settings revert to their *permanent* values.

To make the changes persistent across reboots, apply them again using the **`--permanent`** option. Alternatively, to make changes persistent while **`firewalld`** is running, use the **`--runtime-to-permanent`** **`firewall-cmd`** option.

If you set the rules while **firewalld** is running using only the **--permanent** option, they do not become effective before **firewalld** is restarted. However, restarting **firewalld** closes all open ports and stops the networking traffic.

8.1.4. Modifying settings in runtime and permanent configuration using CLI

Using the CLI, you do not modify the firewall settings in both modes at the same time. You only modify either runtime or permanent mode. To modify the firewall settings in the permanent mode, use the **--permanent** option with the **firewall-cmd** command.

```
~]# firewall-cmd --permanent <other options>
```

Without this option, the command modifies runtime mode.

To change settings in both modes, you can use two methods:

1. Change runtime settings and then make them permanent as follows:

```
~]# firewall-cmd <other options>
~]# firewall-cmd --runtime-to-permanent
```

2. Set permanent settings and reload the settings into runtime mode:

```
~]# firewall-cmd --permanent <other options>
~]# firewall-cmd --reload
```

The first method allows you to test the settings before you apply them to the permanent mode.

NOTE

It is possible, especially on remote systems, that an incorrect setting results in a user locking themselves out of a machine. To prevent such situations, use the **--timeout** option. After a specified amount of time, any change reverts to its previous state. Using this options excludes the **--permanent** option.

For example, to add the **SSH** service for 15 minutes:

```
~]# firewall-cmd --add-service=ssh --timeout 15m
```

8.2. INSTALLING THE FIREWALL-CONFIG GUI CONFIGURATION TOOL

To use the **firewall-config** GUI configuration tool, install the **firewall-config** package as **root**:

```
~]# yum install firewall-config
```

Alternatively, in **GNOME**, use the **Super** key and type **Software** to launch the **Software Sources** application. Type **firewall** to the search box, which appears after selecting the search button in the top-right corner. Select the **Firewall** item from the search results, and click on the **Install** button.

To run **firewall-config**, use either the **firewall-config** command or press the **Super** key to enter the **Activities Overview**, type **firewall**, and press **Enter**.

8.3. VIEWING THE CURRENT STATUS AND SETTINGS OF FIREWALLD

8.3.1. Viewing the current status of firewalld

The firewall service, **firewalld**, is installed on the system by default. Use the **firewalld** CLI interface to check that the service is running.

To see the status of the service:

```
~]# firewall-cmd --state
```

For more information about the service status, use the **systemctl status** sub-command:

```
~]# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
vendor pr
   Active: active (running) since Mon 2017-12-18 16:05:15 CET; 50min ago
     Docs: man:firewalld(1)
 Main PID: 705 (firewalld)
    Tasks: 2 (limit: 4915)
   CGroup: /system.slice/firewalld.service
           └─705 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
```

Furthermore, it is important to know how **firewalld** is set up and which rules are in force before you try to edit the settings. To display the firewall settings, see Section 8.3.2, “Viewing current **firewalld** settings”

8.3.2. Viewing current firewalld settings

8.3.2.1. Viewing allowed services using GUI

To view the list of services using the graphical **firewall-config** tool, press the **Super** key to enter the Activities Overview, type **firewall**, and press **Enter**. The **firewall-config** tool appears. You can now view the list of services under the **Services** tab.

Alternatively, to start the graphical firewall configuration tool using the command-line, enter the following command:

```
~]$ firewall-config
```

The **Firewall Configuration** window opens. Note that this command can be run as a normal user, but you are prompted for an administrator password occasionally.

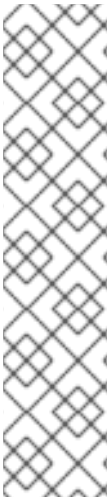
8.3.2.2. Viewing firewalld settings using CLI

With the CLI client, it is possible to get different views of the current firewall settings. The **--list-all** option shows a complete overview of the **firewalld** settings.

firewalld uses zones to manage the traffic. If a zone is not specified by the **--zone** option, the command is effective in the default zone assigned to the active network interface and connection.

To list all the relevant information for the default zone:

```
~]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```



NOTE

To specify the zone for which to display the settings, add the **`--zone=`*`zone-name`*** argument to the **`firewall-cmd --list-all`** command, for example:

```
~]# firewall-cmd --list-all --zone=home
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh mdns samba-client dhcpv6-client
... [output truncated]
```

To see the settings for particular information, such as services or ports, use a specific option. See the **`firewalld`** manual pages or get a list of the options using the command help:

```
~]# firewall-cmd --help

Usage: firewall-cmd [OPTIONS...]

General Options
  -h, --help           Prints a short help text and exists
  -V, --version        Print the version string of firewalld
  -q, --quiet          Do not print status messages

Status Options
  --state              Return and print firewalld state
  --reload             Reload firewall and keep state information
... [output truncated]
```

For example, to see which services are allowed in the current zone:

```
~]# firewall-cmd --list-services
ssh dhcpv6-client
```

Listing the settings for a certain subpart using the CLI tool can sometimes be difficult to interpret. For example, you allow the **SSH** service and **firewalld** opens the necessary port (22) for the service. Later, if you list the allowed services, the list shows the **SSH** service, but if you list open ports, it does not show any. Therefore, it is recommended to use the **--list-all** option to make sure you receive a complete information.

=== To start **firewalld**, enter the following command as **root**:

```
~]# systemctl unmask firewalld
~]# systemctl start firewalld
```

To ensure **firewalld** starts automatically at system start, enter the following command as **root**:

```
~]# systemctl enable firewalld
```

8.4. STOPPING FIREWALLD

To stop **firewalld**, enter the following command as **root**:

```
~]# systemctl stop firewalld
```

To prevent **firewalld** from starting automatically at system start, enter the following command as **root**:

```
~]# systemctl disable firewalld
```

To make sure firewalld is not started by accessing the **firewalld D-Bus** interface and also if other services require **firewalld**, enter the following command as **root**:

```
~]# systemctl mask firewalld
```

8.5. CONTROLLING TRAFFIC

8.5.1. Predefined services

Services can be added and removed using the graphical **firewall-config** tool, **firewall-cmd**, and **firewall-offline-cmd**.

Alternatively, you can edit the XML files in the **/etc/firewalld/services/** directory. If a service is not added or changed by the user, then no corresponding XML file is found in **/etc/firewalld/services/**. The files in the **/usr/lib/firewalld/services/** directory can be used as templates if you want to add or change a service.

8.5.1.1. Disabling all traffic in case of emergency using CLI

In an emergency situation, such as a system attack, it is possible to disable all network traffic and cut off the attacker.

To immediately disable networking traffic, switch panic mode on:

```
~]# firewall-cmd --panic-on
```

IMPORTANT

Enabling panic mode stops all networking traffic. From this reason, it should be used only when you have the physical access to the machine or if you are logged in using a serial console.

Switching off panic mode reverts the firewall to its permanent settings. To switch panic mode off:

```
~]# firewall-cmd --panic-off
```

To see whether panic mode is switched on or off, use:

```
~]# firewall-cmd --query-panic
```

8.5.2. Controlling traffic with predefined services using CLI

The most straightforward method to control traffic is to add a predefined service to **firewalld**. This opens all necessary ports and modifies other settings according to the *service definition file*.

1. Check that the service is not already allowed:

```
~]# firewall-cmd --list-services
ssh dhcpv6-client
```

2. List all predefined services:

```
~]# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client
bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc ceph ceph-
mon cfengine condor-collector ctdb dhcp dhcpv6 dhcpv6-client dns
docker-registry ...
[output truncated]
```

3. Add the service to the allowed services:

```
~]# firewall-cmd --add-service=<service-name>
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.5.3. Controlling traffic with predefined services using GUI

To enable or disable a predefined or custom service, start the **firewall-config** tool and select the network zone whose services are to be configured. Select the **Services** tab and select the check box for each type of service you want to trust. Clear the check box to block a service.

To edit a service, start the **firewall-config** tool and select **Permanent** from the menu labeled **Configuration**. Additional icons and menu buttons appear at the bottom of the **Services** window. Select the service you want to configure.

The **Ports**, **Protocols**, and **Source Port** tabs enable adding, changing, and removing of ports, protocols, and source port for the selected service. The modules tab is for configuring **Netfilter** helper modules. The **Destination** tab enables limiting traffic to a particular destination address and Internet Protocol (**IPv4** or **IPv6**).

NOTE

It is not possible to alter service settings in **Runtime** mode.

8.5.4. Adding new services

Services can be added and removed using the graphical **firewall-config** tool, **firewall-cmd**, and **firewall-offline-cmd**. Alternatively, you can edit the XML files in **/etc/firewalld/services/**. If a service is not added or changed by the user, then no corresponding XML file are found in **/etc/firewalld/services/**. The files **/usr/lib/firewalld/services/** can be used as templates if you want to add or change a service.

To add a new service in a terminal, use **firewall-cmd**, or **firewall-offline-cmd** in case of not active **firewalld**. enter the following command to add a new and empty service:

```
~]$ firewall-cmd --new-service=service-name
```

To add a new service using a local file, use the following command:

```
~]$ firewall-cmd --new-service-from-file=service-name.xml
```

You can change the service name with the additional **--name=*service-name*** option.

As soon as service settings are changed, an updated copy of the service is placed into **/etc/firewalld/services/**.

As **root**, you can enter the following command to copy a service manually:

```
~]# cp /usr/lib/firewalld/services/service-name.xml
/etc/firewalld/services/service-name.xml
```

firewalld loads files from **/usr/lib/firewalld/services** in the first place. If files are placed in **/etc/firewalld/services** and they are valid, then these will override the matching files from **/usr/lib/firewalld/services**. The overriden files in **/usr/lib/firewalld/services** are used as soon as the matching files in **/etc/firewalld/services** have been removed or if **firewalld** has been asked to load the defaults of the services. This applies to the permanent environment only. A reload is needed to get these fallbacks also in the runtime environment.

8.5.5. Controlling ports using CLI

Ports are logical devices that enable an operating system to receive and distinguish network traffic and forward it accordingly to system services. These are usually represented by a daemon that listens on the port, that is it waits for any traffic coming to this port.

Normally, system services listen on standard ports that are reserved for them. The **httpd** daemon, for example, listens on port 80. However, system administrators by default configure daemons to listen on different ports to enhance security or for other reasons.

Opening a Port

Through open ports, the system is accessible from the outside, which represents a security risk. Generally, keep ports closed and only open them if they are required for certain services.

To get a list of open ports in the current zone:

1. List all allowed ports:

```
~]# firewall-cmd --list-ports
```

2. Add a port to the allowed ports to open it for incoming traffic:

```
~]# firewall-cmd --add-port=port-number/port-type
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

The port types are either **tcp**, **udp**, **sctp**, or **dccp**. The type must match the type of network communication.

Closing a Port

When an open port is no longer needed, close that port in **firewalld**. It is highly recommended to close all unnecessary ports as soon as they are not used because leaving a port open represents a security risk.

To close a port, remove it from the list of allowed ports:

1. List all allowed ports:

```
~]# firewall-cmd --list-ports
[WARNING]
====
This command will only give you a list of ports that have been
opened as ports. You will not be able to see any open ports that
have been opened as a service. Therefore, you should consider using
the --list-all option instead of --list-ports.
====
```

2. Remove the port from the allowed ports to close it for the incoming traffic:

```
~]# firewall-cmd --remove-port=port-number/port-type
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.5.6. Opening ports using GUI

To permit traffic through the firewall to a certain port, start the **firewall-config** tool and select the network zone whose settings you want to change. Select the **Ports** tab and click the **Add** button on the right-hand side. The **Port and Protocol** window opens.

Enter the port number or range of ports to permit. Select **tcp** or **udp** from the list.

8.5.7. Controlling traffic with protocols using GUI

To permit traffic through the firewall using a certain protocol, start the **firewall-config** tool and select the network zone whose settings you want to change. Select the **Protocols** tab and click the **Add** button on the right-hand side. The **Protocol** window opens.

Either select a protocol from the list or select the **Other Protocol** check box and enter the protocol in the field.

8.5.8. Opening source ports using GUI

To permit traffic through the firewall from a certain port, start the firewall-config tool and select the network zone whose settings you want to change. Select the **Source Port** tab and click the **Add** button on the right-hand side. The **Source Port** window opens.

Enter the port number or range of ports to permit. Select **tcp** or **udp** from the list.

8.6. WORKING WITH ZONES

Zones represent a concept to manage incoming traffic more transparently. The zones are connected to networking interfaces or assigned a range of source addresses. You manage firewall rules for each zone independently, which enables you to define complex firewall settings and apply them to the traffic.

8.6.1. Listing zones

To see which zones are available on your system:

```
~]# firewall-cmd --get-zones
```

The **firewall-cmd --get-zones** command displays all zones that are available on the system, but it does not show any details for particular zones.

To see detailed information for all zones:

```
~]# firewall-cmd --list-all-zones
```

To see detailed information for a specific zone:

```
~]# firewall-cmd --zone=zone-name --list-all
```

8.6.2. Modifying `firewalld` settings for a certain zone

The Section 8.5.2, “Controlling traffic with predefined services using CLI” and Section 8.5.5, “Controlling ports using CLI” explain how to add services or modify ports in the scope of the current working zone. Sometimes, it is required to set up rules in a different zone.

To work in a different zone, use the **--zone=*zone-name*** option. For example, to allow the **SSH** service in the zone *public*:

```
~]# firewall-cmd --add-service=ssh --zone=public
```

8.6.3. Changing the default zone

System administrators assign a zone to a networking interface in its configuration files. If an interface is not assigned to a specific zone, it is assigned to the default zone. After each restart of the **firewalld** service, **firewalld** loads the settings for the default zone and makes it active.

To set up the default zone:

1. Display the current default zone:

```
~]# firewall-cmd --get-default-zone
```

2. Set the new default zone:

```
~]# firewall-cmd --set-default-zone zone-name
```

NOTE

Following this procedure, the setting is a permanent setting, even without the **--permanent** option.

8.6.4. Assigning a network interface to a zone

It is possible to define different sets of rules for different zones and then change the settings quickly by changing the zone for the interface that is being used. With multiple interfaces, a specific zone can be set for each of them to distinguish traffic that is coming through them.

To assign the zone to a specific interface:

1. List the active zones and the interfaces assigned to them:

```
~]# firewall-cmd --get-active-zones
```

2. Assign the interface to a different zone:

```
~]# firewall-cmd --zone=zone-name --change-interface=<interface-
name>
```

NOTE

You do not have to use the **--permanent** option to make the setting persistent across restarts. If you set a new default zone, the setting becomes permanent.

8.6.5. Assigning a default zone to a network connection

When the connection is managed by **NetworkManager**, it must be aware of a zone that it uses. For every network connection, a zone can be specified, which provides the flexibility of various firewall

settings according to the location of the computer with portable devices. Thus, zones and settings can be specified for different locations, such as company or home.

To set a default zone for an Internet connection, use either the **NetworkManager** GUI or edit the **/etc/sysconfig/network-scripts/ifcfg-*connection-name*** file and add a line that assigns a zone to this connection:

```
ZONE=zone-name
```

8.6.6. Creating a new zone

To use custom zones, create a new zone and use it just like a predefined zone. New zones require the **--permanent** option, otherwise the command does not work.

To create a new zone:

1. Create a new zone:

```
~]# firewall-cmd --new-zone=zone-name
```

2. Check if the new zone is added to your permanent settings:

```
~]# firewall-cmd --get-zones
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.6.7. Creating a new zone using a configuration file

Zones can also be created using a *zone configuration file*. This approach can be helpful when you need to create a new zone, but want to reuse the settings from a different zone and only alter them a little.

A **firewalld** zone configuration file contains the information for a zone. These are the zone description, services, ports, protocols, icmp-blocks, masquerade, forward-ports and rich language rules in an XML file format. The file name has to be ***zone-name*.xml** where the length of *zone-name* is currently limited to 17 chars. The zone configuration files are located in the **/usr/lib/firewalld/zones/** and **/etc/firewalld/zones/** directories.

The following example shows a configuration that allows one service (**SSH**) and one port range, for both the **TCP** and **UDP** protocols.:

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>My zone</short>
  <description>Here you can describe the characteristic features of the
zone.</description>
  <service name="ssh"/>
  <port port="1025-65535" protocol="tcp"/>
  <port port="1025-65535" protocol="udp"/>
</zone>
```

To change settings for that zone, add or remove sections to add ports, forward ports, services, and so on. For more information, see the **firewalld.zone** manual pages.

8.6.8. Using zone targets to set default behavior for incoming traffic

For every zone, you can set a default behavior that handles incoming traffic that is not further specified. Such behaviour is defined by setting the target of the zone. There are three options - **default**, **ACCEPT**, **REJECT**, and **DROP**. By setting the target to **ACCEPT**, you accept all incoming packets except those disabled by a specific rule. If you set the target to **REJECT** or **DROP**, you disable all incoming packets except those that you have allowed in specific rules. When packets are rejected, the source machine is informed about the rejection, while there is no information sent when the packets are dropped.

To set a target for a zone:

1. List the information for the specific zone to see the default target:

```
~]$ firewall-cmd --zone=zone-name --list-all
```

2. Set a new target in the zone:

```
~]# firewall-cmd --zone=zone-name --set-target=
<default|ACCEPT|REJECT|DROP>
```

8.7. USING ZONES TO MANAGE INCOMING TRAFFIC DEPENDING ON A SOURCE

You can use zones to manage incoming traffic based on its source. That enables you to sort incoming traffic and route it through different zones to allow or disallow services that can be reached by that traffic.

If you add a source to a zone, the zone becomes active and any incoming traffic from that source will be directed through it. You can specify different settings for each zone, which is applied to the traffic from the given sources accordingly. You can use more zones even if you only have one network interface.

8.7.1. Adding a source

To route incoming traffic into a specific source, add the source to that zone. The source can be an IP address or an IP mask in the Classless Inter-domain Routing (CIDR) notation.

1. To set the source in the current zone:

```
~]# firewall-cmd --add-source=<source>
```

2. To set the source IP address for a specific zone:

```
~]# firewall-cmd --zone=zone-name --add-source=<source>
```

The following procedure allows all incoming traffic from *192.168.2.15* in the **trusted** zone:

1. List all available zones:

```
~]# firewall-cmd --get-zones
```

2. Add the source IP to the trusted zone in the permanent mode:

```
~]# firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.7.2. Removing a source

Removing a source from the zone cuts off the traffic coming from it.

1. List allowed sources for the required zone:

```
~]# firewall-cmd --zone=zone-name --list-sources
```

2. Remove the source from the zone permanently:

```
~]# firewall-cmd --zone=zone-name --remove-source=<source>
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.7.3. Adding a source port

To enable sorting the traffic based on a port of origin, specify a source port using the **`--add-source-port`** option. You can also combine this with the **`--add-source`** option to limit the traffic to a certain IP address or IP range.

To add a source port:

```
~]# firewall-cmd --zone=zone-name --add-source-port=<port-
name>/<tcp|udp|sctp|dccp>
```

8.7.4. Removing a source port

By removing a source port you disable sorting the traffic based on a port of origin.

To remove a source port:

```
~]# firewall-cmd --zone=zone-name --remove-source-port=<port-
name>/<tcp|udp|sctp|dccp>
```

8.7.5. Using zones and sources to allow a service for only a specific domain

To allow traffic from a specific network to use a service on a machine, use zones and source.

For example, to allow traffic from *192.168.1.0/24* to be able to reach the *HTTP* service while any other traffic is blocked:

1. List all available zones:

```
~]# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

2. Add the source to the trusted zone to route the traffic originating from the source through the zone:

```
~]# firewall-cmd --zone=trusted --add-source=192.168.1.0/24
```

3. Add the *http* service in the trusted zone:

```
~]# firewall-cmd --zone=trusted -add-service=http
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

5. Check that the trusted zone is active and that the service is allowed in it:

```
~]# firewall-cmd --zone=trusted --list-all
trusted (active)
target: ACCEPT
sources: 192.168.1.0/24
services: http
```

8.7.6. Configuring traffic accepted by a zone based on a protocol

You can allow incoming traffic to be accepted by a zone based on a protocol. All traffic using the specified protocol is accepted by a zone, in which you can apply further rules and filtering.

Adding a protocol to a zone

By adding a protocol to a certain zone, you allow all traffic with this protocol to be accepted by this zone.

To add a protocol to a zone:

```
~]# firewall-cmd --zone=zone-name --add-protocol=port-
name/tcp|udp|sctp|dccp|igmp
```

NOTE

To receive multicast traffic, use the **`igmp`** value with the **`--add-protocol`** option.

Removing a protocol from a zone

By removing a protocol from a certain zone, you stop accepting all traffic based on this protocol by the zone.

To remove a protocol from a zone:

```
~]# firewall-cmd --zone=zone-name --remove-protocol=port-
name/tcp|udp|sctp|dccp|igmp
```

-

8.8. PORT FORWARDING

Using **firewalld**, you can set up ports redirection so that any incoming traffic that reaches a certain port on your system is delivered to another internal port of your choice or to an external port on another machine.

8.8.1. Adding a port to redirect

Before you redirect traffic from one port to another port, or another address, you need to know three things: which port the packets arrive at, what protocol is used, and where you want to redirect them.

To redirect a port to another port:

```
~]# firewall-cmd --add-forward-port=port=port-
number:proto=tcp|udp|sctp|dccp:toport=port-number
```

To redirect a port to another port at a different IP address:

1. Add the port to be forwarded:

```
~]# firewall-cmd --add-forward-port=port=port-
number:proto=tcp|udp:toport=port-number:toaddr=IP/mask
```

2. Enable masquerade:

```
~]# firewall-cmd -add-masquerade
```

[[ex-Redirecting_TCP_Port_80_to_Port_88_on_the_Same_Machine_using-configuring-firewalls']
.Redirecting TCP port 80 to port 88 on the same machine

To redirect the port:

1. Redirect the port 80 to port 88 for TCP traffic:

```
~]# firewall-cmd --add-forward-port=port=80:proto=tcp:toport=88
```

2. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

3. Check that the port is redirected:

```
~]# firewall-cmd --list-all
```

8.8.2. Removing a redirected Port

To remove a redirected port:

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=
<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

To remove a forwarded port redirected to a different address:

1. Remove the forwarded port:

```
~]# firewall-cmd --remove-forward-port=port=port-number:proto=
<tcp|udp>:toport=port-number:toaddr=<IP/mask>
```

2. Disable masquerade:

```
~]# firewall-cmd --remove-masquerade
```

NOTE

Redirecting ports using this method only works for IPv4-based traffic. For IPv6 redirecting setup, you need to use rich rules.

To redirect to an external system, it is necessary to enable masquerading. For more information, see Section 8.9, “Configuring IP address masquerading”.

[[ex-Removing_TCP_Port_80_forwarded_to_Port_88_on_the_Same_Machine_using-configuring-firewalls'] .Removing TCP port 80 forwarded to port 88 on the same machine

To remove the port redirection:

1. List redirected ports:

```
~]# firewall-cmd --list-forward-ports
port=80:proto=tcp:toport=88:toaddr=
```

2. Remove the redirected port from the firewall::

```
~]# firewall-cmd  --remove-forward-
port=port=80:proto=tcp:toport=88:toaddr=
```

3. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.9. CONFIGURING IP ADDRESS MASQUERADING

To check if IP masquerading is enabled (for example, for the **external** zone), enter the following command as **root**:

```
~]# firewall-cmd --zone=external --query-masquerade
```

The command prints **yes** with exit status **0** if enabled. It prints **no** with exit status **1** otherwise. If **zone** is omitted, the default zone will be used.

To enable IP masquerading, enter the following command as **root**:

```
~]# firewall-cmd --zone=external --add-masquerade
```

To make this setting persistent, repeat the command adding the **--permanent** option.

To disable IP masquerading, enter the following command as **root**:

```
~]# firewall-cmd --zone=external --remove-masquerade
```

To make this setting persistent, repeat the command adding the **--permanent** option.

8.10. MANAGING ICMP REQUESTS

The **Internet Control Message Protocol** (**ICMP**) is a supporting protocol that is used by various network devices to send error messages and operational information indicating a connection problem, for example, that a requested service is not available. **ICMP** differs from transport protocols such as TCP and UDP because it is not used to exchange data between systems.

Unfortunately, it is possible to use the **ICMP** messages, especially **echo-request** and **echo-reply**, to reveal information about your network and misuse such information for various kinds of fraudulent activities. Therefore, **firewalld** enables blocking the **ICMP** requests to protect your network information.

8.10.1. Listing ICMP requests

The **ICMP** requests are described in individual XML files that are located in the **/usr/lib/firewalld/icmptypes/** directory. You can read these files to see a description of the request. The **firewall-cmd** command controls the **ICMP** requests manipulation.

To list all available **ICMP** types:

```
~]# firewall-cmd --get-icmptypes
```

The **ICMP** request can be used by IPv4, IPv6, or by both protocols. To see for which protocol the **ICMP** request is used:

```
~]# firewall-cmd --info-icmptype=<icmptype>
```

The status of an **ICMP** request shows **yes** if the request is currently blocked or **no** if it is not. To see if an **ICMP** request is currently blocked:

```
~]# firewall-cmd --query-icmp-block=<icmptype>
```

8.10.2. Blocking or unblocking ICMP requests

When your server blocks **ICMP** requests, it does not provide the information that it normally would. However, that does not mean that no information is given at all. The clients receive information that the particular **ICMP** request is being blocked (rejected). Blocking the **ICMP** requests should be considered carefully, because it can cause communication problems, especially with IPv6 traffic.

To see if an **ICMP** request is currently blocked:

```
~]# firewall-cmd --query-icmp-block=<icmptype>
```

To block an **ICMP** request:

```
~]# firewall-cmd --add-icmp-block=<icmptype>
```

To remove the block for an **ICMP** request:

```
~]# firewall-cmd --remove-icmp-block=<icmptype>
```

8.10.3. Blocking ICMP requests without providing any information at all

Normally, if you block **ICMP** requests, clients know that you are blocking it. So, a potential attacker who is sniffing for live IP addresses is still able to see that your IP address is online. To hide this information completely, you have to drop all **ICMP** requests.

To block and drop all **ICMP** requests:

1. Set the target of your zone to **DROP**:

   ```
   ~]# firewall-cmd --set-target=DROP
   ```

2. Make the new settings persistent:

   ```
   ~]# firewall-cmd --runtime-to-permanent
   ```

Now, all traffic, including **ICMP** requests, is dropped, except traffic which you have explicitly allowed.

To block and drop certain **ICMP** requests and allow others:

1. Set the target of your zone to **DROP**:

   ```
   ~]# firewall-cmd --set-target=DROP
   ```

2. Add the ICMP block inversion to block all **ICMP** requests at once:

   ```
   ~]# firewall-cmd --add-icmp-block-inversion
   ```

3. Add the ICMP block for those **ICMP** requests that you want to allow:

   ```
   ~]# firewall-cmd --add-icmp-block=<icmptype>
   ```

4. Make the new settings persistent:

   ```
   ~]# firewall-cmd --runtime-to-permanent
   ```

The *block inversion* inverts the setting of the **ICMP** requests blocks, so all requests, that were not previously blocked, are blocked. Those that were blocked are not blocked. Which means that if you need to unblock a request, you must use the blocking command.

To revert this to a fully permissive setting:

1. Set the target of your zone to **default** or **ACCEPT**:

```
~]# firewall-cmd --set-target=default
```

2. Remove all added blocks for **ICMP** requests:

```
~]# firewall-cmd --remove-icmp-block=<icmptype>
```

3. Remove the **ICMP** block inversion:

```
~]# firewall-cmd --remove-icmp-block-inversion
```

4. Make the new settings persistent:

```
~]# firewall-cmd --runtime-to-permanent
```

8.10.4. Configuring the ICMP filter using GUI

To enable or disable an **ICMP** filter, start the **firewall-config** tool and select the network zone whose messages are to be filtered. Select the **ICMP Filter** tab and select the check box for each type of **ICMP** message you want to filter. Clear the check box to disable a filter. This setting is per direction and the default allows everything.

To edit an **ICMP** type, start the **firewall-config** tool and select **Permanent** mode from the menu labeled **Configuration**. Additional icons appear at the bottom of the **Services** window. Select **Yes** in the following dialog to enable masquerading and to make forwarding to another machine working.

To enable inverting the **ICMP Filter**, click the **Invert Filter** check box on the right. Only marked **ICMP** types are now accepted, all other are rejected. In a zone using the DROP target, they are dropped.

8.11. SETTING AND CONTROLLING IP SETS USING FIREWALLD

To see the list of IP set types supported by **firewalld**, enter the following command as root.

```
~]# firewall-cmd --get-ipset-types
hash:ip hash:ip,mark hash:ip,port hash:ip,port,ip hash:ip,port,net
hash:mac hash:net hash:net,iface hash:net,net hash:net,port
hash:net,port,net
```

8.11.1. Configuring IP set options with the command-line client

IP sets can be used in **firewalld** zones as sources and also as sources in rich rules. In Red Hat Enterprise Linux, the preferred method is to use the IP sets created with **firewalld** in a direct rule.

To list the IP sets known to **firewalld** in the permanent environment, use the following command as **root**:

```
~]# firewall-cmd --permanent --get-ipsets
```

To add a new IP set, use the following command using the permanent environment as **root**:

```
~]# firewall-cmd --permanent --new-ipset=test --type=hash:net
success
```

The previous command creates a new IP set with the name *test* and the **hash:net** type for **IPv4**. To create an IP set for use with **IPv6**, add the **--option=family=inet6** option. To make the new setting effective in the runtime environment, reload **firewalld**. List the new IP set with the following command as **root**:

```
~]# firewall-cmd --permanent --get-ipsets
test
```

To get more information about the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --info-ipset=test
test
type: hash:net
options:
entries:
```

Note that the IP set does not have any entries at the moment. To add an entry to the *test* IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --add-entry=192.168.0.1
success
```

The previous command adds the IP address *192.168.0.1* to the IP set. To get the list of current entries in the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

Generate a file containing a list of IP addresses, for example:

```
~]# cat > iplist.txt <<EOL
192.168.0.2
192.168.0.3
192.168.1.0/24
192.168.2.254
EOL
```

The file with the list of IP addresses for an IP set should contain an entry per line. Lines starting with a hash, a semi-colon, or empty lines are ignored.

To add the addresses from the *iplist.txt* file, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --add-entries-from-
file=iplist.txt
success
```

To see the extended entries list of the IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
192.168.0.2
```

```
192.168.0.3
192.168.1.0/24
192.168.2.254
```

To remove the addresses from the IP set and to check the updated entries list, use the following commands as **root**:

```
~]# firewall-cmd --permanent --ipset=test --remove-entries-from-
file=iplist.txt
success
~]# firewall-cmd --permanent --ipset=test --get-entries
192.168.0.1
```

You can add the IP set as a source to a zone to handle all traffic coming in from any of the addresses listed in the IP set with a zone. For example, to add the *test* IP set as a source to the *drop* zone to drop all packets coming from all entries listed in the *test* IP set, use the following command as **root**:

```
~]# firewall-cmd --permanent --zone=drop --add-source=ipset:test
success
```

The **ipset:** prefix in the source shows **firewalld** that the source is an IP set and not an IP address or an address range.

Only the creation and removal of IP sets is limited to the permanent environment, all other IP set options can be used also in the runtime environment without the **--permanent** option.

WARNING

Red Hat does not recommend using IP sets that are not managed through **firewalld**. To use such IP sets, a permanent direct rule is required to reference the set, and a custom service must be added to create these IP sets. This service needs to be started before firewalld starts, otherwise **firewalld** is not able to add the direct rules using these sets. You can add permanent direct rules with the **/etc/firewalld/direct.xml** file.

8.12. CONFIGURING FIREWALL LOCKDOWN

Local applications or services are able to change the firewall configuration if they are running as **root** (for example, **libvirt**). With this feature, the administrator can lock the firewall configuration so that either no applications or only applications that are added to the lockdown whitelist are able to request firewall changes. The lockdown settings default to disabled. If enabled, the user can be sure that there are no unwanted configuration changes made to the firewall by local applications or services.

8.12.1. Configuring lockdown with the command-line client

To query whether lockdown is enabled, use the following command as **root**:

```
~]# firewall-cmd --query-lockdown
```

The command prints **yes** with exit status **0** if lockdown is enabled. It prints **no** with exit status **1** otherwise.

To enable lockdown, enter the following command as **root**:

```
~]# firewall-cmd --lockdown-on
```

To disable lockdown, use the following command as **root**:

```
~]# firewall-cmd --lockdown-off
```

8.12.2. Configuring lockdown whitelist options with the command-line client

The lockdown whitelist can contain commands, security contexts, users and user IDs. If a command entry on the whitelist ends with an asterisk "*", then all command lines starting with that command will match. If the "*" is not there then the absolute command including arguments must match.

The context is the security (SELinux) context of a running application or service. To get the context of a running application use the following command:

```
~]$ ps -e --context
```

That command returns all running applications. Pipe the output through the **grep** tool to get the application of interest. For example:

```
~]$ ps -e --context | grep example_program
```

To list all command lines that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-commands
```

To add a command *command* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

To remove a command *command* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

To query whether the command *command* is on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --query-lockdown-whitelist-command='/usr/bin/python3 -Es
/usr/bin/command'
```

The command prints **yes** with exit status **0** if true. It prints **no** with exit status **1** otherwise.

To list all security contexts that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-contexts
```

To add a context *context* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-context=context
```

To remove a context *context* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-context=context
```

To query whether the context *context* is on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --query-lockdown-whitelist-context=context
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

To list all user IDs that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-uids
```

To add a user ID *uid* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-uid=uid
```

To remove a user ID *uid* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-uid=uid
```

To query whether the user ID *uid* is on the whitelist, enter the following command:

```
~]$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

To list all user names that are on the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --list-lockdown-whitelist-users
```

To add a user name *user* to the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --add-lockdown-whitelist-user=user
```

To remove a user name *user* from the whitelist, enter the following command as **root**:

```
~]# firewall-cmd --remove-lockdown-whitelist-user=user
```

To query whether the user name *user* is on the whitelist, enter the following command:

```
~]$ firewall-cmd --query-lockdown-whitelist-user=user
```

Prints **yes** with exit status **0**, if true, prints **no** with exit status **1** otherwise.

8.12.3. Configuring lockdown whitelist options with configuration files

The default whitelist configuration file contains the **NetworkManager** context and the default context of **libvirt**. The user ID 0 is also on the list.

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <selinux context="system_u:system_r:virtd_t:s0-s0:c0.c1023"/>
   <user id="0"/>
 </whitelist>
```

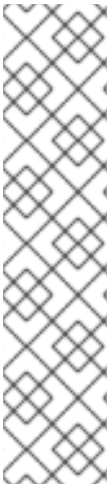
Following is an example whitelist configuration file enabling all commands for the **firewall-cmd** utility, for a user called *user* whose user ID is **815**:

```
<?xml version="1.0" encoding="utf-8"?>
 <whitelist>
   <command name="/usr/bin/python3 -Es /bin/firewall-cmd*"/>
   <selinux context="system_u:system_r:NetworkManager_t:s0"/>
   <user id="815"/>
   <user name="user"/>
 </whitelist>
```

This example shows both **user id** and **user name**, but only one option is required. Python is the interpreter and is prepended to the command line. You can also use a specific command, for example:

```
/usr/bin/python3 /bin/firewall-cmd --lockdown-on
```

In that example, only the **--lockdown-on** command is allowed.



NOTE

In Red Hat Enterprise Linux , all utilities are placed in the **/usr/bin/** directory and the **/bin/** directory is sym-linked to the **/usr/bin/** directory. In other words, although the path for **firewall-cmd** when run as **root** might resolve to **/bin/firewall-cmd**, **/usr/bin/firewall-cmd** can now be used. All new scripts should use the new location. But be aware that if scripts that run as **root** have been written to use the **/bin/firewall-cmd** path, then that command path must be whitelisted in addition to the **/usr/bin/firewall-cmd** path traditionally used only for non-**root** users.

The "*" at the end of the name attribute of a command means that all commands that start with this string will match. If the "*" is not there then the absolute command including arguments must match.

8.13. CONFIGURING LOGGING FOR DENIED PACKETS

With the **LogDenied** option in the **firewalld**, it is possible to add a simple logging mechanism for denied packets. These are the packets that are rejected or dropped. To change the setting of the logging, edit the **/etc/firewalld/firewalld.conf** file or use the command-line or GUI configuration tool.

If **LogDenied** is enabled, logging rules are added right before the reject and drop rules in the INPUT, FORWARD and OUTPUT chains for the default rules and also the final reject and drop rules in zones.

The possible values for this setting are: **all**, **unicast**, **broadcast**, **multicast**, and **off**. The default setting is **off**. With the **unicast**, **broadcast**, and **multicast** setting, the **pkttype** match is used to match the link-layer packet type. With **all**, all packets are logged.

To list the actual **LogDenied** setting with firewall-cmd, use the following command as **root**:

```
~]# firewall-cmd --get-log-denied
off
```

To change the **LogDenied** setting, use the following command as **root**:

```
~]# firewall-cmd --set-log-denied=all
success
```

To change the **LogDenied** setting with the **firewalld** GUI configuration tool, start **firewall-config**, click the **Options** menu and select **Change Log Denied**. The **LogDenied** window appears. Select the new **LogDenied** setting from the menu and click OK.

8.14. ADDITIONAL RESOURCES FOR FIREWALLD

The following sources of information provide additional resources regarding **firewalld**.

8.14.1. Installed documentation

- **firewalld(1)** man page — Describes command options for **firewalld**.
- **firewalld.conf(5)** man page — Contains information to configure **firewalld**.
- **firewall-cmd(1)** man page — Describes command options for the **firewalld** command-line client.
- **firewall-config(1)** man page — Describes settings for the **firewall-config** tool.
- **firewall-offline-cmd(1)** man page — Describes command options for the **firewalld** offline command-line client.
- **firewalld.icmptype(5)** man page — Describes XML configuration files for **ICMP** filtering.
- **firewalld.ipset(5)** man page — Describes XML configuration files for the **firewalld IP** sets.
- **firewalld.service(5)** man page — Describes XML configuration files for **firewalld service**.
- **firewalld.zone(5)** man page — Describes XML configuration files for **firewalld** zone configuration.
- **firewalld.direct(5)** man page — Describes the **firewalld** direct interface configuration file.
- **firewalld.lockdown-whitelist(5)** man page — Describes the **firewalld** lockdown whitelist configuration file.

- **firewalld.richlanguage(5)** man page — Describes the **firewalld** rich language rule syntax.
- **firewalld.zones(5)** man page — General description of what zones are and how to configure them.
- **firewalld.dbus(5)** man page — Describes the **D-Bus** interface of **firewalld**.

8.14.2. Online documentation

- http://www.firewalld.org/ — **firewalld** home page.

8.15. INTRODUCTION TO NFTABLES

The **nftables** framework provides packet classification facilities and it is the designated successor to the **iptables**, **ip6tables**, **arptables**, and **ebtables** tools. It offers numerous improvements in convenience, features, and performance over previous packet-filtering tools, most notably:

- lookup tables instead of linear processing
- a single framework for both the **IPv4** and **IPv6** protocols
- rules all applied atomically instead of fetching, updating, and storing a complete ruleset
- support for debugging and tracing in the ruleset (**nftrace**) and monitoring trace events (in the **nft** tool)
- more consistent and compact syntax, no protocol-specific extensions
- a Netlink API for third-party applications

Similarly to **iptables**, **nftables** use tables for storing chains. The chains contain individual rules for performing actions. The **nft** tool replaces all tools from the previous packet-filtering frameworks. The **libnftnl** library can be used for low-level interaction with **nftables** Netlink API over the **libmnl** library.

In Red Hat Enterprise Linux 8, **nftables** serve as the default **firewalld** backend. Although the **nftables** backend is backward-compatible with the previous **iptables** backend in firewall configurations, you can switch back to **iptables** by setting the **FirewallBackend** option to the **iptables** value in the **/etc/firewalld/firewalld.conf** file.

Additional resources

- The **nft(8)** man page provides a comprehensive reference documentation for configuring and inspecting packet filtering with nftables using the **nft** command-line tool.
- The What comes after iptables? Its successor, of course: nftables article explains why **nftables** replaces **iptables**.
- The Firewalld: The Future is nftables article provides additional information on **nftables** as a default backend for **firewalld**.